VLSI Design

Networks-on-Chip

Guest Editors: Davide Bertozzi, Shashi Kumar,
and Maurizio Palesi

Networks-on-Chip

VLSI Design

Networks-on-Chip

Guest Editors: Davide Bertozzi,
Shashi Kumar, and Maurizio Palesi

Contents

Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 26454, 3 pages
doi:10.1155/2007/26454

Editorial
Networks-on-Chip: Emerging Research Topics and Novel Ideas

Davide Bertozzi,[1] Shashi Kumar,[2] and Maurizio Palesi[3]

[1] *Engineering Department, University of Ferrara, 44100 Ferrara, Italy*
[2] *Department of Electronics and Computer Engineering, School of Engineering, Jönköping University, 55111 Jönköping, Sweden*
[3] *Department of Information Engineering and Telecommunication, University of Catania, 95125 Catania, Italy*

Received 4 April 2007; Accepted 4 April 2007

Networks-on-chip (NoCs) are being devoted intensive research efforts by R&D institutions all around the word, and it is our pleasure to host in this special issue the latest contributions on key design issues at different levels of abstraction, namely, physical link design, architecture design and optimization, performance and power characterization, and design technology. Applying the networking concept to on-chip communication is part of the breakthrough solutions urged by the advances in silicon manufacturing technology (which keeps scaling well beyond 100 nm), by increased time-to-market pressures and by the growing computing requirements of current and future embedded applications. In fact, scalable computation horsepower has been traditionally provided through an increase of clock frequency of monolithic processor cores at each technology node. This trend is, however, running into the barriers of nanoscale technologies, such as heating levels beyond the capability of state-of-the-art packaging and cooling technologies, limited scaling of memory access times and the von Neumann bottleneck. These limitations are being increasingly overcome by breaking up functions into concurrent tasks, assigning them to parallel computational units and operating them at a lower frequency than monolithic cores. This approach paves the way for energy-efficient massively parallel chip-level computation architectures, which are at the core of multiprocessor system-on-chip (MPSoC) technology.

This trend has profound implications on the communication architecture as well, since the communication requirements of an increasing number of processor cores have to be accommodated by the system interconnect. In contrast, state-of-the-art interconnect fabrics will soon incur severe scalability limitations. The International Technology Roadmap for Semiconductors foresees that they will represent the limiting factor for performance and power consumption in next generation SoCs. In the last few years, a number of advances in on-chip interconnect architectures have tried to relieve the limitations of the communication sub-system. First, more parallel topologies have been proposed to increase the amount of delivered bandwidth, such as partial or full crossbars. However, scalability limitations of crossbar-based interconnection fabrics are well known, and they will not be a long-term solution. Second, new communication protocols have been developed, aiming at a more effective exploitation of the available bandwidth. AMBA 3.0 AXI and the open-core protocol (OCP) are examples thereof. Interestingly, these latest protocols provide support for point-to-point communication only (e.g., an IP core with a bus or directly with another IP core) and do not provide any specification on the interconnect fabric, which can (almost) freely evolve in the direction of a higher communication parallelism.

In a short span of seven years, networks-on-chip (NoCs) have been recognized as the most important alternative for the design of modular and scalable communication architectures, providing inherent support to the integration of heterogeneous cores through standard socket interfaces. Not only NoCs relieve system-level integration issues, but are also suitable to deal with the challenges of nanoscale technology. The degradation of the RC propagation delay of signals across global wires is in fact making multiclock cycle signal propagation come true. At the same time, design predictability of global chip-wide structures (like some state-of-the-art system interconnects) is increasingly jeopardized. Through an aggressive path segmentation, NoCs loosen the delay bottleneck of on-chip interconnects and improve design predictability.

Unfortunately, area and power overheads incurred by current NoC prototypes remain still significant in spite of the performance benefits, calling for further research efforts to make this solution more mature and viable from an

industrial viewpoint. Recently, there have been a few books edited describing various aspects of NoC design and implementation. Almost all important international conferences related to electronic design and to its automation have special sessions focusing on NoC-based MPSoC design. This special issue serves the purpose of collecting papers proposing innovative and even exotic solutions to NoC design.

We received twenty-nine submissions, four of which were invited, and a total of ten were accepted. The high level of competition has led to the selection of top-level contributions from renowned academic institutions and industries, covering issues at different levels of the design process. In particular, the topics covered by this special issue include physical link design, NoC architecture design and optimization, power modeling and performance evaluation, mapping and routing strategies. Interestingly, we have challenges associated with nanoscale designs (signal integrity and process variability) addressed at different levels of abstraction, from physical- to system-level design.

NoC physical links are responsible for actual signal propagation among routers and/or network interfaces. The choice of a physical link technology (e.g., serial versus parallel, synchronous versus asynchronous, voltage versus current-mode signaling) has deep implications on even system-level performance, area and power figures due to the relevance of wiring for NoC designs. Moreover, it is at this level that the fundamental challenges of nanoscale technologies have to be primarily tackled. In "Variation-tolerant and low-power source-synchronous multicycle on-chip interconnect scheme," the authors M. Ghoneima et al. address the problem of on-chip communication links whose length makes it impossible to reach the destination in a single clock cycle. They present a variation-tolerant low-power source synchronous multicycle interconnect scheme which is proven to be tolerant to process variations and energy-effective when compared to a traditional pipelined link.

In "High-performance long NoC link using delay-insensitive current mode signaling," the authors E. Nigussie et al. present a high-performance long NoC link implementation based on multilevel current mode signaling and delay-insensitive 1-of-4 encoding. They show that current mode signaling reduces the communication latency of long wires significantly compared to voltage mode signaling, making it possible to achieve high throughput without pipelining and/or using repeaters. In "Online reconfigurable self-timed links for fault tolerant NoC" the authors T. Lehtonen et al. tackle the problem of reliable system design by proposing link structures for NoCs that have properties for effectively tolerating transient, intermittent, and permanent errors. They show how considerable enhancements in fault tolerance can be achieved at the cost of performance and area, and with only a slight increase in power consumption.

At a higher level of abstraction, the behavior of on-chip interconnects, as well as of other NoC building blocks such as switches and network interfaces, can be captured by means of high-level yet accurate models.

In "Area and power modeling for networks-on-chip with layout awareness" the authors P. Meloni et al. present a flow to devise analytical models of area occupation and power consumption of NoC switches for a reference architecture. Such models are parameterized on several architectural, synthesis-related, and traffic variables.

While a lot of prior work focuses on the switch network, protocol interactions between NoC and IP cores should be carefully considered since they introduce message dependencies that affect deadlock properties of the MPSoC as a whole. In "Avoiding message-dependent deadlock in network-based systems on chip" the authors A. Hansson et al. analyse message-dependent deadlock, survey possible solutions, and show that deadlock avoidance, in the presence of higher-level protocols, poses a serious challenge for many current NoC architectures. Finally, the authors evaluate the solutions qualitatively, and for a number of designs they quantify the area cost for the two most economical solutions, namely, strict ordering and end-to-end flow control.

Developing NoC-based systems tailored to a particular application domain is crucial for achieving high-performance, energy-efficient customized solutions. The effectiveness of this approach largely depends on the availability of a design methodology that, starting from a high-level application specification, derives an optimized NoC configuration with respect to different design objectives. Design choices include topology mapping, routing, and assignment of network channel capacity.

In "A unified approach to mapping and routing on a network on chip for both best-effort and guaranteed service traffic," the authors A. Hansson et al. address the problem of spatial mapping of cores and routing of the communication between cores on NoCs. They present a unified single-objective algorithm which couples path selection, mapping of cores and time-division multiplexing time-slot allocation to minimise the network required to meet the constraints of the application. The application of the algorithm to an MPEG decoder SoC results in a quite significant reduction of area, power dissipation, and worst-case latency over a traditional multistep approach.

In "A method for routing packets across multiple paths in NoCs with in-order delivery and fault-tolerance gaurantees," the authors S. Murali et al. present a multipath routing strategy that guarantees in-order packet delivery for NoCs. The strategy is based on the idea of routing packets on partially nonintersecting paths and rebuilding packet order at path reconvergent nodes. The authors present a design methodology that uses the routing strategy to optimally spread the traffic in the NoC to minimize the network bandwidth needs and power consumption. They also integrate support for tolerance against transient and permanent failures in the NoC links in the methodology by utilizing spatial and temporal redundancy for transporting packets.

In "Network delays and link capacities in application-specific wormhole NoCs," the authors Guz et al. consider a NoC-based application-specific SoC scenario, where information traffic is heterogeneous and delay requirements may largely vary. In this context, the individual capacity assignment for each link in the NoC is required. The authors present an analytical delay model for virtual-channeled

wormhole networks with nonuniform links and apply the analysis in devising an efficient capacity allocation algorithm which assigns link capacities such that packet delay requirements for each flow are satisfied.

Finally, the special issue reports two contributions that take a more radical approach to NoC design, based on novel architectures or concepts that represent revolutionary solutions with respect to the common design practice.

In "Comparison of a ring on-chip network and a code-division multiple-access on-chip network," the authors Wang and Nurmi discuss advantages and disadvantages of applying CDMA techniques for on-chip communication, consisting of the multiplexing of data transfers in code domain instead of in time domain. A comparison with a bidirectional ring connection scheme is performed in terms of network structure, data transfer principle, network node design, asynchronous design, and performance.

In "Stochastic communication: a new paradigm for fault-tolerant networks-on-chip," the authors P. Bogdan et al. introduce a novel communication paradigm for SoCs, called stochastic communication, which allows to relax the requirement of 100% correctness for devices and interconnects by providing a high degree of system-level fault-tolerance NoCs. Using this communication scheme, authors show how a large percentage of data upsets, packet losses due to buffers overflow, and severe levels of synchronization failures can be tolerated, while providing high levels of performance.

We sincerely hope you will enjoy this special issue and that it will inspire further research in this very important area of electronic system design.

We would like to thank all authors who submitted papers to this special issue as well as the authors of the invited papers. Special thanks go to the referees for their time and diligence during the review process and for providing us with high-quality reviews. In conclusion, we would like to thank Bernard Courtois, Editor-in-Chief of VLSI Design, for offering us the opportunity to bring about this special issue.

Davide Bertozzi
Shashi Kumar
Maurizio Palesi

Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 95402, 12 pages
doi:10.1155/2007/95402

Research Article

Variation-Tolerant and Low-Power Source-Synchronous Multicycle On-Chip Interconnect Scheme

Maged Ghoneima,[1] Yehea Ismail,[2] Muhammad Khellah,[3] and Vivek De[3]

[1] *VLSI Design Group, NVIDIA Corporation, Santa Clara, CA 95050, USA*
[2] *Electrical Enginering and Computer Science Department (EECS), Northwestern University, Evanston, IL 60208-3118, USA*
[3] *Circuit Research Laboratories, Intel Corporation, Hillsboro, OR 97124, USA*

Received 6 November 2006; Revised 27 February 2007; Accepted 16 March 2007

Recommended by Davide Bertozzi

A variation-tolerant low-power source-synchronous multicycle (SSMC) interconnect scheme is proposed. This scheme is scalable and suitable for transferring data across different clock domains such as those in "many-core" SoCs and in 3D-ICs. SSMC replaces intermediate flip-flops by a source-synchronous synchronization scheme. Removing the intermediate flip-flops in the SSMC scheme enables better averaging of delay variations across the whole interconnect, which reduces bit-rate degradation due to within-die WID process variations. Monte Carlo circuit simulations show that SSMC eliminates 90% of the variation-induced performance degradation in a 6-cycle 9 mm-long 16-bit conventional bus. The proposed multicycle bus scheme also leads to significant energy savings due to eliminating the power-hungry flip-flops and efficiently designing the source synchronization overhead. Moreover, eliminating intermediate flip-flops avoids the timing overhead of the setup time, the flip-flop delay, and the single-cycle clock jitter. This delay slack can then be translated into further energy savings by downsizing the repeaters. The significant delay jitter due to capacitive coupling has been addressed and solutions are put forward to alleviate it. Circuit simulations in a 65-nm process environment indicate that energy savings up to 20% are achievable for a 6-cycle 9 mm long 16-bit bus.

1. INTRODUCTION

Today's system-on-ship (SoC) devices are targeting complex applications, where there is a need for a significant amount of computing power and data transfer. This implies that the number of on-chip modules will increase, and so will the number of on-chip buses connecting these modules. With the continuous scaling of technology, increased die area and faster clock speeds, the delay and power dissipation of on-chip buses are becoming one of the main bottlenecks in current high-performance SoC design [1].

As technology scales and die sizes increase, gate delay improves, but on the contrary, interconnect delay is increasing. This is mainly due to the increase in interconnect length, and due to increasing coupling capacitance which is increasing to dominate the overall interconnect capacitance [2]. Previously, the maximum clock frequency was determined by the longest interconnect, as data was expected to propagate along the entire interconnect length in a single clock cycle.

In order to decouple the maximum system clock frequency from the interconnect delay, latency-insensitive interconnect schemes, such as the multicycle interconnect scheme, were proposed [3–6]. In this multicycle scheme, shown in Figure 1, interconnects are partitioned into segments bounded by flip-flops. The maximum segment length is chosen to satisfy the desired clock speed of the design. Thus, the maximum clock frequency is bounded by the longest segment, and not the whole interconnect. Thus, in this conventional multicycle (CMC) interconnect scheme, the notion of interconnect delay is transformed into interconnect latency, which is measured in clock cycles.

In the CMC scheme, flip-flops are inserted between the interconnect segments to ensure that the interconnect latency is independent of the clock period. This is important to ensure correct system functionality during low-frequency boundary-scan testing [7]. However, these flip-flops are power-hungry cells that dissipate about 20% of the overall bus power dissipation [8]. Moreover, these flip-flops insert a

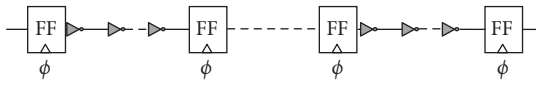


Figure 1: Conventional multicycle interconnect scheme.

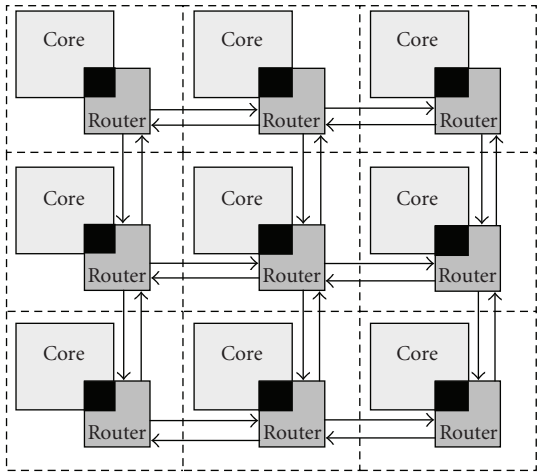


Figure 2: Many-core platform with a network-on-chip (NoC) interconnect fabric.

delay overhead for *each* clock cycle, which leads to upsized repeaters and a higher energy dissipation. The overall clock period T has to encompass all the delay components, such that

$$T \geq t_d + t_{\text{flop}} + t_{\text{setup}} + t_{\text{cycle-jitter}}, \quad (1)$$

where t_d is the interconnect delay (including repeaters), t_{flop} is the flip-flop delay, t_{setup} is the setup delay, and $t_{\text{cycle-jitter}}$ is the clock single-cycle jitter. Moreover, intermediate flip-flops disable cycle-sharing, which makes the CMC scheme very sensitive to process variations.

Thus, an interconnect scheme without intermediate flip-flops is required to reduce the power dissipation and variation sensitivity. At the same time, a certain synchronization scheme is still required to maintain the fixed interconnect latency, and keep it independent of the clock period. In this paper, we propose a source-synchronous multicycle (SSMC) bus scheme with a period-independent latency. The SSMC scheme eliminates the energy dissipation of the intermediate flip-flops and increases the interconnect variation tolerance. The SSMC also appears to be very suitable for *serial links* used for intercore communication in the many-core platform [9], which contains multiple cores, each in a different mesochronous clock domain, as shown in Figure 2. It can be considered a globally asynchronous locally synchronous (GALS) interconnect scheme [10], which solves synchronization issues when crossing different clock domain boundaries. It will also be useful for interdie communication in 3D die stacks [11, 12], shown in Figure 3, because of the limitations imposed on where flip-flops can be added to the vertical through-silicon vias (TSV).

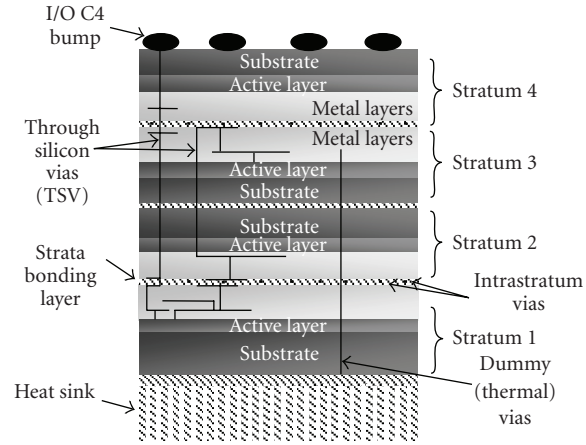


Figure 3: 3D die stack.

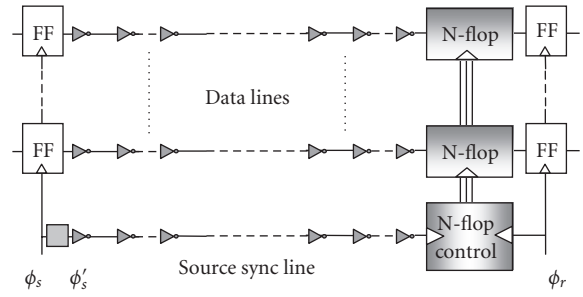


Figure 4: Proposed SSMC scheme.

The rest of the paper is organized as follows. Section 2 presents the proposed SSMC bus scheme in detail, where system constraints are derived and discussed. Section 3 provides the circuit simulation setup and results. Finally, Section 4 provides the conclusion.

2. ON-CHIP SOURCE-SYNCHRONOUS MULTICYCLE INTERCONNECT SCHEME

In this section, the architecture of the proposed source-synchronous multicycle (SSMC) bus architecture and each of its components will be explained. The operation of this scheme will be illustrated, and system design constraints will also be derived and discussed. It should be noted that even though the concept of the proposed architecture is very similar to that used for off-chip communication, many differences exist due to the difference in nature between off-chip and on-chip interconnects. On-chip interconnects are primarily RC dominated and strongly capacitively coupled as opposed to the LC and widely spaced off-chip interconnects.

2.1. SSMC architecture

A schematic of the proposed SSMC scheme is shown in Figure 4. The data lines of the proposed SSMC scheme are exactly the same as that of a conventional CMC scheme, except that the intermediate flip-flops are replaced by a synchronizing element, an N-flop, at the receiver end of the line. The

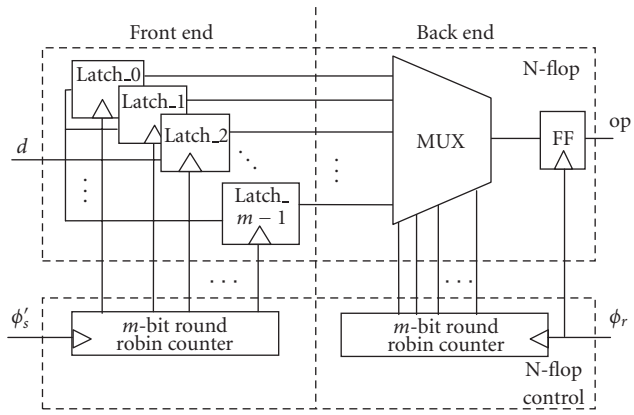


Figure 5: Schematic of an N-flop and the N-flop control.

N-flop is an element that synchronizes the receiver with the transmitter, such that it provides a fixed latency (n-cycles) irrespective of the cycle period T. The N-flop synchronizes the received data using timing information extracted from both the receiver and transmitter clock signals.

Transmitter clock timing information can be delivered to the N-flop in one of two ways: (1) send the transmitter clock signal on a separate line alongside the data, or (2) extract transmitter clock timing information from the received data (clock recovery). The main disadvantage of the first method is the high energy dissipation of sending the clock signal along a dedicated line. However, using some modifications, explained in Section 2.7, the energy dissipation of the source synchronizing line can be significantly reduced. Moreover, as the energy dissipation of this line will be amortized over the number of data bus lines, the energy overhead of this extra line will be considerably small. On the contrary, the second method requires a clock recovery circuit for each bus line, which is a large area and power overhead. Thus, the first option of adding a single separate line to the bus carrying the transmitter clock signal was implemented in the SSMC described in this paper, as shown in Figure 4.

2.2. N-flop

The N-flop is basically a circuit that synchronizes the data sent from a certain clock domain (ϕ_s) with the clock of another clock domain (ϕ_r). In this paper, the two clock domains are considered mesochronous, that is, they have the same frequency f but with a possible phase difference. The N-flop also ensures that data enters the datapath of the second (receiver) domain after a fixed latency of n-cycles, irrespective of the clock frequency.

The schematic of the N-flop and its control logic is shown in Figure 5. Each of the N-flop and its control consists of two main parts. The front part of the N-flop control is a round robin counter fed by the source synchronizing signal ϕ'_s. Only one bit of the m output bits of the counter is "1" during any given cycle, and each line is "1" once every m cycles. Thus, the output of the counters is "one hot" and increments as follows: $0 : \text{“}100\cdots0\text{”} \rightarrow 1 : \text{“}010\cdots0\text{”} \rightarrow 2 : \text{“}001\cdots0\text{”} \rightarrow \cdots m-1 : \text{“}000\cdots1\text{.”}$ The front end of the N-flop is a circular FIFO buffer that has m parallel latches; each of which is activated by one of the counter's output bits. As the front end counter increments, the FIFO latches are individually and sequentially activated. Thus, only one latch is active at any given time and each latch is active once every m cycles.

The back end of the N-flop is an m-to-1 multiplexer, whose inputs are the front end latch outputs as shown in Figure 5. The multiplexer is controlled by another round robin counter fed by the receiver clock signal ϕ_r. The incrementing counter causes the multiplexer to sequentially output its n inputs. An initial offset Δ exists between the values of the front end and back end counters to avoid premature release of the data before the preset latency of n-cycles.

The construction of an N-flop is very similar to the *mesochronous FIFO synchronizer* used in source-synchronous off-chip communication [13]. The N-flop can also be viewed as the general case of a flip-flop that outputs data after a latency of n cycles. Note that when $m = 1$, each of the front end and back end of the N-flop becomes a single latch, whose control (clock) signals are out-of-phase. Note also that the critical path of the N-flop is the same as that of a flip-flop. Moreover, as only one latch is active at any time in each of the N-flop front end and back end, the active energy dissipation of the N-flop is comparable to that of a conventional flip-flop, but its leakage is $m - 1$ times that of a conventional flip-flop.

2.3. SSMC operation

In order to illustrate the operation of the SSMC scheme, let us assume that $n = m = 4$, and that the front end and back end counters are initially reset to $0 : \text{“}1000\text{”}$ and $1 : \text{“}0100\text{,”}$ respectively. This difference in initial counter values is to avoid premature release of data before the required latency in the case when the data propagation delay across the interconnect is less than the clock cycle. This will be apparent from the context of the second example in this subsection.

The maximum clock frequency $f_{\max} = 1/T_{\min}$ is set such that data propagates along the interconnect, goes through the N-flop, and is set up at the output flip-flop in n-clock cycles, that is,

$$nT > t_d + t_{\text{latch}} + t_{\text{mux}} + t_{\text{setup}} + t_{\text{long-jitter}}, \tag{2}$$

where T is the cycle time, t_d is the propagation delay along the data line, t_{latch} is the propagation delay of the N-flop front end, t_{mux} is the propagation delay of N-flop back end (multiplexer), t_{setup} is the setup time of the output flop, $t_{\text{long-jitter}}$ is the n-cycle long-term clock jitter. The source sync signal path is also designed such that the source sync signal edge arrives early enough to increment the front end counter before the arrival of its corresponding data edge.

At maximum frequency operation $f = f_{\max}$, the operation, shown in Figure 6, is as follows: at the first clock edge ($t = 0$), the first source sync signal edge and the first data edge are launched. Meanwhile, the front end counter is still at $0 : \text{“}1000\text{”}$ and latch_0 is active, while the other FIFO

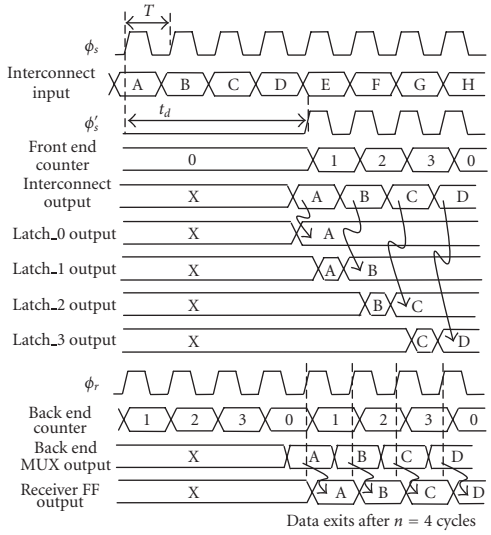


Figure 6: Example of SSMC operation at $f = f_{\max}$.

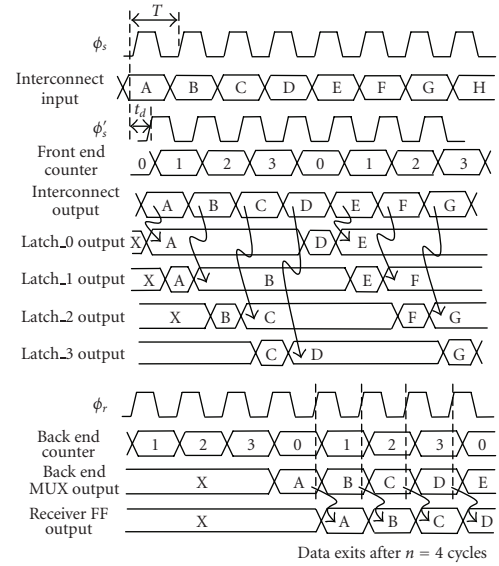


Figure 7: Example of SSMC operation when $t_d < T$.

latches are inactive. The first receiver clock edge will also occur and the back end counter will increment from 1 : "0100" to 2 : "0010," such that the multiplexer is reading the output of latch_2. At $t = T$, the launched data and sync signal edges will have traveled further along the interconnect, but have not reached their destination yet. Also, the next receiver edge will come and increment the back end counter to 3 : "0001." The data and sync signal edge will not reach until after $3T$, meanwhile the back end counter will increment to 0"1000" and then 1 : "0100." Once, the first sync signal edge arrives at the receiver ($4T > t > 3T$), the front end counter will increment from 0 : "1000" to 1 : "0100" and the data will be latched into latch_1. And as the back end counter is also at 1 : "0100," then the multiplexer will read the data which was just latched into latch_1, deliver it to setup at the FF input. Thus, after $n = 4$ clock cycles, the output flip-flop will launch the received data into the receiver datapath.

The other extreme is when operation is at a very low frequency ($T \gg t_d$), which occurs during boundary-scan testing. In this scenario, shown in Figure 7, the first source sync signal edge and the first data edge are launched at the first clock edge ($t = 0$). Meanwhile, the front end counter is still at 0"1000" and latch_0 is active, while the rest is inactive. The first receiver clock edge will also occur and the back end counter will increment from 1"0100" to 2 : "0010." Because $t_d \ll T$, the source sync signal edge and data will arrive before the next clock edge, which causes the front end counter to increment from 0 : "1000" to 1 : "0100," and the received data will get latched to latch_1. Note that if both front end and back end counters were initialized to the same value, the data arriving at the receiver would have been immediately latched to the receiver output, that is, after a latency of 1 cycle and not after the required latency of $n = 4$ cycles. Thus, an initial difference Δ between the front end and back end counters must exist to avoid premature release of data and ensure proper operation at low frequencies. This initial difference Δ will be quantified in (5) and (6).

The data latched in latch_1 will remain there for three more receiver clock edges, such that the back end counters increment from 2 : "0010" to 3 : "0001" to 0 : "1000" to 1 : "0100." At such point, the multiplexer reads the data in latch_1 and delivers it to setup at the flip-flop output. At the following receiver clock edge, $n = 4$ clock cycles have passed, and the first received data is launched into the receiver datapath.

2.4. SSMC advantages

One of the main advantages of the SSMC scheme is that it provides a variation-tolerant characteristic. In the CMC scheme, the delay of some of the interconnect segments may increase due to within-die (WID) parameter variations, which leads to a reduction in the maximum clock frequency of the whole bus. However, SSMC eliminates intermediate flip-flops, and hence, removes timing boundaries throughout an interconnect. Thus, SSMC enables better averaging of delay variations across the whole interconnect, which reduces bit-rate degradation due to WID process variations.

SSMC is also scalable and suitable for data transmission across clock domain boundaries. Another advantage of SSMC compared to a conventional MCB is that it replaces its power-hungry synchronizing flip-flops by a more energy efficient synchronizing scheme. The intermediate flip-flops in CMC are replaced by an N-flop per data line. An N-flop has almost the same active energy dissipation but $n - 1$ times the leakage of a single flip-flop. A shared source sync line and an N-flop control block are also required in the SSMC, but their energy dissipation is amortized over the number of data bus lines. Significant energy dissipation is expected, as shown from the simulation results in Section 3. Note that

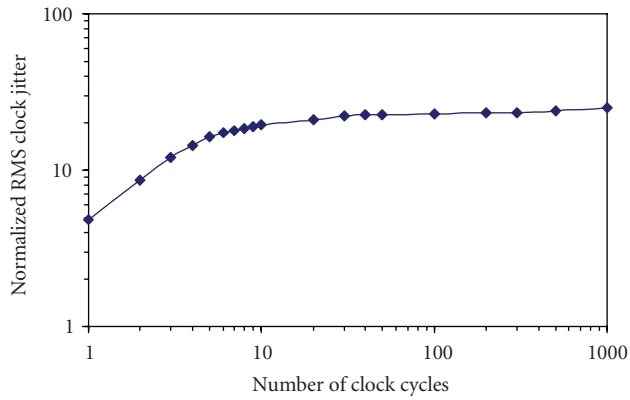


Figure 8: Normalized long-term jitter versus the number of clock cycles.

these energy savings are expected to increase as the number of bus cycles increases, because more flip-flops will be eliminated.

The SSMC scheme also reduces the overhead per cycle due to clock jitter. The dependency of jitter on the number of clock cycles [14, 15] is presented in Figure 8. Single-cycle jitter is defined as the clock edge variation in consecutive clock cycles. Long-term jitter is the accumulated clock edge variation across multiple cycles. For a small number of clock cycles, jitter primarily results from voltage controlled oscillator (VCO) variation, where jitter is proportional to the square root of the number of cycles [14]. For a large number of clock cycles, the phase-locked loop (PLL) phase detector and loop filter sense the VCO error and adjust the VCO input accordingly, leading to a saturation of the long-term jitter. This indicates that the long-term jitter per cycle ($t_{\text{long-jitter}}/n$) monotonically decreases as the number of cycles increases. Hence, the clock jitter overhead per cycle in the SSMC scheme is less than that in the CMC scheme ($t_{\text{long-jitter}}/n < t_{\text{cycle-jitter}}$), and is further reduced as the number of cycle increases, as shown by (1) and (2). Thus, this extra timing slack can be used to further reduce the repeater sizes and energy dissipation.

2.5. System constraints

Several system constraints have been identified to ensure correct system functionality.

2.5.1. Critical path

The maximum clock frequency $f_{\max}$ ensures that data propagates along the interconnect, goes through the N-flop, and is set up at the output flip-flop in n-clock cycles

$$nT > t_d + t_{\text{latch}} + t_{\text{mux}} + t_{\text{setup}} + t_{\text{jitter}}, \tag{3}$$

where T is the cycle time, t_d is the propagation delay along the data line, t_{latch} is the propagation delay of the N-flop front end, t_{mux} is the propagation delay of N-flop back end (multiplexer), t_{setup} is the setup time of the output flop, t_{jitter} is the n-cycle long-term clock jitter.

2.5.2. Avoiding FIFO buffer overrun

In an n-cycle bus, data takes n-cycles to be transferred from the source to the receiver datapath. During the data transfer, the data spends some time on the line and the rest is stored in the N-flop front end (FIFO buffer). In case $T \gg T_{\min}$, the data stays stored in the FIFO buffer most of the time. And in that case, if the FIFO buffer capacity is not large enough, buffer overrun may occur, and data temporarily stored in the buffer may be overwritten before being delivered to the output flip-flop. Thus, if the N-flop FIFO buffer has a depth of m-bits, buffer overrun can be avoided if the buffer can accommodate at least n bits, that is,

$$m \geq n. \tag{4}$$

As explained earlier, an initial offset Δ exists between the front end and back end counters to avoid premature release of the data. And to ensure that data is delivered into the receiver datapath after a latency of exactly n-cycles, then

$$n = m - \Delta + 1. \tag{5}$$

Combining (4) and (5) shows that

$$\Delta \geq 1. \tag{6}$$

However, increasing the depth of the FIFO m leads to more devices and more devices loading, and hence, a higher energy dissipation for the N-flop. Thus, for a low-power design, the FIFO depth should be kept at its minimum $m = n$, and hence a counter offset of $\Delta = 1$.

2.5.3. Reducing intersymbol interference (ISI)

To ensure correct sampling, the bit period has to be large enough to accommodate any data jitter and to account for data attenuation, that is, has an appropriately open "eye," as shown in Figure 9. Thus, given a certain cycle time T, the data line and its repeaters have to be designed such that

$$T \geq \Delta t_d + t_{\text{margin}}, \tag{7}$$

where Δt_d is the data jitter and t_{margin} is an extra time margin to account for signal attenuation (and rise time) at the end of any interconnect segment and ensure full-scale switching.

2.5.4. Ensuring correct sampling

In order to ensure correct sampling, the source sync signal has to arrive at the front end N-flop counter and increment it before the data arrives at the N-flop front end input. In order for this condition to be satisfied in the presence of a certain data jitter $\Delta t_d = t_{d,\max} - t_{d,\min}$ and a certain source sync signal jitter $\Delta t_c = t_{c,\max} - t_{c,\min}$, the worst-case increment of the front end counter has to occur before the earliest arrival of the data, as shown in Figure 10, that is,

$$t_{c,\max} + t_{\text{counter}} \leq t_{d,\min}, \tag{8}$$

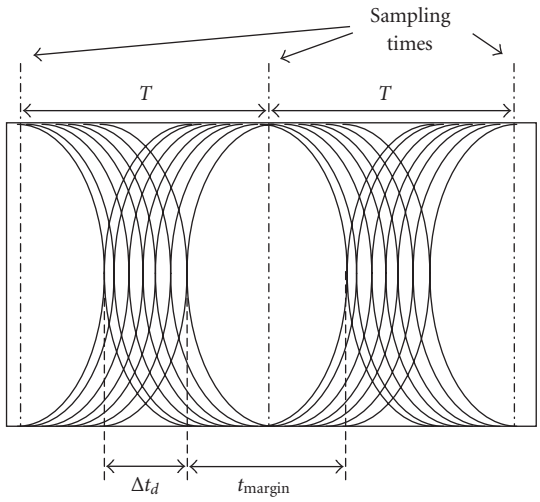


Figure 9: Eye diagram for the data signal at the end of the interconnect.

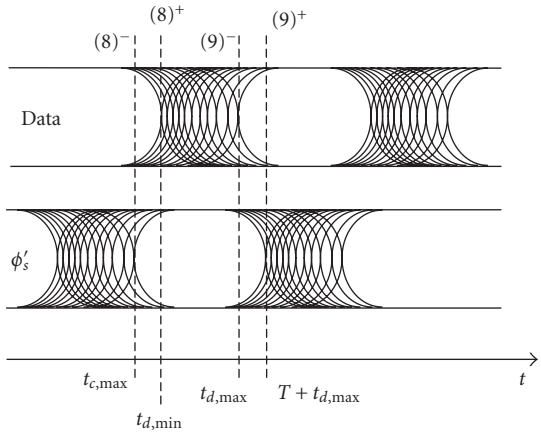


Figure 10: Illustrating the system constraints for correct sampling (8)-(9).

where t_{counter} is the counter delay. Also, the worst-case data edge has to arrive early enough before the following front end counter increment, that is,

$$t_{d,\max} \le T + t_{c,\min} + t_{\text{counter}} - t_{\text{cycle-jitter}}. \tag{9}$$

Rearranging (8) and (9) yields

$$T \ge \Delta t_d + \Delta t_c + t_{\text{cycle-jitter}}, \tag{10}$$

$$\Delta t_c + t_{\text{counter}} \le t_{d,\min} - t_{c,\min} \le T - \Delta t_d + t_{\text{counter}} - t_{\text{cycle-jitter}}. \tag{11}$$

2.5.5. Summary

The system design constraints can be summarized by the following set of equations:

$$n = m - \Delta + 1, \quad \Delta \ge 1. \tag{12}$$

$$\Delta t_c + t_{\text{counter}} \le t_{d,\min} - t_{c,\min} \le T_{\min} - \Delta t_d + t_{\text{counter}} - t_{\text{cycle-jitter}}. \tag{13}$$

$$T \ge \max\left\{\Delta t_d + \Delta t_c + t_{\text{cycle-jitter}}, \Delta t_d + t_{\text{margin}}, \frac{t_{d,\max} + t_{\text{latch}} + t_{\text{mux}} + t_{\text{setup}} + t_{\text{long-jitter}}}{n}\right\}. \tag{14}$$

2.6. Reducing the jitter of the data and the source synchronizing signals

One of the main differences between SSMC and its similar off-chip communication scheme is the nature of the interconnect. On-chip interconnects have smaller cross-sections and narrower spacing as compared to off-chip interconnects. Thus, on-chip interconnects tend to be RC dominated while off-chip interconnects are LC dominated. Hence, on-chip interconnects tend to have higher dispersion and attenuation compared to off-chip interconnects, which leads to a relatively poor pulse response. Also, the delay of on-chip interconnects has a quadratic dependence on length as opposed to the linear relationship for off-chip interconnects. Thus, repeater insertion [16] is required to improve the pulse response and delay-length relationship of on-chip interconnects. Moreover, due to narrower line spacing, on-chip interconnects are more capacitively coupled than off-chip interconnects. Hence, different switching scenarios have different delays, which cause large signal jitter (Δt_d) and degrade the overall bus performance as shown in (14).

The delay of a repeated interconnect can be expressed [16, 17] as

$$t_d = 0.38 r_t c_t \frac{l^2}{k} + 0.69\left(k \cdot R_u C_u + \frac{R_u}{h} c_t l + r_t l C_u h\right), \tag{15}$$

where R_u and C_u are the output resistance and input capacitance of a unit repeater. l is the total interconnect length, k are the number of interconnect sections (i.e., number of repeaters), and h is the repeater sizing multiplier. r_t is the per-unit-length interconnect resistance and c_t is the per-unit-length equivalent interconnect capacitance, which can be expressed as in [18]

$$c_t = c_g + \Sigma \text{MCF}_i c_c, \tag{16}$$

where c_g and c_c are the physical vertical and lateral (coupling) capacitance components, respectively. MCF_i is the Miller coupling coefficient between the line of interest and line i. The MCF indicates the contribution of the coupling capacitance C_c to the overall equivalent capacitance C_t. In the literature, the commonly used values of MCF are zero for similarly switching coupled lines (1) when only one of the two coupled lines switches, and (2) when the coupled lines oppositely switch. Thus, different switching scenarios lead to MCF

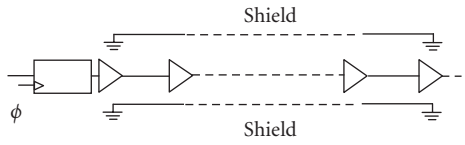


Figure 11: Interconnect shielding technique [19].

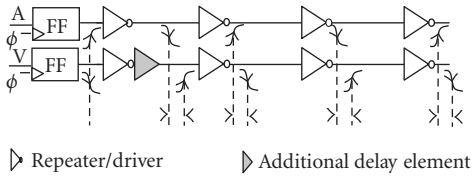


Figure 12: Alternate delayed line (ADL) bus scheme: delay element placed on alternate lines as proposed in [20, 21].

variation ΔMCF, interconnect capacitance variation Δc_t, and eventually a signal jitter Δt_d,

$$\begin{aligned}\Delta t_d &= \left(0.38 r_t \frac{l}{k} + 0.69 \frac{R_u}{h}\right) \cdot \Delta c_t l \\ &= \left(0.38 r_t \frac{l}{k} + 0.69 \frac{R_u}{h}\right) \cdot \Delta\text{MCF} \cdot c_c l.\end{aligned} \tag{17}$$

As shown in (17), the delay jitter can be reduced by reducing the value in parentheses. Thus, the higher ($k \uparrow$) number of upsized ($h \uparrow$) repeaters is required to reduce the signal jitter on-chip interconnects. This solution has the disadvantage of a higher repeater capacitance (hkC_u), which in turn increases the energy dissipation. Another disadvantage is that even if the value within parentheses is reduced, it does not go down to zero, and a jitter Δt_d will remain. This jitter is proportional to length, and in order to achieve a certain bit rate, a limit will be placed on the maximum bus length.

A more energy efficient solution that avoids this bus length limitation is to reduce the MCF variation ΔMCF. This can be performed by using a bus scheme that unifies the MCF for all switching scenarios. One of these scenarios is bus shielding [19], where V_{dd} or ground lines are inserted between the switching lines, as shown in Figure 11. This scheme has the advantage of transforming the MCF for all switching scenarios to 1, and hence eliminates data delay variation $\Delta\text{MCF} = 0$. The disadvantage of this scheme is that it doubles the required bus metal area. Thus, shielding will not be applied to the data lines in our SSMC scheme, but will only be applied to the extra source sync line, as it occurs only once. An area efficient alternative for the data lines will be the alternate delayed line (ADL) scheme [20, 21], which adds a delay element to alternate lines, as shown in Figure 12. This inserted delay separates the switching events of adjacent interconnects, and the MCF for all switching scenarios becomes approximately MCF $\approx$ 1, that is, $\Delta\text{MCF} \approx 0$. And in case of a bidirectional bus, the optimal repeater staggering [18] scheme, shown in Figure 13, can be used to achieve MCF $\approx$ 1 for all switching scenarios, and hence, $\Delta\text{MCF} \approx 0$. Another advantage of these three schemes is that they reduce

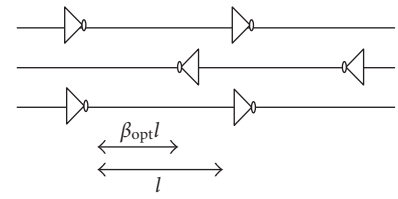


Figure 13: Staggered bidirectional bus [18].

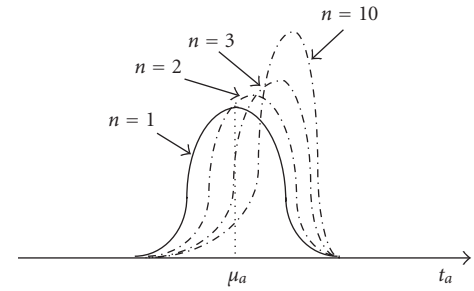


Figure 14: Worst-case arrival time distribution for a CMC.

the worst-case MCF from 2 to 1, which reduces the interconnect delay. This delay slack can be used to downsize repeaters and further reduce energy dissipation.

2.7. Variation tolerance of SSMC

The data arrival time t_a (with respect to the clock edge) at the end of any cycle in the conventional MCB, shown in Figure 1, can be written as

$$t_a = t_{\text{flop}} + t_d. \tag{18}$$

Due to WID process variations, the arrival time t_a will have a distribution with mean μ_a and standard deviation σ_a. In this conventional synchronous interconnect scheme, CMC, the clock cycle has to accommodate the worst-case arrival time $t_{a,w}$ in any bus cycle ($1 \le i \le n$),

$$t_{a,w} = \max\{t_{a,1}, t_{a,2}, \ldots, t_{a,n}\}. \tag{19}$$

The max operation in (19) narrows and negatively skews the worst-case arrival time distribution, as shown in Figure 14, that is,

$$\mu_{a,w} > \mu_a, \qquad \left(\frac{\sigma_{a,w}}{\mu_{a,w}}\right) < \left(\frac{\sigma_a}{\mu_a}\right). \tag{20}$$

This increase in the mean arrival time indicates that a multicycle bus will need a larger minimum cycle time, and hence a lower maximum clock frequency $f_{\max}$. This performance degradation is exacerbated as the number of cycles increases as shown in Figure 14. The main reason behind this degradation is the presence of the intermediate flip-flops, which led to the max operation in (19).

The variation in the maximum clock frequency in SSMC depends on the system design. If the system variables are such

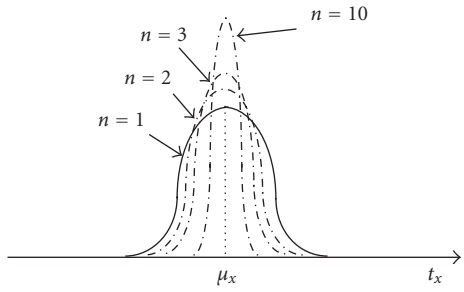


Figure 15: Worst-case arrival time distribution for a datapath delay constrained SSMC.

that the signal jitter on the data and source sync lines is negligible, then the minimum clock period is limited by the datapath delay as shown in (3). Assume that the SSMC scheme has n-cycles and has a length L. Assume also that the delay across the whole datapath is t_x, and that the delay across a single-cycle length L/n is t_s. Assume also that the delay across a single-cycle length L/n has an overall mean μ_s. Assuming that these process variations are Gaussian random variables, then the mean of the delay along the whole interconnect will be

$$\mu_x = E[t_x] = E[t_{s,1} + t_{s,2} + \cdots + t_{s,n}] = \sum_{1}^{n} E[t_{s,i}] = n\mu_s. \tag{21}$$

If all the random delay variables $t_{s,i}$ are uncorrelated, and if the standard deviation for the delay across a single-cycle length (L/n) is σ_{su}, then the standard deviation of the whole datapath delay t_x is

$$\begin{aligned}\sigma_x &= \sqrt{\mathrm{Var}[t_x]} = \sqrt{\mathrm{Var}[t_{s,1} + t_{s,2} + \cdots + t_{s,n}]} \\ &= \sqrt{\sum_{1}^{n} \mathrm{Var}[t_{s,i}]} = \sigma_{su}\sqrt{n}.\end{aligned} \tag{22}$$

As this overall delay represents the delay for n-cycles, then the delay per cycle μ_x/n remains at μ_s *showing no mean performance degradation* with a distribution that narrows as n increases, as shown in Figure 15,

$$\frac{\sigma_x}{\mu_x} = \frac{1}{\sqrt{n}}\frac{\sigma_{su}}{\mu_s}. \tag{23}$$

However, if the variations are correlated with a standard deviation of σ_{sc} for the delay across a single-cycle length (L/n), then the standard deviation of the whole datapath is

$$\begin{aligned}\sigma_x = \sqrt{\mathrm{Var}[t_x]} = \sqrt{\mathrm{Var}[t_{s,1} + t_{s,2} + \cdots + t_{s,n}]} < n\sigma_{sc}, \\ \frac{\sigma_x}{\mu_x} < \frac{\sigma_{sc}}{\mu_s}.\end{aligned} \tag{24}$$

This overall delay variation decreases (i.e., better variation averaging occurs) as the total bus length increases and becomes much greater that the correlation distance, which is typically in the range of $1 \sim 3$ mm in current technologies [1]. Thus, in the presence of correlated and/or uncorrelated process variations, *a datapath delay constrained SSMC outperforms a CMC* as it preserves its mean maximum clock frequency, while narrowing its maximum frequency distribution.

If, on the other hand, the SSMC system variables are such that the minimum cycle time is constrained by data jitter, as in (10), then the minimum cycle time does not depend on the overall interconnect delay, but rather on the minimum temporal separation between two consecutive data edges $D[t]$ and $D[t + T]$. As these two edges travel consecutively on the interconnect, the within-die process variations along the path are exactly the same for both data edges. Hence, the variation in the maximum clock frequency for this jitter-constrained SSMC interconnect scheme will be determined by the variation in data jitter rather than by interconnect delay due to WID variations. And if the techniques explained in Section 2.6 are employed, then data jitter variations can be significantly reduced, and hence, the *performance degradation can also be reduced*.

2.8. Reducing the energy penalty of the source synchronizing overhead

The overhead of the SSMC scheme is the area and energy dissipation of the N-flop control circuitry, and the extra line used to deliver the source sync signal. If the source clock is sent directly on the source sync line, a large energy overhead occurs. The reason is that the clock signal switches twice every cycle, whereas the data signals switch with a ~ 0.1 probability [8]. Thus, the source sync line will incur approximately 20 times the energy dissipation of a single data line. In order to reduce the energy overhead of this line, the source signal has to be altered to reduce its switching activity, while still keeping its required synchronizing edges. The first modification would be to send the clock signal over the source sync line after halving its frequency, as only one edge per cycle is required for synchronization. And in that case the regular flip-flops used in the front end round robin counter can be replaced by double-edge triggered flip-flops [22]. This first modification reduces the activity and energy dissipation of the source sync line by half. The next modification comes from the fact that the source sync is not required if data on the bus is not changing. In that case, a bus activity signal can be generated by XORing the input and output of the driving flip-flop for each bus line, and then ORing all these signals. This bus activity signal is then used to gate the source sync signal [23], and significantly reduce its activity factor and energy dissipation to a value closer to that of a data line. It should be noted that this overhead is shared by all bus lines. Hence, the energy overhead will be amortized over the number of bus lines.

2.9. Dealing with metastability

Whenever there are setup and hold time violations in any flip-flop, it enters a state where its output is unpredictable;

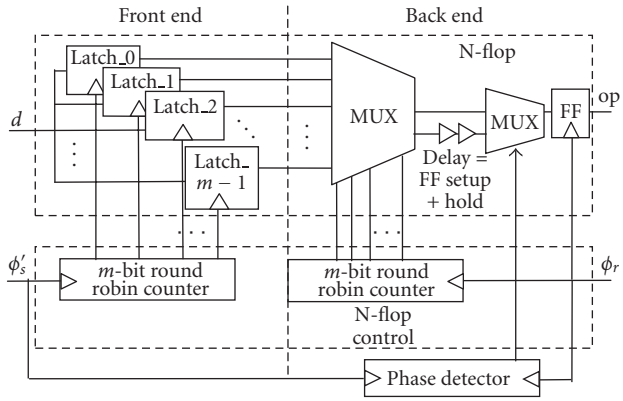


Figure 16: N-flop with metastability reduction overhead (enabled during reset only).

this state is known as metastable state (quasi-stable state); at the end of metastable state, the flip-flop settles down ambiguously to either "1" or "0." This whole process is known as metastability. The proposed scheme uses the same technique used in off-chip mesochronous links to reduce metastability [13]. When most chips are initially powered up, a power-on reset circuit delivers a global reset signal to the whole chip. During this initial reset period, the front and back end counters in the N-flop are reset to their initial values, which remain there until the reset signal is asserted low. Also, during this reset period, which is in the order of a 1 millisecond, metastability possibility is detected and accounted for using a phase detector, a delay element, and a multiplexer, as shown in Figure 16. Metastability occurs if the data at the back end flip-flop input changes during the critical keep out window around the flop's rising edge (setup + hold).

In order to avoid metastability in the SSMC scheme, two measures are usually taken. First, the back end flop is designed in order to reduce its keep out timing window by using high gain latches with high voltage transfer characteristic slopes in the metastable region. Second, if the data arriving at the back end flip-flop input can possibly transition during the back end's flop keep out window, the data is delayed to avoid transitioning in the keepout window and avoid metastability. This is usually implemented by comparing the phase of the received sender clock signal (sync ϕ_s') to the local receiver clock ϕ_r using a phase detector. If the detected phase difference is such that the data at the input of the back end flip-flop switches during this flop's critical keep out window, then the phase detector's latched output will be asserted to select a delayed version of the back end multiplexer output to be fed into the back end flop. Note that as the clock domains are mesochronous (both have constant phase), this phase detection process needs to be performed only. Thus, the phase detector circuit is disabled after the initial system reset period is over. Hence, the power dissipation penalty of the phase detector will have little impact on the reported power results during normal operation. The power dissipation penalty is comprised by the power leakage of the phase detector circuitry and the power dissipation overhead required to upsize interconnect repeaters to account for the extra (negligible) delay of the delay selection multiplexer shown in Figure 16.

2.10. SSMC implementation issues

This subsection will explain some implementation issues faced during the design of a global multicycle interconnect using both the CMC and SSMC schemes. These will be explained in the context of one of the most common design scenarios in microprocessors, GPUs and pipelined architectures, in which the system cycle time T, the interconnect latency n, and the source and destination locations are known a priori.

In the CMC scheme, n flip-flops linking $n - 1$ interconnect segments will need to be inserted between source and destination. The main constraints for CMC interconnect links are the locations of the intermediate flip-flops, which are constrained by positions with active area availability to insert a flip-flop and by the interconnect segment length L_{seg}. The interconnect segment lengths must have a propagation delay $t_{d,\max}$, that satisfies (1), that is,

$$t_d \leq T - t_{\text{flop}} - t_{\text{setup}} - t_{\text{cycle-jitter}}. \tag{25}$$

In most cases, (1) is not satisfied, and repeaters need to be inserted along the interconnect segments. The number and size of repeaters are determined by positions with repeater placement availability, and by the segment propagation delay t_d requirement set by (1).

Besides lower energy dissipation and better variation tolerance, the SSMC has three other main differences, compared to the CMC scheme, which are apparent during implementation. The first difference is the extra metal resources required by the SSMC scheme for the shielded sync signal. However, this penalty is reduced when the SSMC scheme is used to implement wide buses, as the shielded sync line area penalty will be shared by all bus lines. The second difference is that the SSMC scheme does not have any intermediate flip-flops. This implies the advantage of not needing any available active area for intermediate flip-flop placement, and hence, removes the constraints on the global interconnect route. However, it does mandate a larger area at the receiver end for the N-flop and its control circuitry. *The area of the N-flop and its control circuitry is almost the same as that of the n flip-flops in the CMC scheme*. Thus, the SSMC scheme approximately retains the same active area requirement for flip-flops, but instead of constraining the designer to distribute the flip-flops along the interconnect route as in CMC, it lumps it all at the receiver end. It should be noted that minor routing constraints still exist due to the presence of intermediate repeaters, but these constraints can be easily dealt with using back end placement and routing tools. So, in summary, SSMC removes the interconnect global routing constraints due to flip-flop placement, but requires a larger active area (approximately equal to the total CMC flip-flop area) at the receiver side. Removing the intermediate flip-flops in the SSMC scheme gives the advantage of avoiding the necessity of a low-skew global clock network to synchronize the intermediate flip-flops. The SSMC scheme only requires

local clock signals at the source and receiver ends of the interconnect, which reduces the energy dissipation, delay, and metal resource requirement for the low-skew global clock distribution network. It should be noted that in the context of this second difference, the SSMC scheme is suitable to implement serial links in network-on-chips and through-silicon vias (TSV) is 3D stacked chips. Both of these interconnect links cannot have flip-flops distributed along their lengths either due to the lack of a global clock network or due to the infeasibility of connecting active devices to it along its length.

The third difference is that the intermediate interconnect segment length is only limited by its propagation delay, as shown in (1). However, the interconnect length between the source and the N-flop in the SSMC scheme is limited by both the interconnect propagation delay t_d and jitter on the data and sync lines $\Delta t_d + \Delta t_c$, as shown by (3) and (10). In the CMC scheme, if both the latency n and the total global interconnect length are increased such that their ratio is kept the same, the delay constraint (1) is still maintained. However, if the same occurs for the SSMC scheme, the delay constraint given by (3) will remain satisfied, but that given by (10) may be violated as the MCF-induced jitter Δt is length-dependent as shown in (17). In such a case, the interconnect can be segmented in one or more segments by inserting intermediate N-flops along the interconnect to reduce the data jitter, as shown in Figure 17. Note that the latency of each segment does not have to be equal, for example, n_1 does not necessarily have to be equal to n_2 in the example shown in Figure 19. This degree of freedom can be used to pick the best locations with available active area to implement the intermediate N-flops. To illustrate SSMC schemes with more than one segment, both the SSMC and CMC schemes were used to implement an n-cycle 4 GHz 16-bit bus in a 65- nm technology. The length of the bus and the required latency n were varied. Implementation details and normalized energy savings are shown in Table 1. Both implementations had almost the same active area requirement, however, the SSMC scheme required 13% more intermediate metal layer area to implement the shielded sync line. The results shown in Table 1 show that SSMC can still be used to implement very long buses if the interconnects are segmented by inserting intermediate N-flops to reduce MCF-induced data jitter and achieve the required bit rate. The results also show that the normalized energy reduction will be limited by that at the longest interconnect length that requires a single N-flop (i.e., $\%\Delta E_{\max} = 20\%$ which occurs for $L = 9$ mm in Table 1).

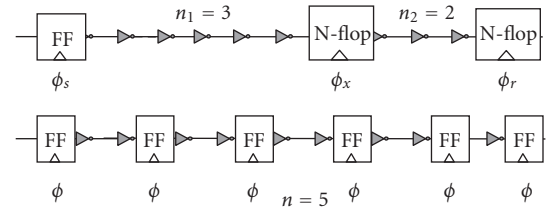


Figure 17: SSMC scheme for a long global multicycle $n = 5$ interconnects compared to its corresponding CMC scheme implementation.

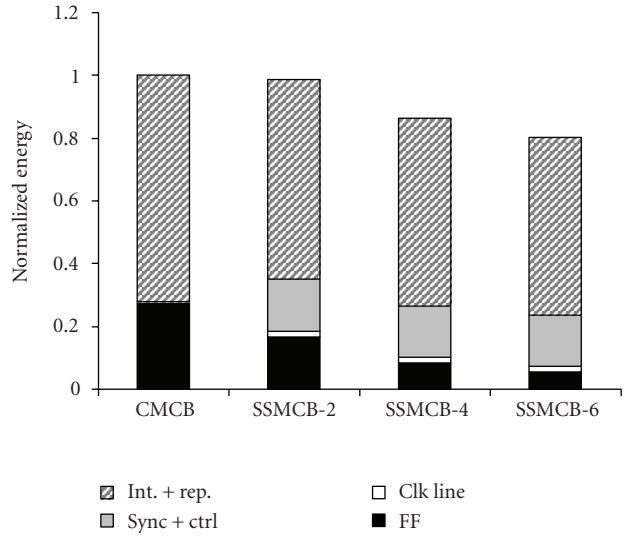


Figure 18: Energy comparison between the CMC and SSMC schemes for different numbers of cycles $n = 2, 4, 6$.

3. SIMULATION SETUP AND RESULTS

3.1. Simulation setup

Sixteen-bit conventional (CMC) and source-synchronous (SSMC) multicycle buses were designed in a 65-nm technology [24]. The flop-to-flop distance was set to 1500 μm and the target bit rate was 4 GHz. The interconnects were all at minimum width and minimum spacing ($w = s = 0.14\,\mu$m). In order to reduce signal jitter on the source sync line of SSMC scheme, the source sync line was fully shielded and placed in the center of the bus. Thus, the SSMC has a metal area overhead equivalent to three lines. The alternate delayed line (ADL) scheme [20, 21], discussed in Section 2, was applied to the data lines of the SSMC scheme to reduce data jitter Δt_d. But as the ADL scheme also reduces worst-case delay, it was applied to the bus lines of the conventional MCB for a fair comparison. Device sizes were optimized using a global optimizer in order to achieve the system constraints (12)–(14), the required bit rate, required signal slopes, and noise margins, while reducing energy dissipation. The energy dissipation overhead on the clock tree due to the flip-flops or the synchronizer was factored in our energy measurements for both schemes. This was all repeated for different numbers of bus cycles n. Both energy reduction techniques explained in Section 2.7 were implemented in the design of the SSMC bus.

Both buses were fed with random uniformly distributed datastreams with a 0.1 activity factor. The bus energy dissipation was recorded and divided by the number of bus lines. Random within-die (WID) variations were then applied to both of these buses. The sources of variations used were channel length, channel width, threshold voltage, metal thickness, metal width, line spacing, and ILD thickness.

Table 1: CMC and SSMC schemes comparison for implementing a 4- GHz 16-bit bus of latency n and length L. All energy values are normalized to the CMC energy dissipation at each (L, n).

L (mm)	n	CMC	SSMC		
		Number of flip-flops	Number of N-flops	Size of N-flops (m)	Normalized energy
3	2	2	1	2	0.98
6	4	4	1	4	0.84
9	6	6	1	6	0.80
12	8	8	2	4, 4	0.85
15	10	10	2	4, 6	0.82
18	12	12	2	6, 6	0.81

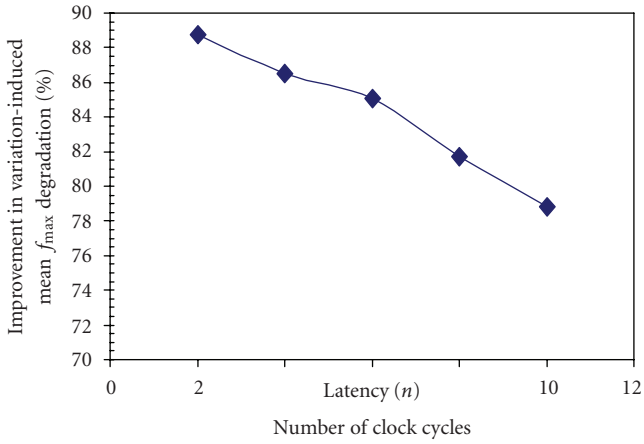


Figure 19: SSMC improvement in bit rate due to its better tolerance to process variations compared to CMC (Ref. bit rate= 4 GHz).

Although the sources of interconnect metal variation are mostly deterministic (e.g., metal thickness depends on metal density), the simulations are performed by assuming a random correlated variation model. Using these WID variations, a Monte Carlo circuit simulation is performed for both bus schemes resulting in a bit rate distribution.

3.2. Energy dissipation results

Table 2 shows the energy dissipation results obtained for 2, 4, and 6-cycle CMC and SSMC buses. All energy values for each n-cycle bus in Table 2 are normalized with respect to the respective total energy dissipation of the CMC bus. Even though SSMC did not achieve much energy reduction for a 2-cycle MCB, a significant 20% energy reduction was achieved as the number of cycles increased to 6. It should also be noted that the energy dissipation of the source sync line and the synchronizer is linear with the number of cycles, and hence their percentage contribution to the overall system remains almost constant as n increases, as shown in Table 2 and Figure 1. The energy reduction in the SSMC scheme increases as n increases, because of two main reasons: (1) the energy dissipation of the initial flip-flops becomes negligible as the bus gets longer; and (2) the timing overhead of the initial flip-flop and the synchronizer gets amortized better as n increases, which reduces the required size and number of repeaters as shown in Figure 18.

3.3. Variation results

Simulation results show that SSMC removes up to ∼ 90% of the maximum clock frequency degradation that occurred in CMC due to WID process variations, as shown in Figure 19. This improvement was reduced as the bus length increased, because the f_{max} degradation of CMC saturated faster than that of SSMC. Note that most of the delay variation is due to data jitter variation, which has a close-to-linear relationship with bus length. The data jitter variations induced by WID process variation can be further reduced by shielding data lines instead of using the ADL bus scheme but at the cost of doubling the required metal area.

4. CONCLUSION

Interconnect links are one of the main bottlenecks in improving the performance of state-of-the-art integrated circuits. Several aspects of recent DSM technologies limit the maximum bit rate that can be sent on the bus as well as degrade the signal integrity. Moreover, new system architectures and technologies such as the "many-core" platform and 3D-IC stacking lead to the presence of many clock domains.

In this paper, a low-power variation-tolerant source-synchronous multicycle (SSMC) interconnect scheme was proposed. This scheme is scalable and suitable for transferring data across different clock domains such as those in "many-core" SoCs and in 3D-ICs. SSMC replaces intermediate flip-flops by a source-synchronous synchronization scheme. Removing the intermediate flip-flops in the SSMC scheme enables better averaging of delay variations across the whole interconnect, which reduces bit rate degradation due to within-die (WID) process variations. Monte Carlo circuit simulations showed that SSMC eliminates ∼ 90% of the performance degradation in a 6-cycle 9 mm-long 16-bit conventional bus in the presence of WID process variations.

The proposed multicycle bus scheme also led to significant energy savings due to eliminating the power hungry flip-flops and efficiently designing the source synchronization overhead. Moreover, eliminating intermediate flip-flops avoided the timing overhead of the setup time, the flip-flop delay, and the single-cycle clock jitter. This delay slack was

Table 2: Energy simulation results. All energy values are normalized to the CMC energy dissipation at each n.

n	CMC			SSMC				
	FF	Int. + rep.	Total	FF	Source sync line	N-flop ctrl	Int. + rep.	Total
2	0.28	0.72	1.00	0.17	0.15	0.02	0.64	0.98
4	0.28	0.72	1.00	0.08	0.14	0.02	0.60	0.84
6	0.28	0.72	1.00	0.06	0.14	0.03	0.57	0.80

then translated into further energy savings by downsizing the repeaters. The significant delay jitter due to capacitive coupling was addressed, and solutions were put forward to alleviate it. Circuit simulations in a 65-nm process environment indicated that energy savings up to 20% are achievable for a 6-cycle 9 mm long 16-bit bus.

REFERENCES

[1] J. D. Meindl, J. A. Davis, P. Zarkesh-Ha, C. S. Patel, K. P. Martin, and P. A. Kohl, "Interconnect opportunities for gigascale integration," *IBM Journal of Research and Development*, vol. 46, no. 2-3, pp. 245–263, 2002.

[2] K. Rahmat, S. Nakagawa, S-Y. Oh, and J. Moll, "A scaling scheme for interconnects in deep-submicron processes," Tech. Rep. HPL-95-77, Hewlett Packard, Palo Alto, Calif, USA, 1995.

[3] L. P. Carloni, K. L. McMillan, and A. L. Sangiovanni-Vincentelli, "Theory of latency-insensitive design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 9, pp. 1059–1076, 2001.

[4] L. Scheffer, "Methodologies and tools for pipelined on-chip interconnect," in *Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD '02)*, pp. 152–157, Freiburg, Germany, September 2002.

[5] R. Lu, G. Zhong, C.-K. Koh, and K.-Y. Chao, "Flip-flop and repeater insertion for early interconnect planning," in *Proceedings of the Design, Automation and Test in Europe Conference (DATE '02)*, pp. 690–695, Paris, France, March 2002.

[6] P. Cocchini, "Concurrent flip-flop and repeater insertion for high performance integrated circuits," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '02)*, pp. 268–273, San Jose, Calif, USA, November 2002.

[7] "IEEE 1149.1-2001: Standard Test Access Port and Boundary-Scan Architecture," Institute of Electrical and Electronics Engineers.

[8] M. Khellah, M. Ghoneima, J. Tschanz, et al., "A skewed repeater bus architecture for on-chip energy reduction in microprocessors," in *Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD '05)*, pp. 253–257, San Jose, Calif, USA, October 2005.

[9] S. Borkar, P. Dubey, K. Kahn, et al., "Platform 2015: Intel Processor and Platform Evolution for the next Decade," 2005, *White Paper*, Intel Corporation.

[10] A. Edman and C. Svensson, "Timing closure through a globally synchronous, timing partitioned design methodology," in *Proceedings of the 41st Design Automation Conference (DAC '04)*, pp. 71–74, San Diego, Calif, USA, June 2004.

[11] B. Black, D. W. Nelson, C. Webb, and N. Samra, "3D processing technology and its impact on IA32 microprocessors," in *Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD '04)*, pp. 316–318, San Jose, Calif, USA, October 2004.

[12] S. Das, A. Chandrakasan, and R. Reif, "Three-dimensional integrated circuits: performance, design methodology, and CAD tools," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (VLSI '03)*, pp. 13–18, Tampa, Fla, USA, February 2003.

[13] W. J. Dally and J. W. Poulton, *Digital Systems Engineering*, Cambridge University Press, Cambridge, UK, 1998.

[14] J. A. McNeill, "Jitter in ring oscillators," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 6, pp. 870–879, 1997.

[15] M. Mansuri and C.-K. K. Yang, "Jitter optimization based on phase-locked loop design parameters," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1375–1382, 2002.

[16] H. B. Bakoglu and J. D. Meindl, "Optimal interconnection circuits for VLSI," *IEEE Transactions on Electron Devices*, vol. 32, no. 5, pp. 903–909, 1985.

[17] T. Sakurai, "Closed-form expressions for interconnection delay, coupling, and crosstalk in VLSI's," *IEEE Transactions on Electron Devices*, vol. 40, no. 1, pp. 118–124, 1993.

[18] M. Ghoneima and Y. Ismail, "Optimum positioning of interleaved repeaters in bidirectional buses," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 3, pp. 461–469, 2005.

[19] A. B. Kang, S. Muddu, E. Sarto, and R. Sharma, "Interconnect tuning strategies for high-performance ICs," in *Proceedings of the Design, Automation and Test in Europe (DATE '98)*, pp. 471–478, Paris, France, February 1998.

[20] K. Nose and T. Sakurai, "Two schemes to reduce interconnect delay in bi-directional and uni-directional buses," in *Proceedings of IEEE Symposium on VLSI Circuits (VLSIC '01)*, pp. 193–194, Kyoto, Japan, June 2001.

[21] K. Hirose and H. Yasuura, "A bus delay reduction technique considering crosstalk," in *Proceedings of the Design, Automation and Test in Europe Conference (DATE '00)*, pp. 441–445, Paris, France, March 2000.

[22] M. Pedram, Q. Wu, and X. Wu, "A new design of double edge triggered flip-flops," in *Proceedings of the 3rd Conference of the Asia and South Pacific Design Automation Conference (ASP-DAC '98)*, pp. 417–421, Yokohama, Japan, February 1998.

[23] R. Bashirullah, W. Liu, R. Cavin, and D. Edwards, "A hybrid current/voltage mode on-chip signaling scheme with adaptive bandwidth capability," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 8, pp. 876–880, 2004.

[24] Predictive Technology model, http://www.eas.asu.edu/~ptm/.

Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 46514, 13 pages
doi:10.1155/2007/46514

Research Article

High-Performance Long NoC Link Using Delay-Insensitive Current-Mode Signaling

Ethiopia Nigussie,[1] Teijo Lehtonen,[1,2] Sampo Tuuna,[1] Juha Plosila,[1,3] and Jouni Isoaho[1]

[1] *Department of Information Technology, University of Turku, 20014 Turku, Finland*
[2] *Turku Centre for Computer Science (TUCS), 20520 Turku, Finland*
[3] *Research Council for Natural Sciences and Engineering, Academy of Finland, 00501 Helsinki, Finland*

Received 1 November 2006; Revised 24 January 2007; Accepted 1 March 2007

Recommended by Maurizio Palesi

High-performance long-range NoC link enables efficient implementation of network-on-chip topologies which inherently require high-performance long-distance point-to-point communication such as torus and fat-tree structures. In addition, the performance of other topologies, such as mesh, can be improved by using high-performance link between few selected remote nodes. We presented novel implementation of high-performance long-range NoC link based on multilevel current-mode signaling and delay-insensitive two-phase 1-of-4 encoding. Current-mode signaling reduces the communication latency of long wires significantly compared to voltage-mode signaling, making it possible to achieve high throughput without pipelining and/or using repeaters. The performance of the proposed multilevel current-mode interconnect is analyzed and compared with two reference voltage mode interconnects. These two reference interconnects are designed using two-phase 1-of-4 encoded voltage-mode signaling, one with pipeline stages and the other using optimal repeater insertion. The proposed multilevel current-mode interconnect achieves higher throughput and lower latency than the two reference interconnects. Its throughput at 8 mm wire length is 1.222 GWord/s which is 1.58 and 1.89 times higher than the pipelined and optimal repeater insertion interconnects, respectively. Furthermore, its power consumption is less than the optimal repeater insertion voltage-mode interconnect, at 10 mm wire length its power consumption is 0.75 mW while the reference repeater insertion interconnect is 1.066 mW. The effect of crosstalk is analyzed using four-bit parallel data transfer with the best-case and worst-case switching patterns and a transmission line model which has both capacitive coupling and inductive coupling.

1. INTRODUCTION

Network-on-Chip (NoC) is the most viable solution for on-chip communication that provides good scalability and enables gigascale integration in single-chip systems. One of the basic reasons for good scalability is that the length of connections is held constant and the signaling is kept local, from one router to another with the maximum distance of few millimeters. However, when the chip size increases, the latency for messages traversing from a processing unit far away from another becomes large. This is either due to the lack of fast paths between remotely situated nodes or due to the type of topology which has long channels. For example, in regular mesh structure to send a data packet between remotely located nodes, the message has to traverse many hops which increases the probability of a message to be blocked. This leads to unpredictable message latencies and difficulty in achieving guaranteed service operations. In [1], it is showed that using a few additional long-range links in a mesh network reduces the average packet latency significantly and improves the achievable throughput substantially. From topological point of view, the end-around channels in torus network are long which results in excessive latency. This problem can be avoided by folding a torus network. However, folding the torus eliminates the long end-around channels at the expense of doubling the length of the other channels [2] and increasing the layout complexity. Thus, it is preferable to use torus without folding if the long end-around channels can be implemented using high-performance signaling techniques. Both of these cases, the mesh structure with additional long links and the torus structure, require the use of long high-performance NoC links that pass over more than one processing element, thus the length of channels is 4 mm or more [3]. The structures of mesh network

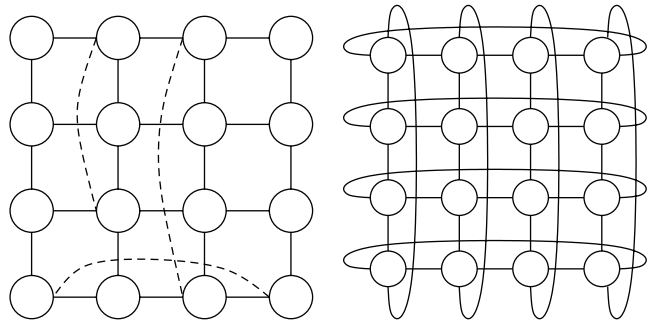


Figure 1: NoC architectures using long links. (a) 4×4 mesh with added long links. (b) 4-ary 2-cube torus.

with additional long distance links and torus network are illustrated in Figure 1.

The physical performance of long wires suffers greatly under technology scaling because the length is not scaled instead even longer wires are needed due to the increase of on-chip size. This makes long-range on-chip communication increasingly expensive [4]. The higher wire resistance, increased length, and decreased wire spacing cause the wire delay to increase considerably compared to the gate delay. In order to control this increase, designers scale down the wire cross-sectional area at a slower rate which prevents the dramatical increase of wire resistance. This ongoing trend of controlling the *RC* delay, combined with the faster rise/fall times and longer wires, results in a situation where the inductive part of the wire impedance can no longer be ignored. Thus, in addition to the capacitive coupling, the inductive coupling also causes crosstalk noise which creates more signal integrity problems.

Furthermore, the impact of process, supply voltage, and temperature variations on the performance and reliability of long on-chip links is expected to increase as technology scales down [5]. These variations cause the signal propagation delay through interconnects to be uncertain which in turn affects the performance and reliability of the system significantly. Moreover, the power dissipation due to global interconnect is increasing compared to the power consumption of the logic.

In order to achieve high-performance on-chip communication, it is necessary to implement efficient signaling technique. Current-mode signaling is faster and has lower dynamic power consumption than voltage-mode signaling. It is also immune to power supply noise and has reduced sensitivity to process-induced variations. Due to these advantages, we use current-mode signaling for the implementation of high-performance long-range NoC links. The delay variations problem can be tackled by using delay-insensitive communication. In this work we combine self-timed 1-of-4 encoded communication protocol with current-mode signaling for achieving a high-performance delay-variation-insensitive long-range on-chip communication.

The paper is organized as follows. We first discuss the principles of self-timed communication and present the delay-insensitive 1-of-4 encoding in Section 2. In Section 3 we discuss the advantages of current-mode signaling compared to voltage mode. In Section 4, brief discussion about multilevel current-mode signaling and its usage in our interconnect design is presented. The implementation of the signaling circuitry for self-timed 2-phase 1-of-4 encoded multilevel current-mode signaling is presented in Section 5 together with the implementations of the two reference 2-phase 1-of-4 encoded voltage-mode signaling circuits. The first reference uses pipelining and the other one uses optimal repeater insertion. In Section 6, first, the wire model used during simulations is presented followed by analysis of the presented current-mode signaling and the reference voltage-mode signaling techniques in latency, throughput, power consumption, and noise tolerance. Section 7 contains discussion about the results and future work, and finally conclusions are presented in Section 8.

2. SELF-TIMED COMMUNICATION

A NoC system consists of many processing blocks which have different timing requirements and can operate at different clock frequencies. Communication between these blocks needs synchronization which is error-prone. Also the clock distribution over a wide chip with low skew and jitter is problematic. A viable solution for this is the use of the globally asynchronous locally synchronous (GALS) design approach, where communication between processing blocks is done asynchronously. Therefore, we base our link on self-timed design principles.

The choice of the handshake protocol affects the throughput of a communication link. The two-phase protocol is often preferred instead of four-phase protocol for long on-chip interconnects to avoid the usage of a time-consuming spacer (return-to-zero phase) between two consecutive data symbols [6]. The use of two-phase protocol also minimizes power consumption since there is less transitions in the control wires.

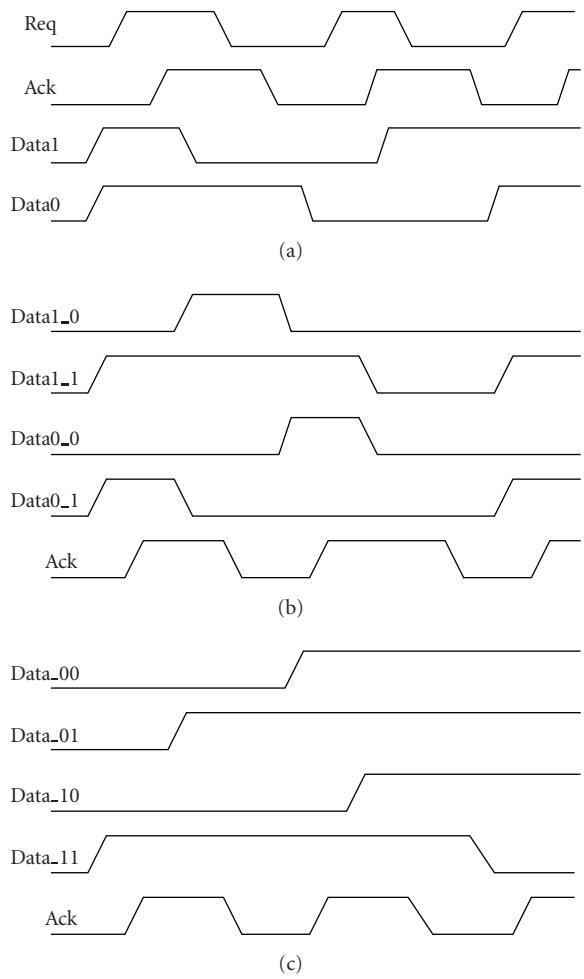


Figure 2: Self-timed communication. (a) 2-phase bundled-data (transmitting data "1101001011"). (b) 2-phase dual-rail (transmitting data "1101001011"). (c) 2-phase 1-of-4 (transmitting data "1101001011").

The communication can be carried out using control wires separately of the data. In this *bundled-data* approach, it is assumed that by the time request arrives, the data have already arrived. 2-phase bundled-data signaling is presented in Figure 2(a). To get rid off the timing constraints, the data validity indicator signal can be included in the data resulting in delay-insensitive communication. The delay-insensitive handshake protocol in which the data validity is transmitted implicitly operates correctly regardless of the delay in the interconnecting wires. The simplest one of the delay-insensitive protocols is the *dual-rail* protocol, which is demonstrated in Figure 2(b). In dual-rail, there are two wires for each bit, one for *zero* and the other for *one*. Either one of these signals is toggled and so at the receiver it can be noticed when all the bits have arrived regardless of their different delays.

In 1-of-4 data encoding, a group of four wires is used to transmit two bits of information per symbol. A symbol is one of the two-bit codes 00, 01, 10, or 11 and it is transmitted through activity on one of the four wires. Since it is possible to detect the arrival of each symbol at the receiver, 1-of-4 encoding is delay-insensitive, as are all the 1-of-N codes [7]. The 1-of-4 signaling is illustrated in Figure 2(c). Delay-insensitive data communication is a viable method to realize robust on-chip interconnects in future nanoscale technologies in which significant signal propagation delay variations are unavoidable. These delay variations occur due to different reasons, for example, due to crosstalk, temperature, supply voltage, and process variations. Besides being delay-insensitive, 1-of-4 encoding has more immunity against crosstalk effects as compared to single-rail (bundled-data) encoding, because the likelihood of two adjacent wires switching at the same time is much smaller. Furthermore, dynamic power consumption due to wire capacitance is smaller for the 1-of-4 code than for the simpler 1-of-2 (dual-rail) code. This is because the 1-of-4 code conveys two bits of information using only a single transition, while the 1-of-2 code requires two transitions for two bits of information. This effect can be seen in Figure 2. Considering these advantages, 2-phase 1-of-4 encoding is used in the proposed multilevel current-mode interconnect.

3. CURRENT-MODE SIGNALING

The signal transmission systems used in CMOS circuits can be broadly classified into two categories: voltage-mode and current-mode signaling. The important difference between the two transmissions systems lies in the type of signal that is forced on the transmission medium. While voltage mode uses voltage as signal, current mode uses current. In voltage mode, the voltage has to swing from rail to rail over the entire length of the wire. This leads to large transient currents consuming more power, larger delay, and it also generates power-supply noise [8]. The optimal repeater insertion technique [9] used in voltage-mode signaling was developed to reduce the wire delay and improve the performance of global interconnections. However, with the increase in number and density of interconnects with technology scaling, the number of repeaters necessary would increase considerably, presenting significant overhead in terms of power and area.

The key to current-mode signal transporting is the low-impedance termination at the receiver which results in reduced signal swings without the need of separate voltage references and increased bandwidth performance. Also this low-impedance termination shifts the dominant pole of the system and leads to a smaller time constant and thus, to a smaller delay. It can operate at a much lower noise margin than the voltage-mode network, and at a much lower swing as well due to its immunity to power supply noise. All these translate into increased bandwidth performance [10], decreased delay and dynamic power dissipation and higher noise immunity. For these reasons, current-mode signaling technique becomes a better alternative than voltage mode for contemporary and future high-speed noise-prone single-chip systems.

Current-mode signaling has already been proven to provide drastic speed enhancements for on-chip signaling [11–13]. It is also shown theoretically in [11] that current-mode

signaling can be three times faster than voltage-mode signaling.

There are three primary sources of power dissipation in current-mode circuits: static, dynamic, and short-circuit power dissipation. In current-mode signaling, static power dissipation is the major component of the total power dissipation that arises from the constant current path from Vdd to ground via the termination. Static power dissipation can be minimized using different circuit techniques which reduce leakage currents. Dynamic power is dissipated when the parasitic capacitance of the wire is charged and discharged. Since current-mode signaling operates at low-voltage swing, dynamic power consumption is not as significant source of power dissipation like in voltage-mode signaling. The third source of power dissipation arises from the finite input signal edge rates that result in short-circuit current. Generally, careful control of input edge rates can minimize the short-circuit current component to within 20% of the total dynamic power dissipation [14].

The other important feature of current-mode signaling is its reduced delay sensitivity due to process-induced variations [15]. Inspired by the advantages explained above, we investigate here the use of current-mode signaling for implementing high-performance delay-insensitive links for NoC long-range communication.

4. MULTILEVEL CURRENT-MODE SIGNALING

In delay-insensitive transmission, the data validity indicator is the transmitted data itself. Due to this, the transmission of every new data needs to be seen in the transmitting wire usually in the form of voltage level or transition depending on the type of handshake protocols. Using transition in current-mode signaling may cause unnecessary power consumption due to the constant current flow in some of the wires which have been previously made a transition to high state. In order to save this power, the presented current-mode interconnect allows current flow in the wires only during the respective symbol transmission. In this power-saving transmission scheme, it is not possible to see the arrival of new data during consecutive same symbol transmission using binary current-mode signaling. Due to this, three current levels are required in the proposed current-mode interconnect, two nonzero current levels to differentiate between consecutive same symbol transmissions and the third current level (zero current) to indicate the wire is idle, that is, there is no data transmission through that wire. The transmitted multilevel current is first detected at the receiver by a detecting circuit based on a current comparator. Then, the encoded voltages are estimated using decoding circuitry.

Multilevel current-mode signaling has been demonstrated to be robust and power-efficient in interchip signaling [16, 17]. In addition, using an analogy between digital communication over a band-limited channel and on-chip signaling, it is shown that for a given bit error rate and data rate, four-level current-mode signaling is the most power-efficient compared to binary voltage and current-mode signaling [18]. In this type of signaling the acceptable number of current levels and the step size between them are limited by the noise margin. In [19], it is shown that radix-8 full-adder design using eight current levels and 10 μA step size gets large enough noise margin.

Due to mismatch, parameter variations, noise, and other nonidealities, the current levels at the receiver input may deviate from the one predefined in the driver. This may lead to decoding error if the steps between different current levels are not enough and the current comparator has low noise margin. In addition, it is necessary to decode out the data in voltage form as fast as possible to fulfill the requirement of high data transmission rate. The key to achieve these is a large current comparator gain which provides sharp transition and greater noise margins which can accommodate all current levels. Lower threshold current values will increase the gain at the expense of greater comparator delay times. The usual approach to counteract the delay penalty is scaling the input current of the comparator lower than the input using current mirror division and then comparing these scaled input currents to reference currents.

Some fast and robust on-chip links based on multilevel current-mode signaling have been proposed [20–22]. In this paper, we present a high-performance interconnect which uses 1-of-4 data encoding and three distinct current levels per data wire. This interconnect has superiority and uses different approaches compared to [20–22]. In [20], 2-color 1-phase dual-rail encoding using four current levels is presented. As stated in Section 2, the dynamic power consumption due to wire capacitance of dual-rail is larger than 1-of-4 encoding since it requires two transitions rather than one to transmit two-bit data. In addition, it is more susceptible to crosstalk effects because there is a larger probability of adjacent wires switching at the same time than that of 1-of-4 encoding. Furthermore, using three current levels instead of four allows our proposed interconnect to have a larger noise margin than in [20]. The presented multilevel current-mode interconnect is also superior to [21] in terms of performance, power consumption, crosstalk, and noise margin. The interconnect in [21] is designed using 2-phase single-rail encoding with delay-insensitive feature and supporting simultaneous bidirectional data transmission. Although this approach decreases the required number of wires by half, it makes the signaling circuitry more complex, dissipates much more power, and has a significant decrease in signal transmission speed. Also due to the requirement of 7 current levels per wire, the noise margin of this interconnect decreases considerably compared to the presented three current-level interconnect. Moreover, the proposed three-current level interconnect has more immunity to crosstalk than [21] since it uses 1-of-4 encoding rather than single-rail. The interconnect proposed in [22] is designed using synchronous approach and allows to transmit two-bit data per wire. Since the reported performance and power consumption result is using 100 nm technology it is difficult to compare with our interconnect results. However, asynchronous interconnect has many advantages over synchronous one especially in the nanotechnology design of on-chip interconnect. The most relevant ones are the avoidance of clock and

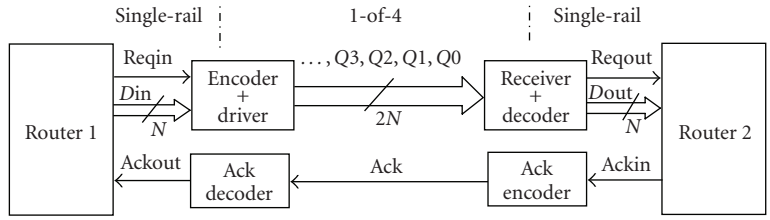


Figure 3: Conversion of single-rail to 1-of-4 and back to single-rail encoding.

clock-related problems and allowing delay-insensitive data transfer.

5. IMPLEMENTATIONS

In the subsequent sections, we present two different on-chip link implementations based on 1-of-4 data encoding. Both of them use two-phase protocol, the difference being that one is implemented using voltage-mode signaling and the other using multilevel current-mode signaling. The most common data encoding in GALS design is single-rail (bundled-data) encoding which uses N wires to transfer N-bit information and two additional handshake wires indicating data validity and acceptance. Since this encoding has a timing constraint between control (data validity) and data wires, communication through long on-chip interconnect becomes sensitive to delay variations. Therefore, converting single-rail encoding to delay-variation-insensitive encoding is mandatory for long on-chip communication where delay variations are unavoidable. The general block diagram of the considered signaling system is shown in Figure 3. We assume that the communicating parties, routers 1 and 2, have voltage-mode bundled-data (i.e., single-rail encoded) interfaces. The bundled-data protocol is then converted into the appropriate delay-insensitive 1-of-4 protocol and back to bundled-data protocol by the encoder/decoder units attached to the routers 1 and 2.

5.1. Two-phase 1-of-4 encoded voltage-mode interconnect (TPVm)

In the TPVm scheme, which serves as the reference for the current-mode implementation, one of the four wires makes transition to indicate the presence of a new two-bit symbol. When this new symbol arrives to the receiving module, the receiver accepts the symbol and sends an acknowledgement to the sender module by changing the state of the acknowledge signal. Since voltage-mode signaling is used, the voltage on the interconnect swings from rail to rail over its entire length. This leads to large dynamic power consumption, large delay, and generation of power supply noise. The usual approach to improve the performance of a voltage-mode interconnect is to insert repeaters or pipeline latches. Inserting repeaters decreases the signal propagation delay at the cost of increasing power consumption and chip area. A higher throughput can be obtained by using pipeline latches instead of repeaters to both amplify the signal and spread the link delay over multiple pipeline stages. This further increases power consumption and area costs compared to the simple repeater approach. We consider here both schemes for the reference voltage-mode 1-of-4 encoded interconnect. The pipelined and repeater-based implementations are called TPVmP and TPVmRep, respectively. In the TPVmP implementation pipeline stages are inserted in every 2 mm along the link wire. This is based on the assumption that the typical distance between two neighbouring (adjacent) routers in the mesh structure is 2 mm [3] and that the local link length can be considered an upper limit for pipeline-free signal transmission [1]. In TPVmRep implementation optimal repeater insertions are used for both data and acknowledgment transmissions. The required optimal number of repeaters and optimal size of the repeater are calculated using [23, equation (36)]. Using this equation, the required number of optimal repeaters becomes 2.22*L and the optimum size of the repeater becomes 76.5*minimum size inverter, where L is the wire length.

The straightforward gate level implementations of the encoder which converts the two-phase single-rail input to the delay-insensitive two-phase 1-of-4 protocol, the pipeline stage, and the decoder and completion detector which converts the delay-insensitive code back to the two-phase single-rail form at the receiver side are shown in Figure 4. The encoder consists of NOR gates which generate the select inputs for the multiplexers depending on the two-bit input codes, double-edge triggered flip-flops which are used to sample the symbol value at both edges of the request signal, and multiplexers each of which allows transition on the corresponding flip-flop output only when the appropriate input symbol is present. The decoder and completion detector circuit consists of XNOR gates which detect the transitions on the wires, NAND gates and an SR latch to decode the data back into the single-rail form, and a four-input XOR gate together with an $N/2$-input C-element for detecting completion. A C-element is a basic building block of self-timed logic. It is a state-holding element, a special kind of latch. When all of its inputs are 0 or 1, the output is set to 0 or 1, respectively. For other input combinations, it preserves its state. Its truth table is shown in Table 1 where t and $t-1$ indicate the current and previous values, respectively, and $-$ indicate, do not care.

Table 1: The truth tables of 2- and 3-input C-elements.

$a_{0,t}$	$a_{1,t}$	c_t
0	0	0
0	1	c_{t-1}
1	0	c_{t-1}
1	1	1

$a_{0,t}$	$a_{1,t}$	$a_{2,t}$	c_t
0	0	0	0
0	0	1	c_{t-1}
0	1	−	c_{t-1}
1	0	−	c_{t-1}
1	1	0	c_{t-1}
1	1	1	1

An inverter is used as both driver and receiver for the transmission of the two-phase acknowledgment signal between the pipeline stages in the TPVmP implementation.

5.2. Pulsed 1-of-4 encoded multilevel current-mode interconnect (PMCm)

The PMCm scheme converts two-phase single-rail voltage-mode signaling into pulsed 1-of-4 multilevel current-mode signaling at the transmitter side. At the receiver side, delay-insensitive current-mode signaling is turned back into single-rail voltage-mode communication. The PMCm scheme is logically equivalent to the TPVm scheme described above, but now information is presented as current rather than voltage transitions. Hence, one of the four data wires draws current to indicate the presence of a new two-bit data symbol. Similarly, an acknowledgement is signaled as current on the acknowledgement wire. As explained in section 3, such current-mode implementation is inherently much faster and more immune against power supply noise and delay variations compared to the voltage-mode implementation. The communication protocol is shown in Figure 5 (from the receiver's perspective) and the signaling circuits are depicted in Figures 6 and 7. The advantage of this link implementation is that high throughput and low latency can be achieved without using pipelining or repeaters.

The multilevel and pulsed nature of the PMCm scheme can be seen in Figure 5. The current detected at the receiver has three different values: 0, I, and $2I$. The values I and $2I$ are used when the voltage-mode request signal *Reqin* at the transmitter side is low and high, respectively, reflecting the adopted two-phase communication protocol. The value 0, in turn, means that there is no symbol on a wire. It is used as the initial value of the data wires and for switching off current on a wire when the 2-bit symbol to be transmitted changes, making current on a wire pulse shaped. This feature reduces the overall power consumption of the current-mode interconnect. The values of I and $2I$ are determined by considering the speed, power consumption, and noise margin of the interconnect. In the following consecutive sections, the implementations of the encoder, decoder, and completion detector are separately discussed.

5.2.1. Encoder and driver

The encoder takes the request and two data bits in the voltage-mode single-rail form and converts this information into multilevel current-mode 1-of-4 signaling. The double-edge triggered flip-flops shown in Figure 6 are used to sample the value of the 2-bit data symbol at each transition of the two-phase request signal *Reqin*. For instance, consider the encoder circuit of the wire $Q3$. Depending on the value of the signal *Reqin*, either transistor *Mn1* or *Mn2* conducts making either current I or $2I$ to flow through the wire $Q3$ when the symbol "11" has arrived from the sender module. To prevent the line from drawing current continuously, the transistor *Mn4* is used to ground the line when other than the symbol "11" is sent. The reset signal *rst* is controlled by the transmitting module. When a data burst is about to begin, *rst* is set to high enabling the sampling flip-flops. When the burst has been completed, *rst* is initialized back to low, meaning that all the data wires become grounded. This is necessary to prevent data wires of the link from drawing current (consuming power) during possibly long idle periods between bursts. In nanometer technology, where NoC is one of the promising candidates, process variation effects are one of the major concerns. Due to process variation effects, the driver output currents may vary from their expected values. In order to minimize this variation, transistors *Mp1* and *Mp2* which operate in the linear region form resistive path from the supply voltage to *Mn1* and *Mn2* which in turn keeps the switching threshold of *Mn1* and *Mn2* transistors constant.

5.2.2. Receiver and current comparator

At the receiver side, consider the current comparator circuit of $Q3$, as depicted in Figure 7. It is composed of the diode-connected input NMOS transistor *Mn2*, the NMOS transistors *Mn3* and *Mn4* connected to replicate this input current, the reference or threshold current generating pair of transistors *Mn1* and *Mp1*, and the PMOS transistors *Mp2* and *Mp3* that replicate the threshold current. In addition to serving as an input transistor, *Mn2* acts also as a termination load. The drains of the PMOS reference current replicating transistors and line current replicating NMOS transistors are connected together to generate the comparator circuit's output voltages. This comparator provides a logical high output voltage when its input current $I(Q3)$ is less than the threshold current and a logical low output voltage when the input current $I(Q3)$ is greater than the threshold current. Here the current comparator compares current on the wire $Q3$ with two different threshold currents, $0.5I$ and $1.5I$, in order to distinguish the three current levels. To be more specific, if $I(Q3) < 0.5I$, both comparator outputs $V(30)$ and $V(31)$ are high (initial state). If $0.5I < I(Q3) < 1.5I$, $V(30)$ and $V(31)$ are low and high, respectively. If $I(Q3) > 1.5I$, both $V(30)$ and $V(31)$ are low.

In nanometer technology, the line current at the input of the receiver may vary from the nominal value due to crosstalk, process variation effects, and other noise sources. However, this does not affect the reliability as long as the current levels are within the specified margins. Since there are only three current levels, it is easily possible to meet the required noise margins at minimal power consumption cost. In addition, the reference current may also vary from its nominal value due to process variations. This affects only

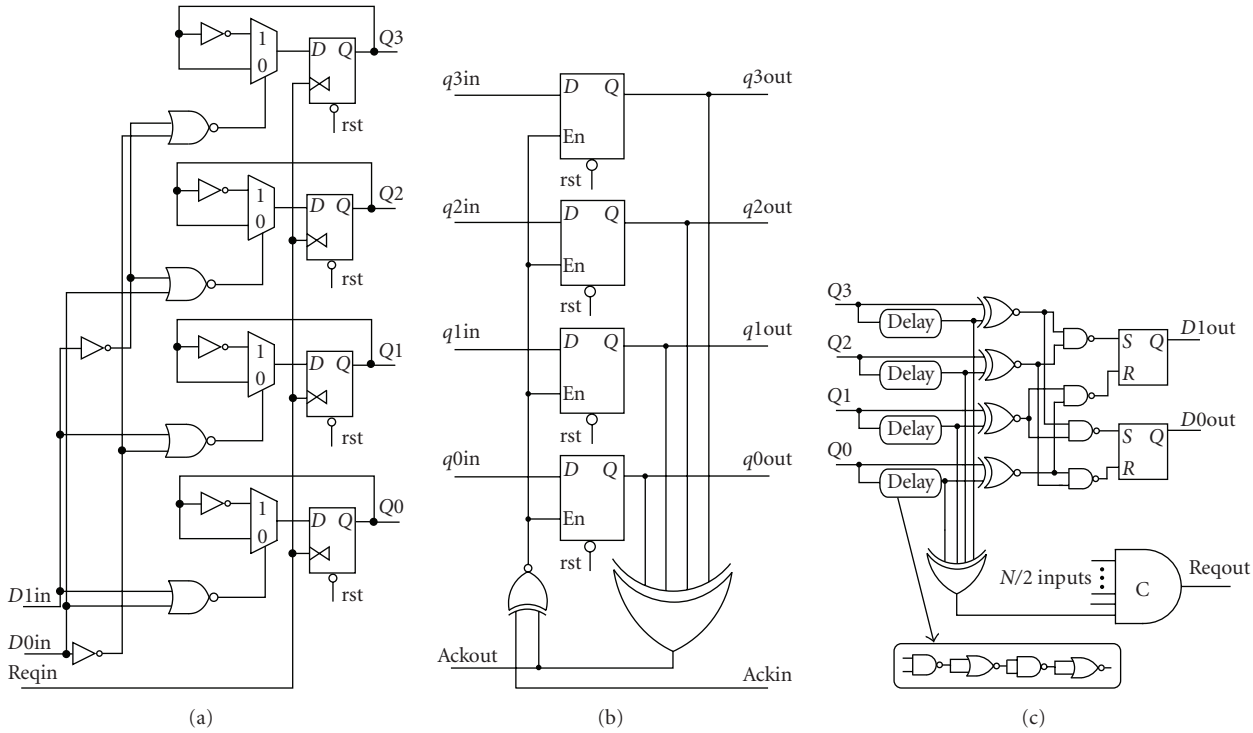


Figure 4: The reference 2-phase 1-of-4 encoded voltage-mode interconnect components. (a) Encoder. (b) Pipeline stage. (c) Decoder and completion detector.

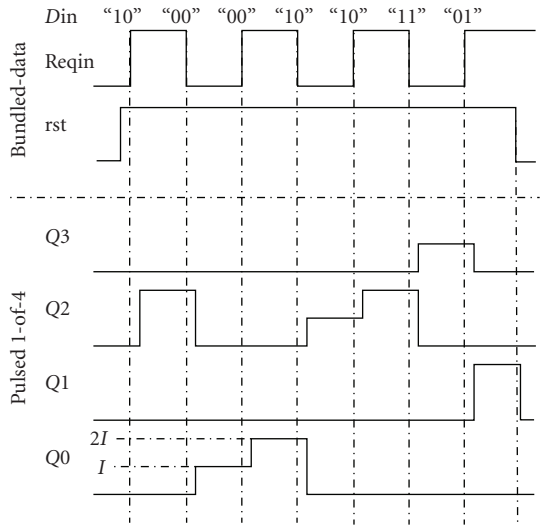


Figure 5: Communication protocol of 1-of-4 encoding in pulsed multilevel current-mode interconnect.

the speed but not the reliability of the communication since delay-insensitive data transfer mechanism is used. For example, if the reference current decreases from its nominal value, the comparison takes place ahead. This shifts the data output point as well as the data validity indicator to the left. Thus there is no threat to the reliability of the communication.

5.2.3. Decoder and completion detector

As shown in Figure 7, the data decoder, composed of three inverters and two OR gates, needs as inputs the outputs of the current comparators of the wires $Q3$, $Q2$, and $Q1$ to reconstruct the two bits ($D1out$, $D0out$) sent from the transmitter module. Only the comparator outputs of the threshold current $0.5I$ (i.e., $V(10)$, $V(20)$, and $V(30)$) are needed for this purpose. Formally, the logic is as follows:

$$\begin{aligned}
&(V(30) =' 0') \wedge (V(20) =' 1') \wedge (V(10) =' 1') \\
&\qquad \Longrightarrow (D1\text{out} =' 1') \wedge (D0\text{out} =' 1'), \\
&(V(30) =' 1') \wedge (V(20) =' 0') \wedge (V(10) =' 1') \\
&\qquad \Longrightarrow (D1\text{out} =' 1') \wedge (D0\text{out} =' 0'), \\
&(V(30) =' 1') \wedge (V(20) =' 1') \wedge (V(10) =' 0') \\
&\qquad \Longrightarrow (D1\text{out} =' 0') \wedge (D0\text{out} =' 1'), \\
&(V(30) =' 1') \wedge (V(20) =' 1') \wedge (V(10) =' 1') \\
&\qquad \Longrightarrow (D1\text{out} =' 0') \wedge (D0\text{out} =' 0').
\end{aligned} \tag{1}$$

The completion detector reads all current comparator outputs as illustrated in Figure 7. For each 4-wire block, the completion detection circuit includes two 4-input NAND gates (N0 and N1), a 2-input NAND gate (N2), and a resettable 2-input C-element (C1). To produce the receiver-side request signal *Reqout*, the completion signals of the $N/2$ 4-wire blocks are combined with an $N/2$-input C-element, where N is the bit-width of the transmitted data. The completion detection process is started by sensing the current

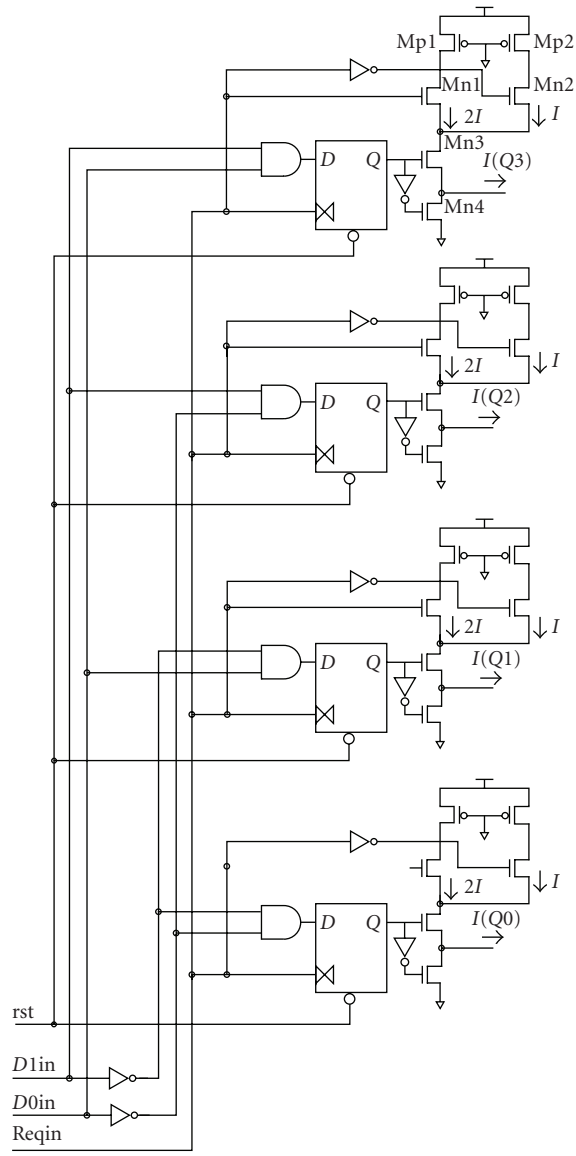


Figure 6: Encoder of pulsed multilevel current-mode interconnect.

values on the four wires. In our pulsed implementation of 1-of-4 encoding, current flows only in one of the four wires. Current through the wire becomes I or $2I$ when the transmitter-side request signal *Reqin* is low or high, respectively. Hence, if the input current of the comparator is greater than the threshold $1.5I$, then the output of the C-element C1 and subsequently the receiver-side request signal *Reqout* go high. Correspondingly, if the comparator input current is between the thresholds $0.5I$ and $1.5I$, the output of C1 and the signal *Reqout* go low. The completion detection logic uses as inputs the current comparator outputs $V(30)$ and $V(31)$ of $Q3$, $V(20)$ and $V(21)$ of $Q2$, $V(10)$ and $V(11)$ of $Q1$, and $V(00)$ and $V(01)$ of $Q0$. For instance, consider again the receipt of the symbol "11" through the wire $Q3$. Assuming that the transmitter-side request signal *Reqin* is high, the current on the wire $Q3$ is $2I$. Consequently, the comparator outputs

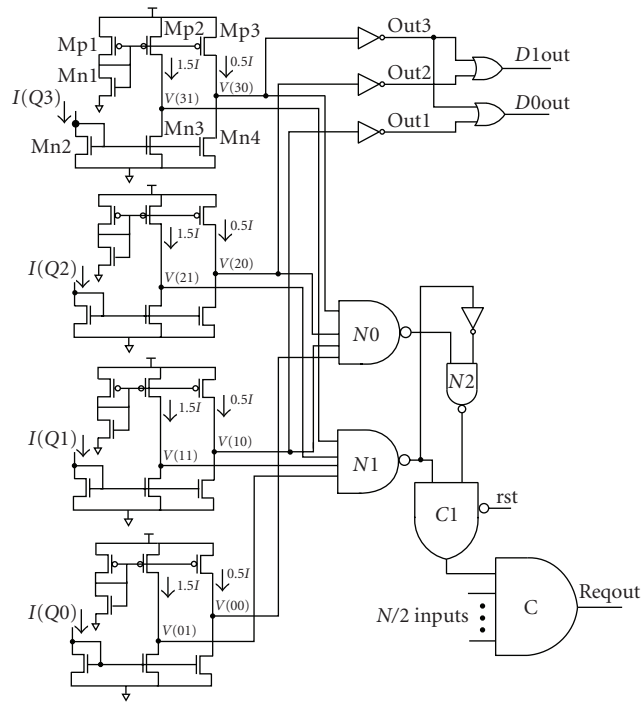


Figure 7: Decoder and completion detector circuits of pulsed multilevel current-mode signaling.

$V(30)$ and $V(31)$ become low, and all the other comparator outputs remain high since no current flows through the wires $Q2$, $Q1$, and $Q0$. This makes the outputs of the NAND gates N1 and N2 high, causing an up-going transition on the output of the C-element C1. Formally, the completion detection logic for the symbol "11" is as follows (we denote the output of a gate X by $O(X)$):

$$\begin{aligned}
&(V(30) =' 0') \wedge (V(31) =' 0') \quad \text{(current is } 2I\text{)} \\
&\quad \Longrightarrow (O(N0) =' 1') \wedge (O(N1) =' 1') \\
&\quad \Longrightarrow (O(N2) =' 1') \\
&\quad \Longrightarrow (O(C1) =' 1'), \\
&(V(30) =' 0') \wedge (V(31) =' 1') \quad \text{(current is } I\text{)} \\
&\quad \Longrightarrow (O(N0) =' 1') \wedge (O(N1) =' 0') \\
&\quad \Longrightarrow (O(N2) =' 0') \\
&\quad \Longrightarrow (O(C1) =' 0').
\end{aligned} \tag{2}$$

The waveforms of $V(30)$ and $V(31)$ are shown in Figure 8.

5.2.4. Acknowledgment transmission

The voltage-mode bundled-data acknowledge signal (*Ackin*), sent by the receiver module, is converted into a current-mode signal during transmission and back into a voltage-mode signal (*Ackout*) at the transmitter side. In this interconnect design transmission of acknowledgment signal also uses multilevel current-mode signaling. The current through

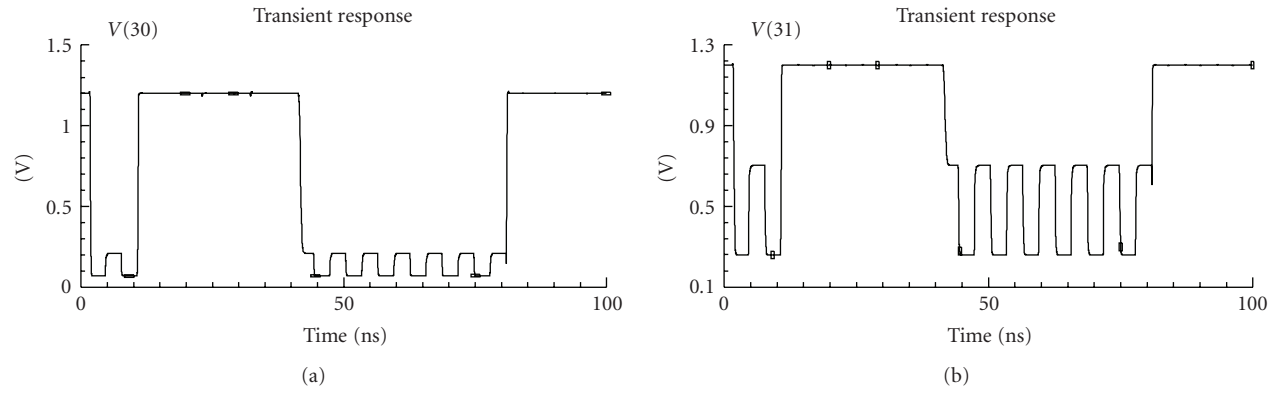


Figure 8: Outputs of current comparator.

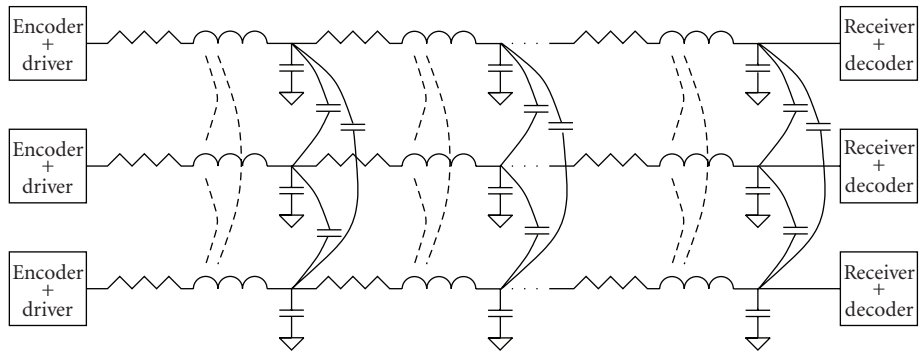


Figure 9: Distributed RLC model for capacitively and inductively coupled wires.

acknowledgment wire becomes I and $2I$ when acknowledgment signal from the receiving module is low and high, respectively. The same current comparator circuit is used to detect the value of the current through acknowledgment wire and output the result in voltage form. Inverter is used as a decoding logic.

6. ANALYSIS

6.1. Wire model

Due to the scaling of technology and increasing operating speeds, accurate modeling of wires has become a necessity. Wires have traditionally been modeled as lumped RC segments, but for long high-speed wires, transmission line modeling is needed. Transmission line modeling needs to be applied when the time of flight across the wire becomes comparable to the signal rise time. A transmission line can be thought as a large number of lumped segments in series so that they represent the distributed nature of the wire.

The importance of modeling inductive effects in wires is increasing because of faster rise times and longer wires. Wide wires used in upper metal layers can be especially susceptible to inductive effects due to their low resistance [24]. Since we are considering high-performance signaling over long wires, we modeled the wires using a distributed RLC model, as shown in Figure 9. In order to accurately model crosstalk noise, both capacitive and inductive coupling between all wires was included.

A 130 nm CMOS technology with metal 4 wires was used. The bus consisted of eight parallel wires. The *RLC* values of the wires were extracted using field solvers. The resistance and inductance matrices were extracted using FastHenry [25], while the capacitance matrices were extracted using Linpar [26]. The wire length was varied in the simulations from 2 mm to 12 mm, which corresponds to 1–6 expected processing unit widths.

6.2. Performance analysis

In this section, we consider latency and throughput as main parameters to analyze the performance of multilevel current-mode on-chip interconnects along with the two reference voltage-mode interconnects. The most common approach to achieve high-performance long-range on-chip communication is using pipelining or inserting repeaters in voltage-mode signaling. Thus in our first reference interconnect, TPVmP, pipeline stages are inserted every 2 mm assuming that the local wire length (between neighbour routers) is

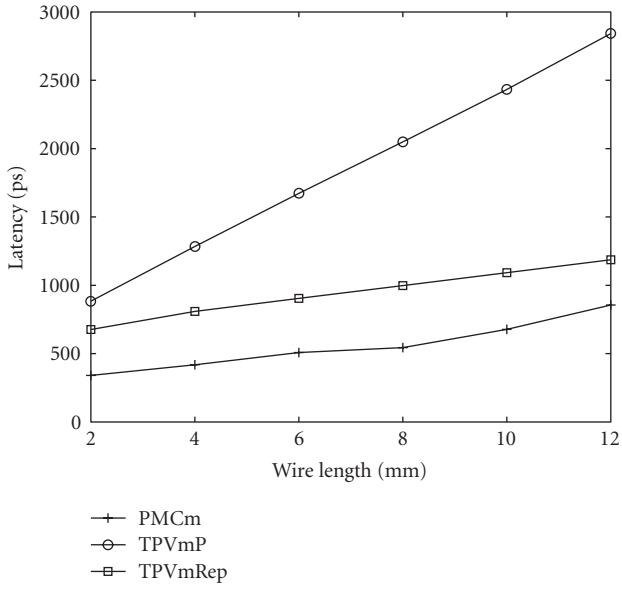


Figure 10: Forward latency of the interconnects.

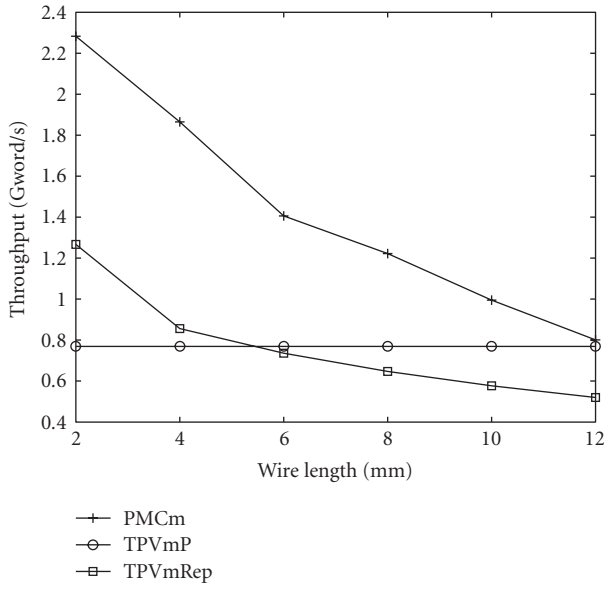


Figure 11: Throughput of the interconnects.

2 mm [3]. This improves the throughput at the expense of increased forward latency, power consumption, and chip area. In the second reference interconnect, TPVmRep, optimal size repeaters are inserted at optimal distances.

Here we define forward latency as the delay from a transition on the bundled-data request signal (*Reqin*) at the transmitter side to the corresponding transition on the bundled-data request signal (*Reqout*) at the receiver side (see Figure 3). In other words, the time required for one packet to traverse from the sending router to its receiving router. The change in the forward latency of the three interconnects when wire length is varied from 2 mm to 12 mm is shown in Figure 10. Since PMCm interconnect uses current-mode signaling, its forward latency is much smaller than the two reference interconnects. At global wire length of 8 mm, PMCm's forward latency was less than one third of TPVmP latency. The latency of pipelined voltage-mode interconnect was much larger than both PMCm and TPVmRep at global lengths of the wire.

The throughput of PMCm, along with the two reference interconnects, is shown in Figure 11 in Gword/s by assuming there is one word packet data transfer between the routers at a time. The throughput of PMCm was greater than the TPVmP and TPVmRep interconnects at all wire lengths (2 to 12 mm) of the interconnect. In case of the reference interconnects, TPVmP achieved a throughput of 769 Mword/s while the throughput of TPVmRep is varied from 1.267 Gword/s to 520 Mword/s when the wire length is varied from 2 to 12 mm. The reported latency and throughput values are for one group of 1-of-4 encoding (2-bit data transfer).

Therefore PMCm interconnect is a better alternative than TPVmP and TPVmRep to realize high-performance long-range NoC links. In addition to achieving high-performance, PMCm circuitry is simpler and takes a smaller chip area compared to pipelined and optimal repeater insertion TPVm. This is because the complexity and required chip area of encoder and decoder of both TPVm and PMCm interconnects are almost the same. However, the number of required pipeline stages and the number of repeaters increase with wire length which makes the two reference TPVm interconnects complex and require larger area for long-range NoC links.

6.3. Power analysis

The average total power consumption for 2-bit data transfer of the proposed current mode and the two reference interconnects when wire length is varied from 2 to 12 mm is shown in Figure 12. The power consumption of PMCm was higher than that of TPVmP at all wire lengths, but its power consumption was lower than that of TPVmRep starting from 6 mm wire length. The power consumption of TPVmP increases at a faster rate with wire length compared to PMCm due to the increase in the number of pipeline stages. Thus at global lengths of wires, the difference in power consumption between these two interconnects decreases considerably. Due to the increase in the number of repeaters inserted at global lengths of the wire, power consumption of TPVmRep is much larger than the other two interconnects.

Here we use a metric called power-throughput ratio which measures the energy consumed per data transmission. This actually corresponds to the power-delay product metric of logic gates. The power-throughput ratio of PMCm is significantly less than that of TPVmRep and slightly greater than that of TPVmP at intermediate and global wire lengths as shown in Figure 13. The voltage-mode interconnect with repeaters has much larger power-throughput ratio than the TPVmP and PMCm interconnects.

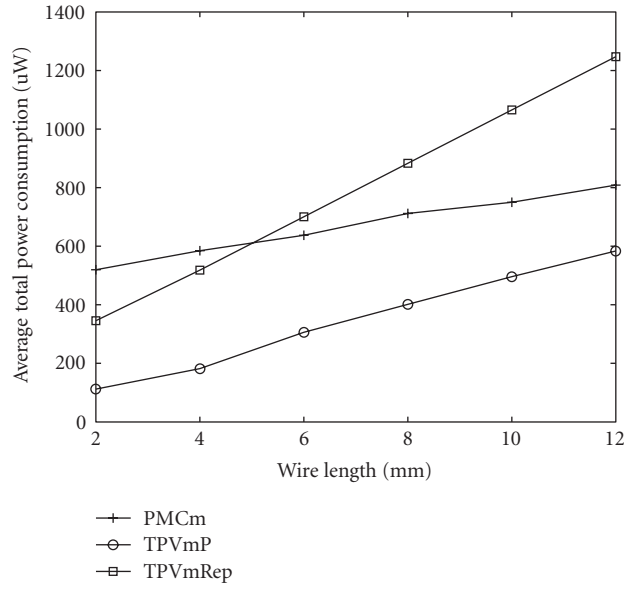


Figure 12: Average total power consumption of the interconnects.

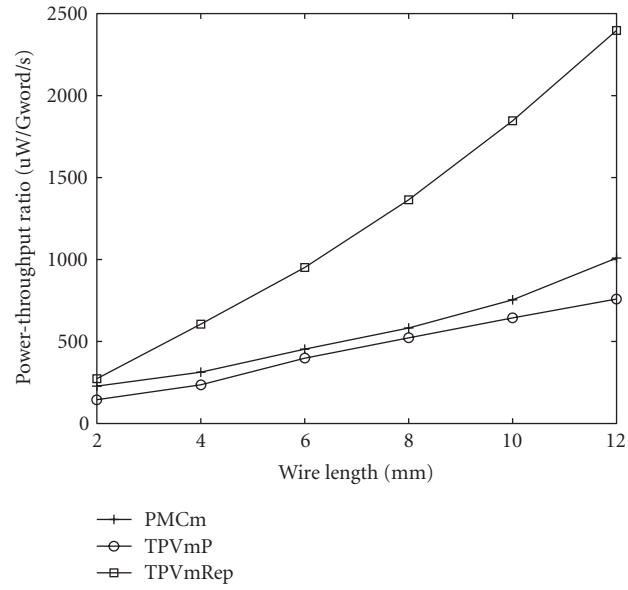


Figure 13: Power consumption per throughput of the interconnects.

6.4. Noise analysis

The impact of crosstalk noise on latency and throughput was also studied. In this analysis, 4-bit parallel data transfer was assumed. This requires 9 (8 parallel data transmission +1 acknowledgment) physical wires since we are using 1-of-4 encoding. The acknowledgment wire was designed as having shielding from the parallel data transmission wires, to counteract the coupling effect. The wires were modeled as transmission lines which have both capacitive and inductive coupling between each other. During this analysis, minimum wire separation distance with minimum global pitch specified in 130 nm technology and 1.2 V supply voltage were used. The delay variation due to both capacitive and inductive coupling was simulated by considering the worst-case and best-case switching patterns. These switching patterns depend on the RLC values of the wire. In our case, we assumed that the capacitive coupling dominates the inductive coupling which is the most usual case in on-chip parallel wires. The effect of crosstalk on latency and throughput when the wire length was varied from 2 mm to 12 mm is shown in Figures 14 and 15, respectively.

During best-case and worst-case switching, the latency variation of TPVmP from the latency without crosstalk effect (nominal latency) was slightly less than the PMCm one. For example, at a wire length of 8 mm, the increase in latency due to best-case switching from the nominal latency of TPVmP and PMCm was 59.8% and 62.3%, respectively. In worst-case switching, the TPVmP and PMCm latency variations were 144% and 147%, respectively, at the same wire length. In fact, these percentage values are rather large because in the nominal case shown in Figure 10 the considered capacitive loads were only to ground. In other words, the nominal case capacitive loads do not consider the coupling capacitances loading effect. The decrease in throughput due to crosstalk was greater for TPVmP than for PMCm, specially at long wire length. For example at 12 mm wire length, the throughput of TPVm was decreased by 38% while the PMCm was only by 30%.

7. DISCUSSION

In order to mitigate real-life applications, we can assume there is a 64-bit data transfer using the long links presented in this work. In the pipelined voltage-mode interconnect, there is a need of completion detection circuits at both sides of the communication; in the receiver side to indicate the validity of the arrived data and in the transmitter side to indicate the acceptance of the transmitted data since an acknowledgment is sent per each 1-of-4 group. This requires 32-input C-element at both sides which creates a considerable delay, because its complexity approximately corresponds to that of a 32-input AND gate with multiple logic levels. However, the pulsed multilevel current-mode interconnect requires only receiver side completion detection since it is possible to use one acknowledgment signal per data transfer. Even though the delay due to the completion detection is reduced by half in the current-mode interconnect, it is necessary to have a fast completion detection mechanism. Thus, our future work will be designing of a fast and area efficient completion detection circuit for the pulsed multilevel current-mode interconnect, for example performing completion detection by sensing currents. This can be done by summing up the currents of all the wires and comparing the sum with a threshold current.

Based on *International Technology Roadmap for Semiconductors* [27], the long on-chip wire length can be even longer than 10 mm in future nanoscale technologies. The proposed current-mode interconnect throughput becomes

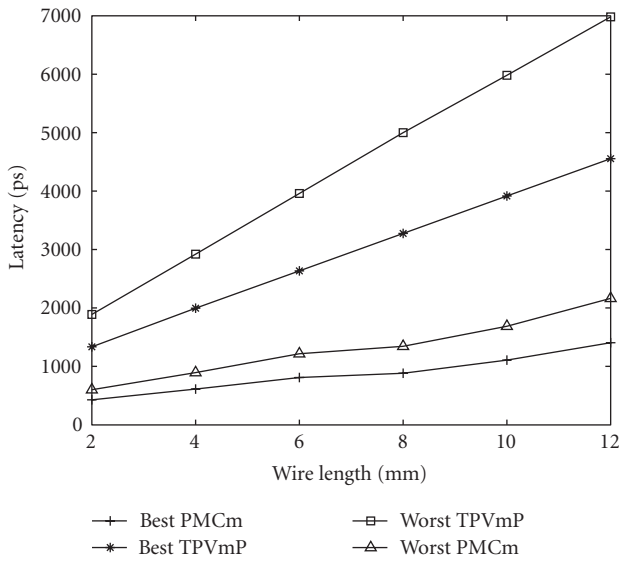


Figure 14: Forward latency of the interconnects in the presence of crosstalk.

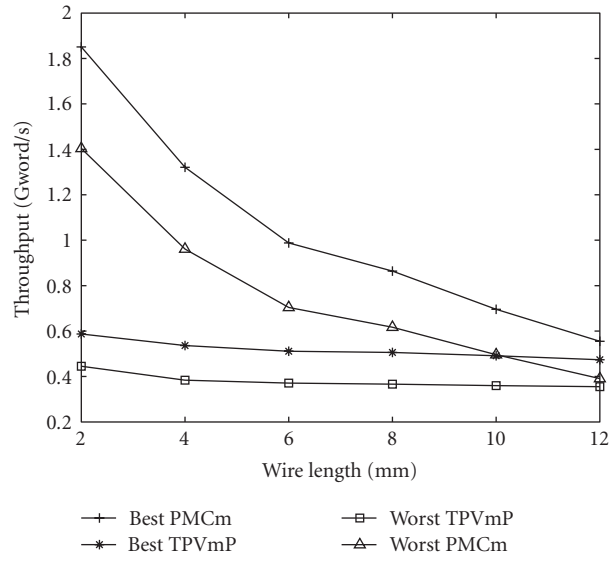


Figure 15: Throughput of the interconnects in the presence of crosstalk.

almost equal and even slightly less than the pipelined voltage mode throughput when the wire length exceeds 10 mm. To maintain the high throughput of the current-mode interconnect, an efficient current-mode pipeline stage could be inserted after every 10 mm wire length. This increases the forward latency but it will not be that significant since a pipeline stage is needed only every 10 mm.

Another direction of our future work is examining how much improvement in overall average latency and throughput of the network can be achieved by using our high-performance current-mode link for end-around torus channels and for additional long channels of mesh network and what are the expenses.

8. CONCLUSION

We presented a high-performance delay-variation-insensitive long on-chip interconnect which uses two-phase 1-of-4 encoding and multilevel current-mode signaling. This interconnect is a promising candidate for long-range NoC communication links since it has low latency, high throughput, and low power-throughput ratio. In addition, its delay-insensitive data transfer ability makes it appropriate for future nanoscale long-range NoC interconnects where delay variations are inevitable. Since the usual way of improving the performance of long on-chip interconnects is using voltage mode signaling along with either repeater insertion or using pipeline stages, we designed two-phase 1-of-4 encoded voltage-mode signaling references, one with pipeline stages and the other with optimally inserted repeaters. These voltage-mode interconnects serve as references to our proposed current-mode interconnect.

The performance analysis shows that the current-mode interconnect has higher throughput and lower latency than the two reference interconnects. It achieves a throughput of 1.222 Gword/s at 8 mm wire length which is 1.58 times higher than the throughput of the pipelined voltage-mode interconnect and 1.89 times higher than the one using optimal repeater insertion. From the power consumption analysis, it is seen that the current-mode interconnect consumes less power than the reference voltage-mode interconnect with optimal repeater insertion starting from the wire length of 6 mm. On the other hand, it consumes more power than the voltage-mode interconnect with pipeline stages for 2 to 12 mm wire length. However, the power consumption difference between these two interconnects becomes smaller when the wire length increases. The power-throughput ratio of the proposed current-mode interconnect is much less than the voltage-mode interconnect with optimal repeaters and slightly greater than the pipelined voltage-mode interconnect. The effects of crosstalk on latency and throughput of the interconnects are also analyzed. The variation in forward latency of the current-mode interconnect was a few percents larger than that of the pipelined voltage-mode interconnect. In case of throughput reduction due to crosstalk, the throughput of pipelined voltage mode is more affected than the current mode.

Therefore, using the proposed multilevel current-mode interconnect for long-range NoC links such as the torus end-around channels allows the network to achieve high throughput and low latency along with delay-variation-insensitive communication. The delay insensitivity makes the communication robust and attains average-case performance rather than worst-case performance which is the situation in communication based on timing constraints.

REFERENCES

[1] U. Y. Ogras and R. Marculescu, ""It's a small world after all": NoC performance optimization via long-range link insertion," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 7, pp. 693–706, 2006.

[2] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann-Elsevier, San Francisco, Calif, USA, 2004.

[3] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 684–689, Las Vegas, Nev, USA, June 2001.

[4] D. Sylvester and K. Keutze, "A global wiring paradigm for deep submicron design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 2, pp. 242–252, 2000.

[5] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proceedings of the 40th Design Automation Conference (DAC '03)*, pp. 338–342, Anaheim, Calif, USA, June 2003.

[6] R. Ho, J. Gainsley, and R. Drost, "Long wires and asynchronous control," in *Proceedings of the 10th International Symposium on Asynchronous Circuits and Systems (ASYNC '04)*, pp. 240–249, Crete, Greece, April 2004.

[7] T. Verhoeff, "Delay-insensitive codes—an overview," *Distributed Computing*, vol. 3, no. 1, pp. 1–8, 1988.

[8] W. J. Dally and J. W. Poulton, *Digital Systems Engineering*, Cambridge University Press, Cambridge, UK, 1998.

[9] D. Pamunuwa and H. Tenhunen, "Repeater insertion to minimise delay in coupled interconnects," in *Proceedings of the 14th IEEE International Conference on VLSI Design*, pp. 513–517, Bangalore, India, January 2001.

[10] R. Bashirullah, W. Liu, and R. K. Cavin III, "Current-mode signaling in deep submicrometer global interconnects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 3, pp. 406–417, 2003.

[11] A. Katoch, E. Seevinck, and H. Veendrick, "Fast signal propagation for point to point on-chip long interconnects using current sensing," in *Proceedings of the 28th European Solid-State Circuits Conference (ESSCIRC '02)*, pp. 195–198, Florence, Italy, September 2002.

[12] A. Katoch, H. Veendrick, and E. Seevinick, "High speed current-mode signaling circuits for on-chip interconnects," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '05)*, vol. 4, pp. 4138–4141, Kobe, Japan, May 2005.

[13] A. P. Jose, G. Patounakis, and K. L. Shepard, "Near speed-of-light on-chip interconnects using pulsed current-mode signalling," in *Proceedings of IEEE Symposium on VLSI Circuits Digest of Technical Papers*, pp. 108–111, Kyoto, Japan, June 2005.

[14] M. K. Gowan, L. L. Biro, and D. B. Jackson, "Power considerations in the design of the Alpha 21264 microprocessor," in *Proceedings of the 35th Design Automation Conference (DAC '98)*, pp. 726–731, San Francisco, Calif, USA, June 1998.

[15] R. Bashirullah, "Reduced delay sensitivity to process induced variability in current sensing interconnects," *Electronics Letters*, vol. 42, no. 9, pp. 531–532, 2006.

[16] J. Q. Zhang, S. I. Long, F. H. Ho, and J. K. Madsen, "Low power current mode multi-valued logic interconnect for high speed interchip communications," in *Proceedings of the 17th Annual IEEE Gallium Arsenide Integrated Circuit Symposium (GaAs IC '95)*, pp. 327–330, San Diego, Calif, USA, October-November 1995.

[17] J.-Y. Sim, Y.-S. Sohn, S.-C. Heo, H.-J. Park, and S.-I. Cho, "A 1-Gb/s bidirectional I/O buffer using the current-mode scheme," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 4, pp. 529–535, 1999.

[18] I. B. Dhaou, M. Ismail, and H. Tenhunen, "Current mode, low-power, on-chip signaling in deep-submicron CMOS technology," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 3, pp. 397–406, 2003.

[19] T. Temel and A. Morgul, "Implementation of multi-valued logic gates using full current-mode CMOS circuits," *Analog Integrated Circuits and Signal Processing*, vol. 39, no. 2, pp. 191–204, 2004.

[20] T. Hanyu, T. Takahashi, and M. Kameyama, "Bidirectional data transfer based asynchronous VLSI system using multiple-valued current mode logic," in *Proceedings of the 33rd International Symposium on Multiple-Valued Logic*, pp. 99–104, Tokyo, Japan, May 2003.

[21] E. Nigussie, J. Plosila, and J. Isoaho, "Delay-insensitive on-chip communication link using low-swing simultaneous bidirectional signaling," in *Proceedings of IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, pp. 217–222, Karlsruhe, Germany, March 2006.

[22] V. Venkatraman and W. Burleson, "Robust multi-Level current-mode on-chip interconnect signaling in the presence of process variations," in *Proceedings of the 6th International Symposium on Quality of Electronic Design (ISQED '05)*, pp. 522–527, San Jose, Calif, USA, March 2005.

[23] R. Venkatesan, J. A. Davis, and J. D. Meindl, "Compact distributed RLC interconnect models—part IV: unified models for time delay, crosstalk, and repeater insertion," *IEEE Transactions on Electron Devices*, vol. 50, no. 4, pp. 1094–1102, 2003.

[24] Y. I. Ismail, E. G. Friedman, and J. L. Neves, "Figures of merit to characterize the importance of on-chip inductance," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 4, pp. 442–449, 1999.

[25] M. Kamon, M. J. Tsuk, and J. K. White, "FASTHENRY: a mutipole-accelerated 3-D inductance extraction program," *IEEE Transactions on Microwave Theory and Techniques*, vol. 42, no. 9, pp. 1750–1758, 1994.

[26] A. R. Djordjevic, M. B. Bazdar, T. K. Sarkar, and R. F. Harrington, *Linpar for Windows: matrix parameters for multiconductor transmission lines*, Software and User Manual, Version 2.0, Artech House Publisher, Norwood, Mass, USA, 1999.

[27] International Technology Roadmap for Semiconductors, 2005, http://public.itrs.net/.

Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 94676, 13 pages
doi:10.1155/2007/94676

Research Article

Online Reconfigurable Self-Timed Links for Fault Tolerant NoC

Teijo Lehtonen,[1,2] Pasi Liljeberg,[2] and Juha Plosila[2,3]

[1] *Turku Centre for Computer Science (TUCS), Joukahaisenkatu 3–5 B, 20520 Turku, Finland*
[2] *Department of Information Technology, University of Turku, 20014 Turku, Finland*
[3] *Research Council for Natural Sciences and Engineering, Academy of Finland, 00501 Helsinki, Finland*

Received 15 October 2006; Accepted 4 March 2007

Recommended by Davide Bertozzi

We propose link structures for NoC that have properties for tolerating efficiently transient, intermittent, and permanent errors. This is a necessary step to be taken in order to implement reliable systems in future nanoscale technologies. The protection against transient errors is realized using Hamming coding and interleaving for error detection and retransmission as the recovery method. We introduce two approaches for tackling the intermittent and permanent errors. In the first approach, spare wires are introduced together with reconfiguration circuitry. The other approach uses time redundancy, the transmission is split into two parts, where the data is doubled. In both structures the presence of permanent or intermittent errors is monitored by analyzing previous error syndromes. The links are based on self-timed signaling in which the handshake signals are protected using triple modular redundancy. We present the structures, operation, and designs for the different components of the links. The fault tolerance properties are analyzed using a fault model containing temporary, intermittent, and permanent faults that occur both as bursts and as single faults. The results show a considerable enhancement in the fault tolerance at the cost of performance and area, and with only a slight increase in power consumption.

1. INTRODUCTION

The move towards nanoscale circuits increases performance and capacities of ICs, but poses new challenges to circuit design. As the dimensions shrink dramatically, it is becoming increasingly difficult to control the variance of physical parameters in the manufacturing process. This results in faults and decreased yield which increases the costs per functioning chip [1, 2]. The yield can be maintained at an acceptable level by admitting some amount of faults in a chip, which then have to be tolerated with dedicated circuit structures. Different types of errors can be tackled using a variety of fault tolerance methods. No single method is sufficient for all types of errors and therefore a sophisticated combination of them is needed [3].

Network-on-chip (NoC) is a structure believed to be the basic platform of future designs [4]. In such a system the communication links between system modules are crucial for the correct operation of system. In this paper, we focus on fault tolerance design of the communication links in NoC architectures. We take into consideration both the permanent and intermittent errors that are commonly originated from the manufacture process or produced by electromigration as well as transient errors caused, for example, by different noise sources and radiation.

The paper is organized as follows. In Section 2 we give background for the presented approach and discuss some related works. The idea is presented in Section 3 followed by the description of the created realizations in Section 4. The results are presented in Section 5 and in Section 6 the fault tolerance properties are analyzed. The power analysis is presented in Section 7, and Section 8 contains discussion about the results and possible enhancements to the design. Finally the conclusions are presented in Section 9.

2. BACKGROUND

An NoC system consists of many processing blocks which have different timing requirements and can operate at different clock frequencies. Communication between these blocks needs synchronization which is error-prone. Also the clock distribution over a wide chip with low skew and jitter is problematic. A viable solution for this is the use of the

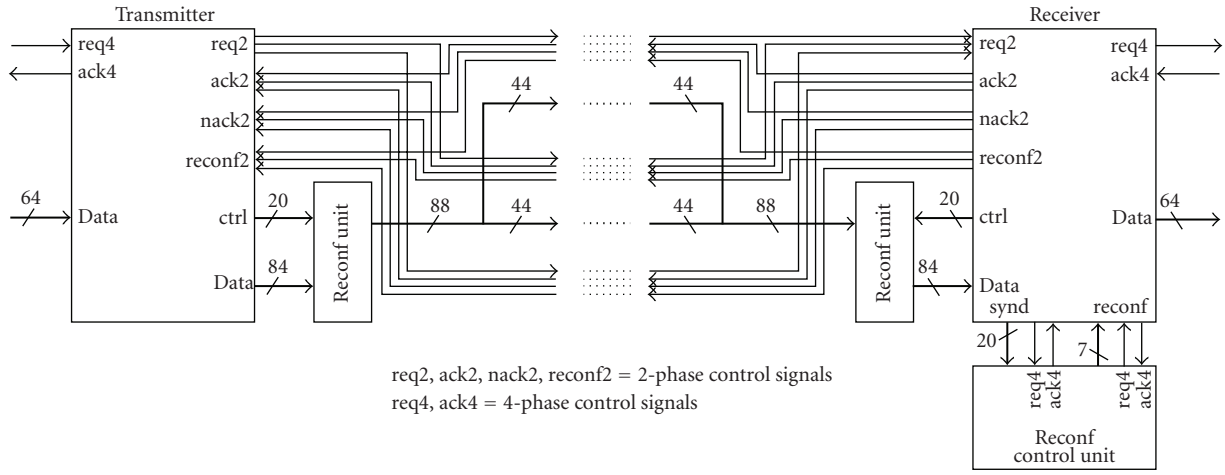


Figure 1: Simplified structure of the proposed link with spare wires and reconfiguration circuits.

globally-asynchronous locally-synchronous (GALS) design approach, where communication between processing blocks is done asynchronously [5–7]. Therefore, we base our link on self-timed design principles [8–10].

When designing a fault tolerant on-chip link for a deep submicron chip, one should first consider the possible fault scenarios. The approach must be capable of tolerating multiple bidirectional errors as well as burst errors [2, 11]. For detecting multiple bidirectional errors the Hamming code is widely used [11–13]. The standard version of it can detect two single errors and with one additional check bit the error detection ability can be extended to three. Cyclic codes can be used to detect burst errors [11]. Another approach for handling burst errors is interleaving [14].

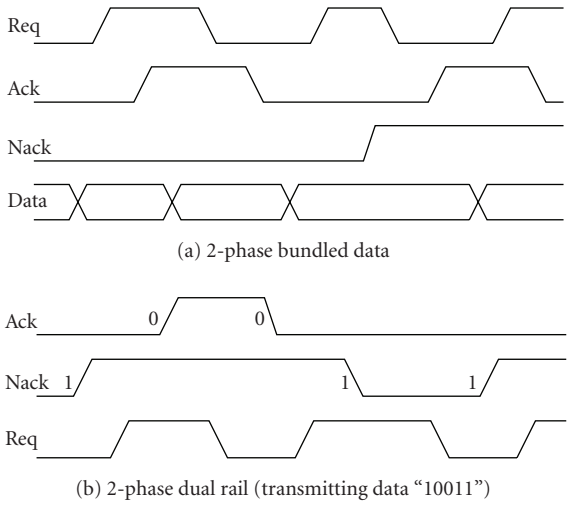
The faults can be categorized into three classes: permanent, intermittent, and temporary or transient faults [1]. Most of the failures (80%) are caused by transient faults [4]. The other way around, up to one fifth of all the failures are originating from permanent or intermittent faults. Thus, the fault tolerance approach must contain elements to not only tolerate the temporary errors but also the ones of more permanent nature. Most of the research has been concentrating on tolerating transient errors. The two methods for this purpose are forward error control (FEC) and automatic repeat query (ARQ) [15], the latter of which is found to be more energy efficient [11]. In FEC the errors are corrected at the receiver based on the information of the check bits while in ARQ the check bits are used to detect errors and a retransmission is requested when necessary. The ARQ will not work in the presence of permanent errors and FEC with commonly used Hamming coding loses its effectiveness already in the presence of a single permanent fault.

Fault tolerance methods besides coding include for instance triple modular redundancy (TMR) and the use of spare components together with error detection [16]. The use of spare wires for NoC interconnects has been presented in [17], where the focus, however, has been on improving yield and no method for fault detection nor reconfiguration protocol is presented.

3. LINK STRUCTURE

The idea in this work is to combine two different fault tolerance methods to achieve a system that is efficient in tolerating transient, intermittent and permanent errors. For transient faults we use Hamming coding and interleaving to detect faults and ARQ as the recovery method motivated by its energy efficiency as reported in [11]. For tolerating permanent and intermittent errors we introduce two methods, one that uses hardware redundancy and the other based on time redundancy. In the first approach spare wires are introduced together with reconfiguration circuits. In the second approach, the data is split into two transmissions and in both of them the transmitted data is doubled. In the receiver the fault-free copy is chosen and the data of the two transmissions are again combined into a whole word. In both structures, the presence of a permanent fault is detected using the same Hamming code as for transient faults. A number of previous error syndromes are stored and if they equal, it indicates a permanent error. The exact error location can be determined by decoding the syndrome. A similar method is used to switch back to the normal operation mode in the split transmission approach. If all the syndromes are zero for the past transmissions, the error has probably been intermittent and change back to the normal mode can be carried out.

The spare wire approach aims at providing unchanged performance in the presence of errors. However, the reconfiguration procedure is allowed to take some time since it is a rare occasion. The split transmission approach on the other hand has a significant impact on the performance and its use is motivated by considering a typical NoC system. An erroneous link could be bypassed through neighboring routers, but this would increase traffic in other links and may result in congestion. Nevertheless, the total latency of links



(a) 2-phase bundled data

(b) 2-phase dual rail (transmitting data "10011")

Figure 2: Self-timed signaling.

and routers on a bypassing route set an objective to the split transmission link design.

The width of the target link is set to 64 bits, which is split into four identical parts. Every part is encoded with Hamming (21, 16) code and the parts are interleaved. In other words, if we name the inputs as $i_{0,\ldots,63}$ and the check bits as $c_{0,\ldots,19}$ the coding proceeds as follows. The check bits c_0, c_4, c_8, c_{12} and c_{16} are calculated from inputs $i_0, i_4, i_8, \ldots, i_{60}$, the check bits c_1, c_5, c_9, c_{13} and c_{17} are calculated from inputs $i_1, i_5, i_9, \ldots, i_{61}$, and the other two interleaving sections correspondingly. Consequently, the system is capable of detecting error bursts affecting up to 8 adjacent wires and at least two simultaneous single faults extending to 8 simultaneous single faults if only two of them affect the same interleaving section.

In the spare wire approach, four spare wires are added to the system, one for each interleaving section. This gives the system tolerance for permanent error bursts affecting up to 4 wires and maximum of 4 single faults if they affect separate interleaving sections. Thus, the total number of wires is 88, from which 64 are data, 20 check bits and 4 spares.

In the split transmission approach the data is split into two parts, two interleaving sections to each. The data in both parts is doubled, preserving the interleaving. Therefore, the error detection capability is 4-bit wide error bursts and two single errors. The minimum tolerance against permanent errors is determined by the wires located physically in the middle of the link, where the two doubled parts have the minimum physical distance to each others. Since there are four control signals between the parts (see end of the section), the minimum permanent error burst tolerance is 6 bits extending up to 36 bits depending on which part of the link the errors affect. The minimum permanent single error tolerance is one, but a system having even 36 single errors works, if the errors occur only in two different interleaving sections.

The timing of the data transfer between transmitter and receiver is realized using two-phase asynchronous bundled data signaling [10, 18], illustrated in Figure 2(a), while internally the transmitter and receiver use 4-phase signaling [10, 19]. Two-phase signaling is chosen in order to minimize the control wire switching activity and the handshaking delay. This is since in the 4-phase protocol four transitions on handshake lines, two on both request and acknowledge wires, are required, while in the 2-phase protocol only two transitions, one on both request and acknowledge wires, are required. Hence, from this perspective 2-phase signaling protocol is a more attractive choice for NoC interconnects, which could possibly have significant physical wire lengths with high capacitive and resistive properties [20] causing considerable signal delays.

Therefore, in addition to the data wires we need request (*req*) and acknowledgment (*ack*) signals to implement the 2-phase signaling protocol between the transmitter and receiver. For signaling the incorrectness of a data transfer from the receiver to transmitter we use a negative acknowledgment (*nack*) signal instead of *ack*. This makes the backward signaling delay-insensitive since there is no need for making timing assumptions. Finally, we need also a signal for indicating the reconfiguration (*reconf*). The actual reconfiguration data in the spare wire approach can be sent from receiver to transmitter serially by using self-timed dual-rail protocol presented in Figure 2(b) [10], where *ack*/*nack* signals are used for data (0/1) and *req* for acknowledgement. This way also the reconfiguration data exchange is delay-insensitive and no additional signals are needed.

The timing signals are crucial for the correct operation of the link and therefore, they have to be protected against errors. For this purpose we use triple modular redundancy (TMR) as proposed in [21]. Furthermore, the three instances of each control signal are physically dispersed to maintain the tolerance against burst errors. The proposed link structure with spare wires and reconfiguration circuits is presented in Figure 1.

4. REALIZATIONS

The links were designed using the *Haste* design language and *timeless design environment (TiDE)* toolset for asynchronous design by Handshake Solutions [22]. *Haste* has support only for 4-phase bundled data signaling so converter circuits were created to transform signaling to 2-phase and vice versa. The converters were designed using standard components and translated to structural VHDL as were also the majority voters needed for TMR.

To find out the area overhead, performance penalties and the impact on power consumption the introduced fault tolerance causes, also a link without any fault tolerance and a design with ARQ but no reconfiguration circuitry were realized. The buffering capacity of these reference designs was set to the same as in the reconfiguration cases.

4.1. Link with spare wires

The structure of the link with spare wires is presented in Figure 3. The system is pipelined to increase the throughput. The transmitter contains first a latch stage (*L1*), which is

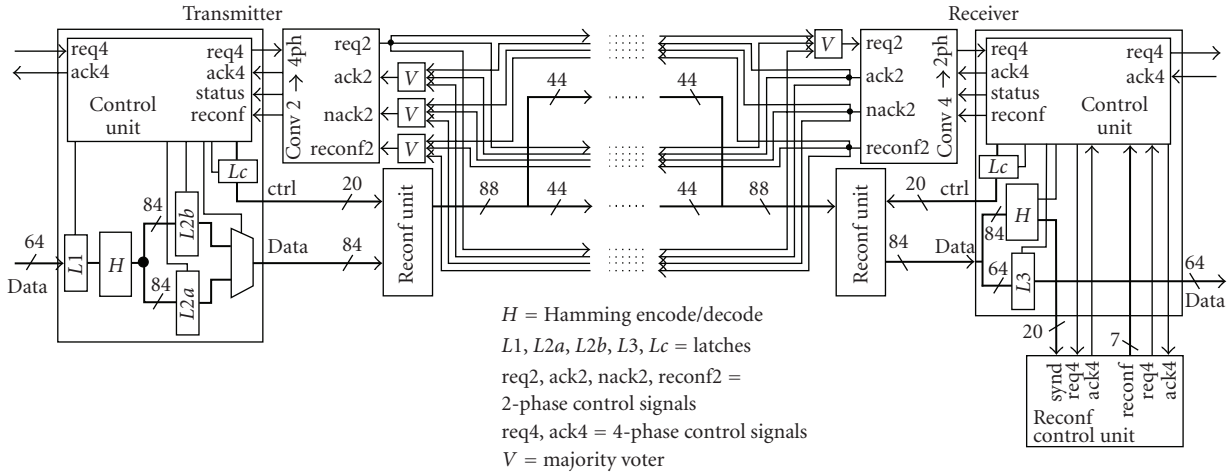


Figure 3: Structure of the designed link with spare wires.

followed by the Hamming encoder circuitry (H). After the encoding the data with check bits is stored in one of the two parallel output latches (*L2a* and *L2b*). These latches are used so that while one is connected to the output channel the next data word can be stored to the other. When *ack* is received to indicate a correct transmission, the output latch can be rapidly changed and a new transmission can begin. In the case of negative acknowledgment (*nack*) the same latch stays connected to the output channel and a retransmission is carried out (see also Figure 2(a)).

In the receiver the error syndrome for incoming data is calculated and the data word is stored to a latch (*L3*). If the syndrome equals zero, *ack* is sent and the data from the latch is forwarded to the receiver output. In the case of a nonzero syndrome, *nack* is sent and the syndrome is passed to the reconfiguration control unit, where it is stored into a 3-place ring buffer, so that the last three nonzero syndromes are found in this buffer. The structure and operation of the reconfiguration control unit is explained in Section 4.3.

When the receiver gets a request together with the reconfiguration vector from the reconfiguration control unit, it switches to the reconfiguration mode. An arbiter is used to guarantee that the mode change cannot occur in the middle of a receive operation. The reconfiguration data exchange between the receiver and the transmitter is illustrated in Figure 4. The receiver sends *reconf* to indicate the mode change to the transmitter. The transmitter acknowledges this through the *req* line. Next, the reconfiguration information (reconfiguration vector, 7 bits) is transferred bitwise using *ack* and *nack* lines and every bit is acknowledged with the *req* line (see Figure 2(b)). After the acknowledgement of the last bit, the receiver sends *reconf* to indicate the mode change back to normal and the transmitter acknowledges it with *req*. Finally, receiver sends *nack*, so that the transmitter sends again the data it was sending when the mode change took place.

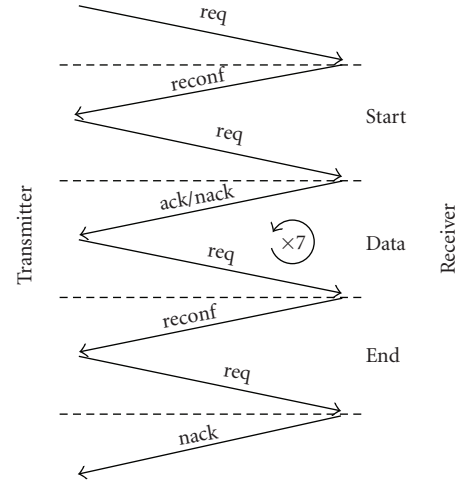


Figure 4: The protocol for sending the reconfiguration information from the receiver to transmitter.

After the transmission of the control word both the receiver and transmitter store the error location into the control register of the correct interleaving section according to the reconfiguration vector. The reconfiguration units were created using structural VHDL. The transmitter side unit is illustrated in Figure 5. Tristate buffers are used to drive the link wires. From the control registers the exact error location is decoded using a 5-to-21 decoder. Based on that value and the fact that no reconfiguration is done above the error location and that all wires below the error location need to be reconfigured, the control signals for the tristate buffers can be solved. By using tristate buffers instead of multiplexers the corrupted wire can be completely isolated from the driving circuit and so, for example, possible short-cuts to power lines do not cause any power leakage. Furthermore, tristate buffers can be easily included in the driving buffers, that are

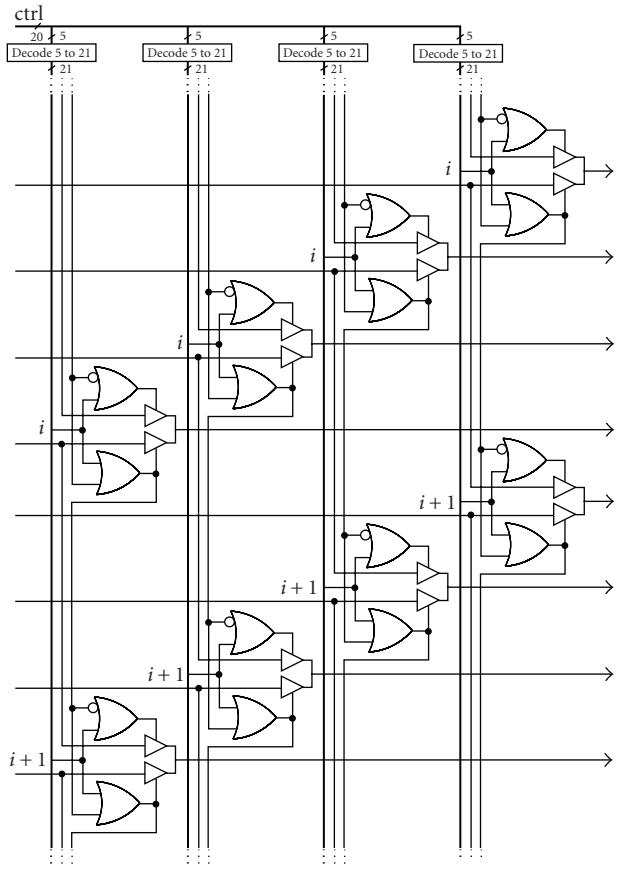


Figure 5: The structure of the reconfiguration unit at the transmitter side.

needed to drive the capacitive on-chip wires. At the receiver side the control is realized similarly with the exception that now multiplexers are used to select the correct wire for each input.

4.2. Link with split transmission

The structure of the link with split transmission is presented in Figures 6 and 7. The system has many similar parts to the one with spare wires. The transmitter differs after the multiplexer *M1*. In the normal mode the output of *M1* is connected to the output of the transmitter via the multiplexer *M3*. In the split mode, indicating that there is a faulty wire in the link, the output of the transmitter is obtained from the multiplexer *M2* (via *M3*). In the Hamming encoder circuitry the data is split into four interleaving sections (every fourth wire per section) and the five check bits are calculated for each section separately and interleaved. In selector *Sel1* either first two or last two sections are chosen and doubled. Splitting the data according to the interleaving sections removes the need of two separate encoding and decoding units. The multiplexer *M2* chooses between the first and second split transmission parts. The part is changed when *ack* is received, in case of *nack* it stays the same and a retransmission is carried out.

In the receiver the error syndrome for incoming data is calculated and the data word is stored into a latch (*L3*) similarly as in the structure with spare wires. The data and check bits are in different parts of the data word depending on the operation mode. This selection is not shown in Figure 6. If the transmission is correct, *ack* is sent to the transmitter and the data from the latch is forwarded to the receiver output via the multiplexer *M5* in case of the normal mode and to the selector *Sel2* in the split mode. The correctness of the transmission is indicated in the normal mode when the syndrome equals zero and in the split mode when either copy of an interleaving section gives a zero-syndrome for both transmitted sections. This means that if the transmitted interleaving sections are marked as 1 and 2 and doubling creates instances *a* and *b*, the transmission is correct if one of the following combinations have zero-syndromes: 1*a* and 2*a*, 1*a* and 2*b*, 1*b* and 2*a*, or 1*b* and 2*b*. In the case of an incorrect transmission, *nack* is sent and the syndrome is passed to the reconfiguration control unit. The syndrome is passed to the reconfiguration control unit also during correct transmissions in the split mode. Based on the syndromes the reconfiguration control unit detects permanent errors and controls the return to the normal mode if the errors have vanished (e.g., they were intermittent). The structure of the reconfiguration control unit and its operation is explained in Section 4.3.

In the split mode, the selector *Sel2* separates the interleaved sections and using the multiplexers *M4a* and *M4b* the correct instances of both transmitted sections are chosen. The control for these multiplexers is obtained from the syndromes calculated by the decoding unit. The first split transmission part is stored to the latch *L4* and when the second part arrives they are interleaved (*Ilv*) and passed to the output via *M5*.

If the receiver is in the normal mode and an error is detected (signal *error* from the reconfiguration control unit) it switches to the split mode. The mode change takes place in the same way in the split mode if the signal *zero* is detected. An arbiter is used to guarantee that the mode change cannot occur in the middle of a receive operation or before a split transmission is completed. The receiver sends *reconf* to indicate the mode change to the transmitter. The transmitter acknowledges this through the *req* line, after which the receiver sends *nack*, in order to get the transmitter to resend the data it was sending when the mode change took place.

4.3. Reconfiguration control

The reconfiguration control unit for the spare wire design is depicted in Figure 8(a) and the unit for the split transmission design in Figure 8(b). Both reconfiguration units have a 3-place syndrome buffer which is divided into four 5-bit segments, one segment for each interleaving section (*LXa*, *LXb* and *LXc*, where *X* is the number of the segment).

In the reconfiguration control unit for the spare wire design, the values of the three syndromes are compared individually in all the four segments and if they equal and are

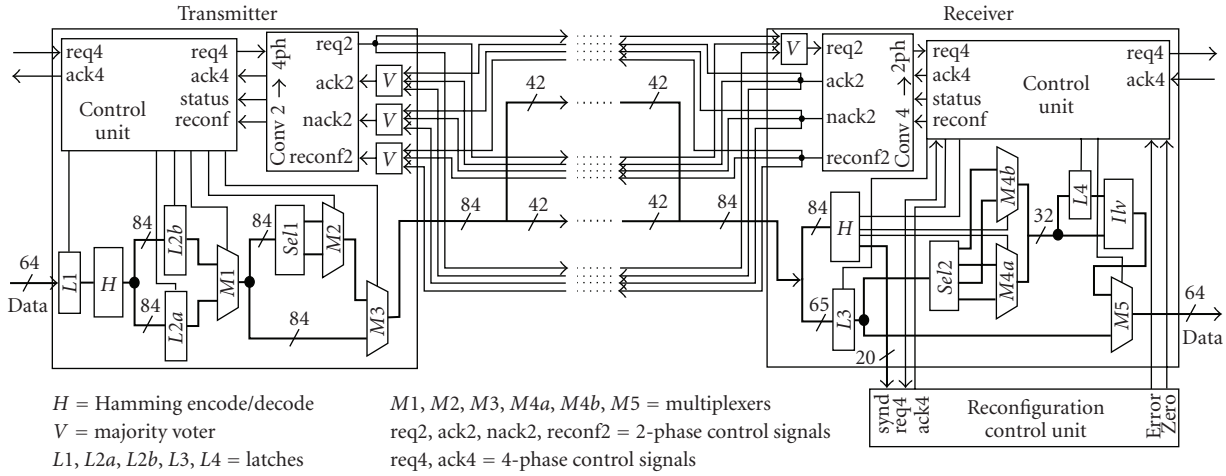


Figure 6: Structure of the designed link with split transmission. Select and interleaver units (*Sel1*, *Sel2*, and *Ilv*) are presented in Figure 7.

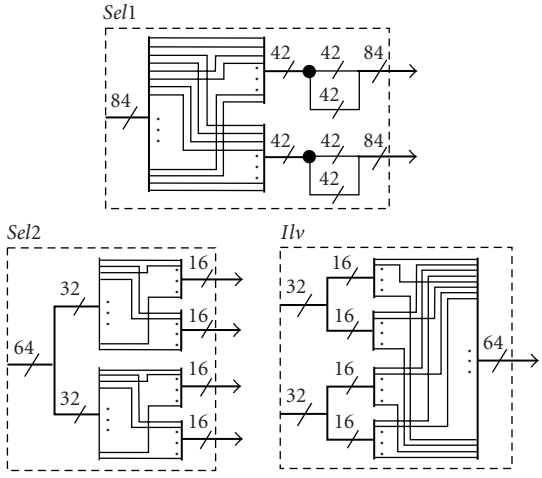


Figure 7: Select and interleaver units.

nonzero, a signal indicating a permanent error in that interleaving section is asserted. Based on these signals the reconfiguration process is commenced. Since there are only one spare wire for each interleaving section, a reconfiguration status flag (latch *Lf*) indicates if the spare has already been used and therefore the reconfiguration cannot take place. Also arbitration between the four error signals is needed to handle the situation, where a permanent error is detected in many sections at the same time (e.g., a burst error).

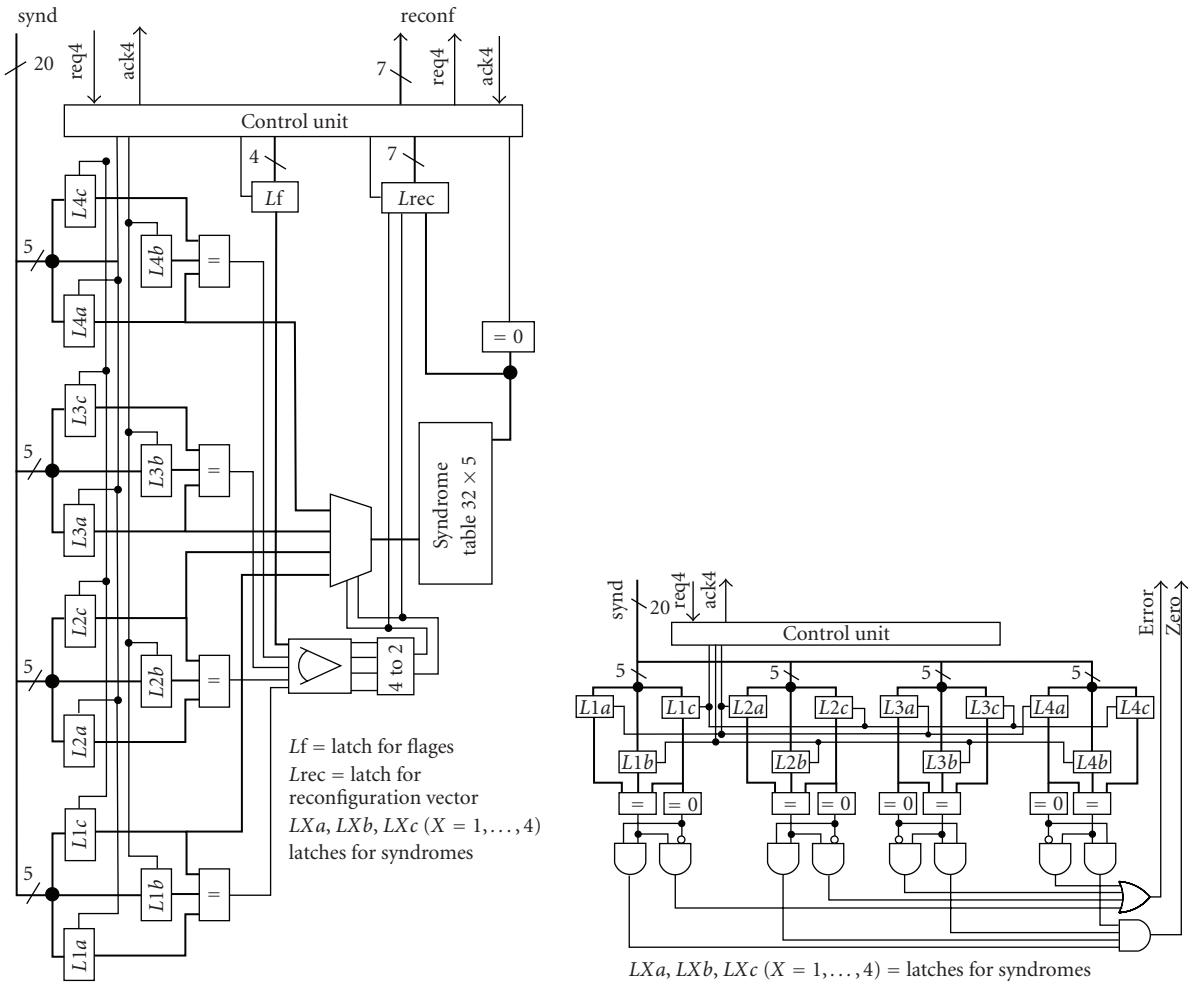
The reconfiguration procedure creates a reconfiguration control vector (latch *Lrec*) to be forwarded to the receiver. This contains 2-bit information indicating the correct interleaving section and 5 bits for pointing out the location of the corrupted wire. The location is fetched from a ROM memory according to the syndrome. In this memory, there is a place for each of the 32 combinations of the 5-bit syndrome. Only 21 of these indicate a single error. The others are set to zero and the circuit checks if the fetched location equals zero. In this case no reconfiguration is done, because the exact error location cannot be solved.

The structure of the reconfiguration control unit for the split transmission design is simpler than the one explained above. An error is indicated via the signal *error* if in one of the segments the three syndromes are equal and nonzero. The signal *zero* on the other hand is asserted if all the syndromes in all the segments are zero.

4.4. Protocol converters

Protocol converters from 2-phase to 4-phase and vice versa in both spare wire and split transmission implementations are shown in Figure 9 [6]. As mentioned earlier, 4-phase signaling [10, 19] is used internally in the transmitter and receiver and in communication between a phase converter and the transmitter/receiver. However, communication between converters, actual communication between the transmitter and receiver, is realized by using a 2-phase handshake protocol to minimize the number of transitions in the link control wires [10, 18]. Hence, protocol converters are required at the boundary of the transmitter and receiver as shown in Figures 3 and 6. Both spare wire and split transmission implementations can utilize the same 2-to-4 and 4-to-2 phase converters.

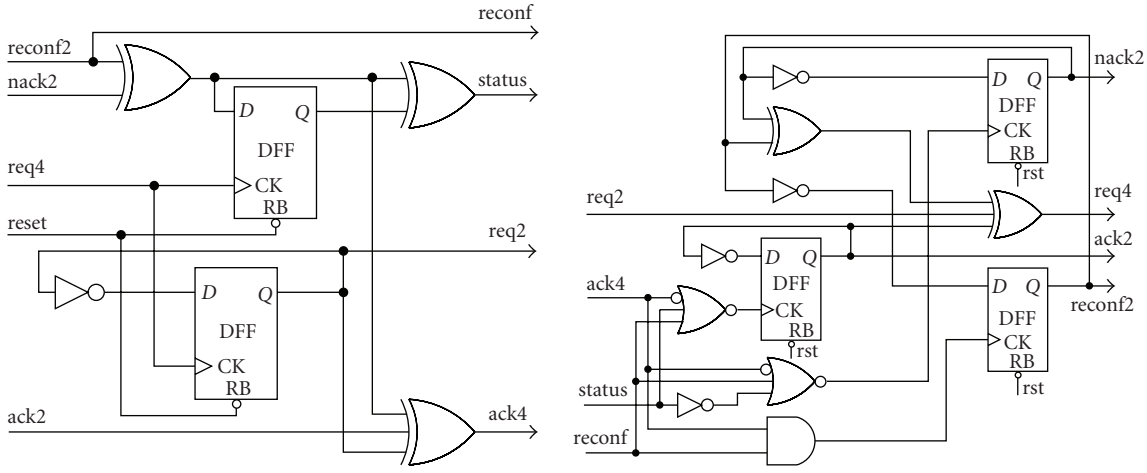
The 4-to-2 phase converter presented in Figure 9(a) is activated by the signal *req4* from the transmitter indicating that a new transaction is going to be commenced. The signal *req4* is connected to the clock input of both flipflops which triggers events in *req2* and *ack4*. The signal *req2* is the actual 2-phase request signal to the receiver indicating availability of new data while the 4-phase signal *ack4* indicates to the transmitter that the data transaction has been started. Immediately after the transmitter has received *ack4* it sets *req4* back to its original level. Observe that the direction of the event in the 2-phase *req2* signal, the actual logic signal level, is not important. Only the change of the signal level is monitored. After the receiver has captured the data it produced an event in the *ack2* signal, which causes *ack4* to go back to its initial



(a) Spare wire design

(b) Split transmission design

Figure 8: Structures of the reconfiguration control units.



(a) 4-to-2 phase converter

(b) 2-to-4 phase converter

Figure 9: Protocol converters.

Table 1: Comparison of the link designs, with wire delay of 500 picoseconds.

Design	No FT	ARQ	Spare	Split
Latency (empty) (ns)	2.67	7.71	8.48	9.65
Latency (full) (ns)	4.93	14.98	17.82	20.31
Throughput (MWord/s)	492.6	204.5	180.5	152.7
Area ($(\mu m)^2$)	6675	12 386	25 377	21 602
Transmitter	4975	8348	12 986	10 518
Receiver	1700	4038	12 391	11 083

Table 2: The burst error probabilities used in simulations.

	Deviation	p_x	Burst length	Explanation
p_0	$\pm[0, \sigma)$	68.3%	1	Single error
p_1	$\pm[\sigma, 2\sigma)$	27.2%	3	+1 adjacent wire in both directions
p_2	$\pm[2\sigma, 3\sigma)$	4.3%	5	+2 adjacent wires in both directions
p_3	$\pm[3\sigma, \infty)$	0.3%	7	+3 adjacent wires in both directions

value informing the transmitter that a new data transaction can be started.

In the case that a retransmission is required, the receiver produces a negative acknowledgment by asserting the signal *nack2*. This changes the state of *status* signal, indicating to the transmitter that the same data should be resent. Events in the signals *ack2* and *nack2* are mutually exclusive, that is, there can be an event only in one of these signals as shown in Figure 2. The signal *reconf2* indicates that the system enters the reconfiguration mode and at the same time causes *ack4* to return to its initial state.

At the receiver side, the 2-to-4 phase converter shown in Figure 9(b), the operation is activated by an event in *req2*, indicating arrival of data. This initiates 4-phase handshaking between the converter and receiver via *req4*. After the receiver has captured data it sets *ack4*, which is connected to the clock inputs of the flipflops through logic gates. At this point, there are three possibilities: *ack2* is sent in the case that data has been properly received, *nack2* is sent if retransmission is required (indicated by *status*), and if reconfiguration is going to take place *reconf2* is sent (indicated by the *reconf* signal) to the transmitter. Regardless of the information sent back to the transmitter, *req4* is initialized back to its initial value which eventually resets *ack4* and hence completes the 4-phase handshake cycle.

5. RESULTS

The designs were mapped to a 130 nm technology, synthesized and simulated using specific VHDL testbenches to find out the best and worst case throughputs and latencies. The interconnect delay was set to 500 picoseconds, which was estimated to model the delay between two routers in an NoC structure including the drivers and possible repeaters. The same wire model was used in all simulations.

The throughputs, latencies and circuit areas are presented in Table 1, where *no FT* means the reference circuit without any fault tolerance structure, *ARQ* the one with ARQ but no reconfiguration properties, *spare* the presented structure with spare wires, and *split* is the structure with the split transmission properties. Latency (full) means the situation, where the buffer capacity of the link is totally occupied when the transmission begins and correspondingly latency (empty) means that there is no data in the buffers. The values listed in Table 1 for the split transmission structure are measured in the normal operation mode. The split transmission latency (full) was measured to be 59.34 nanoseconds and throughput 61.2 Mword/s. The reconfiguration procedure in spare wire design was measured to take in total 29.7 nanoseconds.

6. FAULT TOLERANCE ANALYSIS

In this section the fault tolerance abilities of the presented link structures are analyzed and compared against the reference designs. The used fault model is extended from the one presented in [23] to consider also burst errors as well as intermittent and permanent errors.

According to the model presented in [23] the probability of error ϵ is given by

$$\epsilon = Q\left(\frac{V_{dd}}{2\sigma_N}\right), \tag{1}$$

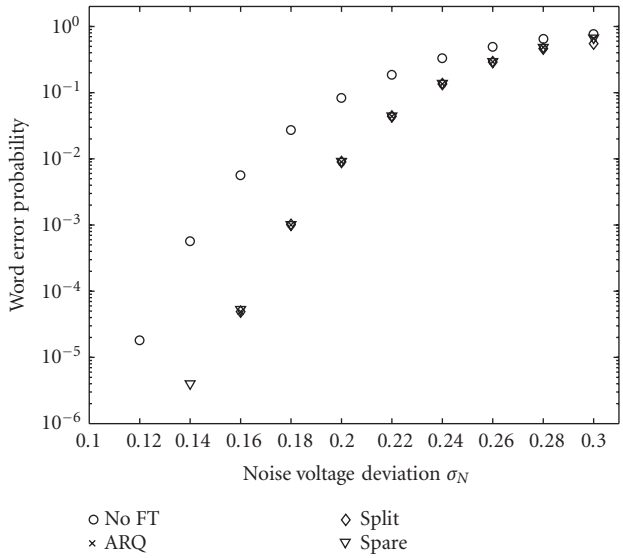
where V_{dd} is the supply voltage, σ_N the deviation of the noise voltage, which is presumed to have normal distribution, and $Q(x)$ is the Gaussian pulse defined as

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-y^2/2} dy. \tag{2}$$

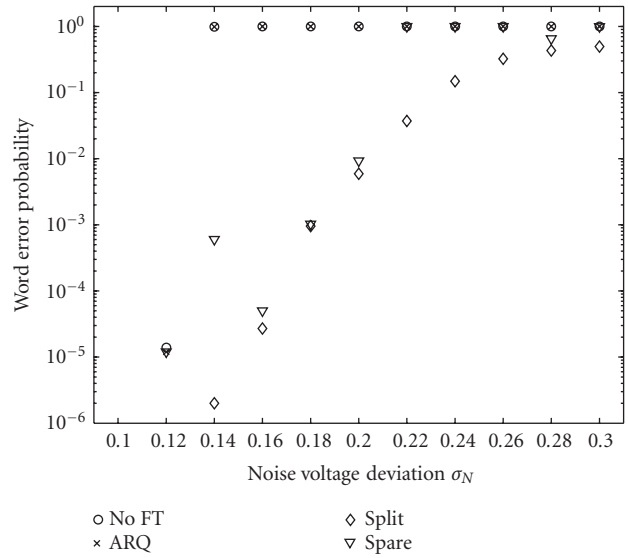
The model assumes that a fault in a single wire is independent of the others. We extend the model by assuming that a fault affects its neighbouring wires in both directions with a certain probability p_1, two neighboring wires with probability p_2 and so on. The probabilities used in the simulations are obtained from the normal distribution in a way that the probability p_0 for a single error is the area under the normal distribution curve for deviations smaller than $\pm\sigma$, p_1 for deviations $\pm[\sigma, 2\sigma)$, and so forth. The probabilities used in the analysis are presented in Table 2.

We add to the model also intermittent and permanent errors. According to [4] 80% of all the errors are transients. We add intermittent and permanent errors respecting this relation and taking the view that half of the added faults are intermittent lasting 10 transmissions and the other half permanent.

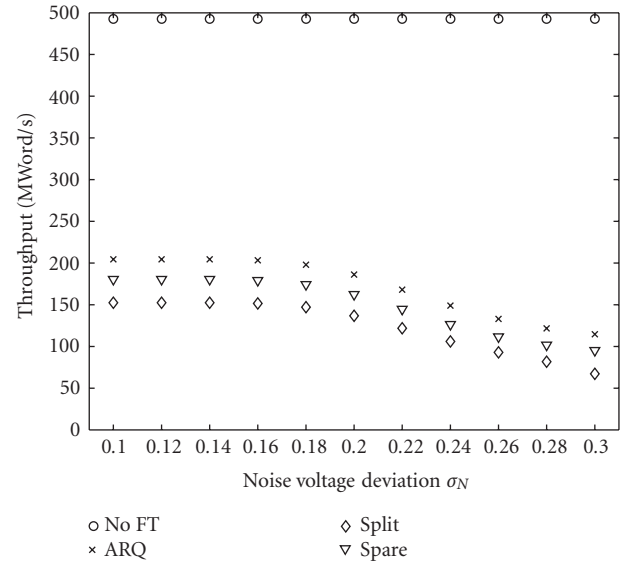
A number of test sets were generated for different values of σ_N while the V_{dd} was held at the constant value 1.2 V. Error sets both with and without intermittent and permanent errors were created. A test data set of one million random data words was generated and simulations were run on the two presented link structures as well as on the two reference structures for the different error sets. The simulation results are shown in Figure 10.



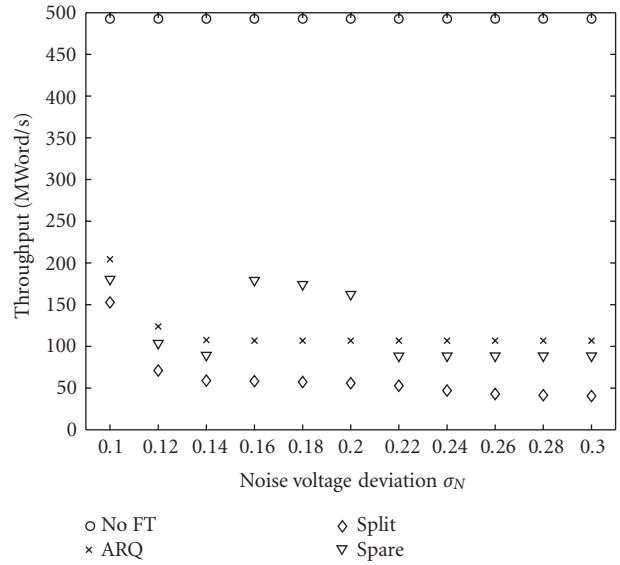
(a) Error probability in the case of only transient errors



(b) Error probability in the case of transient, intermittent, and permanent errors



(c) Throughput in the case of only transient errors



(d) Throughput in the case of transient, intermittent, and permanent errors

Figure 10: The performance of different structures in the presence of different error scenarios.

The word error probability as a function of the noise voltage deviation is shown in Figure 10(a) for the error model without intermittent and transient errors. The fault tolerance properties of the two presented structures match with the reference circuit with ARQ but no reconfiguration. This is quite obvious since the fault tolerance method they use is the same. Quite obvious is also the fact, that the structure without any fault tolerance properties results in a higher word error probability rate as the others.

The corresponding throughputs are shown in Figure 10(c). We see that the throughput decreases as the noise deviation increases for all the other structures except for the one without fault tolerance properties, which have constant throughput regardless of the amount of faults. The ARQ is realized in a way that in the case of a detected fault, the word is retransmitted once, but no more. If there was no limit for the number of retransmissions the circuit would end up in a livelock. The effect of most transient faults, for example, intersymbol interference, radiation induced pulses, crosstalk, and so forth, ceases by just waiting before sampling the values on the line. For this reason the number of retransmissions is limited to one. Unfortunately, our error model does

Table 3: Energy consumption analysis of the link designs.

(a) Energy per transmission for $\sigma = 0.10$

Design	No FT	ARQ	Spare	Split
Transmitter (pJ)	26.28	28.31	29.50	30.94
Receiver (pJ)	7.12	12.42	14.46	17.75
Total (pJ)	33.41	40.74	43.96	48.69
Compared to ARQ			+7.9%	+19.5%

(b) Energy per transmission for $\sigma = 0.20$

Design	No FT	ARQ	Spare	Split
Transmitter (pJ)	26.28	28.98	29.57	34.25
Receiver (pJ)	7.80	21.24	16.30	46.37
Total (pJ)	34.08	50.22	45.86	80.62
Compared to ARQ			−8.7%	+60.5%

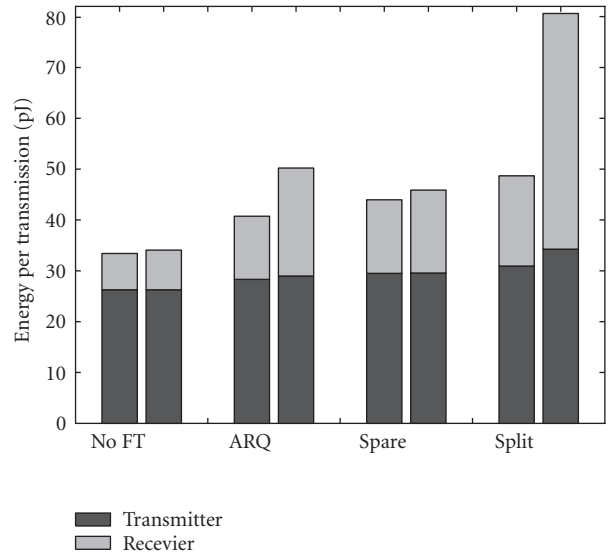


Figure 11: Energy consumption analysis of the link designs.

not take into account this phenomenon but instead all transfers are presumed to be independent of each others. This gives pessimistic values for the fault tolerance of the simulated structures. Nevertheless, they can be compared with the used model since the situation is the same for all of them. The impact of the limited number of retransmissions is seen in the saturation of the throughputs as the noise voltage deviation is increased. The throughput-based ranking order of the structures corresponds to the values presented in Table 1 and it remains the same for all the simulated sets.

The simulation results for the error sets with also intermittent and permanent errors are shown in Figures 10(b) and 10(d). The reference structures lose their operability rapidly as the noise voltage deviation is increased. As expected, they do not work in the presence of permanent faults. The presented online reconfigurable structures maintain their operability although there is more variance in the results. The variance is originated from the randomness of the error locations and the properties of the structures to guarantee tolerance against some limited set of permanent faults while a larger set is tolerated with some conditions. From the two presented structures the one with split transmission shows higher fault tolerance.

The throughputs presented in Figure 10(d) show similar changes as in the simulations without intermittent and permanent faults. For the reference structure without fault tolerance, the throughput is constant and for the reference structure with ARQ the throughput decreases, now more rapidly than without intermittent and permanent faults, and saturates because of the limited number of retransmissions. The spare wire approach has exceptions to this trend. When the reconfiguration has been successfully carried out, the permanently faulty wires have been replaced by spares, the throughput equals the fault-free situation and the word error probability remains low. On the other hand, if the reconfiguration is not successful the fault tolerance is almost as low as for the reference circuit with ARQ, and the throughput has similar values as in the case without intermittent and permanent faults for high noise voltage deviation. The split transmission structure shows a similar curve to the one in the case without intermittent and permanent faults, but with intermittent and permanent faults it saturates to a lower throughput value. This is because the throughput in the split mode is lower and in the presence of a permanent fault, the circuit remains in the split transmission mode all the time.

7. POWER ANALYSIS

The power consumption of the systems was analyzed by using Synopsys Primepower. Simulations were carried out for an input set of ten thousand random words for two different error sets containing transient, intermittent, and permanent errors. The error sets with noise voltage deviations $\sigma = 0.10$ and $\sigma = 0.20$ were chosen. The first one demonstrates a situation of low error probability whereas the second contains a lot of errors. By comparing these two analyses the power efficiency in the presence of errors can be seen. The simulation results are presented in Table 3 and in Figure 11, where *no FT* means the reference circuit without any fault tolerance structures, *ARQ* the one with ARQ but no reconfiguration properties, *spare* the presented structure with spare wires, and *split* is the structure with the split transmission properties. The average power consumption is translated to energy consumption per transmitted word by taking into consideration the throughputs of the different systems and averaging over the random input set.

The energy consumption values do not include the energy taken by the drivers driving the link wires. These values were not included since the main target of the analysis is to see the impact of the added reconfiguration properties in comparison to the reference circuit with ARQ but no reconfiguration. These values are comparable since in all of

these systems the number of active wires is the same or the difference in the wire count can be ignored. In the spare wire approach, the added spare wires do not consume any power and therefore, the number of active data wires is 84 as is the case in the split and reference designs. In the systems with reconfiguration properties, there is one additional tripled control signal (*reconf2*), but it is switched extremely rarely. The power analysis of the majority voters indicates that the power consumption of the majority voters connected to the *reconf2* signals is less than one thousandth of the corresponding values of the ones connected to the *ack*2 signals. The relative differences of the energy consumptions of the spare and split approaches as compared to the reference with only ARQ are presented in Table 3 for the both simulated noise voltage deviations.

The results indicate that the addition of reconfiguration properties results in a small increase in power consumption. For the spare wire approach the increase is less than 8% in the case of few errors, as for the split transmission design the increase is almost 20%. When the noise voltage deviation is increased the values show dramatical but nevertheless expectable results. Now the spare wire approach gives the best power consumption values, almost 9% less than for the reference design. The reasons for this were already discussed in the previous section. The power consumption of the spare wire approach is almost the same as without any permanent or intermittent errors when the reconfiguration is successful. The split transmission design uses a lot more energy than the others, which is rather obvious since it takes many transmission for all the words.

From the power consumption simulations it can be concluded that the spare wire approach is more power efficient than the split transmission approach, and that the addition of spare wires and the needed reconfiguration properties does not cause any considerable changes in the power consumption.

On the other hand, the results also indicate that the addition of fault tolerance properties in general results in a higher power consumption. This observation is further strengthened by the fact that the system without fault tolerance properties contains less wires than the others.

8. DISCUSSION

When comparing the designs with the reconfiguration properties to the ARQ design without them, it can be noticed that there is a performance penalty. In the spare wire design, the latency increases 10%/19% (empty/full) and the throughput decreases 12%, while in the split transmission design the latency increase is 25%/36% and the throughput decrease 25%. Thus, the spare wire approach has a smaller impact on the performance. On the other hand, the area overhead is larger in the spare wire approach. It uses 105% more area than the ARQ design without reconfiguration. The corresponding value for the split transmission is 75%. The area increase is mainly due to the reconfiguration control unit in the receiver. Moreover, the reconfiguration logic in both the transmitter and receiver consume area.

The split transmission performance should be compared to the situation, where the erroneous link is bypassed through neighbouring links and routers in an NoC. The split transmission latency (full) is 14.4 nanoseconds larger than three times the latency of the reference circuit without split transmission properties. The three link latencies corresponds to the shortest route to bypass a link in a mesh-shaped NoC. In addition to the link latencies, the latencies of the two routers contribute to the total delay of the bypass route. The latency of these two routers is comparable to the 14.4 nanoseconds. Therefore, the latency of the split transmission is approximately the same as that of bypassing the erroneous link. The essential difference is that bypassing increases traffic in neighbouring links and may cause serious consequences in the form of congestion. In the worst case this affects the operation of the whole NoC.

The area overhead should be considered in the NoC context. Let us consider a simple mesh structure, where one router is assigned for each resource and the size of a resource is two times two millimeters [24]. There are three bidirectional links per each resource, thus, in total six transmitter/receiver pairs per resource. The area overhead caused by the introduced fault tolerance is only 2.2%–2.8% compared to the situation without any fault tolerance. The router area was not included in this calculation but it further decreases the relative area overhead.

If the link is connected, for example, between NoC routers, the buffering capacity of the link can be taken into account, and so the buffering capacity of the routers can be decreased by the same amount. This obviously reduces the area overhead. In the same way the tristate buffers of the transmitter reconfiguration unit in the spare wire approach can be a part of the link drivers as was already discussed in Section 4.1.

The simulations were carried out using a 130 nm technology. As the dimensions have been scaled below 100 nm the delay of wires has not decreased as is the case with the delay of logic. The gap between the wire and logic delays is expected to increase as scaling deeper into nano regime [4]. Based on this scenario and because the bottleneck at the moment are the receiver and transmitter units, the performance of the presented link structures should increase as the dimensions are scaled down.

The main parts were created using a modelling language and synthesis tools which use a rather conservative approach to the problem. Based on our previous experience we predict that the performance could be significantly enhanced by careful manual design although still using standard gate libraries [6].

In the presented designs the approach for transient fault tolerance was the use of error detection and ARQ with stop-and-wait strategy. The addition of reconfiguration to tackle intermittent and permanent errors does not force sticking at this approach. Quite as well forward error correction and in the spare wire approach also different ARQ strategies [15] could be used. One such strategy is go-back-N, where instead of waiting for an acknowledgment of correct transfer, the transmission proceeds with next word instanteously and in

the case of an error the system returns the necessary amount of words (N) backwards. Another strategy is selective repeat, where the transmission is continued as in go-back-N, but in the presence of an error only the erroneous word is asked to be retransmitted. In the presented analysis the focus has been on the comparison of the reconfigurable structures with the reference structure with ARQ but no reconfiguration. Since all these designs use the same ARQ strategy, the choice of it has not been of vital importance and hence, the simplest one has been used.

The reconfiguration control indicates a permanent error when the same error syndrome has been repeated three times. In the split transmission approach the reconfiguration could be advantageous also under other conditions. In Figure 10(a), we see a slightly smaller error probability for split transmission than for the other designs when $\sigma_N = 0.30$. In this case the circuit has entered the split transmission mode since the temporary error probability has been so large that the same error syndrome has been repeated three times. The lower throughput for this situation confirms the diagnosis. This gives an idea that the change to the split transmission mode might be advantageous if the error occurrence rate crosses some limit.

The tolerance against single permanent faults in the split transmission design is limited by the fixed location of the doubled data parts. If the location of the permanent fault was more exactly isolated, the placement of the doubled data could be controlled and a greatly improved tolerance against single faults could be achieved. The error location could be determined using the syndromes as is done in the spare wire design. This would probably increase the control logic and would also result in performance degradation. The presented approach has the assumption that if there are multiple errors, they most likely occur as bursts, which can be motivated by considering different manufacture defects.

The split transmission design has a property to return to the normal mode if the fault was intermittent. A similar property would be advantageous also to the spare wire design, where the reconfiguration at the moment is irreversible. The circuit is not able to tolerate permanent errors if there have been intermittent errors in the same interleaving section (but at a different wire) although the intermittent fault has vanished. The intermittent faults in practice repeat themselves in the same locations due to, for example, small anomalies in the manufactured chip, which turn into faults under certain environmental conditions. However, the used fault model considers all the intermittent faults independent of each others and therefore gives pessimistic values for the fault tolerance properties of the spare wire approach. The advanced detection of different fault scenarios will be a part of our future research.

The number of spare wires in the presented design was set to four and they were assigned one for each interleaving section. The fault tolerance properties could be enhanced by increasing the number of spare wires. It would also be advantageous to have the spare wires assigned more flexibly to different interleaving sections. One interleaving section may have many erroneous wires while some other has none. The drawback of the increased flexibility is the growth of complexity which probably leads to a performance decrease and area overhead.

In this work, the transmitter and receiver circuits were presumed error-free. When considering fault tolerance of future nanoscale systems, also these circuits should be taken into consideration from the fault tolerance perspective. We will address this issue as a part of our future work.

9. CONCLUSION

We proposed link structures that have properties for tolerating efficiently transient, intermittent, and permanent errors. The protection against transient errors was realized using Hamming coding and interleaving for error detection and ARQ as the recovery method. Two approaches were introduced to tackle the intermittent and permanent errors. Split transmission was an approach utilizing time redundancy while the other structure, which introduced spare wires, was a hardware redundancy approach. Communication in the links was based on asynchronous 2-phase signaling and the control signals for ARQ and reconfiguration were incorporated into these control signals. The control lines were protected using triple modular redundancy.

We presented designs for the different components of the links and proposed the needed reconfiguration control. The designs were implemented, simulated and compared against reference designs. The simulation results show that the performance decrease when comparing to a design with ARQ but no reconfiguration is larger for the split transmission design (latency 31%/throughput 25%) than for the spare wire design (15%/10%) while the area overhead is larger for the spare wire design (105%) as for the split transmission design (75%). The fault tolerance analysis using error models containing temporary, intermittent, and permanent faults that occur as both bursts and single errors, shows the effectiveness of the presented link structures. When there are no intermittent or permanent errors present, the structures perform the same way as the reference design with ARQ but no reconfiguration, and which clearly outperforms the reference design without any fault tolerance. When also intermittent and permanent errors are taken into account, the presented link structures show clearly better results than the reference designs. From the two presented reconfiguration structures, the split transmission design tolerates faults slightly better than the design with four spare wires. On the other hand, the spare wire approach turned out to be more power efficient than the split transmission design. Using the spare wire design the power consumption is approximately the same as with the reference design with ARQ but no reconfiguration. In the presence of an excessive amount of errors the spare wire approach turns out to be even more energy efficient than the reference design.

Our research shows that combining different fault tolerance methods a structure capable of tolerating all different types of errors is achievable. As is always the case with fault tolerance, this does not come for free. There is a clear area overhead, but on the other hand when comparing to the IP

block areas in the NoC context, the overhead is only a couple of percents.

ACKNOWLEDGMENTS

The authors would like to thank The Finnish Cultural Foundation, The Foundation of Technology, and Ulla Tuominen Foundation for their financial support. They also acknowledge the financial support from Academy of Finland.

REFERENCES

[1] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *IEEE Micro*, vol. 23, no. 4, pp. 14–19, 2003.

[2] International Technology Roadmap for Semiconductors, 2005, http://public.itrs.net/.

[3] T. Lehtonen, J. Plosila, and J. Isoaho, "On fault tolerance techniques towards nanoscale circuits and systems," Tech. Rep. 708, Turku Centre for Computer Science (TUCS), Turku, Finland, August 2005.

[4] G. De Micheli and L. Benini, *Networks on Chips*, Morgan Kaufmann Publishers, San Francisco, Calif, USA, 2006.

[5] D. M. Chapiro, *Globally-asynchronous locally-synchronous systems*, Ph.D. thesis, Stanford University, Stanford, Calif, USA, October 1984.

[6] P. Liljeberg, J. Plosila, and J. Isoaho, "Self-timed communication platform for implementing high-performance systems-on-chip," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 43–67, 2004.

[7] J. Muttersbach, T. Villiger, and W. Fichtner, "Practical design of globally-asynchronous locally-synchronous systems," in *Proceedings of the 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC '00)*, pp. 52–59, Eilat, Israel, April 2000.

[8] M. Renaudin, "Asynchronous circuits and systems: a promising design alternative," *Microelectronic Engineering*, vol. 54, no. 1-2, pp. 133–149, 2000.

[9] C. L. Seitz, "System timing," in *Introduction to VLSI Systems*, C. Mead and L. Conway, Eds., chapter 7, Addison-Wesley, Reading, Mass, USA, 1980.

[10] J. Sparsø and S. Furber, *Priciples of Asynchronous Circuit Design—A System Perspective*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001.

[11] D. Bertozzi, L. Benini, and G. De Micheli, "Error control schemes for on-chip communication links: the energy-reliability tradeoff," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 818–831, 2005.

[12] S. R. Sridhara and N. R. Shanbhag, "Coding for system-on-chip networks: a unified framework," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 6, pp. 655–667, 2005.

[13] L. Li, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Adaptive error protection for energy efficiency," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '03)*, pp. 2–7, San Jose, Calif, USA, November 2003.

[14] H. Zimmer and A. Jantsch, "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip," in *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '03)*, pp. 188–193, Newport Beach, Calif, USA, October 2003.

[15] R. E. Ziemer and R. L. Peterson, *Introduction to Digital Communication*, Prentice-Hall, Upper Saddle River, NJ, USA, 2nd edition, 2001.

[16] B. W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, Addison-Wesley, Reading, Mass, USA, 1989.

[17] C. Grecu, A. Ivanov, R. Saleh, and P. P. Pande, "NoC interconnect yield improvement using crosspoint redundancy," in *Proceedings of the 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT '06)*, pp. 457–465, Arlington, Va, USA, October 2006.

[18] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, 1989, The 1988 Turing Award Lecture.

[19] A. Davis and S. M. Nowick, "Asynchronous circuit design: motivation, background and methods," in *Asynchronous Digital Circuit Design*, G. Birtwistle and A. Davis, Eds., pp. 1–49, Springer, New York, NY, USA, 1995.

[20] W. J. Dally and J. W. Poulton, *Digital Systems Engineering*, Cambridge University Press, Cambridge, UK, 1998.

[21] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. De Micheli, "Analysis of error recovery schemes for networks on chips," *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 434–442, 2005.

[22] Handshake Solutions, http://www.handshakesolutions.com/.

[23] R. Hegde and N. R. Shanbhag, "Toward achieving energy efficiency in presence of deep submicron noise," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 4, pp. 379–391, 2000.

[24] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference*, pp. 684–689, Las Vegas, Nev, USA, June 2001.

Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 50285, 12 pages
doi:10.1155/2007/50285

Research Article

Area and Power Modeling for Networks-on-Chip with Layout Awareness

Paolo Meloni,[1] Igor Loi,[1,2] Federico Angiolini,[2] Salvatore Carta,[3] Massimo Barbaro,[1] Luigi Raffo,[1] and Luca Benini[2]

[1] *Department of Electrical and Electronic Engineering (DIEE), University of Cagliari, 09123 Cagliari, Italy*
[2] *Department of Electronics and Information Scince (DEIS), University of Bologna, 40136 Bologna, Italy*
[3] *Department of Mathematics and Information Science, University of Cagliari, 09123 Cagliari, Italy*

Received 1 November 2006; Revised 2 February 2007; Accepted 1 March 2007

Recommended by Maurizio Palesi

Networks-on-Chip (NoCs) are emerging as scalable interconnection architectures, designed to support the increasing amount of cores that are integrated onto a silicon die. Compared to traditional interconnects, however, NoCs still lack well established CAD deployment tools to tackle the large amount of available degrees of freedom, starting from the choice of a network topology. "Silicon-aware" optimization tools are now emerging in literature; they select an NoC topology taking into account the tradeoff between performance and hardware cost, that is, area and power consumption. A key requirement for the effectiveness of these tools, however, is the availability of accurate analytical models for power and area. Such models are unfortunately not as available and well understood as those for traditional communication fabrics. Further, simplistic models may turn out to be totally inaccurate when applied to wire dominated architectures; this observation demands at least for a model validation step against placed and routed devices. In this work, given an NoC reference architecture, we present a flow to devise analytical models of area occupation and power consumption of NoC switches, and propose strategies for coefficient characterization which have different tradeoffs in terms of accuracy and of modeling activity effort. The models are parameterized on several architectural, synthesis-related, and traffic variables, resulting in maximum flexibility. We finally assess the accuracy of the models, checking whether they can also be applied to placed and routed NoC blocks.

1. INTRODUCTION

Current and future Systems-on-Chip (SoCs) achieve increasingly better functionality and performance by integrating larger amounts of processing elements. This growth in computing resources must be matched by a corresponding evolution of the interconnection infrastructure. Traditional communication fabrics exhibit scalability issues in terms of performance and physical circuit design. The Network-on-Chip (NoC) paradigm, which brings networking concepts to the on-chip domain, answers such concerns with a scalable design, at both the architectural and physical levels.

NoC design is a discipline with a large amount of degrees of freedom. While this is an advantage, in that the interconnect can be optimally tailored to match the application at hand, it creates a critical issue when exploring the design space from the hardware overhead point of view. Given the flexibility of NoCs, an exhaustive exploration would require impractical amounts of synthesis runs, and a thorough characterization of switching activity in every candidate topology to properly assess power consumption.

An answer to the NoC customization complexity lays in the deployment of robust CAD tools, introducing the ability for automated design space exploration according to some fast optimization algorithm. In turn, however, such an approach requires the availability of accurate analytical models of NoC area and power consumptions to drive the optimization algorithms. Models allow the designer to quickly preestimate the area requirements and power consumption overhead introduced by the candidate interconnects. However, it must be noted that, as with any hardware component, the hardware cost of an NoC switch depends on several kinds of parameters, including (i) architectural (e.g., amount of buffering), (ii) synthesis tool-related (e.g., target operating frequency), (iii) operating (e.g., traffic flows).

It is also sometimes easy to underestimate the complexity of synthesis flows, which involve multiple tools, increasingly complex libraries, a large amount of heuristics and

several approximations or models of the behavior of physical on-chip devices. Experience shows that one wrong assumption may severely impact the properties of a whole design. For example, in recent years, the increasing importance of wiring resources has sometimes been impacting the final performance of circuits in unexpected ways, for example, by introducing higher parasitic capacitances and therefore power consumption, or by forcing redesign iterations due to higher transition latencies. This scenario demands for careful assessment of the accuracy of any predictive model at the lowest available level of abstraction; if possible, designers should try to validate their assumptions on placed and routed netlists. This obviously requires a large effort and may be impractical.

As a major contribution of this work, we propose an NoC modeling methodology which takes advantage of the designer's knowledge of the target architecture and synthesis library. It is of course impossible to devise an accurate yet fully generic model for the hardware cost, in power and area, of any given NoC. Our focus is instead on how such a model can be built for a specific NoC instance; we will illustrate the challenges and opportunities involved in this flow, in terms of accuracy and characterization time.

For our analysis, we choose the ×pipes NoC switch as a case study due to its parameterizability (Section 3). Key properties of our approach include accuracy and explicit modeling on several parameters of the design, like switch cardinality, flit width, buffering, traffic, and synthesis parameters. These properties make the approach suitable for fast exploration of large parts of the fabric design space, flexible and applicable in real life, for example, by accounting for the behavior of the synthesis tools when the target operating frequency approaches the limits of the design. The characterization is dependent on the target technology library, but can be easily scripted and automated. A major novel feature of our study, improving on our previous work [1], is that we explore the accuracy of our modeling style against placed and routed test instances; we feel that this is an essential step given the uncertainties intrinsic in today's technology processes. We also show that model coefficients can be made even more accurate by using a placed and routed training set for characterization, albeit at a modeling effort cost, and that the remaining inaccuracies can mostly be attributed to the intrinsic variations induced by synthesis tools.

Our approach starts from an existing RTL description of the NoC components, which are then synthesized and characterized under multiple architectural configurations and traffic conditions. A mathematical formulation of the area and power models is derived from empirical evidence and from the designer's knowledge of the NoC. Eventually, the coefficients of the model are fitted to the experimental results, guaranteeing accurate results for the given architecture. We present two different ways of characterizing the coefficients, with varying accuracy/effort tradeoffs, and two models to account for the dependency of synthesis results on the target synthesis frequency. The synthesis process can optionally include the placement and routing step for maximum thoroughness of the assessment.

2. RELATED WORK

NoCs have recently been proposed as a way to overcome the scalability issues of traditional interconnects [2–4]. Research has focused on multiple design levels. From the architectural point of view, a complete scheme is presented for example in [5], while specific topics are tackled in several works, such as quality of service (QoS) provisions [6] and asynchronous implementations [7]. The complexity in tackling the configurability of NoCs has been made clear by [8], where synthesis results for switches show widely different hardware requirements. Hence, the need for the development of algorithms and CAD tool-chains for NoC instantiation and optimization, as found for example in [9, 10]. A requirement of such works is the availability of reliable power and area models.

Power models and simulators for processors and memories have been proposed in an extremely large body of research [11, 12]. Interconnects have also become the focus of research [13], due to their increasing role in the hardware budget of recent and future systems; for example, the on-chip network of the MIT RAW chip multiprocessor is taking 36% of the chip power budget on average [14].

Some models of NoC hardware cost have already been proposed in previous literature. Results in [15] are derived from a mix of results on template circuits and from technology trends, and are specifically aimed at wide applicability. Therefore, even though they have been used for design space exploration [16] and in association with high-level traffic injection models [17], they do not guarantee maximum accuracy within an architecture-specific CAD flow. The main advantage of these techniques is flexibility and fast deployment. We see them as complementary to our approach, especially for initial exploration when the NoC component library is not available yet.

The approach in [18], on the other hand, attempts to build a cycle-accurate power model of a target router instance. However, several major points differentiate our approach. First, we build a model which is parametric not only on traffic-related events, but also on the architectural knobs of the design. Second, we include an area model in the exploration. Third, our model can be more readily adopted within a CAD mapping flow; this is both because we express the model as a function of architectural parameters, and because we provide a high-level dependence on traffic variables, instead of a cycle-by-cycle one. Fourth, we strive to make our approach as applicable as possible in real-world conditions, including the hard-to-model peculiarities of the behavior of synthesis tools when aiming for maximum frequency operation, and placement and routing issues. Fifth, we propose a fast characterization mechanism, by means of which model coefficients can be quickly derived with a minimal amount of synthesis runs.

In [19], a framework for NoC exploration is presented; the framework includes a power modeling flow. The power model features very limited dependence on architectural parameters and does not seem to account for the configuration knobs of synthesis tools. No area model is provided.

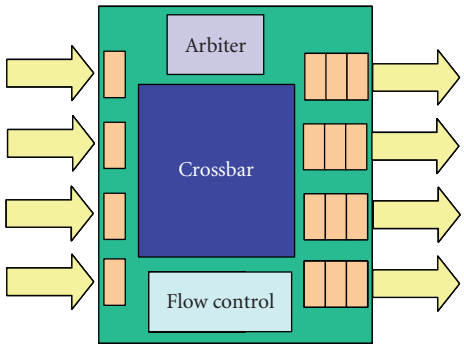


Figure 1: 4 × 4 ×pipes switch architecture.

In [20], a bit energy modeling flow is proposed to compare different switch fabrics in IP network routers. The approach is focused on the cost for transmitting bits from input to output ports, and while bit pattern-accurate, it is only focused on comparing router topologies against each other. The authors of [21] propose a model based on transistor count, while in [22], which is focused on FPGAs, switch cardinality is the main parameter. None of these models is meant for simultaneously accurate, parametric, and fast representation of power consumption, that is, suitable for design space exploration within a CAD environment.

3. THE ×PIPES SWITCH ARCHITECTURE

We choose the switch architecture defined in the ×pipes NoC component library [8, 23] as a case study, due to its customizability. The ×pipes switch (Figure 1) is output buffered; FIFOs of configurable depth are instantiated at each output, while inputs feature a single register. The flit width can be arbitrarily set. The number of input and output ports is also a parameter; full connectivity is provided in the central crossbar. An arbiter is attached to each output port to handle contention issues. We test the switch with its default ACK/NACK flow control mechanism, which leverages the output buffer resources. Since ×pipes performs source routing, the switch does not include routing LUTs.

4. PROPOSED MODELING METHODOLOGY

Our modeling activity is composed of five main phases. First, we devise a set of parameters that are relevant to the accuracy of any model which aims at practical applicability. Second, we define a general model formula for area and for power, relying on the knowledge of the target architecture as explained in Section 3. Third, we synthesize several configurations (*training set*) of the target switch architecture in a 0.13 μm technology library with Design Compiler [24], and measure the corresponding area and power consumption. The configurations are chosen so as to uniformly but sparsely cover the design space of interest, therefore allowing for an accurate yet quick construction of the model. Fourth, we use experimental results to numerically quantify the coefficients of the model. As outlined later, we propose two different ways of performing this step. Fifth, we assess the quality of our models against configurations (*test set*) outside of the training set. The first four steps will be covered in Sections 4.1, 4.2, 4.2.3, 4.3, while the fifth will be discussed in Section 5. An outline of how we handle steps 3–5 is provided in Figure 2.

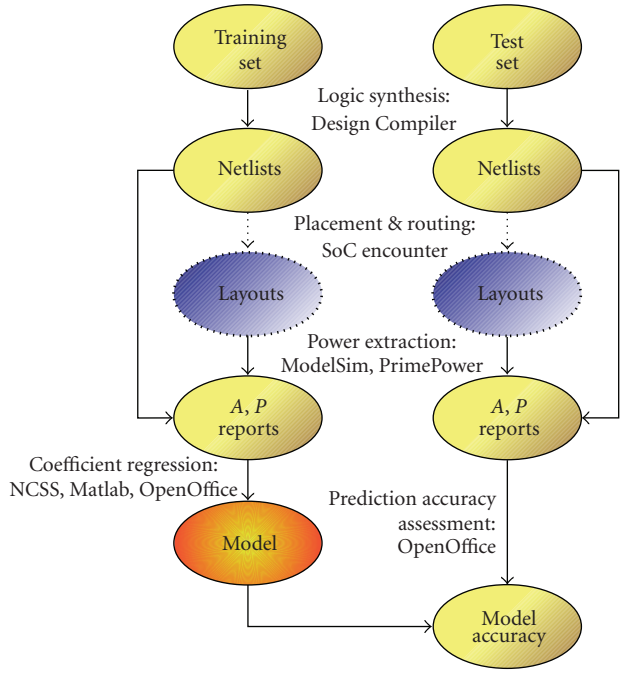


Figure 2: Outline of our characterization activity. The placement and routing step is optional both for the training and the test set.

4.1. Parameters of interest

A key phase of the approach is devising a model that matches the architecture under consideration and its properties. However, considering the architecture alone does not guarantee that the model will be applicable and accurate enough in practice. For example, synthesis tools play a primary role in defining the area and power efficiency of a component. Therefore, we first summarize the parameters of interest when assembling our model.

4.1.1. Architectural Parameters

The main parameters are

(i) switch cardinality (number of ports); to account for rectangular switches, we separately consider the amount of input ports (np_i) and output ports (np_o);
(ii) amount of buffering devoted to flow control handling and performance optimization, also called buffer depth (bd) (expressed in terms of single-flit buffering elements);
(iii) number of bits of the incoming and outgoing elementary data blocks, also called flit width (fw).

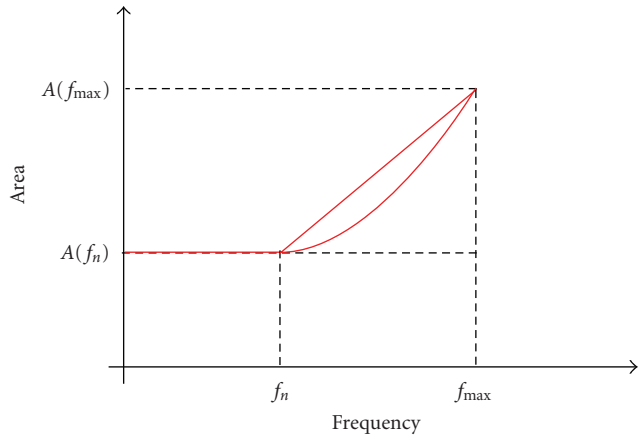


Figure 3: Area requirements versus target operating frequency.

4.1.2. Implementation flow parameters

It is possible to tune synthesis tools, among other things, for

(i) target frequency of operation;
(ii) target area;
(iii) target power consumption.

Tuning these parameters differently in the synthesis tools yields, as expected, a widely different quality of results. For example, when performance demands are extreme, synthesis tools are forced to create netlists containing large amounts of buffers and fast gates, which are not area- and power-efficient. To mimic a typical industrial flow, where an application performance constraint must be satisfied, we impose as the primary objective a certain target operating frequency (which is a parameter of our model), while area and power minimization are given to the tool as secondary optimization objectives. As a result, area and power requirements, expressed as a function of the target operating frequency, exhibit a characteristic flat behavior followed by a steeply rising trend after an inflection point. This trend is well known, and can be explained by the fact that, above some target operating frequency which can be achieved with minimal circuitry, synthesis backends are forced to insert extra gates to comply with increasing performance demands.

Figure 3 shows a linearized and a parabolic approximation of this trend, and at the same time summarizes the ways we modeled this effect. For each device configuration (e.g., 4×4 32-bit switches with 6-deep FIFO buffers), a "native" frequency f_n can be identified. This frequency is that achieved by the synthesizer with relaxed timing constraints. Under this condition, the tool is free to fully pursue its secondary objectives, hence creating minimum area ($A(f_n)$) and power ($P(f_n)$) netlists. Configuring the tools for target frequencies lower than f_n does not result in further decreases of area or power dissipation. For each switch instance, it is also possible to find a frequency f_{max}, that corresponds to the fastest achievable synthesis result. Under this timing constraint, the module has $A(f_{max})$ area and $P(f_{max})$ power consumption. We approximate the dependency of area and power overheads as linear or parabolic in the range $(f_n; f_{max})$. This assumption allows us to characterize devices only twice, at f_n and f_{max} (under various combinations of the other architectural parameters), while being able to estimate results over the whole range of frequencies achievable by the module. Since this analysis is not correlated to other model parameters, in the following, for simplicity of notation, we will not explicitly mention the dependency of coefficients on the synthesis target frequency; the characterization of this parameter will be implicitly assumed.

The linearized or parabolic approximation is a way of abstracting away from low-level details of the logic synthesis process, which are impossible to capture in a high-level model. The experimental results that will be shown in Section 5 will be based on a test set which is also spread in terms of target operating frequency, therefore providing a metric of the accuracy of such a model. Section 5.5 will compare the accuracy of the linear versus the parabolic models.

Please note that developing area and power models which are a function of the target frequency of operation up to f_{max} also implies making available a model of the timing properties of the switches.

4.1.3. Traffic condition parameters

These parameters are only relevant to power models, since area models are clearly static. They include downstream congestion and internal congestion (i.e., arbitration conflicts). They will be explained in more detail in Section 4.2.2.

4.2. Area and power models

4.2.1. Area model

In general, the area equation must be of the form:

$$A = f(bd, fw, np_o, np_i). \tag{1}$$

We identify as suitable the area model:

$$A(fw, bd, np) = A_1 \cdot np_o \cdot fw \cdot bd + A_2 \cdot np_i \cdot fw + A_3 \cdot np_o \cdot np_i + A_4 \cdot fw \cdot np_o \cdot np_i. \tag{2}$$

The rationale of this formula is that the area of the target switch can be rendered as the sum of four contributions (Section 3): (i) output buffers, (ii) input buffers, (iii) arbitration and flow control logic, (iv) crossbar. Each contribution strongly depends on a known combination of architectural parameters as follows.

(i) Output buffers, which are dominated by flip-flop area, can be supposed to depend linearly on flit width fw and buffer depth bd (×pipes switches are output-buffered), which, respectively, represent the width and depth of the buffer (Figure 4), there are np_o such buffers.
(ii) Input buffers are similar to the case above, but since they have a constant depth, they do not depend on bd. Obviously np_i is used in place of np_o.

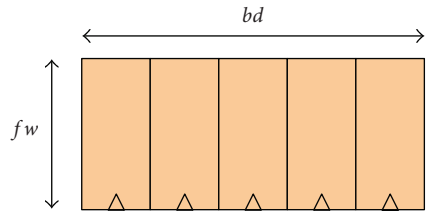


Figure 4: Dependency of the output buffer area on fw, bd.

(iii) Since a distributed arbitration technique is used in the target switch, one arbiter is instantiated at each output port. Each arbiter has a complexity proportional to the number of candidate input ports np_i, therefore the overall contribution is the product of the input and output cardinalities. The arbiter logic is clearly independent of datapath parameters such as flit width and buffer depth.

(iv) The area overhead due to the crossbar must have a linear dependency on flit width, must be independent of the buffering resources, and must have a linear dependency on the product of input and output cardinalities.

4.2.2. Power model

The power consumption of a module depends on the switching activity of the cells, so, to express the power consumption of an NoC switch, a term that accounts for traffic conditions must be present. The most general way to model the power consumption thus becomes

$$P = f(bd, fw, np_o, np_i, T) \tag{3}$$

with T being a generic variable that summarizes the traffic conditions. Since sequential components exhibit a power consumption even if they are not performing computation, due to the clock switching, a static (traffic-independent) term must appear. After analyzing the possible traffic flows in the ×pipes router, we propose (4) as a general power model:

$$\begin{aligned} P(bd, fw, np_o, np_i, T) = {} & P_A(\cdots) + \sum_{j=1}^{np_o} [P_B(\cdots) \cdot T_{Oj}] \\ & + \sum_{j=1}^{np_o} [P_C(\cdots) \cdot T_{OCj}] \\ & + \sum_{j=1}^{np_i} [P_D(\cdots) \cdot T_{ICj}], \end{aligned} \tag{4}$$

where the dots express dependencies on bd, fw, np_o, np_i which will be analyzed in more depth in the following. The first term models the power dissipated by inactive, but still clocked, registers. The remaining terms depend on traffic conditions. An accurate representation of the traffic conditions requires a separate analysis of the state of each input and output port. Therefore, we define np_o traffic variables T_{Oj} and T_{OCj}, to model the lack or presence of external congestion, and np_i traffic variables T_{ICj}, to model internal contention for resources. More specifically, we define the following.

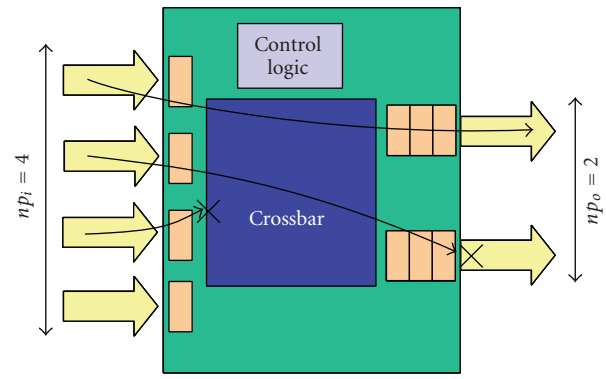


Figure 5: Example traffic in a 4×2 switch.

(i) T_{Oj}: percentage of time during which the output port j is successfully transmitting flits. This coefficient models traffic in absence of congestion.

(ii) T_{OCj}: percentage of time during which the output port j is trying to transmit, but flits are rejected. This coefficient models external congestion due to traffic spikes.

(iii) T_{ICj}: percentage of time during which the input port j of the switch is trying to transmit flits through one of the output ports, but arbitration is denied by the switch logic. This coefficient models the contention for the same output port inside of the switch.

This set of traffic percentages is linearly independent, since the complex arbitration and flow control patterns within an NoC switch make it very easy for some of these time windows to overlap. Please consider the following.

Example 1. A 4×2 switch (see Figure 5) may feature one established input-to-output connection where traffic is freely flowing (which is expressed by the condition T_{O1}), another established input-to-output connection which is stuck due to congestion in the downstream switch (modeled within T_{OC2}), while the third input port is unsuccessfully trying to transmit to one of the two output ports, which in this example are already busy (T_{IC3}), and the fourth is simply idle (this contribution is therefore included in the coefficient P_A).

The coefficients P_A, P_B, P_C, P_D depend on architectural parameters, as for the area model. They account for the power consumption in the traffic states, described above, as follows.

(i) P_A accounts for the static power dissipated by the switch and it is due to the noncombinational logic in

Table 1: Dependency on architectural parameters of the static power coefficient P_A.

Contribution	Dep. on fw	Dep. on bd	Dep. on np_i	Dep. on np_o
Output buffering	Linear	Linear	None	Linear
Input buffering	Linear	None	Linear	None
Spare registers	None	None	Linear	Linear

the design. Therefore, it simply depends linearly on the number of flip-flops in the design, which are

(a) input buffers;
(b) output buffers;
(c) state registers in the control logic;

whose dependencies on architectural parameters are summarized in Table 1.

(ii) P_B accounts for the dynamic power dissipated by flowing packet streams, due to the enabled registers and to the switching activity of combinational logic. We identify four contributions to the power dissipation:

(a) output buffers: in these buffers, during every cycle, one of the flit registers (fw bits wide) samples a new piece of data; a $bd \times 1$ multiplexer then brings a flit to the output port therefore, this contribution is itself the sum of two terms;
(b) input buffers;
(c) control logic;
(d) selected crossbar branch.

The dependencies of these contributions on the architectural parameters are summarized in Table 2.

(iii) P_C accounts for the dynamic power dissipated by the switch under a scenario where downstream congestion is preventing a free flow of packets. Although numerically different, the P_C coefficient is similar to P_B, in that it still involves an established input-to-output channel, and therefore its dependency on architectural parameters is the same (see Table 2).

(iv) P_D accounts for the power dissipated by the switch when an incoming stream requires the access to an output port, but the arbitration is denied. The contributions to this portion of the power consumption are related to the following logic blocks:

(a) input buffers;
(b) control logic.

We can summarize the dependencies on this contributions as shown in Table 3.

The dependencies of the power coefficients are thus summarized in Table 4.

We would like to stress that some coefficients, which could be intuitively expected to quadratically depend on parameters, are instead linearly dependent, because they characterize a single input or output port. The quadratic behavior is indirectly restored by the summation symbols in (4).

4.2.3. Choice of a relevant training set

To characterize the coefficients of our area and power models, we define a training set, composed of switch configurations chosen in such a way as to uniformly cover the relevant design space for the particular NoC under study. In the case of the ×pipes NoC, which is focused on the highest customizability of topologies, we choose to study a design space spanning over a large variety of cardinalities (np_i and np_o of 4, 10, 16, and 20). Since ×pipes is also focused on the best performance/overhead tradeoff point [23], and therefore on low hardware cost, we consider moderate buffer depths bd of 5 and 7 FIFO locations and flit widths fw of 21, 28, and 38 bits.

In the modeling approach called full factorial design, all the possible permutations of the values of the independent design parameters should be studied to create the training set. This is often impractical due to the quick rise in the number of instances as soon as new design knobs are added, leading to approaches to select only a subspace of the characterization set (fractional factorial design). In our case, based on the knowledge of the target architecture, we choose a very simple way of pruning the training set. The rationale is based on the observation that rectangular switches add a smaller amount of information to the training set; for example, when studying the power consumption, a rectangular switch is by design unable to simultaneously feature traffic flows on all of its input and output ports (see Figure 5), and is therefore behaving similarly to a square switch of smaller cardinality. Our preliminary internal testing confirms this property, at least for the ×pipes NoC. Therefore, we simply choose to coalesce the np_i and np_o axes for the generation of the training set, and only include 4×4, 10×10, 16×16, and 20×20 instances.

We finally permutate all the possible parameter values, resulting in 24 (4 cardinalities times 2 buffer depths times 3 flit widths) configurations being synthesized.

4.3. Fitting model coefficients

4.3.1. Fitting area model coefficients

To estimate A_1, A_2, A_3, A_4, we propose two different methods.

(i) Methodology 1: coefficients can be derived directly from synthesis reports, which hierarchically list every switch subblock. For example, once the area cost of an output buffer which is bd_0 flits deep and fw_0 bits wide is gathered from one report, it can be called $A_{\text{obuf}}|_{bd_0, fw_0}$. Since A_1 is expected to increase linearly with both bd and fw, it can be approximately derived as:

$$A_1 = \frac{A_{\text{obuf}}|_{bd_0, fw_0}}{bd_0 \cdot fw_0}. \tag{5}$$

Other coefficients can be similarly computed.

Table 2: Dependency on architectural parameters of the dynamic power coefficients P_B, P_C.

Contribution	Dep. on fw	Dep. on bd	Dep. on np_i	Dep. on np_o
Output buffer (reg.)	Linear	None	None	None
Output buffer (mux)	Linear	Linear	None	None
Input buffer	Linear	None	None	None
Control logic	None	None	Linear	None
Crossbar branch	Linear	None	Linear	None

Table 3: Dependency on architectural parameters of the dynamic power coefficient P_D.

Contribution	Dep. on fw	Dep. on bd	Dep. on np_i	Dep. on np_i
Input buffer	Linear	None	None	None
Control logic	None	None	Linear	Linear

Table 4: Dependency of power coefficients on architectural parameters.

Model coeffcient	Dep. on fw	Dep. on bd	Dep. on np_i	Dep. on np_o
P_A	Linear	Linear	Linear	Linear
P_B	Linear	Linear	Linear	None
P_C	Linear	Linear	Linear	None
P_D	Linear	None	Linear	Linear

Advantages: with this methodology, each contribution in the formula keeps a strict physical meaning. Only one synthesis run is needed to extrapolate coefficients for any switch instance; we arbitrarily choose a 10×10, 28-bit switch as a reference. This instance is close to the center of the design space of interest (see the previous subsection); its choice will be further discussed in Section 5. Disadvantages: this simplified approximation discards any constant offset that may be present in the coefficients. Further, the nature of synthesis tools introduces unpredictable fluctuations in the netlist area and power trends under different architectural configurations. This noise does not have any easily characterizable property. Thus, the model incurs a nonnegligible error when compared against actual switch instances. Moreover, the choice of the specific switch instance for characterization might skew the computed coefficient values.

(ii) Methodology 2: coefficients can be derived by leveraging the multivariate nonlinear regression algorithms natively provided by several mathematical and statistical packages. In this case, the input is a set of characterization syntheses (the training set described in the previous subsection). The target polynomial for the regression is chosen based on insight of the dependency of area on the architectural design parameters (see (2)). Advantages: the model fits actual synthesis results better. Disadvantages: longer characterization time; with a thorough characterization set like that chosen in Section 4.2.3, experiments must be performed in 24 device instances, against just one. The actual improvement in accuracy depends on the smoothness of the native behavior of the synthesis tools. Some coefficients may lose their physical meaning (e.g., they may become negative).

Both methodologies can be readily adapted to any parameterizable NoC architecture.

4.3.2. *Fitting power model coefficients*

To characterize the P_A, P_B, P_C, P_D coefficients, we first inject traffic into the switch netlists under test, one at a time. This is achieved by ModelSim [25] simulation of the Verilog netlists (please refer to Figure 2), to which traffic generators are attached. The traffic generators are configured to inject into the switch one of the four patterns described above (idle, free flow, downstream congestion, internal contention) at a time. The switching activity is logged and fed as an input to Synopsys PrimePower [26], which provides a hierarchical report of the power consumption of the switch subblocks. For each netlist of the training set, four hierarchical reports are therefore generated.

At this point, the power model coefficients are determined by using either of the techniques just outlined for the area models. The P_A, P_B, P_C, P_D scenarios are separately accounted for; the fitting polynomials are directly derived from Table 4. For each of them, either by direct derivation or by nonlinear regression, we extract the coefficients modeling the dependency on architectural parameters.

5. EXPERIMENTAL RESULTS

To evaluate the accuracy of the proposed techniques, we first randomly choose a test set of 70 switch configurations spread across the design space of interest (both in terms of architectural parameters and target synthesis frequencies), and not

overlapping with the training set previously used for characterization. Each switch is synthesized with Design Compiler to extract its area requirements, then stimulated with traffic streams within ModelSim and studied in PrimePower to evaluate its power consumption (Figure 2). A reference set of experimental results is therefore collected. The area and power consumption of the same set of switches is then estimated according to our methodology, and the statistical distribution of the resulting error is plotted to study the behavior of both coefficient fitting strategies.

The implementation flow of any circuit design is composed of several steps, among which the two major ones are logic synthesis (i.e., mapping functionality onto elementary cells from a technology library), which results in a netlist, and placement and routing (i.e., placing and interconnecting the netlist within a target floorplan), generating a layout as the outcome. Netlists can be generated in a relatively short time, but they do not include any information about the placement of the cells, and thus do not give any information about the length of the wires needed for the interconnections. This is a key missing piece of information, especially as designs become wire-dominated. Therefore, logic synthesis tools try to model the effect of wires by means of predictive models provided by the vendor of the technology library. These models are necessarily simplistic, and therefore may impact the accuracy of any area and power evaluation at the netlist level. On the other hand, creating the layout of a complex circuit provides more accurate estimations of its area and power cost, but this extra step is at least as time-consuming as the initial logic synthesis. Therefore, designers would clearly like to avoid performing this extra phase repeatedly during a modeling activity, if at all possible.

To assess the usefulness of our models, we investigate their inference and their application to both netlists and layouts. This can be seen in Figure 2, where the placement and routing step is optional.

5.1. *Experiments with netlist-based models and a netlist-level test set*

In this subsection, we generate our models starting from synthesized (but not placed and routed) switch instances, and check their accuracy against a test set which is also at the netlist level. The results are depicted in Figure 6, where the vertical axis reports the number of occurrences of inaccuracies comprised in the ranges listed on the horizontal axis. As can be seen, in around 80% of the cases, our models result in an error margin less then 10% of the actual value. Sporadically, relatively higherror rates of up to 20% are detected; however, as can be seen for example in Figure 7, the distribution of the errors is quite randomly spread over the design space, and comprises both under- and overestimations. The figure reports modeling inaccuracy for a subspace having as axes the flit width and the switch cardinality; these numbers are thus only a subset of the whole test set. Similar plots can be derived for varying buffer depths and target synthesis frequencies, and we omit them due to space constraints. Therefore, we can attribute inaccuracies to the

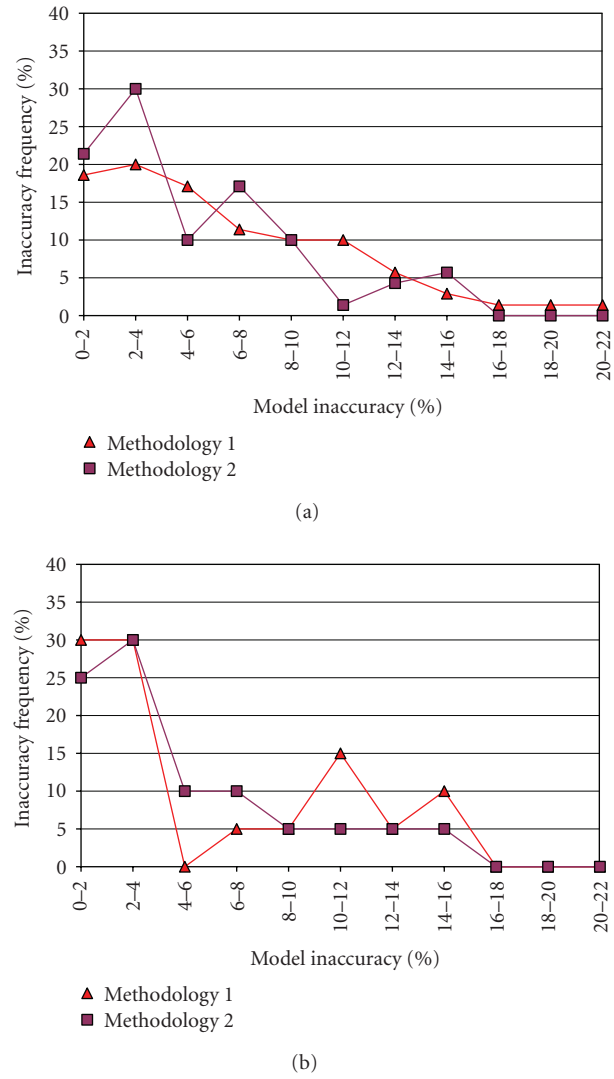


Figure 6: Modeling inaccuracy (percentage deviation among predicted and actual values of (a) area, (b) power, for the test set) under two different characterization policies for the coefficients. Vertical axis represents the occurrence frequency of a given inaccuracy range. Models and test set are at the netlist level.

unpredictability which is intrinsic in the behavior of synthesis tools, and not to a problem of our modeling approach.

Comparing the results of the two techniques for coefficient fitting presented in Section 4.3, we see that the tails of the inaccuracy distributions drop more sharply for Methodology 2, indicating a lower chance of large modeling errors. However, Methodology 1 exhibits just marginally worse average inaccuracy rates: 6.26% against 5.30% for power models and 5.97% against 5.45% for area models. In terms of characterization effort, in our experience, we can roughly assume that one hour may be needed in average for the analysis of an instance of the training set; therefore, Methodology 1 requires one hour of runtime, while Methodology 2 needs 24 hours to provide numerical values of coefficients (the actual time depends on how thoroughly the design space

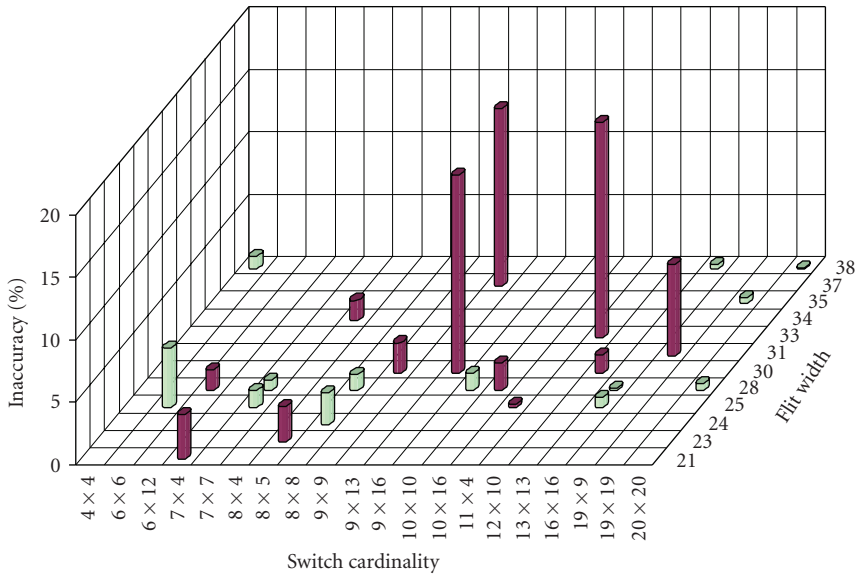


Figure 7: Distribution of the area modeling inaccuracy over a subset of the design space for Methodology 2. Dark color: underestimations; light color: overestimations.

is covered). Due to the drastically lower effort, Methodology 1 becomes a natural candidate for fast yet accurate modeling. However, this approach leverages upon a single switch instance to characterize all the coefficients. The choice of the reference switch configuration is therefore key, and may impact the robustness of the flow. Internal testing, that we omit due to space constraints, shows that coefficients are quite accurately rendered under a wide range of possible choices of the reference switch. However, when manually picking an "outlier" instance as the reference, errors over the whole design space turn out to be large. As a possible workaround, Methodology 1 could be applied to multiple switch instances to minimize the chance of choosing bad references; outliers could be effectively discarded. This hybrid approach provides better reliability, but requires a modeling effort which is progressively closer to that of Methodology 2 as its robustness is increased. Methodology 2 remains the most accurate and reliable, and its characterization time can still be assumed to be fully acceptable for both academic and industrial environments.

5.2. Test case: a complete NoC topology

To further validate the most complex part of our methodology, that is, the power modeling, we study a whole NoC topology, such as a 5×3 mesh. The mesh includes switches with three different cardinalities of 4×4, 5×5, and 6×6. We then inject functional traffic, namely, that required to drive a multimedia application from the MPARM [27] suite, in the topology, and compare the resulting power consumption against that predicted by our model (characterized with Methodology 2). Traffic patterns in the mesh are irregular, due to application needs, causing the switches to spend variable amounts of time in each possible state. The results are

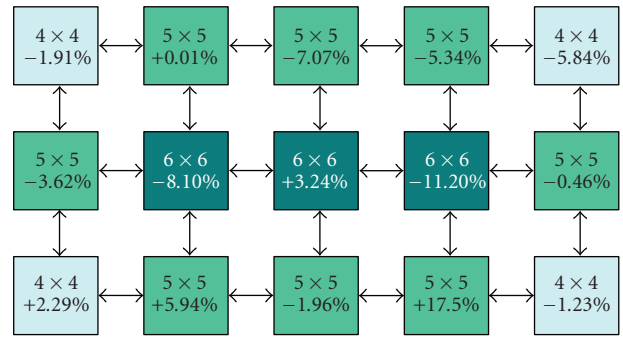


Figure 8: Distribution of the power modeling inaccuracy for the switches of a 5×3 NoC mesh.

plotted in Figure 8. The average inaccuracy is 5%, with only two switches out of fifteen (about 13%) exhibiting inaccuracies greater than 10%. Since the power consumption of some switches is overestimated while that of others is underestimated, the margin of error on the consumption of the whole mesh is as low as 1.3%. This result confirms the usefulness of our modeling strategy for integration within a CAD mapping and design space exploration flow.

5.3. Experiments with netlist-based models and layout-level test set

We try to apply the previously mentioned models, which are based on netlist-level analyses, to a layout-level test set, by placing and routing the test set described above. This activity generates a very realistic test set, and is a demanding metric for the accuracy of the models, since extra unpredictable

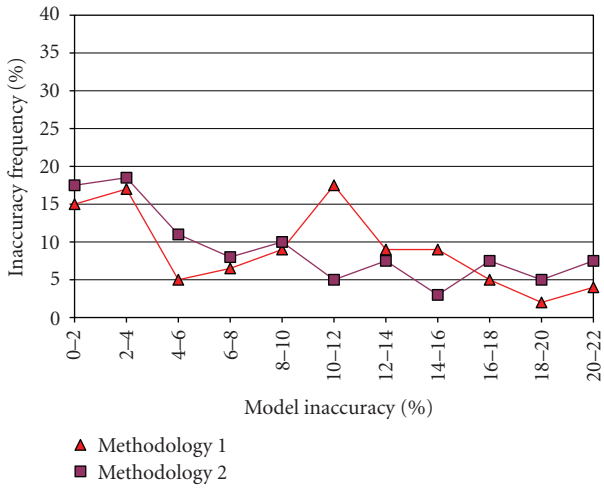


Figure 9: Modeling inaccuracy (percentage deviation among predicted and actual values of power for the test set) under two different characterization policies for the coefficients. Vertical axis represents the occurrence frequency of a given inaccuracy range. Models are at the netlist level, test set is at the layout level.

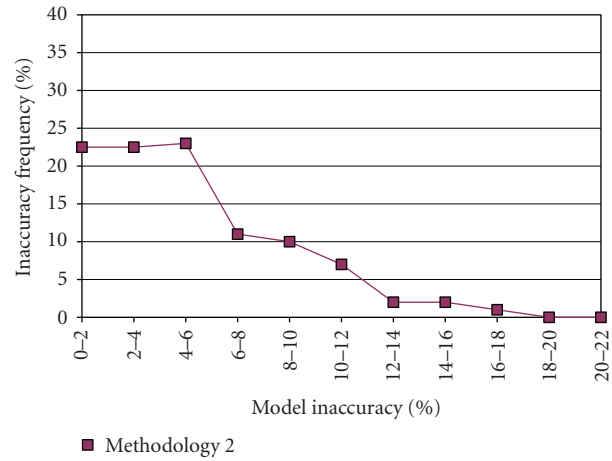


Figure 10: Modeling inaccuracy (percentage deviation among predicted and actual values of power for the test set) with Methodology 2 characterization policy for the coefficients. Vertical axis represents the occurrence frequency of a given inaccuracy range. Models and test set are at the layout level.

noise is added. The results we get are presented in Figure 9, which should be compared to Figure 6(b). The two plots exhibit a comparable trend and errors of roughly the same magnitude, even though the average modeling error for the layout-level test set is about 3% higher. This means that models developed by only taking netlists into account still show good accuracy even with respect to layout-level power evaluation. The added noise also blurs the accuracy difference between Methodology 1 and Methodology 2, both in maximum error (26% versus 23%) and average error (8.8% versus 8.35%). While Methodology 2 remains marginally more accurate, these results seem to suggest that the unpredictability introduced by the logic synthesis process is somewhat unrelated to that introduced by the placement and routing phase. In other words, even though Methodology 2, thanks to its interpolation of results, can compensate for some of the nonidealities of the logic synthesis process better than Methodology 1, this compensation is less effective when trying to predict the power consumption after placement and routing.

5.4. Experiments with layout-based models and a layout-level test set

In an attempt to check whether more accurate models can be built, we recompute the numerical coefficients starting from a layout-level version of the training set and applying Methodology 2. This model is very close to an ideal reference point, since it is derived from a regression on experimental results which already encompass most of the unpredictable elements of the synthesis flow. However, the time required to build the model coefficients is noticeably longer. In our experience, both logic synthesis and placement steps require a computation time which is not easy to predict, as it largely depends on many factors, such as the switch cardinality and the target operating frequency. However, as a rule of thumb, the two steps are about equally time consuming; therefore, the modeling time is approximately doubled.

The error distribution resulting from the usage of the layout-level test set when validating the model coefficients achieved from a layout-level training set is shown in Figure 10.

As can be noticed, and as expected, the average error and the maximum error values both noticeably decrease when compared to Figure 9. However, the decrease is not huge. We attribute the remaining inaccuracies in Figure 10 to the intrinsic unpredictability of the synthesis tools. Even after taking into account all the systematic behaviors in the synthesis flow, the trend is the result of residual instance-to-instance variations due to heuristics in the CAD tools and to degrees of freedom which can only vary in a discrete fashion.

The accuracy improvement guaranteed by a layout-level characterization is associated to a doubling of the runtime overhead, and still does not completely eliminate the presence of some "outlier" instances. The designer may certainly choose to adopt our methodology to characterize devices at the layout level for maximum accuracy. However, we feel that a result that can be derived from our experiments is that, at least at the 0.13 μm technology node, it is still feasible to use accurate netlist-based models in order to save characterization time.

5.5. Experiments with a parabolic model for the dependency on the target synthesis frequency

We conclude our experiments by checking whether a linear model is accurate enough to characterize the dependency of synthesis results on the target synthesis frequency (see Figure 3). We leverage a parabolic model as a potentially more accurate approximation of the actual dependency of model coefficients on the target frequency, then recheck the

Table 5: Accuracy of the linear versus parabolic models for the dependency of synthesis results on the target synthesis frequency. Coefficients derived with Methodology 2.

Experiment		Linear approx.	Parabolic approx.
Netlist training set,	Average error	5.19%	6.32%
Netlist test set	Maximum error	14.61%	15.27%
Netlist training set,	Average error	8.23%	6.57%
Layout test set	Maximum error	22.88%	19.94%
Layout training set,	Average error	5.04%	9.23%
Layout test set	Maximum error	16.10%	22.23%

model accuracy on the test set. The results are reported in Table 5.

These results do not seem to indicate a strong bias towards any of the alternatives. The linear approximation seems to cope much better with a netlist-level or layout-level test set when the model is derived from experiments on a training set at the same level, but the parabolic model is quite a bit better at predicting layout-level results starting from netlist-level models. We attribute this behavior to the impact of noise. In other words, although synthesis results do clearly change depending on the target frequency, the choice of a linear or parabolic model to describe this trend does not matter much, since the nonidealities introduced by the synthesis flow induce enough noise to blur the distinction. Overall, the usage of the linear model, which is simpler, seems to be justified.

6. CONCLUSIONS AND FUTURE WORK

We presented a methodology for characterization of NoC switch area and power requirements. The approach we propose is based on thorough parameterization on several architectural, deployment, and runtime variables. This guarantees excellent applicability within an NoC CAD flow for topology mapping and/or design space exploration. The area and power models for the ×pipes case study turn out to be very accurate within the limits allowed by the nonidealities of synthesis tools, even when applied to a whole NoC topology with irregular traffic flows. Our experiments show that, at least at the 0.13 μm node, applying our methodology to netlist-level devices yields an acceptable approximation of the actual behavior even after placement and routing, but that even greater precision can be achieved, if desired, by applying the same technique at the layout level. We also show that another tradeoff among accuracy and modeling effort exists, namely, coefficients can be extracted based on a single (or on just a few) device instances, by normalization of the synthesis report, or on several of them, by an interpolation process.

Future work includes minimizing the characterization effort, testing how well our technique scales to finer process technologies, and creating similar models for other NoC components, such as network interfaces (NIs). At the 0.13 μm node, we observe that the power consumption of NoCs is still largely dominated by dynamic switching activity [23], therefore we do not consider wire loads in our modeling approach for switches. Since this is expected to change in future technologies, techniques to establish models for NoC links may also become important.

Besides the absolute accuracy of the models, we also plan on quantifying the accuracy of our model when used from within a CAD flow to establish relative cost assessments of alternative NoC topologies; early results in this activity are encouraging, with good agreement between CAD expectations and actual measurements.

ACKNOWLEDGMENTS

This work has been supported by STMicroelectronics and by the Semiconductor Research Corporation (SRC) under contract 1188.

REFERENCES

[1] P. Meloni, S. Carta, R. Argiolas, L. Raffo, and F. Angiolini, "Area and power modeling methodologies for networks-on-chip," in *Proceedings of the 1st International Conference on Nano-Networks (Nano-Net '06)*, Lausanne, Switzerland, September 2006.

[2] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 684–689, Las Vegas, Nev, USA, June 2001.

[3] L. Benini and G. de Micheli, "Networks on chips: a new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[4] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '00)*, pp. 250–256, Paris, France, March 2000.

[5] F. Karim, A. Nguyen, S. Dey, and R. Rao, "On-chip communication architecture for OC-768 network processors," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 678–683, Las Vegas, Nev, USA, June 2001.

[6] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *Journal of Systems Architecture*, vol. 50, no. 2-3, pp. 105–128, 2004.

[7] T. Bjerregaard and J. Sparsø, "scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip," in *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC '05)*, pp. 34–43, New York, NY, USA, March 2005.

[8] F. Angiolini, P. Meloni, D. Bertozzi, L. Benini, S. Carta, and L. Raffo, "Networks on chips: a synthesis perspective," in *Proceedings of the International Conference on Parallel Computing (ParCo '05)*, pp. 745–752, Malaga, Spain, September 2005.

[9] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proceedings of the Conference on Asia and South Pacific Design Automation (ASP-DAC '03)*, pp. 233–239, Kitakyushu, Japan, January 2003.

[10] S. Murali and G. de Micheli, "SUNMAP: a tool for automatic topology selection and generation for NoCs," in *Proceedings of the 41st Design Automation Conference (DAC '04)*, pp. 914–919, San Diego, Calif, USA, June 2004.

[11] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA '00)*, pp. 83–94, Vancouver, BC, Canada, June 2000.

[12] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "The design and use of simplepower: a cycle-accurate energy estimation tool," in *Proceedings of the 37th Conference on Design Automation (DAC '00)*, pp. 340–345, Los Angeles, Calif, USA, June 2000.

[13] A. Bona, V. Zaccaria, and R. Zafalon, "System level power modeling and simulation of high-end industrial network-on-chip," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 3, pp. 318–323, Paris, France, February 2004.

[14] J. S. Kim, M. B. Taylor, J. Miller, and D. Wentzlaff, "Energy characterization of a tiled architecture processor with on-chip networks," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '03)*, pp. 424–427, Seoul, Korea, August 2003.

[15] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *Proceedings of the 35th Annual ACM/IEEE International Symposium on Microarchitecture (MICRO '02)*, pp. 294–305, IEEE Computer Society Press, Istanbul, Turkey, November 2002.

[16] H.-S. Wang, L.-S. Peh, and S. Malik, "A technology-aware and energy-oriented topology exploration for on-chip networks," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE '05)*, vol. 2, pp. 1238–1243, Munich, Germany, March 2005.

[17] N. Eisley and L.-S. Peh, "High-level power analysis of on-chip networks," in *Proceedings of the 7th International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES '04)*, pp. 104–115, Washington, DC, USA, September 2004.

[18] J. Chan and S. Parameswaran, "NoCEE: energy macro-model extraction methodology for network on chip routers," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '05)*, pp. 254–259, San Jose, Calif, USA, November 2005.

[19] G. Palermo and C. Silvano, "PIRATE: a framework for power/performance exploration of network-on-chip architectures," in *Proceedings of the 14th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS '04)*, pp. 521–531, Santorini, Greece, September 2004.

[20] T. T. Ye, L. Benini, and G. de Micheli, "Analysis of power consumption on switch fabrics in network routers," in *Proceedings of the 39th Design Automation Conference (DAC '02)*, pp. 524–529, New Orleans, La, USA, June 2002.

[21] C. S. Patel, S. M. Chai, S. Yalamanchili, and D. E. Schimmel, "Power constrained design of multiprocessor interconnection networks," in *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD '97)*, pp. 408–416, Austin, Tex, USA, October 1997.

[22] H. Zhang, M. Wan, V. George, and J. Rabaey, "Interconnect architecture exploration for low-energy reconfigurable single-chip DSPs," in *Proceedings of the IEEE Computer Society Workshop on VLSI '99 (IWV '99)*, pp. 2–8, Orlando, Fla, USA, April 1999.

[23] F. Angiolini, P. Meloni, S. Carta, L. Benini, and L. Raffo, "Contrasting a NoC and a traditional interconnect fabric with layout awareness," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE '06)*, vol. 1, pp. 124–129, Munich, Germany, March 2006.

[24] Synopsys Inc., "Design Compiler," http://www.synopsys.org/.

[25] Mentor Graphics, "ModelSim," http://www.model.com/.

[26] Synopsys Inc., "PrimePower," http://www.synopsys.org/.

[27] M. Loghi, F. Angiolini, D. Bertozzi, L. Benini, and R. Zafalon, "Analyzing on-chip communication in a MPSoC environment," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 2, pp. 752–757, Paris, France, February 2004.

Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 95859, 10 pages
doi:10.1155/2007/95859

Research Article

Avoiding Message-Dependent Deadlock in Network-Based Systems on Chip

Andreas Hansson,[1] Kees Goossens,[2,3] and Andrei Rădulescu[3]

[1] *Department of Electrical Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands*
[2] *Computer Engineering, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2600 GA Delft, The Netherlands*
[3] *SOC Architectures and Infrastructure, Research, NXP Semiconductors, 5656 AE Eindhoven, The Netherlands*

Received 16 November 2006; Accepted 6 February 2007

Recommended by Maurizio Palesi

Networks on chip (NoCs) are an essential component of systems on chip (SoCs) and much research is devoted to deadlock avoidance in NoCs. Prior work focuses on the router network while protocol interactions between NoC and intellectual property (IP) modules are not considered. These interactions introduce *message dependencies* that affect deadlock properties of the SoC as a whole. Even when NoC and IP dependency graphs are cycle-free in isolation, put together they may still create cycles. Traditionally, SoCs rely solely on request-response protocols. However, emerging SoCs adopt higher-level protocols for cache coherency, slave locking, and peer-to-peer streaming, thereby increasing the complexity in the interaction between the NoC and the IPs. In this paper, we analyze *message-dependent deadlock*, arising due to protocol interactions between the NoC and the IP modules. We compare the possible solutions and show that deadlock avoidance, in the presence of higher-level protocols, poses a serious challenge for many current NoC architectures. We evaluate the solutions qualitatively, and for a number of designs we quantify the area cost for the two most economical solutions, *strict ordering* and *end-to-end flow control*. We show that the latter, which avoids deadlock for *all* protocols, adds an area and power cost of 4% and 6%, respectively, of a typical Æthereal NoC instance.

1. INTRODUCTION

Networks on chip (NoCs) have emerged as the design paradigm for design of scalable on-chip communication architectures, providing better structure and modularity while allowing good wire utilisation through sharing [1–4]. By providing *services* for intermodule communication [5] over a mix of different sockets, NoC enables intellectual property (IP) reuse [3, 6, 7] and enhances system-level composability [6]. The services must be implemented robustly and efficiently.

Deadlock is catastrophic to as SoC and a serious threat to the robustness of the communication services offered by the NoC. Therefore, the importance of deadlock-free operation is stressed as a key research problem in NoC design [8] and much work is focused on providing deadlock-free routing in NoCs [9–11].

Deadlock freedom in the router network, henceforth just network, relies on the *consumption assumption* [12]: the network accepts and delivers all messages sent by the network interfaces (NIs) as long as they promise to consume all messages from the network when they are delivered. Routing algorithms that rely on this assumption, which to the best of our knowledge is true for all nonlossy routing algorithms currently used in NoCs, are still susceptible to deadlock arising from protocol interactions in the NIs. The IP blocks create *message dependencies* between buffers in the NIs that, when transferred to the router network, can lead to *message-dependent deadlocks* [12].

The SoC comprises IP modules with two different types of ports: masters (initiators) and slaves (targets) [3]. Masters initiate *transactions* by issuing *requests*. One or more slaves receive and execute each transaction. Optionally, a transaction also includes a *response*, returning data, or an acknowledgement from the slave to the master. This transaction model subsumes both a distributed shared memory (DSM) and message passing (MP) communication paradigm. As we will see, this model of on-chip communication can lead to *four types* of message dependencies, *request-response*, *response-request*, *request-request*, and *response-response*, depending on the behavior of the IP modules.

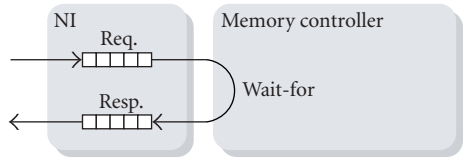


Figure 1: Request-response dependency at a memory.

These dependencies arise as a consequence of the IP modules' *desired* behavior. For example, the memory controller in Figure 1 is expected to respond to requests, and thus creates a request-response dependency.

Even when NoC and IP dependency graphs are cycle-free in isolation, put together they may still create cycles due to these dependencies. Traditional NoC architectures rely solely on request-response protocols, and consequently only have to address these dependencies. However, higher-level protocols are being adopted in emerging SoCs for cache coherency [13, 14], slave locking [14, 15], and peer-to-peer streaming [16]. These higher-level protocols introduce additional dependencies that must be addressed to provide deadlock-free operation.

The main contribution of this paper is an analysis of the message-dependent deadlocks that commonly used programming models and coherency schemes can cause network-based SoCs. We evaluate the possible solutions and show that many NoCs do not consider [6, 17–20] or only partially solve [21–23] the problem. These NoCs can only guarantee deadlock-free operation for a limited set of protocols. Furthermore, we show how the Æthereal [3] and FAUST [24] NoC, both employing *credit-based end-to-end flow control* [25], avoid message-dependent deadlock irrespective of the communication protocols used. Alternative approaches, for example multiple networks, have not been shown for NoCs.

For a number of designs, we quantify the area cost for the two most economical solutions, *strict ordering* and end-to-end flow control. We show that the latter, which avoids deadlock for *all* protocols, has an area and power costs of 4% and 6%, respectively, of a typical Æthereal NoC instance.

Related work is introduced in Section 2. The architectural platform is presented in Section 3. Next, the problem is introduced in Section 4 and the different message dependencies are covered in depth in Section 5. Solutions used in NoCs are presented in Section 6. An evaluation of the different solutions is given in Section 7 together with a quantitative analysis of the two prominent techniques, strict ordering, and end-to-end flow control, applied to Æthereal. Finally, Section 8 concludes the paper and presents directions for future work.

2. RELATED WORK

Key research problems in NoC design are presented in [8]. The authors stress the importance of deadlock-free operation but identify it only as a routing problem, not considering the protocol interactions between the IPs and the NoC at the network endpoints.

Deadlock recovery is a popular resort in parallel computers [12] and is used in the Proteo [26] NoC that drops packets on overflow. The majority of NoCs [3, 6, 17–24], however, avoid deadlock, as deadlock detection and recovery mechanisms are expensive [8] and complicate the provision of guarantees. Deadlock avoidance is also the focus of this paper.

An NI that offers high-level services is presented in [3]. End-to-end flow control, important as we will see in Section 6.2, is part of the basic functionality offered by the design and the added bandwidth for an MPEG-2 decoder is evaluated. However, as with [20, 24] that also use end-to-end flow control, message-dependent deadlock is not discussed.

Many NoCs [21–23] break request-response dependencies by introducing separate physical networks for the two message types. Virtual, instead of physical, networks are used in [27, 28] to avoid deadlock in a higher-order configuration protocol and a forwarding multicast protocol, respectively. All the solutions are protocol-specific and none address the dependencies that can arise when IPs have both master and slave ports.

The possibility of considering message types in the topology synthesis is explored in [29]. The work presents a methodology that tailors the NoC to a particular application behavior while taking message-dependent deadlock into account. In contrast to what we advocate in this work, the NoC architecture is inherently coupled to the application and assumes that the NoC can be redesigned if the application or its binding to the NoC should change.

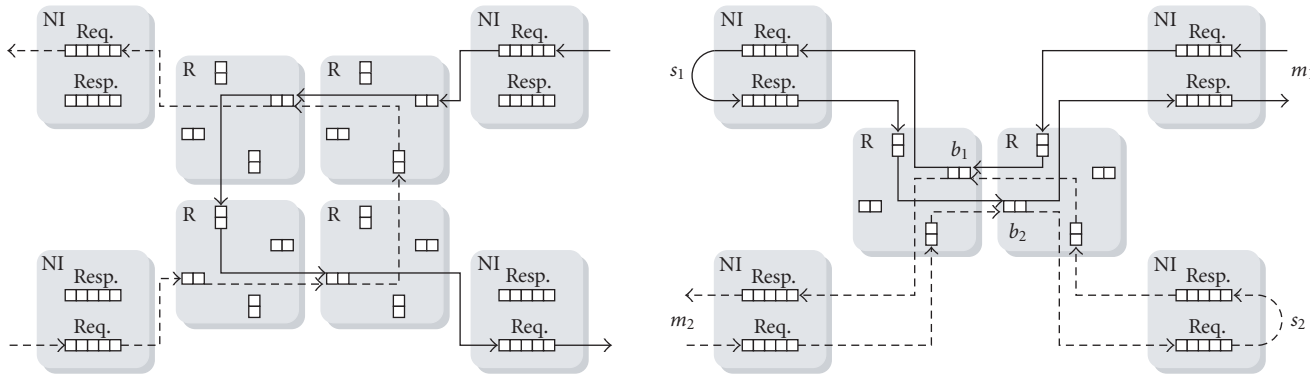
A comprehensive survey on methods for handling message-dependent deadlocks in parallel computer systems is given in [12]. In contrast with the computer networks and multiprocessor environments studied in the work, NoC storage and computation resources are relatively more restricted, and the protocol stack is entirely implemented in hardware. Hence, design constraints and optimization goals are fundamentally different.

In this work, we present the implications regarding deadlock that arise in a network-based SoC due to the *interactions between the NoC and the IP modules*. Furthermore, we evaluate the area and power cost of a NoC architecture, applied to a number of representative SoCs, that avoid *all* potential message-dependent deadlocks through the use of credit-based end-to-end flow control.

3. ARCHITECTURAL PLATFORM

We assume that NoCs comprise two components: routers (R) and network interfaces (NIs), as depicted in Figure 2. The routers can be randomly connected amongst themselves and to the NIs, that is, there are no topology constraints, although the routing is assumed to be deadlock-free. The routers transport packets of data from one NI to another.

The NIs enable end-to-end services [3] to the IP modules and are keys in decoupling computation from communication [6, 7]. The NI allows the designer to simplify communication issues to local point-to-point transactions at IP module boundaries, using protocols natural to the IP [7], for example, AXI [15] or OCP.



(a) Routing-dependent deadlock

(b) Message-dependent deadlock

Figure 2: Different levels of deadlock.

Master and slave IP ports are connected to slave and master NI ports, respectively. The term *connection* is used throughout this paper to denote a unidirectional peer-to-peer interconnection between a master and a slave, either carrying requests from master to slave, or responses from slave to master, but not both. In Section 6.4, we return to the differences with the *looped containers* [18] of Nostrum.

Throughout this paper, data integrity, lossless data delivery, and in-order data delivery are assumed to be services inherent to the router network. Freedom of reassembly deadlock and resequence deadlock is thus guaranteed [30]. In Proteo [26] that is lossy, and Nostrum [18] that uses adaptive (hot-potato) routing, additional care must be taken to recover from and avoid deadlock, respectively.

4. PROBLEM DESCRIPTION

In this paper, we assume freedom of *routing-dependent deadlock* [12], depicted in Figure 2(a). All NoCs we are aware of solve this kind of deadlock, mostly by assuring acyclic resource dependencies in the router network [3, 6, 17–24]. A dependency cycle involving *only* the routers, as shown in the figure, can hence not occur.

Although acyclic routing algorithms assert that no deadlock occurs, they do so under the *consumption assumption*. This assures that delivered messages are, in a finite time, sunk by the NIs. By induction, because the network dependencies are acyclic, all buffers are eventually emptied.

Unconditional consumption requires that delivery of one message is not coupled to the injection or reception of another message [12]. Regardless of whether the DSM or MP communication paradigm is used, IP modules often violate this assumption as a result of their *normal desirable behavior*, for example, a slave module that responds to incoming requests and thereby introduces a request-response dependency. Together with the dependencies of the network, the message dependencies can again cause dependency cycles and introduce message-dependent deadlock, as shown in Figure 2(b).

Taxing the IP modules with the responsibility of correctness (e.g., by employing end-to-end flow control on the application level) is not desired as it necessitates modification of existing IPs [31] and frustrates reuse [7]. Therefore, the onus of consumption is placed on the NIs. In the following sections, we show how the IP behavior determines the type of dependencies that arise, and in Section 6 we present solutions that guarantee consumption in their presence.

Besides the router-dependent and message-dependent deadlocks, we also address *application deadlock* [16]. This third level of deadlock, involving the IPs only, is as important as the two lower levels. It is, however, independent of the behavior of the NoC and is out of the scope of this paper.

5. MESSAGE DEPENDENCIES

We adopt the terminology used in [12]. A *message dependency chain* represents a partially ordered list of message types m_1 through m_n, where $m_i \prec m_j$ if and only if m_j can be generated by a node receiving m_i. The *chain length* denotes the number of types n in the chain. We refer to a protocol with such a message dependency chain as a n-way protocol. A message of type m_n is said to be *terminating* as it does not introduce any new messages.

5.1. Request-response dependency

A dependency that is frequently occurring in contemporary SoCs is the *request-response dependency*. As we have seen, this dependency arises in a slave module, such as a memory controller, that awaits a request and upon reception processes the request and sends a response. The protocol is clearly two-way with a message dependency chain *request* $\prec$ *response*.

The coupling between reception of request and generation of responses introduces a dependency between the request and response buffers in the NI, as depicted in Figure 2(b). Transferred to the network, this dependency can cause deadlock. In the figure, two master and slave pairs communicate via two shared input buffered routers. The two connections between m_1 and s_1 are drawn with continuous

lines and the connections of m_2 and s_2 with dashed lines. Note that dimension-ordered routing is used and that the network is clearly acyclic. Moreover, the individual master and slave pairs do not introduce cycles as there is only a message dependency on the slave side. However, a dependency cycle is formed over the two slave modules. Responses from s_1 enter the network, turn east, and end up in b_2. This buffer is shared by responses destined for m_1 and requests going to s_2. From b_2, the dependencies continue through the slave s_2, and the shared buffer b_1, back to s_1, closing the cycle. As a result, a deadlocked situation, where none of the involved connections make progress, can occur.

As we will see in Section 6.3, one way to resolve the dependencies of b_1 and b_2 is to use separate request and response networks, or at least separate buffer classes.

5.2. Response-request dependency

In contrast to what most NoC designs suggest, many protocols create more than just request-response dependencies. For example, when a master reacts on the response from a slave by sending an additional request, it creates a *response-request dependency*. Consider for example an implementation of atomic access through read-modify-write (RMW) [14, 15]. A read request is issued by the master which acquires exclusive ownership and receives a response from the slave. Finally, the master issues a write request which upon completion releases the lock. This protocol creates a message chain $read \prec response \prec write$.

Examples of more specialized protocols that have response-request dependencies are given in [27, 28, 32]. In these works, interconnections and multicast groups are established through a three-way resource reservation protocol: (1) a master sends a setup request, (2) the slave responds with a positive or negative acknowledgement, and in the latter case (3) the master restores the reservations done by sending a tear-down. The message dependency chain thus comprises three types: $setup \prec ack/nack \prec teardown$.

5.3. Request-request and response-response dependencies

The aforementioned dependencies involve only dedicated master and slave modules. This is also an assumption made by most existing solutions to message-dependent deadlock in NoCs [21–23, 27]. With the introduction of IP modules with both master and slave ports, for example, a processor or direct memory access (DMA) engine, two additional dependencies may arise: *request-request* and *response-response*.

Request-request dependencies, as depicted in Figure 3, are created when reception of a request on the slave side is coupled to the generation of a request on the master side. This occurs when IP modules process a certain input that is sent to them by the preceding module and then write their output to the succeeding module, as done in *peer-to-peer streaming* and in protocols that, in the interest of performance, use *forwarding* [12, 33].

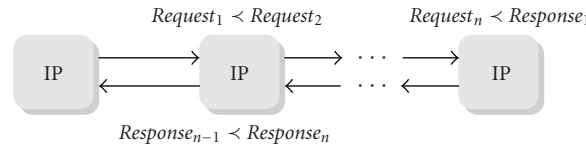


Figure 3: Message forwarding.

5.3.1. Forwarding

In a forwarding protocol, an initial request passes through a number of intermediate IPs, generating new requests until the final destination is reached. Potentially, a response is travelling in the other direction, creating response-response dependencies on the way back. Two prominent examples of forwarding protocols are *cache coherency protocols* [33] and *collective communication* [34], such as multicast and narrowcast [3, 28].

Cache coherency in network-based SoCs is typically implemented using a directory-based protocol as the medium does not lend itself to snooping [13, 14]. These protocols, in general, do not adhere to strict request-response protocols, as they strive to reduce the number of network transactions generated per memory operation [33]. Both *reply forwarding* and *intervention forwarding* manifest request-request dependencies, and the latter introduces also response-response dependencies.

Multicast and narrowcast are used in NoCs to implement DSM on a single interconnection [3], and in parallel systems also for cache invalidation, acknowledgement collection, and synchronization [34]. These higher-order interconnections give rise to both request-request and response-response dependencies when implemented using forwarding [28]. The latter is used to avoid sending a unicast message for every destination, which causes congestion at the source [35].

5.3.2. Streaming

A streaming protocol, where data is pushed from producer to consumer, is beneficial in *dataflow applications* [16, 36] comprising a chain of modules, such as the video pixel processing pipeline [37] depicted in Figure 4.

The advantage of pushing (writing) data instead of pulling it from the producer is that it greatly reduces the impact of network latency. When pulling, as suggested by [38], then first a read request is sent whereafter the producer responds with the data, thereby doubling the latency by traversing the network twice. Note that the latter approach, where every IP reads and writes its input and output, respectively, reduces the protocol to strict request-response but has several drawbacks, further discussed in Section 6.3. An example of a SoC employing peer-to-peer streaming is presented in [39] where a commercially available SoC for picture improvement is extended with a NoC.

The peer-to-peer streaming protocol where IPs write their output to the next module, illustrated in Figure 5, has a message chain that is built only from (forwarded) requests:

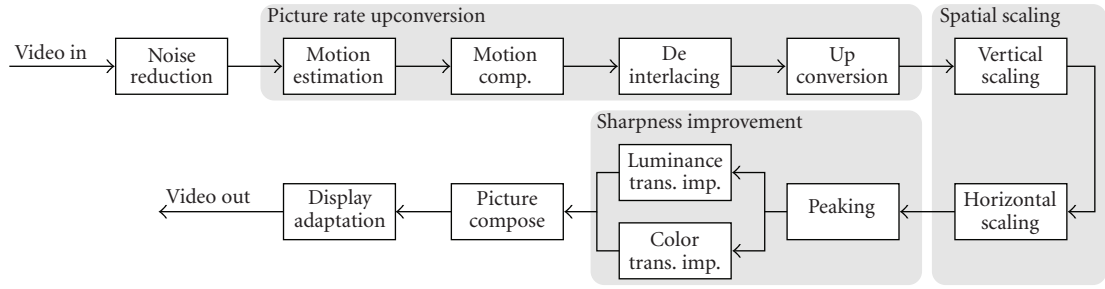


Figure 4: Video pixel processing application.

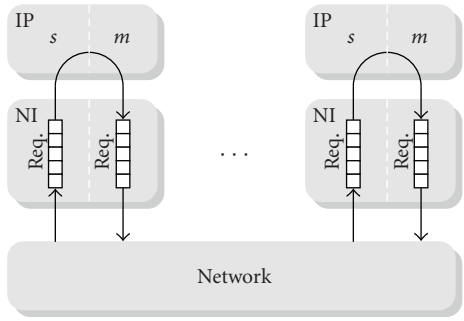


Figure 5: Dependencies created by peer-to-peer streaming.

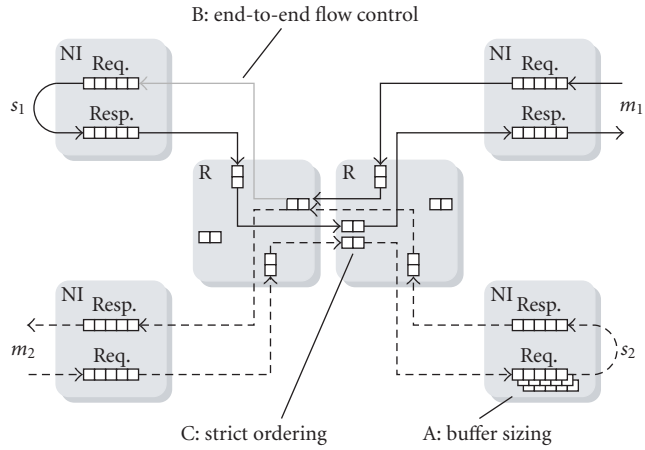


Figure 6: Various solutions.

$request_1 \prec \cdots \prec request_n$, where n is the number of modules in the processing chain. Consider for example the pipeline in Figure 4 that has 12 different types of request messages if all communications are implemented by peer-to-peer streaming.

6. SOLUTIONS

To provide a deadlock-free NoC, the consumption assumption must be fulfilled. As a first requirement, messages must be separated into different NI buffers based on their type. Having a separate NI buffer per message type is a *necessary but not a sufficient condition* to avoid deadlocks [12]. Message-dependencies together with dependencies in the router network can still introduce cycles.

As already outlined, the avoidance-based solutions to this problem fall within two categories. First, the consumption assumption can be implemented by designing the NIs such that NI buffers are guaranteed to consume all messages sent to them, regardless of the IPs. Buffer sizing (Section 6.1) and end-to-end flow control (Section 6.2) are instances of this technique. Alternatively, the NoC must guarantee that messages of one type never preclude the advances of its subordinate types indefinitely. Thereby, messages of the terminating type (guaranteed to sink upon arrival) reach their destination and its dominant types can follow suit. This technique is referred to as strict ordering (Section 6.3), and virtual circuits (Section 6.4) is a special case.

6.1. Buffer sizing

A first way to solve the deadlock problem is to ensure enough space by (over-)sizing the buffers. This requires a generous storage budget, determined by the maximum bounds on packet size and the number of outstanding transactions. The concept is shown in Figure 6 that revisits the case of a request-response protocol.

While extensively used in parallel computers [12], this method is prohibitively expensive in NoCs and is not used in any known architecture.

6.2. End-to-end flow control

Instead of adapting the buffer size to the maximum requirements, end-to-end flow control does the other way around: it assures that no more is ever injected than what can be consumed. This approach, end-to-end flow control, is used in the Æthereal and FAUST NoC. As illustrated in Figure 6, it removes a dependency edge from the network to the NI.

The simplest form of end-to-end flow control is the so-called *local flow control* [30], assuring that enough space is available to fit the response before sending the request. This local check solves response-response and response-request dependencies.

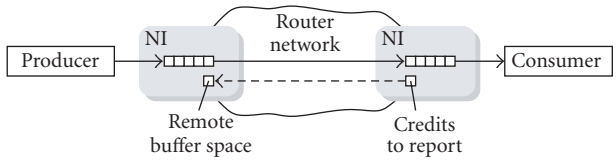


Figure 7: Credit-based end-to-end flow control per connection.

Requests-request and request-response dependencies are caused by transactions initiated by remote parties, and thus require end-to-end flow control. As buffer space is the critical resource, a windowing mechanism must be used. An example of such a mechanism is credit-based flow control, as illustrated in Figure 7. A rate-based mechanism, such as the one used in [40], is insufficient as it does not bound buffer usage.

Just as buffer sizing, end-to-end flow control solves *all* potential message dependencies. It does so without placing any restrictions on the amount of sharing in the router network. Furthermore, routers need not know message types or the number of connections, and can thereby be simplified in complexity and optimized for other important or otherwise critical features [12]. However, credit-based end-to-end flow control carries three major downsides.

First, it requires extra buffering to hide the round-trip latency of the credits. The amount of buffering is determined by the performance requirements [41] and it is evaluated in Section 7.1.

Second, communication of credits consumes bandwidth and hence power. The closed-loop nature requires state to be communicated between the producer and consumer NI. The additional bandwidth, quantified in Section 7.1, can be reduced with 20% by piggybacking credits on the data packets [3].

Third, it requires dedicated NI buffers per connection. Alternatively, if many sources share a common destination buffer they need collective knowledge of the destination and each other, something that cannot be implemented in a cost-efficient way.

6.3. Strict ordering

Another way of assuring freedom of message deadlock is by ordering network resources. This is done by introducing logically independent networks, physical or virtual, for each message type. Arteris [21], STbus [22], and SonicsMX [23] fit in the first category by having two physical networks for requests and responses, respectively. The methods used to break request-response dependencies in [27, 28] fit in the latter category as they both use one buffer class per message type. This approach is illustrated in Figure 6 where a buffer is added to break the dependency cycle.

A major drawback of the strict ordering is that buffers cannot be shared between the different message classes, increasing the amount of buffering required. The partitioning into logical networks leads to inefficient utilization of network resources [33] and increased congestion due to unbalance [12]. These effects increase with the number of networks required. In [22], the authors argue that the size of the request and response networks can be made different. The size is however static, and use-cases (modes) with different traffic characteristics magnify the problem.

Having virtual instead of physical networks mitigates the aforementioned problem. However, the router complexity increases as it must forward messages considering message type [12].

The major limitation with strict ordering is the inherent coupling between the NoC and the IP modules. A NoC with n logical networks can only be used by IP modules employing protocols with n or fewer message types. In multiprocessor designs, like the Alpha 21364 [42], this entanglement of concerns is not an issue. The router network is tailored to the protocol with seven virtual networks, one for each message type. For a NoC design, however, the coupling between IPs and the NoC architecture severely limits the reusability. Consider for example the implementation of a forwarding protocol [28] where the number of buffers determines the maximum number of multicast groups.

Higher-order protocols require either a redesign of the NoC or a reduction of the protocol to n ways. IP modules using peer-to-peer streaming communication hence cannot use the NoCs in [21–23] as they only support two-way protocols. The protocol has to be reduced to pure request-response and communication must go via memory. This adds complexity, requires additional bandwidth, introduces latency, increases congestion, and consumes more power.

6.4. Virtual circuits

Virtual circuits represent the extreme case of strict ordering as every connection has its own logical network. This way of implementing unconditional delivery is found in the guaranteed service networks of Æthereal [3], MANGO [43], and Nostrum [18]. The implementations differ, but all rely on predetermined spatial and/or temporal multiplexing.

The deadlock freedom comes at a price of exclusively reserved resources coupled with decreased utilization. Furthermore, in all three NoCs the maximum number of circuits supported by a router and NI is decided at design time. For all three NoCs, the number of buffers in the NI sets an upper bound for the number of circuits. The router is limited by the number of virtual channels (buffers) in MANGO, by the slot table size in Æthereal and by the number of temporally disjoint networks in Nostrum.

The lack of resource sharing and potentially low utilization is the major drawback of all three implementations. However, Nostrum is more severely limited than the other two, due to the *looped containers* [18] that do a round trip and reserve an equal amount of resources in both directions. As the progress guarantee requires that no circuit carries more than one message type, requests and responses must use separate circuits. A circuit may therefore only carry messages from master to slave or slave to master, not both. Thereby, Nostrum is, if no additional measures are taken, limited to maximally 50% utilization as only the forward

Table 1: Avoidance techniques used in NoCs.

NoC	Technique	2-way	n-way
aSOC	—	–	–
MANGO BE	—	–	–
Nostrum BE	—	–	–
×pipes	—	–	–
Arteris	Strict ordering	+	–
SonicsMX	Strict ordering	+	–
STbus	Strict ordering	+	–
MANGO GS	Virtual circuits	+	+
Nostrum GS	Virtual circuits	+	+
Æthereal GS	Virtual circuits	+	+
SPIN	End-to-end flow control	–	–
FAUST	End-to-end flow control	+	+
Æthereal BE	End-to-end flow control	+	+

path may carry messages. An alternative is to enforce a maximum number of outstanding transactions and a maximum transaction size and then size the buffers accordingly, as discussed in Section 6.1.

7. EVALUATION

As seen in Table 1, the best-effort network in MANGO and Nostrum, together with aSOC [17] and ×pipes [19], do not address message dependencies at all, leaving these networks susceptible to deadlock (livelock in the case of Nostrum). Hence, not even a two-way protocol can be safely implemented on these architectures without further measures.

Arteris, SonicsMX, and STbus all have separate request and response networks, which allows them to handle two-way protocols without deadlock. However, peer-to-peer streaming protocols or forwarding multicast cannot be used by the IP modules unless the NoCs are extended with additional logical networks. The pipeline in Figure 4, for example, requires ten more networks. Even then, the maximal pipeline length is still limited by the architecture. Furthermore, if one IP fails to consume its messages it can bring the entire network to a stall.

The guaranteed-service network in Æthereal, MANGO, and Nostrum all avoid message-dependent deadlocks, but do so at the price of (1) reduced resource sharing, and (2) a fixed number of connections supported by the router and NI architecture.

SPIN [20], FAUST, [24] and the best-effort network in Æthereal all employ credit-based end-to-end flow control. However, only the latter two fulfil the consumption assumption as SPIN issues more credits than the capacity of the receiving buffer. The additional credits are introduced to reduce latency, and the only consequence is said to be an increased possibility of contention in the network. However, consumption can no longer be guaranteed making the system susceptible to message-dependent deadlock. In FAUST and Æthereal, the consumption assumption is fulfilled and no message-dependency chain can introduce deadlock. The router architecture is oblivious to message types and the number of connections, but the latter is instead limited by the number of buffers in the NIs.

Table 2: Buffer cost (words).

	MPEG	$s_1m_1p_2$	$s_1m_2p_2$	$s_8m_1p_2$	$s_8m_2p_2$
Total	242	339	615	450	801
Per conn.	5.8	3.2	3.0	3.5	3.3

7.1. Cost analysis

In this section, we evaluate the cost associated with the two most resource-efficient solutions, namely strict ordering and end-to-end flow control. This is done for five different use-cases. The *MPEG* use-case is an MPEG codec SoC with 16 IP modules, tied together by 42 connections. The remaining four use-cases are internal video processing designs, all having a hot spot around a limited set of IPs (external memories) and 100 to 250 connections. These connections deliver a total bandwidth of 1-2 Gbyte/s to 75 ports distributed across 25 IP modules.

For each use-case, a NoC is dimensioned using the UMARS algorithm [44]. Given the performance requirements, NI buffer sizes are then calculated in two individual parts: (1) the amount required to decouple the IP and NI consumption and production without introducing stalls, and (2) the number of words that must be added to hide the round-trip latency of flow control [45]. The contribution of the latter is presented in Table 2.

As seen in Table 2, the average cost is merely three to six words per connection. The addition to the total NoC area is shown in Figure 8. The silicon area requirements are based on the model presented in [46], for a 0.13 μm CMOS process with full-custom FIFOs. The added NoC area is below 4% for all the applications. The mean value is 3.2%. Thus, in a network-based SoC, such as the one presented in [39], the area cost of end-to-end flow control is no more than 0.2% of the whole SoC.

To put the area cost of end-to-end flow control in contrast with strict ordering, we calculate an approximate cost of such an implementation. This is done by introducing an additional best-effort router network, identical to the one in place, thus having one network for requests and one for responses. Although we have an approximation, the results in Figure 8 suggest that the two methods are comparable in cost. The MPEG and $s_8m_2p_2$ designs have a more evenly distributed communication and less NIs per router than the other designs. As a result, close to 20% of the area is attributable to the routers in these two cases, which affects the cost of strict ordering negatively. The average area cost for strict ordering is slightly less than 3.9% of the NoC, only negligibly different from what is achieved with end-to-end flow control.

Note that we add only one router network for the comparison as all the use-cases employ a strict request-response protocol. With the introduction of higher-order

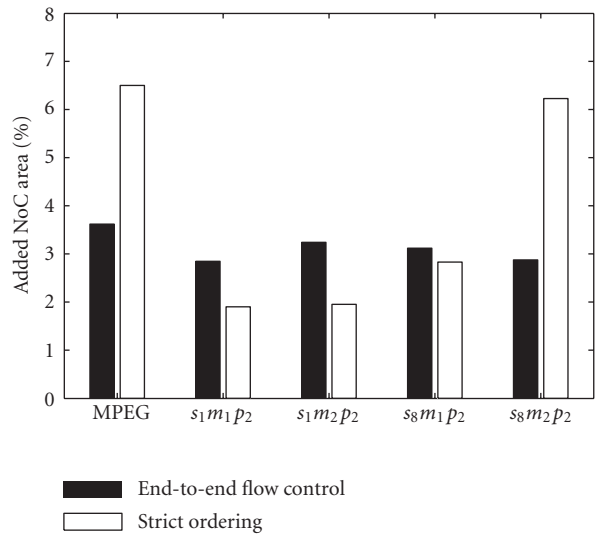


Figure 8: Comparison of added network area.

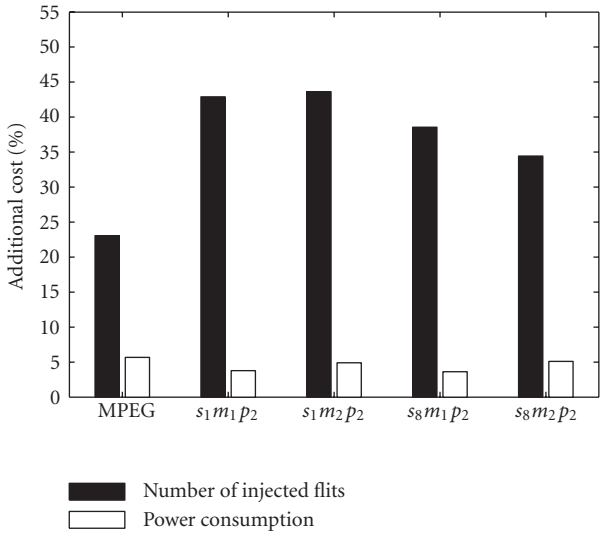


Figure 9: Additional traffic and power consumption.

protocols, the cost of end-to-end flow control remains constant, whereas the cost of strict ordering *increases linearly* with the number of logical networks (protocol stages). This is under the assumption that all network components are designed to handle all different message types. As proposed in [29], it is, for a given application, possible to reduce the cost by only introducing the additional buffer classes where strictly needed.

To asses the cost of the traffic introduced by the end-to-end flow control, we simulate each design 3×10^6 clock cycles in a flit-accurate SystemC simulator of the Æthereal NoC, using traffic generators to mimic core behavior. Figure 9 shows the additional cost in terms of injected flits and power consumption.

The additional amount of injected flits ranges from 23% up to 44%. The MPEG design has an average bandwidth (76 Mbyte/s) three times higher than the other designs, which results in less flits carrying only credits. A higher bandwidth (and larger burst size) increases the opportunities for piggybacking credits on data-carrying packets [3]. Furthermore, it also leads to a more bursty delivery of credits with more credits per packet. As a result, buffers grow (see Table 2), but less credit-carrying flits are injected.

As more flits are injected and routed through the network, also the power consumption increases. The contribution added by the credit-carrying flits is depicted in Figure 9. Note that the power estimation, calculated according to the model in [47], covers only the router network (without the NIs). In the reference case with no flow control, the flits that carry only credits and no data are treated as empty. Despite the amount of flits, the additional cost in power consumption is consistently below 6%, with an average of 4.6%.

8. CONCLUSION AND FUTURE WORK

In this paper we analyze *message-dependent deadlock*, arising due to protocol interactions between the NoC and the IP modules. We compare the possible solutions and show that deadlock avoidance, in the presence of higher-level protocols, for example, cache coherency, slave locking and peer-to-peer streaming, poses a serious challenge for many current NoC architectures.

Furthermore, we show how a NoC, such as the Æthereal and FAUST NoCs, employing credit-based end-to-end flow control, provides robust communication services for *all* potential communication protocols used. We show that the associated area and power cost represent 4% and 6%, respectively, of a typical Æthereal NoC instance.

Future work includes a more in-depth analysis of the costs associated with the various solutions in the presence of streaming peer-to-peer protocols.

REFERENCES

[1] L. Benini and G. de Micheli, "Networks on chips: a new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 684–689, Las Vegas, Nev, USA, June 2001.

[3] A. Rădulescu, J. Dielissen, S. Gonzàlez Pestana, et al., "An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 4–17, 2005.

[4] M. Sgroi, M. Sheets, A. Mihal, et al., "Addressing the system-on-a-chip interconnect woes through communication-based design," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 667–672, Las Vegas, Nev, USA, June 2001.

[5] M. Coppola, S. Curaba, M. D. Grammatikakis, G. Maruccia, and F. Papariello, "OCCN: a network-on-chip modeling and simulation framework," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 3, pp. 174–179, Paris, France, February 2004.

[6] T. Bjerregaard, S. Mahadevan, R. G. Olsen, and J. Sparsø, "An OCP compliant network adapter for GALS-based SoC design using the MANGO network-on-chip," in *Proceedings of International Symposium on System-on-Chip (SOC '05)*, pp. 171–174, Tampere, Finland, November 2005.

[7] D. Wingard, "Socket-based design using decoupled interconnects," in *Interconnect-Centric Design for SoC and NoC*, J. Nurmi, H. Tenhunen, J. Isoaho, and A. Jantsch, Eds., Kluwer, Dordrecht, The Netherlands, 2004.

[8] U. Y. Ogras, J. Hu, and R. Marculescu, "Key research problems in NoC design: a holistic perspective," in *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES, ISSS '05)*, pp. 69–74, Jersey City, NJ, USA, September 2005.

[9] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 729–738, 2000.

[10] J. Hu and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '03)*, pp. 688–693, Munich, Germany, March 2003.

[11] J. Hu and R. Marculescu, "DyAD—smart routing for networks-on-chip," in *Proceedings of the 41st Design Automation Conference (DAC '04)*, pp. 260–263, San Diego, Calif, USA, June 2004.

[12] Y. H. Song and T. M. Pinkston, "A progressive approach to handling message-dependent deadlock in parallel computer systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 3, pp. 259–275, 2003.

[13] T. T. Ye, L. Benini, and G. de Micheli, "Packetized on-chip interconnect communication analysis for MPSoC," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '03)*, pp. 344–349, Munich, Germany, March 2003.

[14] F. Pétrot, A. Greiner, and P. Gomez, "On cache coherency and memory consistency issues in NoC based shared memory multiprocessor SoC architectures," in *Proceedings of the 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD '06)*, pp. 53–60, Dubrovnik, Croatia, August-September 2006.

[15] *AMBA AXI Protocol Specification*, ARM, June 2003.

[16] M. Bekooij, R. Hoes, O. Moreira, et al., "Dataflow analysis for real-time embedded multiprocessor system design," in *Dynamic and Robust Streaming in and between Connected Consumer-Electronics Devices*, P. van der Stok, Ed., Kluwer, Dordrecht, The Netherlands, 2005.

[17] E. Beigné, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An asynchronous NOC architecture providing low latency service and its multi-level design framework," in *Proceedings of International Symposium on Asynchronous Circuits and Systems (ASYNC '05)*, pp. 54–63, New York, NY, USA, March 2005.

[18] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 2, pp. 890–895, Paris, France, February 2004.

[19] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, and G. de Micheli, "×pipes lite: a synthesis oriented design library for networks on chips," in *Proceedings of Design, Automation and Test in Europe (DATE '05)*, vol. 2, pp. 1188–1193, Munich, Germany, March 2005.

[20] P. Guerrier, "Un réseau d'interconnexion pour systémes intégrés," Ph.D. dissertation, Universite Paris VI, Paris, France, 2000.

[21] Arteris, "A comparison of network-on-chip and busses," White paper, 2005.

[22] S. Murali and G. de Micheli, "An application-specific design methodology for STbus crossbar generation," in *Proceedings of Design, Automation and Test in Europe (DATE '05)*, vol. 2, pp. 1176–1181, Munich, Germany, March 2005.

[23] *SonicsMX Datasheet*, Sonics, 2005, http://www.sonicsinc.com/.

[24] Y. Durand, C. Bernard, and D. Lattard, "FAUST: on-chip distributed architecture for a 4g baseband modem SoC," in *Proceedings of IP Based SoC Design Conference and Exhibition (IPSOC '05)*, Grenoble, France, December 2005.

[25] A. Tanenbaum, *Computer Networks*, Prentice-Hall, Upper Saddle River, NJ, USA, 1996.

[26] I. Saastamoinen, M. Alho, and J. Nurmi, "Buffer implementation for Proteo networks-on-chip," in *Proceedings of International Symposium on Circuits and Systems (ISCAS '03)*, pp. 113–116, Bangkok, Thailand, May 2003.

[27] B. Gebremichael, F. Vaandrager, Z. Miaomiao, K. Goossens, E. Rijpkema, and A. Rădulescu, "Deadlock prevention in the Æthereal protocol," in *Proceedings of the 13th IFIP WG 10.5 Advanced Research Working Conference Correct Hardware Design and Verification Methods (CHARME '05)*, pp. 345–348, Saarbrücken, Germany, October 2005.

[28] Z. Lu, B. Yin, and A. Jantsch, "Connection-oriented multicasting in wormhole-switched networks on chip," in *Proceedings of IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, pp. 205–210, Karlsruhe, Germany, March 2006.

[29] S. Murali, P. Meloni, F. Angiolini, et al., "Designing message-dependent deadlock free networks on chips for application-specific systems on chips," in *Proceedings of IFIP International Conference on Very Large Scale Integration*, pp. 158–163, Nice, France, October 2006.

[30] M. Gerla and L. Kleinrock, "Flow control: a comparative survey," *IEEE Transactions on Communications Systems*, vol. 28, no. 4, pp. 553–574, 1980.

[31] P. Bhojwani and R. Mahapatra, "Interfacing cores with on-chip packet-switched networks," in *Proceedings of 16th International Conference on VLSI Design*, pp. 382–387, Las Vegas, Nev, USA, June 2003.

[32] K. Goossens, J. Dielissen, and A. Rădulescu, "Æthereal network on chip: concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, 2005.

[33] D. E. Culler, J. P. Singh, and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann Publishers, San Francisco, Calif, USA, 1999.

[34] R. Sivaram, R. Kesavan, D. K. Panda, and C. B. Stunkel, "Where to provide support for efficient multicasting in irregular networks: network interface or switch?" in *Proceedings of International Conference on Parallel Processing (ICPP '98)*, pp. 452–459, Minneapolis, Minn, USA, August 1998.

[35] R. V. Boppana, S. Chalasani, and C. S. Raghavendra, "Resource deadlocks and performance of wormhole multicast routing algorithms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 6, pp. 535–549, 1998.

[36] G. Kahn, *Information Processing*, North-Holland, New York, NY, USA, 1974, ch. The Semantics of a Simple Language for Parallel Processing.

[37] O. P. Gangwal, J. Janssen, S. Rathnam, E. Bellers, and M. Duranton, "Understanding video pixel processing applications for flexible implementations," in *Proceedings of Euromicro Symposium on Digital System Design*, pp. 392–401, Belek-Antalya, Turkey, September 2003.

[38] H. Nikolov, T. Stefanov, and E. Deprettere, "Multi-processor system design with ESPAM," in *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis (CODES, ISSS '06)*, pp. 211–216, Salzburg, Austria, September-October 2006.

[39] F. Steenhof, H. Duque, B. Nilsson, K. Goossens, and R. Peset Llopis, "Networks on chips for high-end consumer-electronics TV system architectures," in *Proceedings of Design, Automation and Test in Europe (DATE '06)*, vol. 2, pp. 1–6, Munich, Germany, March 2006.

[40] S. Murali, L. Benini, and G. de Micheli, "Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '05)*, vol. 1, pp. 27–32, Shanghai, China, January 2005.

[41] O. P. Gangwal, A. Rădulescu, K. Goossens, S. Gonzàlez Pestana, and E. Rijpkema, "Building predictable systems on chip: an analysis of guaranteed communication in the Æthereal network on chip," in *Dynamic and Robust Streaming in and between Connected Consumer-Electronics Devices*, Kluwer, Norwell, Mass, USA, 2005.

[42] S. S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb, "The Alpha 21364 network architecture," *IEEE Micro*, vol. 22, no. 1, pp. 26–35, 2002.

[43] T. Bjerregaard and J. Sparsø, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proceedings of Design, Automation and Test in Europe (DATE '05)*, vol. 2, pp. 1226–1231, Munich, Germany, March 2005.

[44] A. Hansson, K. Goossens, and A. Rădulescu, "A unified approach to constrained mapping and routing on network-on-chip architectures," in *Proceedings of International Conference on Hardware/Software Codesign and System Synthesis (CODES, ISSS '05)*, pp. 75–80, Jersey City, NJ, USA, September 2005.

[45] M. Coenen, S. Murali, A. Rădulescu, K. Goossens, and G. de Micheli, "A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control," in *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis (CODES, ISSS '06)*, pp. 130–135, Seoul, Korea, October 2006.

[46] S. Gonzàlez Pestana, E. Rijpkema, A. Rădulescu, K. Goossens, and O. P. Gangwal, "Cost-performance trade-offs in networks on chip: a simulation-based approach," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 2, pp. 764–769, Paris, France, February 2004.

[47] J. Dielissen, A. Rădulescu, and K. Goossens, "Power measurements and analysis of a network-on chip," Tech. Rep. NL-TN-2005-0282, Philips Research Laboratories, Eindhoven, The Netherlands, 2005.

Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 68432, 16 pages
doi:10.1155/2007/68432

Research Article

A Unified Approach to Mapping and Routing on a Network-on-Chip for Both Best-Effort and Guaranteed Service Traffic

Andreas Hansson,[1] Kees Goossens,[2,3] and Andrei Rădulescu[3]

[1] *Department of Electrical Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands*
[2] *Computer Engineering, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2600 GA Delft, The Netherlands*
[3] *SOC Architectures and Infrastructure, Research, NXP Semiconductors, 5656 AE Eindhoven, The Netherlands*

Received 15 October 2006; Accepted 4 March 2007

Recommended by Davide Bertozzi

One of the key steps in Network-on-Chip-based design is spatial mapping of cores and routing of the communication between those cores. Known solutions to the mapping and routing problems first map cores onto a topology and then route communication, using separate and possibly conflicting objective functions. In this paper, we present a unified single-objective algorithm, called Unified MApping, Routing, and Slot allocation (UMARS+). As the main contribution, we show how to couple path selection, mapping of cores, and channel time-slot allocation to minimize the network required to meet the constraints of the application. The time-complexity of UMARS+ is low and experimental results indicate a run-time only 20% higher than that of path selection alone. We apply the algorithm to an MPEG decoder System-on-Chip, reducing area by 33%, power dissipation by 35%, and worst-case latency by a factor four over a traditional waterfall approach.

1. INTRODUCTION

System(s)-on-Chip (SoC) grow in complexity with the advance of semiconductor technology enabling integration of dozens of cores on a chip. The continuously increasing number of cores calls for a new communication architecture as traditional architectures are inherently nonscalable, making communication a bottleneck [1, 2].

System architectures are shifting towards a more communication-centric methodology [2]. Growing SoC complexity makes *communication* subsystem design as important as *computation* subsystem design [3]. The communication infrastructure must efficiently accommodate the communication needs of the integrated computation and storage elements, for example, processors, coprocessors, DSPs, hardware accelerators, memory blocks, and I/O blocks.

Network(s)-on-Chip (NoC) have emerged as the design paradigm for design of scalable on-chip communication architectures, providing better structure and modularity than its predecessors [1, 2, 4, 5]. Although NoCs solve the interconnect scalability issues, SoC integration is still a problem. Even in a situation where the building blocks of the system are already designed and validated, much tedious work is traditionally required to validate the complete system.

To enable cores to be designed and validated independently, computation and communication must be decoupled [6]. Decoupling requires that the services cores use to communicate are well defined [7]. Furthermore, many cores also have inherent real-time performance requirements, such as minimum throughput or maximum latency, making time-related service guarantees essential [6]. An NoC delivering Quality-of-Service (QoS) guarantees, adhering to the non-functional (timing) requirements of the application, is key to enable independent design and validation of the SoC building blocks [5]. While this eases the task of the SoC integrator, additional constraints are placed on the NoC design.

Creating a NoC-based system requires efficient mapping of cores and distribution of NoC resources [8, 9]. Additionally, the resource allocation must fulfil the application constraints and guarantee deadlock freedom. Design choices include core port to network port binding, routing of communication between cores and allotment of network channel

capacity over time. As we will see in Section 6, these choices greatly affect the energy, area, and performance metrics of the system [8].

The main contribution of this work is a methodology extending spatial routing (path selection) to span also mapping and temporal routing[1] (time-slot allocation). This enables the aforementioned requirements to be formulated as path selection constraints and optimization goals. We present a unified algorithm, called Unified MApping, Routing and Slot allocation (UMARS+), that couples mapping, path selection and time-slot allocation, accommodating both guaranteed service and best-effort traffic. UMARS+ allows any NoC topology, guarantees deadlock-free routing, has a low complexity and yields a NoC with reduced area, power dissipation and communication latency.

As an example of the efficacy of the suggested methodology, we apply UMARS+ to an MPEG decoder SoC, reducing NoC area by 33%, power dissipation by 35%, and worst-case latency by a factor four over a traditional waterfall approach.

This paper is organized as follows. Related work is introduced in Section 2. The problem domain is described in Section 3 and formalized in Section 4. The UMARS+ algorithm, which solves the unified allocation problem under application constraints, is described in Section 5. Experimental results are shown in Section 6. Finally, Section 7 concludes this work and outlines directions for future research.

2. RELATED WORK

QoS routing objectives are discussed in [12, 13] and implications with common-practize load-balancing solutions are addressed in [14]. In addition to spatial, temporal characteristics are included in path selection in [15–17].

The problem of mapping cores onto NoC architectures and routing communication is addressed in [5, 8, 18–21]. In all works, the mapping and routing is functionally decomposed into modules on the basis of a flowchart, as depicted in Figure 1. The order in time in which processing is expected to take place is used in making the decomposition into modules. Each module has its separate constraints and optimization goals.

In [8, 18–21], mapping is perceived as a special case of the NP-complete *quadratic assignment problem* (QAP) [22]. Intuitively, the QAP can be described as the problem of assigning a set of cores to a set of locations with given distances between the locations and given weights of the communication between the cores. The goal is then to map the cores onto locations in such a way that the sum of the products between communication weights and distances is minimal. Due to the intractability of the QAP, all works use suboptimal approximation methods that iteratively evaluate potential solutions as indicated by the *iteration* arrow in Figure 1.

[1] The scope of this work is the TDM-based Æthereal NoC but the concept is more widely applicable [10, 11].

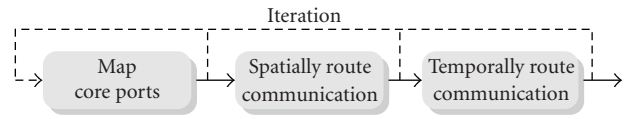


Figure 1: Mapping followed by routing with iteration.

The solution space traversal method used to solve the QAP in [8, 18] is a restricted *branch-and-bound* [22] algorithm. The algorithm maps cores onto a tile-based architecture, aiming to minimize energy while throughput constraints are satisfied. The latter is accomplished by making the distance parameter in the QAP model the energy consumed when transporting bits from one location to the other. Static *xy* routing is used in [18]. In [8] the algorithm is extended to route with the objective to balance network load.

In [19–21] a heuristic improvement method is used. An initial mapping is derived with objectives such as minimizing communication delay, area or power dissipation. This is succeeded by routing according to a predefined routing function. Routing and evaluation is repeated for pair-wise swaps of nodes in the topology, thereby exploring the design space in search for an efficient mapping. In [21] the algorithm is extended to integrate physical planning and the design space exploration is improved with robust tabu search.

In all presented iterative algorithms [8, 18–21], optimality refers to a cost function that evaluates the routes produced by the routing algorithm on a given mapping. Mapping decisions therefore anticipate and rely on the abilities of the routing algorithm to find optimal (and deadlock-free) routes between the locations decided by the mapping algorithm.

Known mapping and routing algorithms that incorporate QoS guarantees [15, 16, 21] either assume static communication flows [15, 16], where message injection times are known at design time, or do not derive any analytical bounds on throughput and latency [21].

TDM-based NoC architectures are presented in [5, 10, 11]. However, only [5] address the resource allocation on such architectures. A greedy noniterative algorithm first maps cores based on clustering whereafter communication is routed by static *xy* routing. Finally, temporal routing allocates TDM time-slots on the network channels such that QoS is guaranteed. This waterfall approach divides the allocation in three distinct phases with no coupling or feedback. While having a low run-time, this methodology pushes responsibility forward where it can be costly or even impossible to undo mistakes from earlier phases.

Aforementioned works address only regular topologies and use routing algorithms that are restricted to such topologies, for example, dimension-ordered routing [23], *north-last* [24], *odd-even* [25], and *DyAD* [26]. However, algorithms supporting irregular topologies are scarce. Benini [27] outline a design flow for application specific NoCs using a turn-prohibition algorithm that supports irregular topologies [28]. No details are however given as to how turns are selected or how path finding is done on the prohibited network.

This work, being an extension of [29], unifies the three resource allocation phases: spatial mapping of cores, spatial routing of communication, and the restricted form of temporal mapping that assigns time-slots to these routes. The hierarchically decomposed model, depicted in Figure 2, is fundamentally different from [5, 8, 18–21] in that mapping is no longer done *prior to* routing but instead *during* it.

The main goal of our methodology is to enable efficient application-specific NoCs for both best-effort and guaranteed service traffic, thus extending and elaborating on the methodology proposed in [29]. The key ideas and contributions of UMARS+ that allow us to achieve this goal are:

(i) mapping is transformed into a path selection problem,
(ii) temporal load (TDM slot tables) is included in the path selection objective function,
(iii) differentiation is made between best-effort and guaranteed service traffic,
(iv) deadlock is avoided by adaptive turn-prohibition, enabling efficient use of residual resources on any network topology.

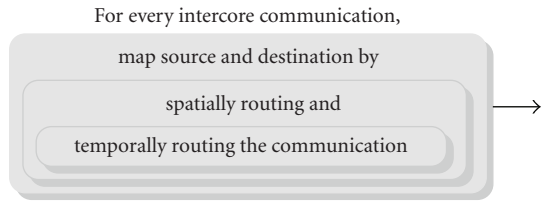


Figure 2: Mapping coupled with routing, hierarchically decomposed.

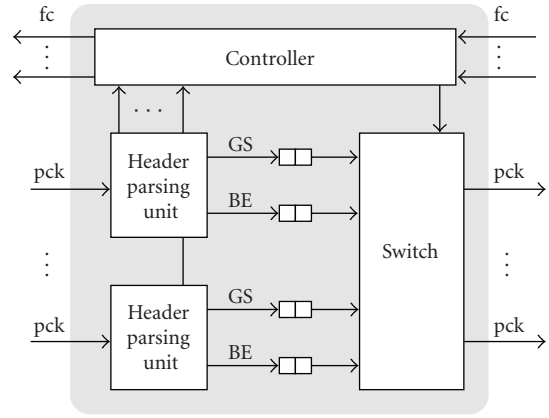


Figure 3: Router architecture where every unidirectional physical channel is shared by two virtual channels, one for guaranteed service (GS) and one for best-effort (BE).

3. BACKGROUND

3.1. Application

We assume that the application is mapped onto cores using existing tools such as [30]. The cores are computational and storage elements of the SoC, such as processors, coprocessors, DSPs, hardware accelerators, memory blocks, I/O blocks. Communication between cores is characterised as *flows*, or sequences of packets, from a source to a destination port.

We distinguish between guaranteed and best-effort services. Guaranteed services (GS) are used for real-time critical traffic and best-effort (BE) for noncritical traffic. Despite the name, even the BE traffic enjoys a number of *qualitative* QoS attributes [31], namely:

(i) *data integrity*, meaning that data is delivered uncorrupted;
(ii) *loss-less delivery*, which means no data is dropped in the network;
(iii) *in-order delivery*, guaranteeing that data arrive at the destination in the order it was sent by the source.

BE services are typically designed for average-case scenarios and require no resource reservations. As a consequence, BE services use resources efficiently. The main disadvantage of BE services is the unpredictability regarding arrival times. In the best case, if sufficient boundary conditions are assumed, a statistical performance bound can be derived [32].

GS adds *flow isolation* to the list of qualitative QoS attributes. Thus the network protects each flow against other (malicious) flows. Moreover, GS introduce a number of *quantitative* QoS attributes, incurring time-related bounds on throughput and latency. To deliver those quantitative guarantees, traffic characteristics must be known in advance [33]. Minimum throughput and maximum latency constraints of the application flows are therefore determined beforehand by means of static analysis or simulation.

3.2. Network

The Æthereal network comprises interconnected *routers* and *network interfaces* (NI). The topology can be regular, such as a mesh, torus, hypercube, or Clos. Irregular topologies are also supported to enable dedicated solutions [27, 34–36].

NIs provide communication services to the cores at the level of the transport layer in the OSI reference model [6]. This is the first layer that offers end-to-end services, hiding the network details [1]. The physical, data-link and network layers of the protocol stack, all being network specific, are thereby not visible to the cores. The NI does not implement any switching functionality. As a consequence, a flow control digit, or flit, destined for a different port on the same NI must turn around in the router network.

The task of a router is to forward data flows from source port to destination port, solving contention when necessary. The architecture, shown in Figure 3, uses *wormhole routing* [37]. No routing decisions are taken by the routers as we employ *source routing*. The NIs contain programmable registers that associate every flow with a path. Upon flit injection, the source NI encodes the path in the header flit and the routers merely execute the decisions already taken.

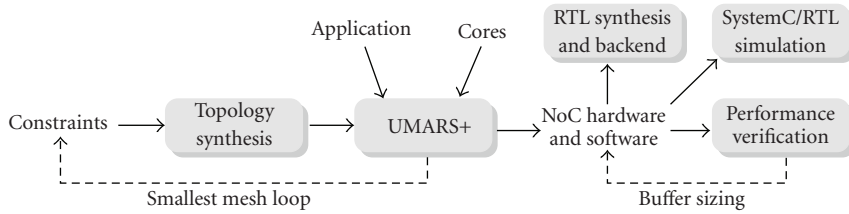


Figure 4: A top-level view of the complete proposed flow.

Two *virtual channels* [37], one guaranteed service channel and one best-effort channel, share each physical channel. By dissociating the buffers from the actual physical channels, a blocked packet in one virtual channel does not block packets residing on other virtual channels [38]. This mechanism affords a division of the entire physical network into two disjoint logical networks [39, 40]. The logically isolated networks use different policies for communication management and can be treated as conceptually separated entities.

The arbitration mechanism that multiplexes between the virtual channels is based on a two-level arbitration scheme. The first level gives priority to GS flows. These flows are thereby isolated from all BE flows as blocking in the BE network can never violate given guarantees. In the second level, two different schemes are used for BE and GS flows, respectively.

3.2.1. Best-effort arbitration

Best-effort flows require contention resolution on the granularity of flits, as multiple packets can request the same output channel and flit arrival cannot be predicted. This contention is resolved individually in each router using a nonoptimal *i*Slip [41] algorithm (round-robin switch-matrix scheduling). The dynamic contention resolution leads to unpredictable storage requirements and delays. Moreover, if a flit is blocked due to busy resources all the trailing flits of that packet are also halted, thereby blocking the resources they occupy in terms of channels and buffers. This can result in chained blocking [42] where the resources of a blocked packet again causes other packets to block, a property that makes wormhole routing very susceptible to deadlock [37, 43].

We address deadlock by means of avoidance, the prominent strategy in NoCs [3, 5, 8, 35, 44]. (Progressive [10, 45] and regressive [46] deadlock recovery techniques exist but are relatively uncommon.) Avoidance-based deadlock-free routing relies on restrictions on resource allocation [37, 43]. In contrast to [35], that advocates the use of virtual channels, we do not add any hardware but solely restrict the BE routing.

3.2.2. Guaranteed service arbitration

Guarantees are implemented by solving contention on the flow level, using TDM-based *virtual circuits*. Every *channel* in the network is multiplexed in time, thereby enabling a single channel to carry several flows. By controlling channel arbitration through a TDM *slot table* in such a way that two flows never compete for the same time-slot, *contention-free routing* is achieved. In other words, once a flit is injected in the router network it never waits for another flit. The slot table is also used to divide bandwidth between the different flows. Note that deadlock is not possible for GS flits as contention is resolved at design-time.

3.3. Problem description

Our problem is to

(1) map the application cores onto any given NoC topology,
(2) statically route the communication flows, and
(3) allocate TDM time-slots on network channels so that application constraints are met.

Two important requirements can be identified and the onus is, in both cases, on the mapping and routing phases. First, the constraints of individual flows must be satisfied. These constraints must hence be reflected in the selection of mapping, path and time slots such that proper resources are reserved. Second, all flows must fit within the available network resources without causing deadlock. Failure in allocating a flow is attributable to nonoptimal previous allocations or insufficient amounts of network resources. This calls for conservation of the finite pool of resources, namely the channels and their time-slots.

This work shows how path selection can be extended to span also mapping and time-slot allocation. This enables the aforementioned requirements to be formulated as path selection constraints and optimization goals.

Figure 4 shows the top-level NoC design flow [5] and the role of UMARS+ in the generation of the NoC hardware and software. The end result is a SystemC model and synthesisable RTL VHDL, compliant with the NXP back-end flow.

4. PROBLEM FORMULATION

4.1. Application

The services are given by the set of valid service classes.

Definition 1. The set of valid service classes $Q = \{GS, BE\}$ represents guaranteed and best-effort service, respectively.

Both service classes provide data-integrity, loss-less delivery and in-order delivery. GS extend those fundamental services with flow isolation and quantitative guarantees on minimum throughput and maximum latency.

The application is characterized by an application graph, comprised of communicating core ports.

Definition 2. An *application graph* is a directed multigraph, $A(P, F)$, where the vertices P represent the set of *core ports*, and the arcs F represent the set of *flows* between the ports. The set of flows comprises two mutually exclusive subsets, F_{GS} and F_{BE}, containing GS and BE flows, respectively. More than a single flow is allowed to connect a given pair of ports. Every core port is source or destination of at least one flow, leaving no node isolated. Each flow $f \in F$ is associated with a service class, $q(f) \in Q$, a minimum throughput, $b(f) \in \mathbb{R}$, and a maximum latency constraint, $l(f) \in \mathbb{R}$. Let $s(f)$ denote the source node of f and let $d(f)$ denote the destination node.

An example application, containing five core ports, is shown in Figure 5. The ports are interconnected through six flows with diverse service requirements. Bandwidth measures are given in Mbps by the designer, as described in [5]. These numbers are, in a preprocessing stage, translated into a real number of TDM slots.

To be able to constrain mapping according to physical layout requirements (e.g., subsystem grouping), we allow grouping of the core ports in P and map groups instead of individual ports. UMARS+ is thereby forced to map ports in a group to the same spatial location (NI).

Definition 3. The *mapping groups* P_M, is a partition of P where the elements are jointly exhaustive and mutually exclusive.

An example of such a partition on a set of core ports $P = \{p_0, p_1, p_2\}$ is shown in Figure 7 where $P_M = \{\{p_0, p_1\}, \{p_2\}\}$. The union of the elements in P_M is clearly the entire P, making the partition jointly exhaustive. Moreover, the elements of P_M are mutually exclusive as no $p \in P$ exists in more than one of them.

A partition according to Definition 3 corresponds to an equivalence relation where two elements in P are considered equal if they must be mapped to the same spatial location. The equivalence class of a core p is hereafter denoted by $[p]$. In the example shown in Figure 7, $[p_0] = [p_1] = \{p_0, p_1\}$ whereas $[p_2] = \{p_2\}$.

4.2. Network

Time-division of network channel capacity is governed by slot tables.

Definition 4. A *slot table* is a sequence of elements in $F^0 = F \cup \{\varnothing\}$. Slots are either occupied by a flow $f \in F$ or empty, represented by $\varnothing$. The number of empty slots in a slot table t is denoted $\sigma(t)$. The same slot table size S_T is used in all the tables of the network.

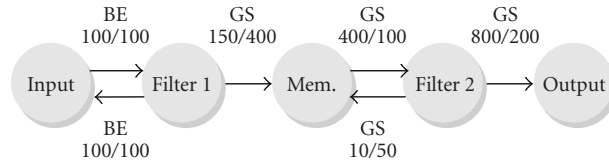


Figure 5: Example application consisting of five core ports and six flows with diverse service requirements. The labels on the edges denote throughput/latency requirements in Mbps and ns, respectively.

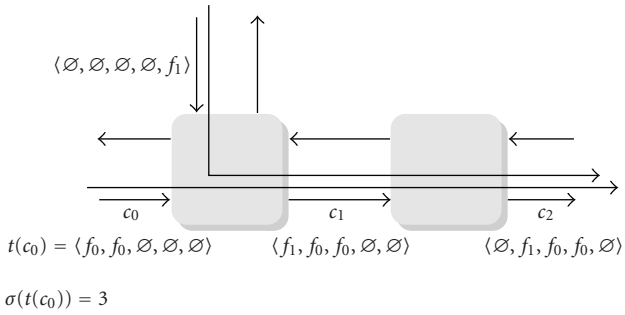


Figure 6: Two flows, f_0 and f_1 are allocated one and two time-slots, respectively, on the paths indicated by the arrows.

To improve latency and reduce buffering requirements, the virtual circuits are pipelined. Pipelining requires a logical notion of router synchronicity, which is possible in the Æthereal NoC. If a slot i is reserved for a flow f on a channel, then slot $i+1$ (modulo the table size) must be reserved on the next channel along the path as depicted in Figure 6.

NoCs are represented by interconnection network graphs.

Definition 5. An *interconnection network graph* I is a strongly connected directed multigraph, $I(N, C)$. The set of vertices N is composed of three mutually exclusive subsets, N_R, N_{NI} and N_P containing *routers*, *network interfaces* (NI), and *core-port mapping nodes* as shown in Figure 7. The latter are dummy nodes to allow unmapped core ports to be integrated in the interconnection graph. The number of core-port mapping nodes in I is equal to the number of mapping groups, $|N_P| = |P_M|$.

The set of arcs C is composed of two mutually exclusive subsets, C_R and C_P containing physical network channels and dummy mapping channels. Channels in C_R represent the physical network architecture and interconnect nodes in N_R and N_{NI}. The channels in C_P interconnect *every* node in N_P to *all* nodes in N_{NI}. This construction allows all cores to be mapped to any NI. No direct interconnections are made between nodes in N_R and N_P.

More than a single physical channel is allowed to connect a given pair of routers. However, an NI node n_{NI} is always connected to a single router through exactly one egress channel $c_E(n_{NI}) \in C_R$ and exactly one ingress channel $c_I(n_{NI}) \in C_R$, as depicted in Figure 7.

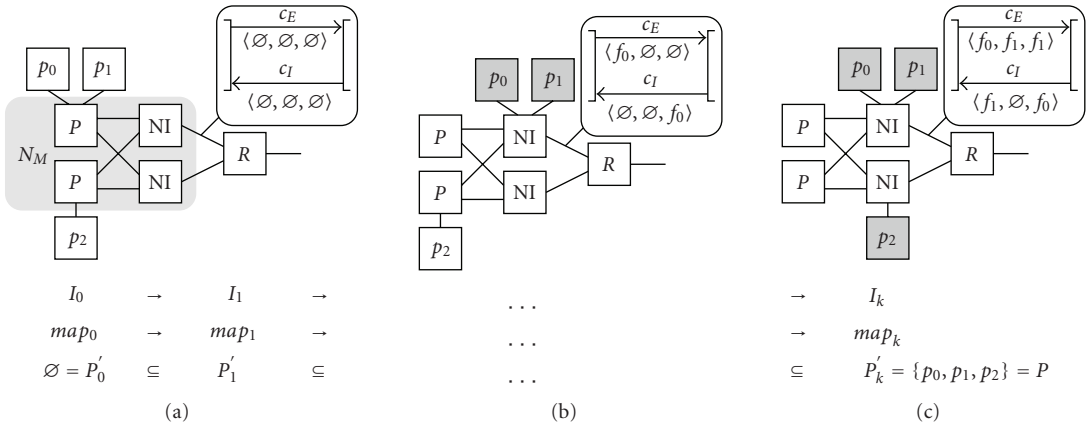


Figure 7: Successive refinement of mapping and interconnection network.

Each channel $c \in C$ has a bandwidth not yet reserved (residual bandwidth) measured in number of slots, $\beta(c) \in \mathbb{R}$, a discretized ditto, $\overline{\beta}(c) \in \mathbb{N}$, and a slot table, $t(c)$. Let $s(c)$ denote the source node of c and let $d(c)$ denote the destination node.

4.3. Path selection

Definition 6. A *path* $\pi \in \operatorname{seq}_1 C$ from source $n_s \in N$ to destination $n_d \in N$ is a nonempty sequence of channels $\langle c_0, \ldots, c_k \rangle$ such that

(1) $d(c_i) = s(c_{i+1})$ for $0 \le i \le k-1$,
(2) $s(c_0) = n_s$ and $d(c_k) = n_d$.

Definition 7. For a source and destination node $n_s, n_d \in N$, $\Pi(n_s, n_d)$ is the set of all possible paths from n_s to n_d.

4.4. Time-slot allocation

When allocating time-slots on a given path $\pi = \langle c_0, \ldots, c_k \rangle$, we first determine the set of available time-slots relative to c_0. To do so we aggregate the individual slot tables through shift and union operations on the slot tables.

Definition 8. The *left shift operator* L^i is an unary operator that shifts a slot table i steps cyclically to the left, $i \in \mathbb{N}^+$:

$$L^i \langle t_0, \ldots, t_k \rangle = \langle t_{i+1}, \ldots, t_k, t_0, \ldots, t_i \rangle, \qquad L \stackrel{\text{def}}{=} L^1. \tag{1}$$

Definition 9. The *union operator* $|$ is a binary operator that joins two equally sized slot tables: $\langle t_0, \ldots, t_k \rangle \mid \langle t_0', \ldots, t_k' \rangle = \langle t_0'', \ldots, t_k'' \rangle$ where

$$t_i'' = \begin{cases} t_i & \text{if } t_i \neq \varnothing, \\ t_i' & \text{if } t_i = \varnothing. \end{cases} \tag{2}$$

Hence, for every position $0 \le i \le k$ in the sequence, the item in the left hand side slot table, t_i, is preferred if that slot is reserved by a flow, $t_i \in F$. If that slot is empty, $t_i = \varnothing$, then t_i' is used instead. As a consequence, a slot on position i in the joined table is empty if and only if both t_i and t_i' are empty.

With the shift and union operator we can formulate a slot table aggregation function.

Definition 10. An *aggregated slot table function* $t : \operatorname{seq} C \to \operatorname{seq} F^0$ maps a sequence of channels $\langle c_0, \ldots, c_k \rangle$ to an aggregated slot table,

$$\begin{aligned} t(\langle\rangle) &= \langle \varnothing, \ldots, \varnothing \rangle, \\ t(\langle c_0, \ldots, c_k \rangle) &= L^0 t(c_0) \,|\, L^1 t(c_1) \,|\, \ldots \,|\, L^k t(c_k). \end{aligned} \tag{3}$$

Every channel slot table $t(c_i)$, is shifted cyclically i steps left and thereafter joined by the union operator. A slot in $t(\langle c_0, \ldots, c_k \rangle)$ is empty if and only if it is empty in *all* shifted slot tables [6]. By definition the empty sequence of channels is associated with the empty slot table of size S_T.

Consider, for example, allocating slots on the path $\langle c_0, c_1, c_2 \rangle$ in Figure 6. From the figure we get $t(c_0) = \langle f_0, f_0, \varnothing, \varnothing, \varnothing \rangle$, $t(c_1) = \langle f_1, f_0, f_0, \varnothing, \varnothing \rangle$ and $t(c_2) = \langle \varnothing, f_1, f_0, f_0, \varnothing \rangle$. To derive the set of empty slots, we start with the slot table of c_0, $t(\langle c_0 \rangle) = t(c_0) = \langle f_0, f_0, \varnothing, \varnothing, \varnothing \rangle$. We continue by adding $L^1 t(c_1)$ followed by $L^2 t(c_2)$ and get

$$\begin{aligned} t(\langle\rangle) &= \langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing \rangle, \\ t(\langle c_0 \rangle) &= \langle f_0, f_0, \varnothing, \varnothing, \varnothing \rangle, \\ t(\langle c_0, c_1 \rangle) &= \langle f_0, f_0, \varnothing, \varnothing, f_1 \rangle, \\ t(\langle c_0, c_1, c_2 \rangle) &= \langle f_0, f_0, \varnothing, \varnothing, f_1 \rangle. \end{aligned} \tag{4}$$

Note that the addition of c_2 does not change the aggregated slot table as $t(c_2)$ is merely $t(c_1)$ shifted one step to the right.

4.5. Deadlock avoidance

To guarantee deadlock freedom and at the same time offer maximum routing flexibility we use a *turn-prohibition* algorithm [28] built on the *turn model* [47].

Definition 11. A *turn* is an ordered pair of directed channels (c_i, c_j), $c_i \neq c_j$ such that $d(c_i) = s(c_j)$. That is, a pair of channels such that c_i is entering the node that c_j is leaving.

Definition 12. $T(I)$ denotes the set of *possible turns* between channels in the interconnection network I.

Definition 13. $T^-(I)$ denotes the set of *prohibited turns* and $T^+(I) = T(I) \setminus T^-(I)$ the set of *permitted turns*.

We introduce a restricted routing function for BE traffic to assert deadlock freedom. This function prohibits any turn not in the set of permitted turns $T^+(I)$. The latter is derived by using any cycle-breaking algorithm with support for the topology of the network I.

Definition 14. A *turn-prohibiting routing function* R' is of the form $R' : T^+(I) \times C \times N \to \mathcal{P}(C)$, where $\mathcal{P}(C)$ is the power set of C. That is, when a BE flow destined for n_d enters a node through one of its input channels c_i, $R'(T^+(I), c_i, n_d)$ supplies a nonempty set of channels $\{c_1, c_2, \ldots, c_k\} \subset C$ through which the flow may be routed to its next hop enroute to n_d without causing deadlock.

4.6. Mapping

The NIs and core port mapping nodes together form the set of nodes to which the port groups can be mapped.

Definition 15. The set of *mappable nodes*, $N_M = N_{NI} \cup N_P$ as shown in Figure 7(a) contains all nodes to which the elements of P_M can be mapped.

The actual mapping from core ports to mapping nodes is captured by a function.

Definition 16. A *mapping function*, $map : P_M \to N_M$, maps sets of ports (the elements in P_M) to mappable nodes.

Both the interconnection network I and the mapping function are refined or iterated over. We therefore subscript them with an index. Our starting point is an initial mapping, map_0, where every $[p] \in P_M$ is mapped to a unique $n_P \in N_P$. Similarly, I_0 denotes the initial network where no channel capacity is reserved, $\beta(c) = \overline{\beta}(c) = S_T$, and all slots in $t(c)$ are empty for every channel $c \in C$.

As seen in Figure 7(a), the range of map_0 covers only N_P. As the algorithm progresses (b), the range of map_i covers both N_P and N_{NI} partially. Successive iterations of map_i progressively replace elements of N_P with elements of N_{NI} until a final mapping is derived (c), where the range of map_k contains elements of N_{NI} exclusively.

Definition 17. The set of *mapped core ports* $P'_i = \{p \in P \mid map_i([p]) \in N_{NI}\}$ denotes the elements in P which are mapped to NIs in iteration i.

From our definition of map_0 it follows that $P'_0 = \varnothing$. Later we show that there exists a mapping map_k with all ports mapped to elements in N_{NI}, hence $P'_k = P$.

(1) Allocate all flows in F_{GS}
(2) Derive the set of permitted turns $T^+(I_i)$ by turn-prohibiting the current network I_i
(3) Allocate all flows in F_{BE}

Algorithm 1: Allocation of all flows in F.

4.7. UMARS+ contribution

We introduce a major change from previous work and formulate mapping and path selection problem as a pure path selection problem.

Given an interconnection network I_0 and an application graph A, we must select a path π for every flow $f \in F$ such that throughput (5) and latency (6) requirements of the flow are met (for GS flows), without over-allocating the network channels (7),

$$\text{bandwidth of } t(\pi) \geq b(f), \tag{5}$$

$$\text{latency of } t(\pi) \leq l(f), \tag{6}$$

$$\beta(c) \geq 0, \quad \forall c \in C. \tag{7}$$

The theory required to derive worst-case throughput and latency from a slot table is covered in [48].

Note that UMARS+ does not consider physical-level issues such as floorplan congestion and wire length. It does, however, enable the designer to (1) construct any regular or irregular topology as input to the algorithm, (2) group the core ports and thus force them to be mapped to the same NI, and (3) partially (or even fully) specify the core port mapping.

5. UNIFIED MAPPING AND ROUTING

In this section, we present the UMARS+ algorithm. The methodology is described in Sections 5.1 through 5.4. In Section 5.5, we prove algorithm termination, whereafter we conclude in Section 5.6 with a discussion on UMARS+ time-complexity.

The outmost level of UMARS+ is outlined in Algorithm 1 and briefly introduced here. We start by allocating (map and route) all guaranteed-service flows of the application in Step (1). In Step (2), a set of permitted turns is derived using the current interconnection network. Finally, all best-effort flows are allocated in Step (3), just as was done with the guaranteed-service flows. Allocation of F_{GS} and F_{BE} is further explained in Section 5.2.

5.1. Turn-prohibition

Turn-prohibition is traditionally based purely on the network topology [8, 24, 27, 34]. The turn-prohibition can then be done prior to the allocation of GS flows and depends only on the NoC topology. Thereby, it implicitly assumes uniform channel capacities. By delaying this step, we can incorporate

(1) Let the set of unallocated flows $F'_i = F_q$
(2) While $F'_i \neq \varnothing$:
  (a) Get flow $\arg\max_{f \in F''} b(f)$
  (b) Select a path $\pi \in \Pi(s(f), d(f))$
  (c) $F'_{i+1} = F'_i \setminus \{f\}$

Algorithm 2: Allocation of a set of flows F_q.

knowledge of residual bandwidth in the prohibition algorithm.

After allocating F_{GS}, the residual capacity on the network channels, which is what is available for the flows in F_{BE}, is not uniform. Employing a traditional turn-prohibition algorithm, we risk prohibiting those turns where there is capacity left. We address this by using the algorithm proposed in [28] with $b(f)$ as channel weight. Besides being applicable to any topology, this algorithm bounds the accumulated prohibited turn weight to 1/2 of the total weight. Hence, by using the nondiscretized residual bandwidth as channel weights we assure that no more than half the residual turn bandwidth is prohibited.

5.2. Allocation of a set of flows

Allocation of all flows F_q belonging to a certain service class q is done according to Algorithm 2. A brief explanation follows and we detail it further in Sections 5.3 and 5.4.

In Step (2)(a), a flow f is selected based on bandwidth requirements. UMARS+ iterates over the monotonically decreasing set of unallocated flows F'_i and never back-tracks to reevaluate an already allocated flow. This results in low time-complexity at the expense of optimality. A path π is selected for f in Step (2)(b). By initially mapping cores to the core mapping nodes, connected to all NIs, the first and last channel traversed implicitly determine what NI $s(f)$ and $d(f)$ are mapped to, respectively. If $q(f) = GS$ then time-slots are allocated to f on π. Thereafter, map_i and I_i are updated to reflect the new state. The procedure is repeated until all flows are allocated.

5.3. Flow traversal order

We order the flows based on bandwidth requirements, in Step (2)(a) of Algorithm 2, as it

(i) helps in reducing bandwidth fragmentation [14],
(ii) is important from an energy consumption and resource conservation perspective as the benefits of a shorter path grow with communication demands [8], and
(iii) gives precedence to flows with a more limited set of possible paths [8].

Ordering by $b(f)$ alone may affect resource consumption negatively as communication chains are disregarded. That is, clusters of interconnected cores are not mapped in sequence. This may increase average hop-count as communicating cores risk being mapped far apart due to resource saturation. For this reason, the selection is limited to flows having $s(f)$ or $d(f)$ mapped to a node in N_{NI}. Every cluster of communicating cores then have their flows allocated in sequence. A similar approach is employed in [19, 20], where the next core is selected based on communication to already placed cores.

Due to the nature of the least-cost path selection algorithm, explained in Section 5.4.2, we restrain the domain even more and only consider flows where $s(f) \in P'_i$. This additional restriction can be removed if path selection is done also in the reverse direction, from destination to source, which is not the case in the current implementation.

The next flow in Algorithm 2 is chosen according to (8), where $f \in F''_i$ if and only if $f \in F'_i \wedge s(f) \in P'_i$. When the latter condition is not fulfilled by any flow, the entire F'_i is used as the domain,

$$\arg\max_{f \in F''} b(f). \tag{8}$$

5.4. Path selection

When a flow f is chosen, we proceed to Step (2)(b) of Algorithm 2 and select a path for f. This is done according to Algorithm 3, briefly presented here, followed by in-depth discussions in Sections 5.4.1 through 5.4.5.

Path selection for f is composed of three major tasks.

(1) Speculative bandwidth reservations for f are removed from egress and ingress channels in Steps (1) and (2) to have I_i reflect what resources are available to f prior to its allocation. Speculative reservations are required as interdependent flows are not allocated simultaneously and are further discussed in Section 5.4.1.
(2) A path from $s(f)$ to $d(f)$ is selected in Steps (3) and (5), a procedure elaborated on in Section 5.4.2. If $s(f)$ or $d(f)$ are not yet mapped to NIs, these steps include refinement of map_i, which is covered in Section 5.4.4. If map_i is refined, then bandwidth reservations are made on ingress and egress channels for flows other than f, as they now have their source or destination mapped to an NI.
(3) If $q(f) = GS$, then a set of time-slots is selected. Resources used by f are then reserved on the resulting path π, as discussed in Section 5.4.5.

5.4.1. Bandwidth reservation

When $s(f)$ for a flow f is mapped to an NI, the communication burden placed on the ingress and egress channels of the NI is not determined by f only. As every p in $[s(f)]$ is fixed to this NI, the aggregated communication burden of all flows incident to those cores is placed on the ingress channel. The egress channel similarly has to accommodate all flows emanating from those cores. When $d(f)$ is mapped, all flows to or from $[d(f)]$ are accounted for accordingly.

Failing to address the above may result in overallocation of network resources. Numerous flows, still not allocated, may be forced to use the ingress and egress channel due to an already fixed mapping. An NI may thereby be associated with an implicit load, not accounted for when evaluating possible paths. We make this load explicit by exploiting knowledge of ingress-egress pairs, as in [49].[2] We define a function that estimates how much bandwidth (measured in slots) a flow reserves in the network.

Definition 18. The *bandwidth requirement estimation function* $b' : F \to \mathbb{R}^+$ supplies an estimate of required network bandwidth for a flow f as

$$b'(f) = \begin{cases} b(f) & \text{if } q(f) = BE \\ \lceil b(f) \rceil & \text{if } q(f) = GS. \end{cases} \tag{9}$$

Although we have no knowledge of exactly what time slots are needed by future guaranteed service flows, we can estimate the bandwidth required by $b'(f)$ and incorporate estimated average load in the cost function, further discussed in Section 5.4.3.

Steps (1) and (2) of Algorithm 3 restore the speculative reservations for f on egress and ingress channel to have I_i reflect what resources are available prior to its allocation.

The corresponding bandwidth reservations on egress and ingress channels are carried out in Steps (4)(b), (4)(c) and Steps (6)(b), (6)(c) for source and destination NI, respectively.

5.4.2. Selecting constrained least-cost path

Steps (3) and (5) of Algorithm 3 select a constrained least-cost path using Dijkstra's algorithm.

Three modifications are done to the standard relaxation procedure, where π_p denotes the partial path from $s(f)$ to the current node.

(1) Best-effort flows must obey the turn-prohibiting routing function R'. Therefore, only channels in $R'(T^+(I_i), d(last\ \pi_p), d(F))$ are evaluated further. We use the *turn net* approach described in [50], as the original Dijkstra's algorithm cannot find least-cost paths on a turn-prohibited network.

(2) The search space is pruned by discarding emanating channels that cannot meet bandwidth constraints. For best-effort flows we discard channels where $\beta(c) < b'(f)$. Guaranteed service flows do a similar control on the discretized residual bandwidth $\overline{\beta}(c) < b'(f)$ but also prune channels where $\sigma(t(\pi_p) \mid Lt(c)) < b'(f)$. Channels that cannot meet bandwidth constraints or do not have enough free slots, given $t(\pi_p)$, are thereby omitted.

(3) As the final path must contain only physical network resources, channels in C_P may only be the first or last element of a path. Hence, if $d(last\ \pi_p) \in N_P$, then all channels emanating from $d(last\ \pi_p)$ are discarded.

(1) If $s(f) \in P'_i$, restore bandwidth reservation on egress channel by adding $b(f)$ to $\beta(c_E(map_i([s(f)])))$ and $b'(f)$ to $\overline{\beta}(c_E(map_i([s(f)])))$.
(2) If $d(f) \in P'_i$, restore bandwidth reservation on ingress channel by adding $b(f)$ to $\beta(c_I(map_i([d(f)])))$ and $b'(f)$ to $\overline{\beta}(c_I(map_i([d(f)])))$.
(3) Select a constrained least-cost path π_s from $map_i([s(f)])$ to a router $n_R \in N_R$.
(4) If $s(f) \notin P'_i$, then
  (a) Refine $map_{i+1} = map_i \oplus \{[s(f)] \mapsto d(head\ \pi_s)\}$
  (b) Reserve egress bandwidth for all unallocated flows emanating from $[s(f)]$ by subtracting $\sum_{f_E \in F_E} b(f_E)$ from $\beta(c_E(d(head\ \pi_s)))$ and $\sum_{f_E \in F_E} b'(f_E)$ from $\overline{\beta}(c_E(d(head\ \pi_s)))$ where $f_E \in F_E$ if and only if $f_E \in F'_i$, $s(f_E) \in [s(f)]$ and $f_E \neq f$
  (c) Reserve ingress bandwidth for all unallocated flows incident to $[s(f)]$ by subtracting $\sum_{f_I \in F_I} b(f_I)$ from $\beta(c_I(d(head\ \pi_s)))$ and $\sum_{f_I \in F_I} b'(f_I)$ from $\overline{\beta}(c_I(d(head\ \pi_s)))$ where $f_I \in F_I$ if and only if $f_I \in F'_i$ and $d(f_I) \in [s(f)]$.
(5) Select a constrained least-cost path π_d from $d(last\ \pi_s)$ to $map_i([d(f)])$
(6) If $d(f) \notin P'_i$, then
  (a) Refine $map_{i+1} = map_i \oplus \{[d(f)] \mapsto s(last\ \pi_d)\}$
  (b) Reserve egress bandwidth for all unallocated flows emanating from $[d(f)]$ by subtracting $\sum_{f_E \in F_E} b(f_E)$ from $\beta(c_E(s(last\ \pi_d)))$ and $\sum_{f_E \in F_E} b'(f_E)$ from $\overline{\beta}(c_E(s(last\ \pi_d)))$ where $f_E \in F_E$ if and only if $f_E \in F'_i$ and $s(f_E) \in [d(f)]$
  (c) Reserve ingress bandwidth for all flows incident to $[d(f)]$ by subtracting $\sum_{f_I \in F_I} b(f_I)$ from $\beta(c_I(s(last\ \pi_d)))$ and $\sum_{f_I \in F_I} b'(f_I)$ from $\overline{\beta}(c_I(s(last\ \pi_d)))$ where $f_I \in F_I$ if and only if $f_I \in F'_i$, $d(f_I) \in [d(f)]$ and $f_I \neq f$.
(7) If $q(f) = GS$, then select a constrained set of slots T_S in $t(\pi)$ for the complete path $\pi = \pi_s \frown \pi_d$ and update $t(c)$, for all $c \in \pi$.
(8) Do a final bandwidth reservation by subtracting $b(f)$ from $\beta(c)$, for all $c \in \pi$. If $q(f) = GS$ then subtract $|T_S|$ from $\overline{\beta}$, for all $c \in \pi$ correspondingly.

Algorithm 3: Path selection for a given f.

The NI architecture requires a path to incorporate at least one physical channel $c \in C_R$ as flows cannot turn around inside an NI. If a flow has both source and destination mapped to the same NI we must hence traverse the egress channel, turn around in the router and return to the NI through the ingress channel. From a least-cost perspective the best path from an NI to itself is the empty path and we force the algorithm into leaving the NI by doing path selection in

[2] The authors suggest selecting paths that interfere least with future requests through a heuristic called *minimum interference routing algorithm (MIRA)*. The algorithm does not only consider the ingress and egress channels but also calculates an interference metric for every intermediate channel in the network.

two steps. (An alternative with a higher time complexity is A*Prune [51] that enables both this constraint and the turn-prohibitions to be formulated as path constraints.)

The first part of the path π_s is selected in Step (3) of Algorithm 3. We know by the definition of I that it is possible to find a path to a router from $s(f)$ and stop at the one with the lowest cost. If several routers share the same path cost, then we pick the one with highest arity. This heuristic maximises routing flexibility throughout the entire allocation procedure. It also makes sure the source node of the first flow (the one with highest communication volume) is mapped to the NI connected to the router with highest arity, a strategy suggested in [20].

The second part of the path π_d is selected in Step (5), starting where π_s ended. From there we continue to the location where $d(f)$ is currently mapped. The complete path is then just the two parts concatenated, $\pi = \pi_s \frown \pi_d$.

Deriving π like suggested above may, without further care, lead to a path which is not the least-cost path in $\Pi(s(f), d(f))$ as minimization is done on the parts in isolation.[3] However, if a flow f has $s(f) \in P'_i$, then there is only one possible least-cost router. This follows from every NI being connected to exactly one router and all channel costs being non-negative. Hence, there is only one possible π_s and as this π_s is a part of any path in $\Pi(s(f), d(f))$ and π_d is a least-cost path, π is a least-cost path in $\Pi(s(f), d(f))$. To mitigate the effect of partial minimization, we prefer allocating flows where $s(f) \in P'_i$, as discussed in Section 5.3.

5.4.3. Choice of cost function

The cost function used plays an essential role in meeting the requirements introduced in Section 1. It should hence reflect resource availability and resource utilization. A good heuristic to maximise the probability of successful flow allocation is to select a path with low contention. At the same time we must keep the path length short not to consume unnecessarily many resources. Similar heuristics are suggested in [13, 52, 53].

Double objective path optimization in general is an intractable problem [12]. Combining objectives in one cost function allows for tractable algorithms at the cost of optimality. We therefore argue for a linear combination of the two cost measures, where two constants Γ_c and Γ_h control the importance (and normalisation) of contention and hop-count, respectively.

Contention is traditionally incorporated by making channel cost inversely proportional to residual bandwidth. Although proved to produce good results in many applications [13, 53], this cost measure has two major drawbacks. First, as the value always is greater than zero its contribution to total path cost grows with distance even if there is no contention on the channels traversed. Second, contention cost grows exponentially, thereby disturbing the balance between contention and hop-count importance. We desire control

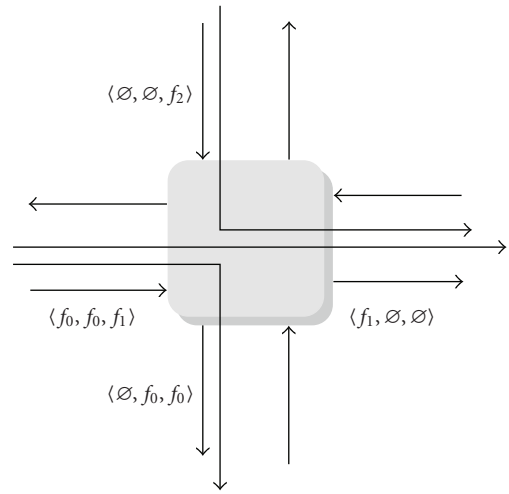


Figure 8: Scenario where average load is insufficient as metric and leads to a path on which slot allocation invariably fails.

over the relative importance of contention and hop-count through Γ_c and Γ_h and therefore use (10) to determine channel cost when allocating a flow with $q(f) = BE$. The contention measure, $S_T - \beta(c)$, makes the cost contribution proportional to the occupied bandwidth. It is zero for an unoccupied channel and grows linearly as bandwidth is reserved,

$$\Gamma_c(S_T - \beta(c)) + \Gamma_h. \tag{10}$$

When allocating guaranteed service flows, the cost measure in (10) fails to reflect a number of important aspects involved in deciding what is an optimal path.

(i) Using only average load when determining contention cost ignores the temporal alignment of the available capacity. Not only must the slots be free, we also require them to be aligned properly to be usable, about which more presently.

(ii) It bases the cost on nondiscretized residual bandwidth, thereby looking at the actual bandwidth available without accounting for TDM discretisation artifacts.

When using pipelined virtual circuits [6], average load is not reflecting what resources are available to the current flow. Not even the slot table $t(c)$ itself provides an accurate view. The set of available slots for a flow, f, on a channel, c, is a function of the slot tables of all channels preceding c in the path traversed from the location where $s(f)$ is mapped to the channel c itself.

Consider the example in Figure 8 where a flow f_2 arrives a router already used by flows f_0 and f_1. If we look only at residual bandwidth, f_2 prefers the channel going east over the one heading south. However, if we consider not only the number of free slots but also their temporal alignment, south is actually a better choice than east. Even though the filling of the slot table is higher for the south-going channel, the alignment compared to the input channel makes it a better choice.

[3] Compare a sum of minima to the minimum of a sum.

We exploit knowledge of the partial path π_p traversed so far and determine contention cost for a channel c by how much $t(c)$ reduces the amount of available slots compared to $t(\pi_p)$ if c is traversed. Discretized available bandwidth is incorporated by taking the maximum of the two as contention measure, according to (11).

$$\Gamma_c \max\left\{S_T - \overline{\beta}(c), \sigma(t(\pi_p)) - \sigma(t(\pi_p) \mid Lt(c))\right\} + \Gamma_h. \tag{11}$$

Channels in C_P must not contribute to the path cost, as they are not physical interconnect components. We therefore make them *zero-cost* channels.

5.4.4. Refining mapping function

When a path π_s is selected for a flow f, we check in Step (4)(a) of Algorithm 3, whether $s(f)$ is not yet mapped to an NI. If not, π_s decides the NI to which the core is to be mapped. We therefore refine the current mapping function with the newly determined mapping to a node in N_{NI} as seen in Step (6)(a). This refinement is fixed and every core in $[s(f)]$ is now in P'_i.

Correspondingly, we check if $d(f)$ is not yet mapped to an NI in Step (6) and if not, refine the mapping according to π_d in Step (6)(a).

5.4.5. Resource reservation

When the entire path π is determined, we perform a slot allocation in Step (7) of Algorithm 3 if the flow requires guaranteed services. The slots available to f are deduced by looking at $t(\pi)$. From the empty slots we select a set of slots T_S such that bandwidth and latency requirements of f are met [48]. All channels $c \in \pi$ are then updated with a new $t(c)$ to reflect what slots are reserved to f.

Step (8) ends the procedure by removing the resources reserved for f from $\beta(c)$ and $\overline{\beta}(c)$ for all channels in the path.

5.5. Algorithm termination

With each refinement of map_i, zero, one or two additional sets of cores are moved to elements of N_{NI} from N_P, hence $P'_{i+1} \supseteq P'_i$, as depicted in Figure 7.

Theorem 1. $(\exists k) P'_k = P$: *there exists a k such that all core ports are mapped to NIs.*

Proof. When a flow is f allocated, map_i is refined in Steps (4)(a) and (6)(a) of Algorithm 2 so that $s(f)$ and $d(f)$ are guaranteed to be in P'_i. For every allocated flow $f \notin F'_i$ we hence know that $s(f), d(f) \in P'_i$.

From Step (2)(c) of Algorithm 2 we know that $F'_{i+1} \subset F'_i$, that is, the set of unallocated flows, monotonically decreases. Hence, $\exists k$ such that all flows are allocated, $F'_k = \varnothing$. We know that, for this k, $s(f)$ and $d(f)$, for all $f \in F$ are in P'_k. As no isolated cores are allowed in A it follows that $P = P'_k$. □

5.6. Algorithm complexity

Due to the greedy nature of UMARS+, time-complexity is very low as seen in (12), where d denotes the maximum degree of any node in N. The expression is dominated by the first term that is attributable to Dijkstra's algorithm, used for path selection. The second term stems from the turn prohibition and varies depending on the choice of algorithm. Finally, the last term covers the selection of next flow, bandwidth reservations and slot allocation. Experiments indicate that UMARS+ run-time is only 20% higher than that of load-balancing path selection alone,

$$\mathcal{O}\left(|F|\left(|C|+|N|\log|N|\right)\right)+\mathcal{O}\left(|N|^2 d\right)+\mathcal{O}\left(|F|\left(|F|+|P|+S_T\right)\right). \tag{12}$$

6. EXPERIMENTAL RESULTS

To evaluate the performance of our methodology, we apply it to a range of SoC designs. The *MPEG* use-case is a MPEG codec SoC, further described in Section 6.3. The *uniform* use-case features distributed communication with 24 cores. Each core has a randomly selected set of inter-connected peers with a total aggregated bandwidth of 750 Mbps. The remaining use-cases are internal set-top box designs, each having hot-spots around a limited set of SDRAM ports and 100 to 250 connections. These connections deliver a total bandwidth of 1-2 Gbps to 75 ports distributed across 25 IP modules.

6.1. Deadlock avoidance

The turn-prohibition algorithm's ability to preserve residual resources is evaluated by allocating the *uniform* benchmark to a fixed 3×4 mesh with a varying degree of BE and GS flows. We study the relative success rate compared to what is achievable without routing restrictions, that is, when deadlock can occur. The results of three different turn-prohibition algorithms are compared. First, *xy* routing, second, traditional oblivious turn prohibition not taking residual bandwidth into account, and third, the adaptive turn prohibition that we propose.

In Figure 9, we see that the adaptive algorithm consistently outperforms the other two algorithms with a relative success rate constantly above 92%. While the oblivious turn-prohibition algorithm offers a qualitative advantage over *xy* routing by being applicable to any topology, the adaptive algorithm adds also a significant quantitative advantage.

6.2. Evaluation experiments

A cost function where $\Gamma_c = 1$ and $\Gamma_h = 1$ is used throughout the experiments. Those values favour contention-balancing over hop-count as the slot table size is an order of magnitude larger than network diameter in all use-cases.

All results are compared with the traditional multistep algorithm in [5], referred to as *waterfall*. Only mesh topologies are evaluated, as the aforementioned algorithm is limited to this class of networks. For a given slot table size S_T, all

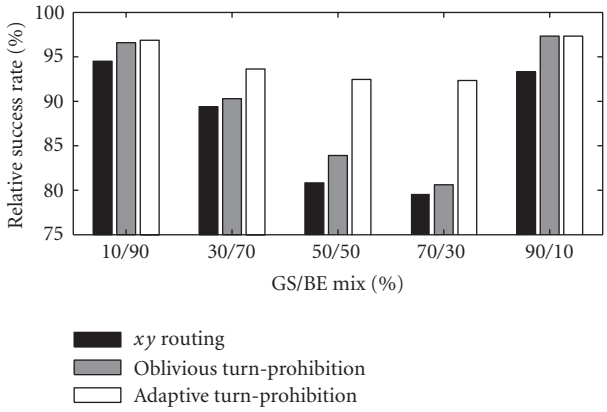


Figure 9: Relative success rate for the different turn-prohibition algorithms.

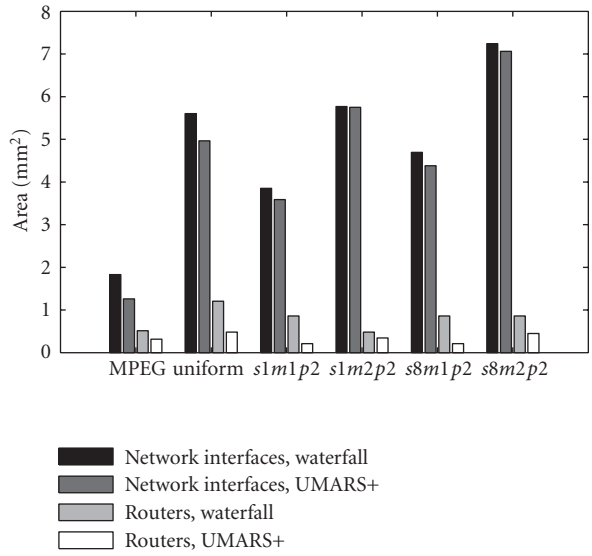


Figure 10: Comparison of area requirements.

unique $n \times m$ router networks with less than 25 routers were generated in increasing order of size. For every such router network, one, two, or three NIs were attached to each router until all application flows were allocated, or allocation failed. Slot table size was incremented until allocation was successful.

The run time of UMARS+ is in the order of a few milliseconds on a Linux workstation and the whole topology exploration loop finishes in a matter of seconds for the example SoC designs.

Each design is simulated during 3×10^6 clock cycles in a flit-accurate SystemC simulator of the Æthereal NoC, using traffic generators to mimic core behaviour.

All the presented use-cases feature applications with guaranteed service flows only. These flows use all three parts of the algorithm (mapping, routing, and slot allocation) and have more allocation constraints than best-effort flows. The latter makes it more difficult to find a working configuration and stresses the importance of objective unification in all three allocation phases.

6.2.1. Analytic benchmarks

Silicon area requirements are based on the model presented in [54], assuming a 0.13 μm CMOS process. Figure 10 shows that area requirements can be significantly reduced. Up to 33% in total area reduction is observed for the experiment applications. Slot table sizes are reduced in all use-cases, leading to lower buffer requirements, analytically derived as described in [5]. Area savings up to 31% are observed for the NIs but the *s1m2p2* use-case is hardly improved at all, showing only a 0.5% decrease. However, the router network is consistently smaller, with an area decrease between 30% and 75%.

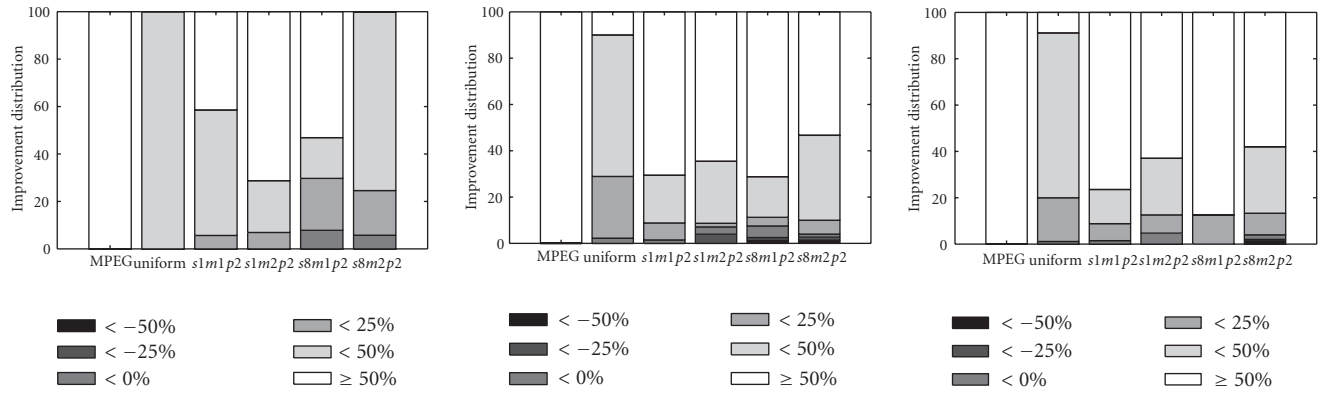
The distribution of improvement on analytical worst-case latency is shown in Figure 11(a). For every flow the worst-case latency is derived using the model in [5]. The latency achieved using UMARS+ and *waterfall* are compared on a flow basis and the distribution of these improvement figures are plotted in the diagram. Although a few flows suffer from latency increase (negative improvement) in the *s8m1p2* and *s8m2p2* use-cases, the majority of flows have significant improvements on worst-case latency. In the *MPEG* example, every single flow has its worst-case latency reduced by 50% or more.

6.2.2. Simulation benchmarks

Relative energy consumption of the router network (without NIs), calculated according to the model in [55] is depicted in Figure 12. As the application remains the same and hence essentially the same bits are being communicated, the savings in energy consumption are attributable to flows being allocated on paths with fewer hops. The correlation between energy saving ratio and relative reduction in number of routers is clearly visible. However, as the smaller router network is used more extensively, energy is reduced less than the number of routers.

Figure 13 shows the average utilization of channels emanating from NIs and routers, respectively. As expected, utilization increases as router network size is reduced and UMARS+ consequently improves both NI and router utilization. Time-division-multiplexed circuits imply bandwidth discretisation, leading to inevitable over-allocation and complicating the task of achieving high utilization. This together with unbalanced hot-spot traffic, leaving some parts of the network lightly loaded and others congested, lead to inherent low utilization in some of the example use-cases. Note that utilization is only to be optimized after all constraints are met.

The distribution of improvement on average and maximum latency is shown in Figures 11(b) and 11(c), respectively. As with the analytical latency comparison we see that a few flows face an increased latency. The bias towards latency improvement is clear though, and in all nonsynthetic



(a) Analytical worst-case delay (b) Average delay in simulation (c) Maximum delay in simulation

FIGURE 11: Distribution of improvement on flow network delay. For every flow, latency of UMARS+, l_{UMARS+}, is compared to that of *waterfall*, $l_{waterfall}$, as $1 - l_{UMARS+}/l_{waterfall}$. These improvement measures are divided into bins of 25% width whereafter the relative frequency of the bins is plotted on a per application basis.

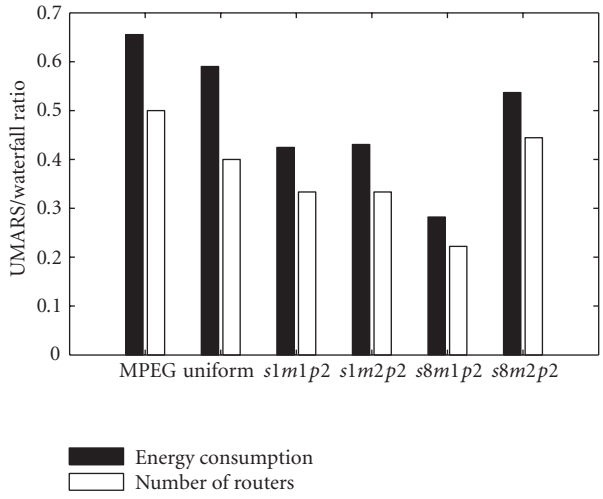


FIGURE 12: Comparison of energy consumption.

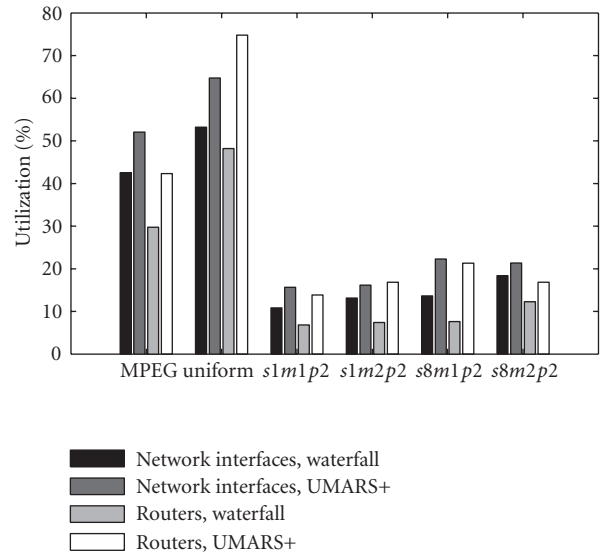


FIGURE 13: Comparison of NoC resource utilization.

use-cases the latency is reduced with 50% or more for more than half of the flows.

6.3. An MPEG application

An existing MPEG codec SoC with 16 cores constitutes our design example and results are shown in Table 1. The architecture uses a single external SDRAM with three ports to implement all communication between cores. A total of 42 flows tie the cores together. Using the design flow presented in [5][4] results in a 2×3 mesh, referred to as *clustering* in Table 1, with a total estimated area of 2.35 mm^2. For comparison, a naive mapping with one core partition per NI is almost double in size, whereas the worst-case write latency remains more or less unaffected.

A manually optimized mapping was produced which managed to reduce the network area with 21% and an almost four-fold reduction of average worst-case write latency was observed [5].

TABLE 1: Comparison of MPEG NoCs.

Generation	Mesh	Slots	NI area	Router area	Total area	Area diff	Avg wc latency
Clustering	2×3	128	1.83	0.51	2.35	ref	1570 ns
Naive	3×6	128	2.17	2.32	4.49	+91%	1583 ns
Optimized	1×3	8	1.51	0.35	1.86	−21%	399 ns
UMARS+	1×3	8	1.26	0.32	1.57	−33%	383 ns

[4] Clustered mapping, *xy* routing and greedy slot allocation.

UMARS+ arrives at a mesh of equal size to what was achieved using the manually optimized mapping. Fewer NIs are needed leading to reductions in router area. Smaller buffer requirements, attributable to less bursty time-slot allocation, results in reduced NI area. Total NoC area is reduced by 17% and average worst-case latency by 4% compared to the optimized handcrafted design. The solution was achieved in less than 100 milliseconds on a Linux workstation. Only a 20% increase in run-time was observed when compared to a pure load-balancing path selection, without mapping and slot allocation.

7. CONCLUSION AND FUTURE WORK

We conclude this work by summarizing our contributions in Section 7.1 and finally presenting directions for future work in Section 7.2.

7.1. Contributions

In this paper, we consider the problem of mapping cores onto any given NoC topology and statically route the communication between these cores. We present the UMARS+ algorithm which integrates the three resource allocation phases: spatial mapping of cores, spatial routing of communication and TDM time-slot assignment.

As the main contribution we show how mapping can be fully incorporated in path selection. This allows for formulation of a single consistent objective function that is used throughout all allocation phases. The objective is reflecting two important goals, namely, fulfilment of application constraints and conservation of network resources while guaranteeing deadlock freedom.

We show how the pruning and the cost metric used in path selection can be extended beyond one channel to capture the nature of virtual circuits. By incorporating also the traversed path in cost calculations we derive a metric that reflects how suitable a channel is when used *after* the channels already traversed.

We show how a highly flexible turn-prohibition algorithm can be used to provide maximum adaptiveness in routing of best-effort flows. The proposed algorithm bases the prohibitions on residual resources such that best-effort flows can use what is not required by guaranteed-service flows.

The time-complexity of UMARS+ is low and experimental results indicate a run-time only 20% higher than that of path selection alone.

We apply the algorithm to an MPEG decoder SoC, improving area 33%, power dissipation 35% and worst-case latency by a factor four over a traditional waterfall approach.

7.2. Future work

We compare UMARS+ only to [5], and a more extensive comparison with traditional algorithms [8, 18–21] is of value.

To allow a more extensive design space exploration for both mapping and routing, UMARS+ can be extended to a k-path algorithm, enabling a trade-off between complexity and optimality. This extension can also be used for traffic splitting, spatially distributing the load of guaranteed service flows over multiple paths.

UMARS+ fully supports any topology, thereby enabling application-specific NoC generation. To exploit those capabilities, a valuable extension is to incorporate the algorithm into a more refined topology generation tool. Topologies can then be tailored for an application and physical layout.

REFERENCES

[1] L. Benini and G. de Micheli, "Networks on chips: a new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] M. Sgroi, M. Sheets, A. Mihal, et al., "Addressing the system-on-a-chip interconnect woes through communication-based design," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 667–672, Las Vegas, Nev, USA, June 2001.

[3] D. Bertozzi, A. Jalabert, S. Murali, et al., "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113–129, 2005.

[4] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 684–689, Las Vegas, Nev, USA, June 2001.

[5] K. Goossens, J. Dielissen, O. P. Gangwal, S. Gonzàlez Pestana, A. Rădulescu, and E. Rijpkema, "A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification," in *Proceedings of Design, Automation and Test in Europe Conference and Exposition (DATE '05)*, pp. 1182–1187, Munich, Germany, March 2005.

[6] E. Rijpkema, K. Goossens, A. Rădulescu, et al., "Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip," *IEE Proceedings: Computers and Digital Techniques*, vol. 150, no. 5, pp. 294–302, 2003.

[7] K. Keutzer, S. Malik, J. M. Rabaey, and A. Sangiovanni-Vincentelli, "System-level design: orthogonalization of concerns and platform-based design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 12, pp. 1523–1543, 2000.

[8] J. Hu and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," in *Proceedings of Design, Automation and Test in Europe Conference and Exposition (DATE '03)*, pp. 688–693, Munich, Germany, March 2003.

[9] U. Y. Ogras, J. Hu, and R. Marculescu, "Key research problems in NoC design: a holistic perspective," in *Proceedings of International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '05)*, pp. 69–74, Jersey City, NJ, USA, September 2005.

[10] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 2, pp. 890–895, Paris, France, February 2004.

[11] A. Laffely, J. Liang, R. Tessier, and W. Burleson, "Adaptive system on a chip (aSoC): a backbone for power-aware signal processing cores," in *Proceedings of International Conference on Image Processing (ICIP '03)*, vol. 3, pp. 105–108, Barcelona, Spain, September 2003.

[12] R. A. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '97)*, vol. 3, pp. 1903–1908, Phoenix, Ariz, USA, November 1997.

[13] R. Widyono, "The design and evaluation of routing algorithms for real-time channels," Tech. Rep. TR-94-024, International Computer Science Institute & University of California, Berkeley, Calif, USA, June 1994.

[14] I. Matta and A. Bestavros, "A load profiling approach to routing guaranteed bandwidth flows," in *Proceedings of the 17th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '98)*, vol. 3, pp. 1014–1021, San Francisco, Calif, USA, March-April 1998.

[15] R. A. Guerin and A. Orda, "Networks with advance reservations: the routing perspective," in *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00)*, vol. 1, pp. 118–127, Tel Aviv, Israel, March 2000.

[16] W. H. Ho and T. M. Pinkston, "A methodology for designing efficient on-chip interconnects on well-behaved communication patterns," in *Proceedings of the 9th International Symposium on High-Performance Computer Architecture (HPCA '03)*, pp. 377–388, Anaheim, Calif, USA, February 2003.

[17] S. Stuijk, T. Basten, M. Geilen, A. H. Ghamarian, and B. Theelen, "Resource-efficient routing and scheduling of time-constrained network-on-chip communication," in *Proceedings of the 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD '06)*, pp. 45–52, Dubrovnik, Croatia, August-September 2006.

[18] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '03)*, pp. 233–239, Kitakyushu, Japan, January 2003.

[19] S. Murali and G. de Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proceedings of Design, Automation and Test in Europe Conference and Exposition (DATE '04)*, vol. 2, pp. 896–901, Paris, France, February 2004.

[20] S. Murali and G. de Micheli, "SUNMAP: a tool for automatic topology selection and generation for NoCs," in *Proceedings of the 41st Design Automation Conference (DAC '04)*, pp. 914–919, San Diego, Calif, USA, June 2004.

[21] S. Murali, L. Benini, and G. de Micheli, "Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '05)*, vol. 1, pp. 27–32, Shanghai, China, January 2005.

[22] P. M. Pardalos, F. Rendl, and H. Wolkowicz, "The quadratic assignment problem: a survey and recent developments," in *Quadratic Assignment and Related Problems*, P. M. Pardalos and H. Wolkowicz, Eds., vol. 16 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 1–42, American Mathematical Society, Providence, RI, USA, 1994.

[23] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *Computer*, vol. 26, no. 2, pp. 62–76, 1993.

[24] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *Journal of the ACM*, vol. 41, no. 5, pp. 874–902, 1994.

[25] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 729–738, 2000.

[26] J. Hu and R. Marculescu, "DyAD: smart routing for networks-on-chip," in *Proceedings of the 41st Design Automation Conference (DAC '04)*, pp. 260–263, San Diego, Calif, USA, June 2004.

[27] L. Benini, "Application specific NoC design," in *Proceedings of Design, Automation and Test in Europe Conference and Exposition (DATE '06)*, pp. 491–495, Munich, Germany, March 2006.

[28] D. Starobinski, M. Karpovsky, and L. A. Zakrevski, "Application of network calculus to general topologies using turn-prohibition," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 411–421, 2003.

[29] A. Hansson, K. Goossens, and A. Rădulescu, "A unified approach to constrained mapping and routing on network-on-chip architectures," in *Proceedings of 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '05)*, pp. 75–80, Jersey City, NJ, USA, September 2005.

[30] S. J. Krolikoski, F. Schirrmeister, B. Salefski, J. Rowson, and G. Martin, "Methodology and technology for virtual component driven hardware/software co-design on the system-level," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '99)*, vol. 6, pp. 456–459, Orlando, Fla, USA, May-June 1999.

[31] I. Stoica, "Stateless core: a scalable approach for quality of service in the Internet," Ph.D. dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pa, USA, December 2000, also as Tech. Rep. CMU-CS-00-176.

[32] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, vol. 83, no. 10, pp. 1374–1396, 1995.

[33] C. M. Aras, J. F. Kurose, D. S. Reeves, and H. Schulzrinne, "Real-time communication in packet-switched networks," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 122–139, 1994.

[34] U. Y. Ogras and R. Marculescu, "Application-specific network-on-chip architecture customization via long-range link insertion," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '05)*, pp. 246–253, San Jose, Calif, USA, November 2005.

[35] K. Srinivasan and K. S. Chatha, "A low complexity heuristic for design of custom network-on-chip architectures," in *Proceedings of Design, Automation and Test in Europe Conference and Exposition (DATE '06)*, pp. 130–135, Munich, Germany, March 2006.

[36] J. Xu, W. Wolf, J. Henkel, and S. Chakradhar, "A design methodology for application-specific networks-on-chip," *ACM Transactions on Embedded Computing Systems*, vol. 5, no. 2, pp. 263–280, 2006.

[37] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. 36, no. 5, pp. 547–553, 1987.

[38] P. Mohapatra, "Wormhole routing techniques for directly connected multicomputer systems," *ACM Computing Surveys*, vol. 30, no. 3, pp. 374–410, 1998.

[39] D. H. Linder and J. C. Harden, "An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes," *IEEE Transactions on Computers*, vol. 40, no. 1, pp. 2–12, 1991.

[40] J. Rexford and K. G. Shin, "Support for multiple classes of traffic in multicomputer routers," in *Proceedings of the 1st International Workshop on Parallel Computer Routing and Communication (PCRCW '94)*, pp. 116–130, Seattle, Wash, USA, May 1994.

[41] N. W. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. dissertation, University of California, Berkeley, Calif, USA, May 1995.

[42] W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, 1992.

[43] E. Fleury and P. Fraigniaud, "A general theory for deadlock avoidance in wormhole-routed networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 7, pp. 626–638, 1998.

[44] T. Bjerregaard and J. Sparsø, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proceedings of Design, Automation and Test in Europe Conference and Exposition (DATE '05)*, vol. 2, pp. 1226–1231, Munich, Germany, March 2005.

[45] P. Guerrier, "Un réseau d'interconnexion pour systémes intégrés," Ph.D. dissertation, Université Paris VI, Paris, France, 2000.

[46] I. Saastamoinen, M. Alho, and J. Nurmi, "Buffer implementation for Proteo networks-on-chip," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '03)*, vol. 2, pp. 113–116, Bangkok, Thailand, May 2003.

[47] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *Proceedings of the 19th International Symposium on Computer Architecture (ISCA '92)*, pp. 278–287, Gold Coast, Queensland, Australia, May 1992.

[48] O. P. Gangwal, A. Rădulescu, K. Goossens, S. Gonzàlez Pestana, and E. Rijpkema, "Building predictable systems on chip: an analysis of guaranteed communication in the Æthereal network on chip," in *Dynamic and Robust Streaming in and between Connected Consumer-Electronics Devices*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2005.

[49] K. Kar, M. Kodialam, and T. V. Lakshman, "Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2566–2579, 2000.

[50] M. Fidler and G. Einhoff, "Routing in turn-prohibition based feed-forward networks," in *Proceedings of the 3rd IFIP-TC6 Networking Conference (Networking '04)*, vol. 3042 of *Lecture Notes in Computer Science*, pp. 1168–1179, Athens, Greece, May 2004.

[51] G. Liu and K. G. Ramakrishnan, "A *Prune: an algorithm for finding K shortest paths subject to multiple constraints," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '01)*, vol. 2, pp. 743–749, Anchorage, Alaska, USA, April 2001.

[52] K. Kowalik and M. Collier, "Should QoS routing algorithms prefer shortest paths?" in *Proceedings of IEEE International Conference on Communications (ICC '03)*, vol. 1, pp. 213–217, Anchorage, Alaska, USA, May 2003.

[53] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in *Proceedings of the International Conference on Network Protocols (ICNP '97)*, pp. 191–202, Atlanta, Ga, USA, October 1997.

[54] S. Gonzàlez Pestana, E. Rijpkema, A. Rădulescu, K. Goossens, and O. P. Gangwal, "Cost-performance trade-offs in networks on chip: a simulation-based approach," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 2, pp. 764–769, Paris, France, February 2004.

[55] J. Dielissen, A. Rădulescu, and K. Goossens, "Power measurements and analysis of a network-on-chip," Tech. Rep. NL-TN-2005-0282, Philips Research Laboratories, Eindhoven, The Netherlands, 2005.

Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 37627, 11 pages
doi:10.1155/2007/37627

Research Article

A Method for Routing Packets Across Multiple Paths in NoCs with In-Order Delivery and Fault-Tolerance Gaurantees

Srinivasan Murali,[1] David Atienza,[2,3] Luca Benini,[4] and Giovanni De Micheli[3]

[1] *Computer Systems Lab, Stanford University, Stanford, CA 94305-9040, USA*
[2] *Departamento Arquitectura de Computadores y Automatica, Universidad Complutense de Madrid, 28040 Madrid, Spain*
[3] *Laboratoire des Systèmes Intégrés, Ecole Polytechnique Federale de Laussanne, 1015 Lausanne, Switzerland*
[4] *Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, 40126 Bologna, Italy*

Received 16 October 2006; Revised 21 January 2007; Accepted 6 February 2007

Recommended by Maurizio Palesi

Networks on Chips (NoCs) are required to tackle the increasing delay and poor scalability issues of bus-based communication architectures. Many of today's NoC designs are based on single path routing. By utilizing multiple paths for routing, congestion in the network is reduced significantly, which translates to improved network performance or reduced network bandwidth requirements and power consumption. Multiple paths can also be utilized to achieve spatial redundancy, which helps in achieving tolerance against faults or errors in the NoC. A major problem with multipath routing is that packets can reach the destination in an out-of-order fashion, while many applications require in-order packet delivery. In this work, we present a multipath routing strategy that guarantees in-order packet delivery for NoCs. It is based on the idea of routing packets on partially nonintersecting paths and rebuilding packet order at path reconvergent nodes. We present a design methodology that uses the routing strategy to optimally spread the traffic in the NoC to minimize the network bandwidth needs and power consumption. We also integrate support for tolerance against transient and permanent failures in the NoC links in the methodology by utilizing spatial and temporal redundancy for transporting packets. Our experimental studies show large reduction in network bandwidth requirements (36.86% on average) and power consumption (30.51% on average) compared to single-path systems. The area overhead of the proposed scheme is small (a modest 5% increase in network area). Hence, it is practical to be used in the on-chip domain.

1. INTRODUCTION

Future *Systems on Chips (SoCs)* will have multiple processing cores and memories with each core having high performance and frequency of operation. There is a growing trend towards integrating multiple applications onto the same SoC, so that the user needs to carry only one device that performs a host of operations. As the computational load of the SoC increases, so does the load on the communication architecture. Scalable micro networks (or *Networks on Chips (NoCs)*) are needed to provide high bandwidth communication infrastructure for the SoCs [1–7]. With technology scaling, on-chip wires are increasingly susceptible to various error sources such as crosstalk, coupling noise, ground bounce, soft errors, process variations, and interconnect failures. The use of NoCs facilitate the application of network error resiliency techniques to tolerate such transient and permanent errors in the interconnects.

The routing scheme used in the NoC can be either static or dynamic in nature. In static routing, one or more paths are selected for the traffic flows in the NoC at design time. In the case of dynamic routing, the paths are selected based on the current traffic characteristics of the network. Due to its simplicity and the fact that application traffic can be well characterized for most SoC designs, static routing is widely employed for NoCs [8]. When compared to static single-path routing, the static multipath routing scheme improves path diversity, thereby minimizing network congestion and traffic bottlenecks. When the NoC is predesigned, with the NoC having a fixed operating frequency, data width, and hence bandwidth (bandwidth available on each network link is the product of the link data width and the NoC operating frequency), reducing congestion results in improved network performance. For most SoC designs, the NoC operating frequency can be set to match the application requirements. In this case, reducing the traffic bottlenecks leads to lower

Table 1: Reorder buffer overhead.

Design	Area (sq mm)	Power (mW)
(1) Base NoC	1.04	183.75
(2) 2 buffers/core	1.14	201
(3) 10 buffers/core	1.65	281.25

required NoC operating frequency, as traffic is spread evenly in the network, thereby reducing the peak link bandwidth needs. A reduced operating frequency translates to a lower power consumption in the NoC. As an example, consider an MPEG video application mapped onto a 4×3 mesh NoC. Detailed analysis of the application and the performance of traditional single path schemes and the proposed multipath scheme are presented later in this work (in Section 7.2). When the NoC operating frequency for the schemes is set so that both schemes provide the same performance level (same average latency for traffic streams), the multipath scheme results in 35% reduction in network operating frequency, leading to 22.22% reduction in network power consumption (after accounting for the overhead involved in the multipath scheme). Another important property of the multipath routing strategy is that there is spatial redundancy for transporting a packet in the on-chip network. A packet can be sent across multiple paths for achieving resiliency against transient or permanent failures in the network links.

Many of today's NoC architectures are based on static single path routing. This is because, with multipath routing, packets can reach the destination in an out-of-order fashion due to the difference in path lengths or due to difference in congestion levels on the paths. For many applications, such out-of-order packet delivery is not acceptable and packet reordering is needed at the receivers. As an example, in chip multiprocessor applications for maintaining coherency and consistency, packets reaching the destination need to be in-order. In video and other multimedia applications, packet ordering needs to be maintained for displays and for many of the processing blocks in the application.

With multipath routing, packet reorder buffers can be used at the receiver to reorder the arriving packets. However, the reorder buffers have large area and power overhead and deterministically choosing the size of them is infeasible in practice. In Table 1, the area and power consumption of a 4×3 mesh NoC (the area-power values include the area power of the switches, links, Network Interfaces (NIs)) with different numbers of packet buffers in the receiving NIs is presented. The network operating frequency is assumed to be 500 MHz with 50% switching activity at the subcomponents. The flit size is assumed to be 16 bits with the base switch/NI having 4 flit queuing buffers at each output. The base NoC component area, power values are obtained from synthesizing the component designs that are based on the ×pipes architecture [9] (refer to Section 7 for a description of the architecture) using UMC 0.13 μm technology library. As seen from the table, a NoC design with 10 packet reorder buffers/core has 59% higher NoC area and 43% higher NoC power consumption when compared to the base NoC without reorder buffers. Another important point is that at design time it is not possible to size the reorder buffers to prevent packets from being dropped at the receiver. As an example, if a packet travels a congested route and takes an arbitrarily long time to reach the destination, several subsequent packets that take a different route can reach the destination before this packet. In this case, the reorder buffers, unless they have infinite storage capacity, can be full for a particular scenario and can no longer receive packets. This leads to dropping of packets to recover from the situation and requires end-to-end ACK/NACK protocols for resuming the transaction.

End-to-end ACK/NACK protocols are used in most macro networks for error recovery and in such networks, these protocols are extended to handle this packet buffering problem as well [10]. However, such protocols have significant overhead in terms of network resource usage and congestion. Thus, they are not commonly used in the NoC domain [10, 11]. Moreover, the performance penalty to recover from such a situation can be very high and most applications cannot tolerate such variations in performance. This motivates the need to find efficient solutions to the packet reordering problem for the on-chip domain.

To the best of our knowledge, this is the first work that presents a multipath routing strategy with guaranteed in-order packet delivery (without packet dropping) for on-chip networks. It is based on the idea of routing packets on *non-intersecting* paths and rebuilding packet order at *path reconvergent* nodes. By using an efficient flow control mechanism, the routing strategy avoids the packet dropping situation that arises in the traditional multipath routing schemes. We present algorithms to find the set of paths in a NoC topology to support the routing strategy and present a method to split the application traffic across the paths to obtain a network with minimum power consumption. We explore the use of temporal and spatial redundancy during multipath routing to provide resilience against temporary and permanent errors in the NoC links. When sending multiple copies of a packet, it is important to achieve the required reliability level for packet delivery with minimum data replication. We integrate reliability constraints in our multipath design methods to provide a reliable NoC operation with least increase in network traffic. Experiments on several benchmarks show large power savings for the proposed scheme when compared to traditional single-path schemes and multipath schemes with reorder buffers. The area overhead of the proposed scheme is small (a modest 5% increase in network area). Hence, it is practical to be used in the on-chip domain.

2. PREVIOUS WORK

Several researchers have been focusing on NoC design issues and a variety of NoC architectures and platforms have been proposed [1–7]. Many works on mapping of applications onto NoC architectures have considered the routing problem during the NoC design phase [8, 12–15]. In [16], a low latency router architecture for supporting dynamic routing is presented. In [17], a routing scheme that switches between

deterministic and adaptive modes, depending on the application requirements, is presented. All these works assume that the architectural support needed for such routing schemes (such as packet reorder buffers) are available in the NoC.

Several works in the multiprocessor field have focused on the design of efficient routing strategies [18]. In the Avici router [19], packets that need to be in-order at the receiver are grouped together into a *flow*. Packets of a single flow follow a single path, while different flows can use different paths. In the IBM SP2 network [20], source-based oblivious routing is used for a multistage interconnection network. In [21], the authors present a source-based dynamic routing algorithm for multistage networks. In both works, a single path is used for packets that require in-order delivery and multiple paths are used for packets that do not require ordering.

Several research works have focused on designing reliable NoC systems [11, 22–28]. In [24], fault-tolerant stochastic communication for NoCs is presented. The use of nonintersecting paths for achieving fault-tolerant routing has been utilized in many designs, such as the IBM Vulcan [18]. The use of temporal and spatial redundancy in NoCs to achieve resilience from transient failures is presented in [28].

Unlike earlier works, we assume that each packet can be routed on different paths. We present algorithms and design methods to support the multipath routing strategy. We explore the use of temporal and spatial redundancy during multipath routing to provide resilience against temporary as well as permanent errors in the NoC links. The routing methods can either be applied after NoC topology mapping and design or during the mapping process. In this work, we only present the details of the design of the routing strategy. We refer the interested readers to existing works [8, 12–15] on mechanisms to integrate different static routing methods with NoC topology mapping. The basic multipath-routing strategy has been presented by us in [29]. In this work, we integrate the routing mechanism into a design methodology which makes it effective for applying the scheme to NoCs.

3. MULTIPATH ROUTING WITH IN-ORDER DELIVERY

In this section, we present the conceptual idea of the multipath routing strategy with in-order packet delivery. For analysis purposes, we define the NoC topology by the NoC topology graph.

Definition 1. The topology graph is a directed graph $G(V, E)$ with each vertex $v_k \in V$ representing a switch/NI in the topology and the directed link (or edge) $e_l \in E$ representing a direct communication between two switches/NIs. We represent the traffic flow between a pair of cores in the NoC as a commodity i with the source switch/NI of the commodity being s_i and the destination of the commodity being d_i. Let the total number of commodities be I. The rate of traffic transferred by commodity i is represented by r_i.

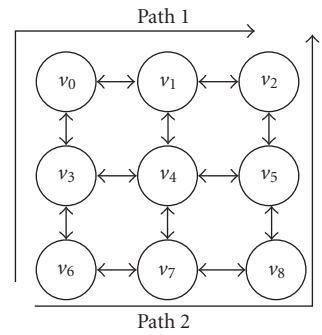


Figure 1: Example mesh topology.

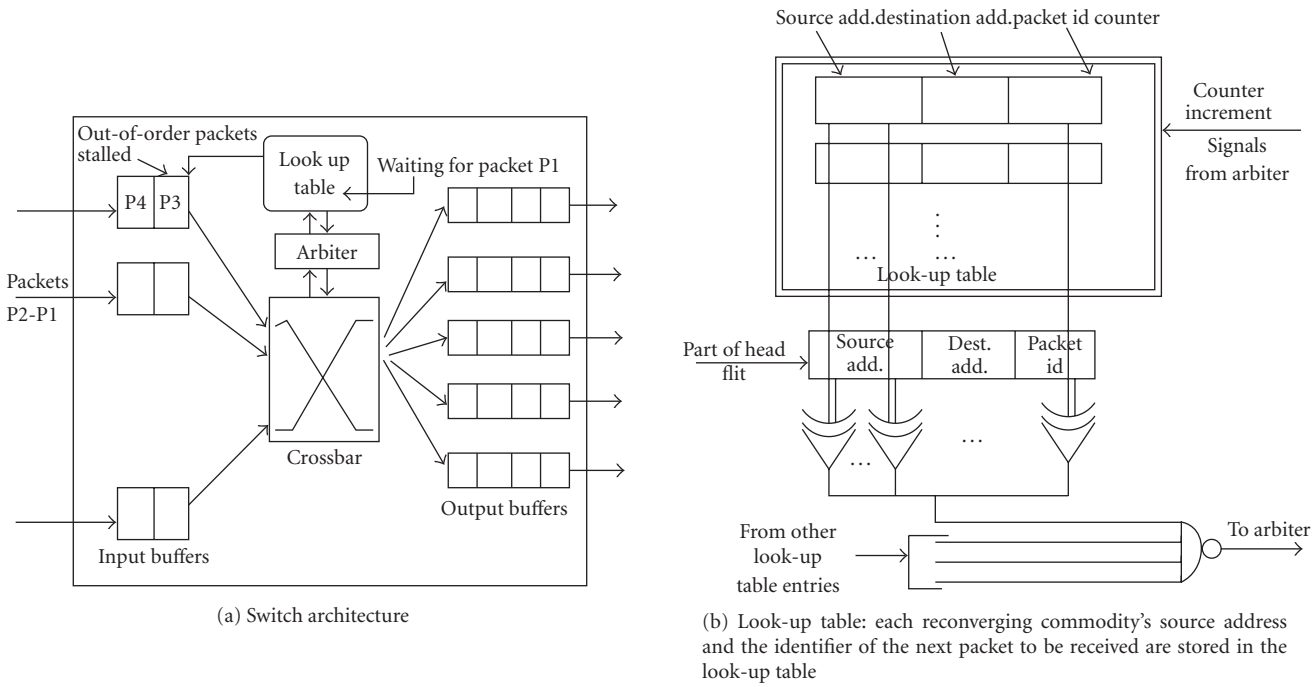
An example NoC topology graph for a 3×3 mesh NoC is shown in Figure 1. Since in the mesh network each NI is connected to only one switch, we represent the topology graph by using only the switches. For topologies where each NI is connected to multiple switches, the NIs are also taken as part of the topology description.

The traffic rate for each commodity (r_i) can either be the average rate of communication between the source and destination of the commodity or can be obtained in an efficient manner that considers the *Quality-of-Service (QoS)* provisions for the application. In this work, we assume that the efficient numbers for the rates are obtained as presented in [14]. The traffic rates are computed considering the peak and average bandwidth needs for the commodity, burstiness in traffic, deadlines and slacks associated with the bursts [14]. We define the paths for the traffic flow of a commodity as follows.

Definition 2. Let the set SP_i represent the set of all paths for the commodity i, for all $i \in 1 \cdots I$. Let P_i^j be an element of SP_i, for all $j \in 1 \cdots |\mathrm{SP}_i|$. Thus P_i^j represents a single path from the source to destination for commodity i. Each path P_i^j consists of a set of links.

Example paths for the commodity (traffic flow) from source vertex v_6 to destination v_2 are shown by the solid lines in Figure 1. We define a set of paths to be *nonintersecting* if the paths originate from the same source vertex but do not intersect each other in the network, except at the destination vertex. The two paths shown in Figure 1 are *nonintersecting*. Consider packets that are routed on the two *nonintersecting* paths. Note that with worm-hole flow control [18], packets of a commodity on a particular path are in-order at all time instances. However, packets on the two different paths can be out-of-order. As an example, if packets 1, 2 are sent on *path* 1 and packets 3, 4 are sent on *path* 2 and if *path* 2 is faster (either because it is shorter or because of lower congestion), then packets 3, 4 can reach the destination before packets 1, 2. Therefore, we need a mechanism to reorder the packets at the *reconvergent* nodes to maintain the packet ordering.

To implement the reordering mechanism at network reconvergent nodes, the following architectural changes to the



(a) Switch architecture

(b) Look-up table: each reconverging commodity's source address and the identifier of the next packet to be received are stored in the look-up table

Figure 2: Switch design to support multipath routing with in-order packet delivery.

switches/NIs of the NoC are required (shown in Figure 2). We assume that the packet is divided into multiple flow control units called *flits*. The first flit of the packet (known as the *header* flit) has the routing information for the packet. To support multipath routing, individual packet identifiers are used for packets belonging to a single commodity. At the *reconvergent* switch, we use a look-up table to store the identifier of the next packet to be received for the commodity. Initially (when the NoC is reset), the identifiers in the look-up tables are set to 1 for all the commodities. When packets arrive at the input of the *reconvergent switch*, the identifier of the packet is compared with the corresponding look-up table entry. If the identifiers match, the packet is granted arbitration and the look-up table identifier value for this commodity is incremented by 1. If the identifiers do not match, then this is an out-of-order packet and access to the output is not granted by the arbiter circuit, and it remains at the input buffer.

As the packets on a particular path are in-order, the mechanism only stalls packets that would also be out-of-order if they reach the switch. Due to the disjoint property of the paths reaching the switch, the actual packet (matching the identifier on the look-up table) that needs to be received by the switch is on a different path. As a result, such a stalling mechanism (integrated with *credit-based* or *on-off* flow control mechanisms [18]) does not lead to packet dropping, which is encountered in traditional schemes when the reorder buffers at the receivers are full. Note that routing-dependent deadlocks that can occur in the network can be avoided using virtual channel flow control [18]. Other routing dependent deadlock free methods, such as presented in [30], can also be used in conjunction with the methods presented in this work.

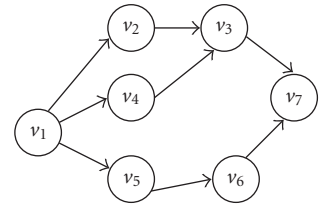
4. PATH SELECTION ALGORITHM

In this section, we describe the algorithms that can be used to efficiently find *nonintersecting paths* for each commodity of the NoC. As in general, the number of paths between a source and destination vertex of a graph is exponential, we present heuristic algorithms to compute the paths [31]. For each commodity, we first find the set of all possible paths for the commodity. Then, from the chosen paths, we find those paths that are nonintersecting. We use such a two-phase approach to achieve fast heuristic solutions to tackle the exponential problem complexity.

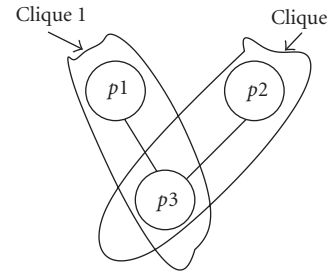
Consider a source vertex s_i (which corresponds to the source core/NI that sends a packet) and destination vertex d_i of a commodity i. Algorithm 1 is used to find the set of possible paths between the two vertices. Example 1 presented below illustrates how the working algorithm works. The objective of the algorithm is to find maximum number of paths possible, so that large path diversity is available for the traffic flow. In the algorithm, after finding a path, we remove one of the edges of the path so that the same path is not considered in further iterations. As most NoC vertices have only a small degree, we remove one of the middle edges (instead of the edges at the source and destination), because it helps in increasing the number of paths found. As in each iteration of the algorithm we remove an edge, the number of iterations (and hence the maximum number of paths found) is at most $|E|$ for one pair of source and destination vertices.

(1) Choose a path from the source to destination of the commodity using Depth First Search (DFS).
(2) Remove one of the middle edges of the chosen path.
(3) Repeat the above steps until there are no paths between the vertices.

Algorithm 1: Path selection algorithm for a single commodity.



(a) Example graph for path selection



(b) Compatibility graph for the example

Figure 3: Path selection and compatibility graph generation.

Example 1. Consider the NoC topology graph presented in Figure 3(a). The vertices represent switches/NIs in the NoC. Let v_1 and v_7 be the source and destination vertices of a traffic flow. In the first iteration of the algorithm, one of the paths (e.g., the path $v1 \rightarrow v2 \rightarrow v3 \rightarrow v7$) is chosen and the middle edge (edge from $v2 \rightarrow v3$) is removed. In the next iteration of the algorithm, another path ($v1 \rightarrow v4 \rightarrow v3 \rightarrow v7$) is chosen and the edge $v4 \rightarrow v3$ is removed. In the last iteration $v1 \rightarrow v5 \rightarrow v6 \rightarrow v7$ is chosen and the edge $v5 \rightarrow v6$ is removed, after which no more paths exist from $v1$ to $v7$. Note that if we had removed the edge $v3 \rightarrow v7$ in the first iteration, we would have obtained only two paths (instead of three paths).

The paths resulting from the algorithm may converge at one or more vertices. In order to obtain *nonintersecting* paths, we form a *compatibility graph* with each vertex of the graph representing a path. An edge between two vertices in the graph implies that the corresponding paths do not intersect. An example *compatibility graph* for the paths from Example 1 is shown in Figure 3(b). The objective is to obtain the maximum number of nonintersecting paths from the set of paths. This is equivalent to finding the maximum size clique[1] in the *compatibility graph*, which is a well-known NP-Hard problem [31]. We use a commonly used heuristic algorithm for finding the maximum clique (see Algorithm 2) [32]. The working of the algorithm is illustrated in Example 2. We repeat the two algorithms for all the commodities in the NoC. When applying Algorithm 1 for each commodity, we start with the original topology graph.

Example 2. The compatibility graph for the 3 paths from Example 1 is shown in Figure 3(b). The vertex $p1$ represents the path $v1 \rightarrow v2 \rightarrow v3 \rightarrow v7$, $p2$ represents the path $v1 \rightarrow v4 \rightarrow v3 \rightarrow v7$, and $p3$ represents the path $v1 \rightarrow v5 \rightarrow v6 \rightarrow v7$. As the paths $v1 \rightarrow v2 \rightarrow v3 \rightarrow v7$ and $v1 \rightarrow v5 \rightarrow v6 \rightarrow v7$ do not intersect each other, there is an edge between $p1$ and $p3$ in the compatibility graph. Similarly, for the paths $v1 \rightarrow v4 \rightarrow v3 \rightarrow v7$ and $v1 \rightarrow v5 \rightarrow v6 \rightarrow v7$, there is an edge between $p2$ and $p3$. There are two maximum size cliques in the graph (formed by $p1, p3$ and $p2, p3$) and one of them is arbitrarily chosen (say, $p1, p3$). Thus, the paths $v1 \rightarrow v2 \rightarrow v3 \rightarrow v7$ and $v1 \rightarrow v5 \rightarrow v6 \rightarrow v7$ are used for the traffic flow between vertices $v1$ and $v7$.

[1] Clique of a graph is a fully connected subgraph.

The time complexity of each iteration in Algorithm 1 is dominated by the *Depth-First Search (DFS)* procedure and is $O(|E| + |V|)$ [31]. The number of iterations is $O(|E|)$. The time complexity of the maximum clique calculation step is $O(|E|^2)$. The algorithms are repeated once for each commodity. Therefore, the run time of the nonintersecting path finding algorithms is $O(|I||E|(|E| + |V|))$. In practice, the run time of the algorithms is less than few minutes for all the experimental studies we have performed (presented in Section 7).

In cases where we are interested in having as many minimum paths as possible, we can modify the call to DFS in Algorithm 1 to a call to Dijkstra's shortest path algorithm, choosing shorter paths first. Then, Algorithm 2 can also be modified to first choose the minimum paths that are nonintersecting and then choosing nonminimum paths that are nonintersecting with each other and with the chosen minimum paths. Note that in situations where we only need few paths for each commodity (to have small route look-up tables), we can select the needed number of paths from the above algorithms. Similarly, for networks that require deadlock avoidance using restricted routing functions, we can use turn models to select the paths [8, 18], with only a marginal increase in the complexity of the presented algorithms.

5. MULTIPATH TRAFFIC SPLITTING

Once we have obtained the set of nonintersecting paths for each commodity, we need to determine the amount of flow of each commodity across the paths that minimizes congestion. Then, we can assign probability values for each path of every commodity, based on the traffic flow across that path for the commodity. At run time, we can choose the path for each packet from the set of paths based on the probability values assigned to them. To achieve this traffic splitting, we

(1) Build a compatibility graph for the paths and initialize the set MAX_CLIQUE to NULL.
(2) Add vertex with largest degree to MAX_CLIQUE.
(3) From remaining vertices, choose vertex that is adjacent to all vertices in set MAX_CLIQUE and add it to the set.
(4) Repeat the above step until no more vertex can be added.

Algorithm 2: Determining non-intersecting paths for a single commodity.

use a *Linear-Programming (LP)*-based method to solve the corresponding multicommodity flow problem. The objective of the LP is to minimize the maximum traffic on each link of the NoC topology, satisfying the bandwidth constraints on the links, and routing the traffic of all the commodities in the NoC. Our LP is represented by the following set of equations:

$$\min : \ t, \tag{1}$$

$$\text{s.t} \sum_{\forall j \in 1 \cdots |\ \text{SP}_i|} f_i^j = r_i, \quad \forall i, \tag{2}$$

$$\sum_{\forall i} \sum_{\forall j, e_l \in P_i^j} f_i^j = \text{flow}_{e_l}, \quad \forall e_l,$$
$$\text{flow}_{e_l} \leq \text{bandwidth}_{e_l}, \quad \forall e_l, \tag{3}$$

$$\text{flow}_{e_l} \leq t \quad \forall e_l \in P_i^j, \ \forall i, j, \tag{4}$$

$$f_i^j \geq 0. \tag{5}$$

In the objective function we use the variable t to represent the maximum flow on any link in the NoC (equations (1), (4)). Equation (2) represents the constraint that the NoC has to satisfy the traffic flow for each commodity with the variable f_i^j representing the traffic flow on the path P_i^j of commodity i. The flow on each link of the NoC and the bandwidth constraints are represented by (3). Other objectives (such as minimizing the sum of traffic flow on the links) and constraints (like latency constraints for each commodity) can also be used in the LP. As an example, the latency constraints for each commodity can be represented by the following equation:

$$\frac{\sum_{\forall j \in 1 \cdots |\text{SP}_i|} \left(f_i^j \times l^j\right)}{\sum_{\forall j \in 1 \cdots |\text{SP}_i|}} f_i^j \leq d_i, \tag{6}$$

where d_i is the hop delay constraint for commodity i and l^j is the hop delay of path j. Once the flows on each path of a commodity are obtained, we can order or assign probability values to the paths based on the corresponding flows.

6. FAULT-TOLERANCE SUPPORT WITH MULTIPATH ROUTING

The errors that occur on the NoC links can be broadly classified into two categories: transient and permanent errors.

6.1. Resilience against transient errors

To recover from transient errors, error detection or correction schemes can be utilized in the on-chip network [11]. Forward error correcting codes such as Hamming codes can be used to correct single-bit errors at the receiving NI. However, the area-power overhead of the encoders decoders, and control wires for such error correcting schemes increases rapidly with the number of bit errors to be corrected. In practice, it is infeasible to apply forward error correcting codes to correct multi-bit errors [11]. To recover from such multi-bit errors, switch-to-switch (link-level) or end-to-end error detection and retransmission of data can be performed. This is applicable to normal data packets. However, control packets such as interrupts carry critical information that need to meet real-time requirements. Using retransmission mechanisms can have significant latency penalty that would be unacceptable to meet the real-time requirements of critical packets. Error resiliency for such critical packets can be achieved by sending multiple copies of the packets across one or more paths. At the receiving switch/NI, the error detection circuitry can check the packets for errors and can accept an error free packet. When sending multiple copies of a packet, it is important to achieve the required reliability level for packet delivery with minimum data replication. We formulate the mathematical models for the reliability constraints and consider them in the LP formulation presented in previous section, as follows.

Definition 3. Let the transient *Bit-Error Rate (BER)* encountered in crossing a path with maximum number of hops in the NoC be β_t. Let the bit-width of the link (also equal to the flit-width) be W.

The maximum probability of a single-bit error when a flit reaches the destination is given by

$$P(\text{Single-bit error in a flit}) = C_1^W \times \beta_t^1 \times (1 - \beta_t)^{W-1}. \tag{7}$$

That is, any one of the W bits can have an error, while the other bits should be error free.

We assume a single-bit error correcting Hamming code is used to recover from single-bit errors in the critical packets and packet duplication is used to recover from multi-bit errors. The probability of having two or more errors in a flit received at the receiving NI is given by

$$P(\geq 2 \text{ errors}) = \gamma_t = \sum_{k=2}^{W} C_k^W \times \beta_t^k \times (1 - \beta_t)^{W-k}. \tag{8}$$

We assume that the error rates on the multiple copies of a flit are statistically independent in nature, which is true for many transient noise sources such as soft errors (for those transient errors for which such statistical independence cannot be assumed, we can apply the method for recovering from permanent failures presented later in this section). When a flit is transmitted n_t times, the probability of having

two or more errors in all the flits is given by

$$\theta_t = \gamma_t^{n_t}. \tag{9}$$

As in earlier works [11, 22, 23], we assume that an undetected or uncorrected error causes the entire system to crash. The objective is to make sure that the packets received at the destination have a very low probability of undetected/uncorrected errors, ensuring the system operates for a predetermined *Mean Time To Failure (MTFF)* of few years. The acceptable residual flit error-rate, defined as the probability of one or more errors on a flit that can be undetected by the receiver is given by the following equation:

$$\mathrm{Err}_{\mathrm{res}} = \frac{T_{\mathrm{cycle}}}{(\mathrm{MTTF} \times N_c \times \mathrm{inj})}, \tag{10}$$

where T_{cycle} is the cycle time of the NoC, N_c is the number of cores in the system and inj is the average flit injection rate per core. As an example, for 500 MHz system with 12 cores, with an average injection rate of 0.1 flits/core and MTTF of 5 years, the $\mathrm{Err}_{\mathrm{res}}$ value is 1.07×10^{-17}. Each critical packet should be duplicated as many times as necessary to make the θ_t value to be greater than the $\mathrm{Err}_{\mathrm{res}}$ value, that is,

$$\theta_t = \gamma_t^{n_t} \geq \mathrm{Err}_{\mathrm{res}}, \quad \text{i.e., } n_t \geq \frac{\ln(\mathrm{Err}_{\mathrm{res}})}{\ln(\gamma_t)}. \tag{11}$$

The minimum number of times the critical packets should be replicated to satisfy the reliability constraints is given by

$$n_t = \left\lceil \frac{\ln(\mathrm{Err}_{\mathrm{res}})}{\ln(\gamma_t)} \right\rceil. \tag{12}$$

To consider the replication mechanism in the LP, the traffic rates of the critical commodities are multiplied by n_t and (2) is modified for such commodities as follows:

$$\sum_{\forall j \in 1 \cdots |\mathrm{SP}_i|} f_i^j = n_t \times r_i \quad \forall i, \text{critical}. \tag{13}$$

6.2. Resilience against permanent errors

To recover from permanent link failures, packets need to be sent across multiple nonintersecting paths. The nonintersecting nature of the paths makes sure that a link failure on one path does not affect the packets that are transmitted on the other paths. As in the transient error recovery case, the critical packets can be sent across multiple paths. For noncritical packets, we assume that hardware mechanisms such as presented in [20] exist to detect and inform the sender of a permanent link failure in a path. Then, the sender does not consider the faulty path for further routing and retransmits the lost flits across other nonintersecting paths. The probability of a path failure in the NoC is given by

$$P(\text{path failure}) = \gamma_p = \sum_{k=1}^{W} C_k^W \times \beta_p^k \times (1 - \beta_p)^{W-k}, \tag{14}$$

where β_p is the maximum permanent bit-error rate of any path in the NoC.

The maximum number of permanent path failures for each commodity (denoted by n_p) can be obtained similar to the derivation of n_t, and is given by

$$n_p = \left\lceil \frac{\ln(\mathrm{Err}_{\mathrm{res}})}{\ln(\gamma_p)} \right\rceil. \tag{15}$$

Let the total number of paths for a commodity i be denoted by $n_{\mathrm{tot},i}$. Once the number of possible path failures is obtained, we have to model the system such that for each commodity, any set of $(n_{\mathrm{tot},i} - n_p)$ paths should be able to support the traffic demands of the commodity. Thus, even when n_p paths fail, the set of other paths would be able to handle the traffic demands of the commodity and proper system operation would be ensured. We add a set of $n_{\mathrm{tot},i}!/(n_p! \times (n_{\mathrm{tot},i} - n_p)!)$ linear constraints in place of (2) for each commodity in the LP with each constraint representing the fact that the traffic on $(n_{\mathrm{tot},i} - n_p)$ paths can handle the traffic demands of the commodity. As an example, when $n_{\mathrm{tot},i}$ is 3 and n_p is 1 (which means that any 1 path can fail from the set of 3 paths) for a commodity i, we need to add the following 3 constraints:

$$\begin{aligned} f_i^1 + f_i^2 &\geq r_i, \\ f_i^2 + f_i^3 &\geq r_i, \\ f_i^1 + f_i^3 &\geq r_i. \end{aligned} \tag{16}$$

Thus, the paths of each commodity can support the failure of n_p paths for the commodity, provided more than n_p paths exist. When we introduce these additional linear constraints, the impact on the run time of the LP is small (for our experiments, we did not observe any noticeable delay in the run time). This is due to the fact that the number of paths available for each commodity is usually small (less than 4 or 5) and hence only few tens of additional constraints are introduced for each commodity. Note that we can modify the mapping procedures to ensure that each commodity has more than n_p paths available for data transfer. In cases where the mapping procedure cannot produce more than n_p paths for some commodities, we can introduce additional links between switches to get such additional paths for the commodity. Modifying the NoC mapping and topology design processes to achieve these effects is beyond the scope of this paper.

7. SIMULATION RESULTS

7.1. Area, power, and timing overhead

Even though the methodology presented in this paper is general, for illustrative purposes we assume that the component designs are based on the ×pipes NoC architecture. The backbone of the NoC is composed of switches, whose main function is to route packets from sources to destinations. Switches provide buffering resources to lower congestion and improve performance; in ×pipes, output buffering is chosen, that is, FIFOs are present on each output port. Switches also handle

flow control issues and resolve conflicts among packets when they overlap in requesting access to the same physical links.

An NI is needed to connect each IP core to the NoC. NIs convert transaction requests/responses into packets and vice versa. In ×pipes, two separate NIs are defined, an *initiator* and a *target* one, respectively associated to system masters and system slaves. A master/slave device will require an NI of each type to be attached to it. The interface among IP cores and NIs is point-to-point as defined by the OCP 2.0 [33] specification, guaranteeing maximum reusability. NI *Look-Up Tables (LUTs)* specify the path that packets will follow in the network to reach their destination (source routing).

The estimated power overhead, based on gate count and synthesis results for switches/NIs, to support the multipath routing scheme for the 4×3 mesh network considered earlier (in Table 1) is found to be 18.09 mW, which is around 5% of the base NoC power consumption. For the power estimation, without loss of generality, we assume that 8 bits are used for representing the source and destination addresses and 8-bit packet identifiers are utilized. The baseline architecture already supports the use of source address in the header flit. With the use of 4-flit packets, with 32-bit for each flit in the baseline architecture, the multipath scheme results in a 13% increase in packet size. The power overhead accounts for this increase in packet size and for the look-up tables and combinational logic associated with the multipath routing scheme. The numbers assume a 500 MHz operating frequency for the network, using UMC 0.13 μm technology library. The estimated area overhead (from gate and memory cell count) for the multipath routing scheme is low (less than 5% of the NoC component area). The maximum possible frequency estimate of the switch design with support for the multipath routing tables is above 500 MHz.

7.2. Case study: MPEG decoder

We assume that the tasks of the MPEG application are assigned to processor/memory cores and the best mapping (minimizing the average communication hop delay) onto a mesh network is obtained using the tool presented in [15]. The communication between the various cores and the resulting mapped NoC are presented in Figures 4, 5. The various paths obtained using the algorithms presented earlier, for some of the commodities, are presented in Table 2. Applying the LP procedure to split the traffic across the obtained paths results in 35% reduction in the bandwidth requirements for the application when compared to single-path routing (both dimension-ordered and minimum-path routing). The bandwidth savings translate to 35% reduction in the required NoC operating frequency. For 16-bit link data width, the multipath routing requires 300 MHz frequency for the NoC to support the application traffic, while the single-path routing schemes require a NoC frequency of 405 MHz. The NoC frequency reduction leads to 22.22% reduction in power consumption of the NoC for the multipath scheme compared to single-path schemes, after accounting for the power overhead of the multipath scheme. The average packet latencies incurred for the MPEG NoC for *dimension ordered (Dim)*, *minimum path (Min)* and our proposed *multipath (Multi)* strategy for the MPEG NoC is presented in Figure 6(a). The simulations are performed on a flit-accurate NoC simulator designed in C++. The multipath routing strategy results in reduced frequency requirements to achieve the same latency as the single-path schemes for a large part of the design space.

When compared to the multipath routing scheme with reorder buffers (10 packet buffers/receiver), the current scheme results in 28.25% reduction in network power consumption.

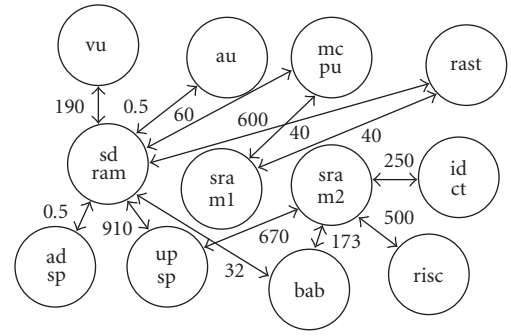


Figure 4: A MPEG decoder application.

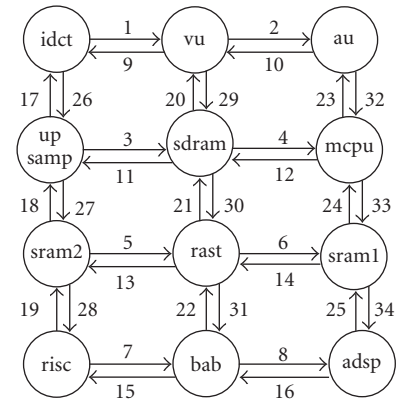


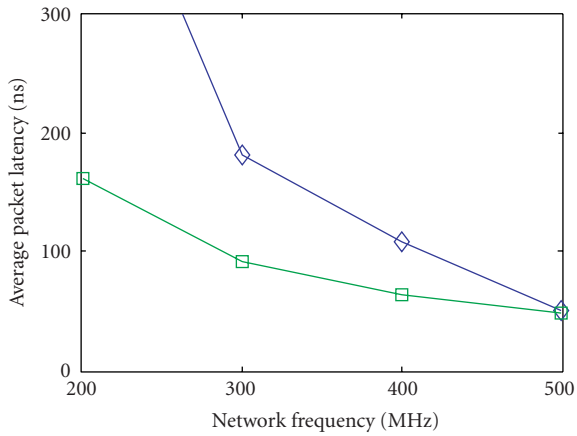
Figure 5: Mapped onto a mesh NoC. The edges of the mesh are numbered.

Table 2: Sample paths.

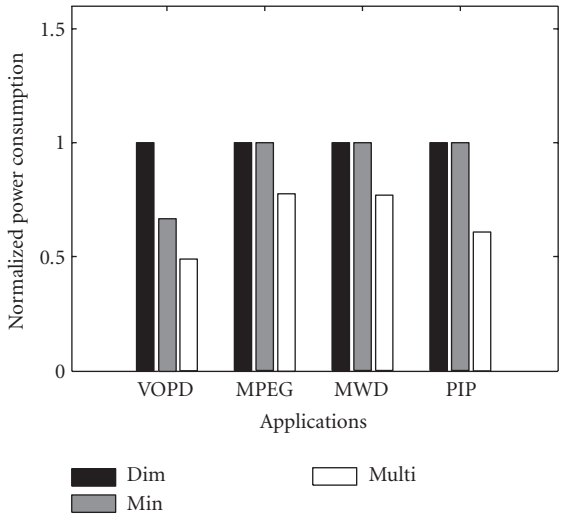
Comm.	Source	Dest.	Paths (edges traversed)
1	au	sdram	32-12, 10-29
2	mcpu	sdram	12, 23-10-29, 33-14-21
3	upsamp	sram2	27, 3-30-13
4	risc	sram2	19, 7-22-13

7.3. Comparisons with single-path routing

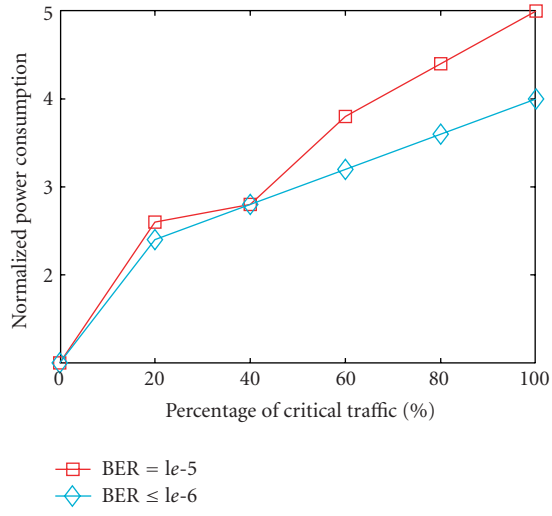
The network power consumption for the various routing schemes for different applications is presented in Figure 6(b).



(a) Average latency for MPEG NoC



(b) Routing effects on applications



(c) Effect of error resiliency

FIGURE 6: (a) Performance of routing schemes for MPEG NoC. (b), (c) Effect of routing and fault-tolerance on NoC power consumption.

The numbers are normalized with respect to the power consumption of dimension-ordered routing. We use several benchmark applications for comparison: *Video Object Plane Decoder (VOPD-mapped onto 12 cores)*, *MPEG decoder (MPEG-12 cores)*, *Multi-Window Display application (MWD-12 cores)* and *Picture-in-Picture (PIP-8 cores) application*. Description of the traffic characteristics of the applications is presented in [15]. Without loss of generality, we assume that the applications are mapped onto mesh topologies using the tool from [15], although the multipath routing strategy can be used for any topology. By using the proposed routing scheme, on average we obtain 33.5% and 27.52% power savings compared to the *dimension ordered* and *minimum path routing*, respectively. The total run time for applying our methodology (includes the run time for path selection algorithms for all commodities and for solving the resulting LP) is less than few minutes for all the benchmarks, when run on a 1 GHz Sun workstation.

7.4. Effect of fault-tolerance support

Adding support for resiliency against a single-path permanent failure for each commodity of the MPEG NoC resulted in a 2.33× increase in power consumption of the base NoC. Please note that the power overhead reported is for the worst-case scenario, where every communication flow experiences a single path failure. The amount of power overhead incurred in achieving fault-tolerance against temporary errors depends on the transient bit-error rate (β_t) of each link and the amount of data that is critical and needs replication. The effect of both factors on power consumption for the MPEG decoder NoC is presented in Figure 6(c). The power consumption numbers are normalized with respect to the base NoC power consumption (when no fault-tolerance support is provided). As the amount of critical traffic increases, the power overhead of packet replication is significant. Also, as the bit-error rate of the NoC increases (higher BER value in the figure, which imply a higher probability of bit-errors happening in the NoC), the amount of power overhead increases. We found that for all BER values lower than or equal to 1*e*-6, having a single duplicate for each packet was sufficient to provide the required MTTF of 5 years.

8. CONCLUSIONS

Routing packets across multiple paths can reduce network congestion and bottlenecks, which translates to reduced NoC frequency and power requirements or improved performance. Traditional multipath schemes require large packet reorder buffers at the receivers to provide in-order delivery. The reorder buffers have large area, power overhead, and deterministically sizing them is infeasible in practice. In this work, we have presented a multipath routing strategy that guarantees in-order packet delivery at the receiver. We introduced a methodology to find paths for the routing strategy and to split the application traffic across the paths to obtain a network operation with minimum power consumption. With technology scaling, reliable operation of on-chip

wires is also rapidly deteriorating and various transient and permanent errors can affect them. With the proposed multi-path routing strategy, we explored the use of spatial and temporal redundancy to tolerate transient as well as permanent errors occurring on the NoC links. Our method results in large NoC power savings for several SoC designs when compared to traditional single-path systems. In the future, we plan to extend the methods to implement source-based dynamic routing strategies that do not require reorder buffers at the receiver.

ACKNOWLEDGMENTS

This work is supported by the US National Science Foundation (NSF, contract CCR-0305718), the Swiss National Science Foundation (FNS, Grant 20021-109450/1), and Spanish Government Research Grant (TIN2005-5619). It is also supported by a Grant by STMicroelectronics for University of Bologna. The authors would also like to thank Alexandru Susu of EPFL for his useful comments on the work.

REFERENCES

[1] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] D. Wingard, "MicroNetwork-based integration for SOCs," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 673–677, Las Vegas, Nev, USA, June 2001.

[3] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '00)*, pp. 250–256, Paris, France, March 2000.

[4] K. Goossens, J. Dielissen, O. P. Gangwal, S. G. Pestana, A. Rădulescu, and E. Rijpkema, "A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '05)*, vol. 2, pp. 1182–1187, Munich, Germany, March 2005.

[5] S. Kumar, A. Jantsch, J.-P. Soininen, et al., "A network on chip architecture and design methodology," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI '02)*, pp. 105–112, Pittsburgh, Pa, USA, April 2002.

[6] F. Karim, A. Nguyen, S. Dey, and R. Rao, "On-chip communication architecture for OC-768 network processors," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 678–683, Las Vegas, Nev, USA, June 2001.

[7] D. Bertozzi, A. Jalabert, S. Murali, et al., "NoC synthesis flow for customized domain specific multi-processor systems-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113–129, 2005.

[8] H. Jingcao and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '03)*, pp. 688–693, Munich, Germany, March 2003.

[9] F. Angiolini, P. Meloni, S. Carta, L. Benini, and L. Raffo, "Contrasting a NoC and a traditional interconnect fabric with layout awareness," in *Proceedings of Design, Automation and Test in Europe (DATE '06)*, vol. 1, pp. 124–129, Munich, Germany, March 2006.

[10] V. Karamcheti and A. A. Chien, "Do faster routers imply faster communication?" in *Proceedings of the 1st International Workshop on Parallel Computer Routing and Communication (PCRCW '94)*, vol. 853 of *Lecture Notes in Computer Science*, pp. 1–15, Springer, Seattle, Wash, USA, May 1994.

[11] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. De Micheli, "Analysis of error recovery schemes for networks on chips," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 434–442, 2005.

[12] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 2, pp. 896–901, Paris, France, February 2004.

[13] A. Hansson, K. Goossens, and A. Rădulescu, "A unified approach to constrained mapping and routing on network-on-chip architectures," in *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and Systems Synthesis (CODES+ISSS '05)*, pp. 75–80, Jersey City, NJ, USA, September 2005.

[14] S. Murali, L. Benini, and G. De Micheli, "Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees," in *Proceedings of the 12th Asia and South Pacific Design Automation Conference (ASP-DAC '05)*, pp. 27–32, Shanghai, China, January 2005.

[15] S. Murali and G. De Micheli, "SUNMAP: a tool for automatic topology selection and generation for NoCs," in *Proceedings of Design Automation Conference (DAC '04)*, pp. 914–919, San Diego, Calif, USA, June 2004.

[16] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, "A low latency router supporting adaptivity for on-chip interconnects," in *Proceedings of the 42nd Design Automation Conference (DAC '05)*, pp. 559–564, Anaheim, Calif, USA, June 2005.

[17] H. Jingcao and R. Marculescu, "DyAD - smart routing for networks-on-chip," in *Proceedings of the 41st Design Automation Conference (DAC '04)*, pp. 260–263, San Diego, Calif, USA, June 2004.

[18] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, San Francisco, Calif, USA, 2003.

[19] W. J. Dally, P. P. Carvey, and L. R. Dennison, "Architecture of the avici terabit switch/router," in *Proceedings of Hot-Interconnects VI*, pp. 41–50, Stanford, Calif, USA, August 1998.

[20] C. B. Stunkel, D. G. Shea, B. Abali, et al., "The SP2 communication subsystem," Tech. Rep., IBM, Yorktown Heights, NY, USA, August 1994.

[21] Y. Aydogan, C. B. Stunkel, C. Aykanat, and B. Abali, "Adaptive source routing in multistage interconnection networks," in *Proceedings of the 10th International Parallel Processing Symposium (IPPS '96)*, pp. 258–267, Honolulu, Hawaii, USA, April 1996.

[22] R. Hegde and N. R. Shanbhag, "Towards achieving energy-efficiency in presence of deep submicron noise," *IEEE Transactions on VLSI Systems*, vol. 8, no. 4, pp. 379–391, 2000.

[23] D. Bertozzi, L. Benini, and G. De Micheli, "Error control schemes for on-chip communication links: the energy-reliability tradeoff," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 818–831, 2005.

[24] R. Marculescu, "Networks-on-chip: the quest for on-chip fault-tolerant communication," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI '03)*, pp. 8–12, Tampa, Fla, USA, February 2003.

[25] H. Zimmer and A. Jantsch, "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip," in *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '03)*, pp. 188–193, Newport Beach, Calif, USA, October 2003.

[26] F. Worm, P. Thiran, P. Ienne, and G. De Micheli, "An adaptive low-power transmission scheme for on-chip networks," in *Proceedings of the 15th International Symposium on System Synthesis (ISSS '02)*, pp. 92–100, Kyoto, Japan, October 2002.

[27] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Fault tolerant algorithms for network-on-chip interconnect," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI '04)*, pp. 46–51, Lafayette, La, USA, February 2004.

[28] S. Manolache, P. Eles, and Z. Peng, "Fault and energy-aware communication mapping with guaranteed latency for applications implemented on NoC," in *Proceedings of the 42nd Design Automation Conference (DAC '05)*, pp. 266–269, Anaheim, Calif, USA, June 2005.

[29] S. Murali, D. Atienza, L. Benini, and G. De Micheli, "A multi-path routing strategy with guaranteed in-order packet delivery and fault-tolerance for networks on chip," in *Proceedings of the 43rd ACM/IEEE Design Automation Conference (DAC '06)*, pp. 845–848, San Francisco, Calif, USA, July 2006.

[30] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, "A methodology for design of application specific deadlock-free routing algorithms for NoC systems," in *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '06)*, pp. 142–147, Seoul, Korea, October 2006.

[31] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, Cambridge, Mass, USA, 1990.

[32] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, New York, NY, USA, 1994.

[33] http://www.ocpip.org/.

Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 90941, 15 pages
doi:10.1155/2007/90941

Research Article

Network Delays and Link Capacities in Application-Specific Wormhole NoCs

Zvika Guz, Isask'har Walter, Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny

Electrical Engineering Department, Technion–Israel Institute of Technology, Technion city, Haifa 32000, Israel

Received 15 November 2006; Accepted 6 February 2007

Recommended by Maurizio Palesi

Network-on-chip- (NoC-) based application-specific systems on chip, where information traffic is heterogeneous and delay requirements may largely vary, require individual capacity assignment for each link in the NoC. This is in contrast to the standard approach of on- and off-chip interconnection networks which employ uniform-capacity links. Therefore, the allocation of link capacities is an essential step in the automated design process of NoC-based systems. The algorithm should minimize the communication resource costs under Quality-of-Service timing constraints. This paper presents a novel analytical delay model for virtual channeled wormhole networks with nonuniform links and applies the analysis in devising an efficient capacity allocation algorithm which assigns link capacities such that packet delay requirements for each flow are satisfied.

1. INTRODUCTION

Network-on-Chip (NoC) is a novel communication paradigm for MultiProcessor Systems-on-Chip (MPSoCs). NoCs provide enhanced performance and scalability, in comparison with previous communication architectures (e.g., dedicated point-to-point signal wires, shared buses, or segmented buses with bridges) [1, 2]. The advantages of NoC are achieved thanks to efficient sharing of wires and a high level of parallelism. Many MPSoCs use specialized application-specific computational blocks, and they require heterogeneous, application-specific communication fabrics. These systems operate under typical precharacterized information-flow patterns, which can often be classified into a few "use-cases" [3] where throughput and delay requirements are specified for data-flows from sources to destinations in the system.

Application-specific NoC generation has been addressed previously [4–10]. The NoC can be customized for a particular application-specific MPSoC through an automatic network design phase [4, 10–14]. Most previous research on NoC customization deals with the selection of network topology, module placement and routing scheme to accommodate the expected application-specific data traffic patterns, assuming that all links in the NoC are identical (e.g., [15–18]). In this paper, the application-specific customization is extended to include allocation of bandwidth for each link in the network according to expected traffic flows and their delay requirements. It should be noted that the problem of capacity allocation is independent of the aforementioned NoC challenges and can be used in conjunction with other optimization procedures.

The use of links with uniform capacities, typical of multicomputer interconnect, is unsuitable for application-specific NoC. In multicomputer systems, there is often little or no knowledge about the dynamic bandwidth and timing requirements of the tasks being executed, or about their assignment to a physical processing unit. In addition, in an off-chip network dynamic routing has an acceptable cost and loads can be equally distributed among multiple paths.

SoC traffic is usually more heterogeneous, in terms of both bandwidth and delay requirements. SoC is also subjected to detailed specifications describing the traffic and timing restrictions of each communicating pair, hence in many cases all communication characteristics are known at design time. At the same time, due to more restrictive cost considerations, solutions such as dynamic routing and load distribution are less appealing and instead fixed shortest-path routing is typically preferred in order to minimize router and network interface logic and communication power dissipation [4, 10, 12, 13, 15, 19]. As a result, different links in the NoC must carry significantly different data

rates and therefore different capacities should be allocated to them. In addition, different latency requirements for different data flows should affect the allocation of extra capacity to some critical links.

The goal of capacity allocation is to minimize the network *cost* (in terms of *link area* and *power*) while maintaining acceptable *packet delays* for the specific system communication demands [4]. This step in the design process is similar to timing closure in traditional chip design, where critical path drivers are often upsized and noncritical drivers are downsized for saving power. However, in an NoC, critical timing paths cannot be uniquely matched to dedicated signal wires, since multiple messages share the network links. In an NoC, timing closure must be achieved by adequate bandwidth allocation to each of the network links: insufficient allocation will not meet performance requirements, while lavish allocation will result in excessive power consumption and area. During the design process, the network architect can control the physical link capacity by setting the number of parallel wires it is composed of, or by tuning its clock frequency. If the network is implemented asynchronously, repeaters can be placed to regulate the throughput of the link.

Wormhole routing [20] is an increasingly common interconnect scheme for NoC as it minimizes communication latencies, requires small buffer space and is relatively simple to implement. However, performance evaluation and customization process of wormhole-based NoCs heavily rely on simulations as no existing analysis accounts for the combination of heterogeneous traffic patterns and virtual channels [21]. Unfortunately, these are both fundamental characteristics of NoC interconnect. The use of simulations makes the task of searching for efficient capacity allocation computationally extensive and does not scale well with the size of the problem. On the other hand, a detailed exact analytical solution of a complex NoC is intractable and simplistic approximations may lead to inaccurate results.

Two contributions are described in this paper. First, a novel delay analysis for wormhole-based NoCs is presented. The model approximates the network behavior under a wide range of loads. Given any system (in terms of topology, routing, and link capacities) and its communication demands (in terms of packets length and generation rate), the analysis estimates the delay experienced by every source-destination pair. To the best of our knowledge, this is the first analysis of a wormhole network with nonuniform link capacities. Second, an algorithm that applies the delay analysis for efficiently allocating capacities to network links is described. Simulation is only used for final verification and fine tuning of the system. This methodology considerably decreases the total NoC cost and significantly improves the speed of the customization process.

The rest of this paper is organized as follows: in Section 2, the capacity allocation problem is formalized. In Section 3, we present and evaluate a delay model for heterogeneous wormhole networks with multiple virtual channels and different link capacities. The capacity allocation algorithm is discussed in Section 4, while Section 5 presents design examples and simulation results. Finally, Section 6 concludes the paper. In the appendix, we describe QNoC, a QoS-oriented architecture for on-chip networks that was used for evaluation of our analysis and capacity allocation technique.

Given:
F
$\forall f \in F$: m^f, λ^f, T^f_{REQ}
$\forall$ *link* j, assign link capacity (C_j) s.t.:
$\forall f \in F$: $T^f \leq T^f_{\text{REQ}}$
$\sum C_j$ is minimal

Figure 1: Capacity assignment minimization problem.

2. THE LINK CAPACITY ALLOCATION PROBLEM

If all links in an application-specific NoC are given the same capacity, some of them might be much larger than needed, consuming unnecessary power and area. Using an analogy to timing closure in classical VLSI design, this resembles instantiating the strong drivers that are needed in the critical path, throughout the entire chip.

Alternatively, an efficient capacity allocation assigns just enough capacity to each of the network links such that all flows in the system meet their delay requirements while the total capacity invested in the network is minimized.

In order to formalize the minimization problem, we introduce the following notation.

F = *The set of all flows, from every source module.* $1 \leq s \leq N$ *to every destination module* $1 \leq d \leq N$.
f^i = *A flow from the set* F.
m^i = *The mean packet length (number of flits) of flow.* $f^i \in F$ *[flits]*.
λ^i = *Average packet generation rate of flow* $f^i \in F$. *[packets/second]*.
T^f_{REQ} = *The required mean packet delivery time for flow f.*
C_j = *Capacity of link j [bits/second]*.

The capacity assignment minimization problem can be formalized as in Figure 1.

Naïve allocations, based only on flow bandwidth requirements, do not yield good solutions. For example, if the capacity assigned to each link is equal to the average data rate of the flows it carries, any temporary deviation from the average will cause network congestion and unacceptable delays. Therefore, the links must be over.3provisioned with bandwidth and should be designed to operate at a fairly low average utilization rate. If utilization rates are too high, network delays will grow and the latency of some flows will be too long. Assignment of link capacities such that the utilization rates of all links are equal may be a reasonable allocation heuristic for homogeneous systems, in which all flows have similar characteristics. However, such an allocation might lead to extreme waste of resources in a heterogeneous scenario. For example (Figure 2), consider a flow f^1 which injects 1 Gb/s into some network link (00→01),

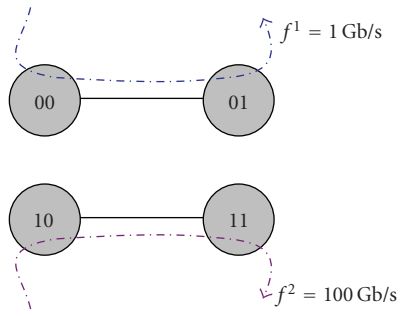


Figure 2: Simple example of a system in which equal utilization capacity assignment is wasteful.

but has a critical latency, such that a bandwidth of 10 Gb/s is required in that link (10% utilization). Assume that another flow f^2 injects 100 Gb/s into another link, and its latency requirement permits 200 Gb/s (50% utilization) in that link (10→11). Enforcing equal link utilization throughout the network will require 10% utilization everywhere and will unnecessarily increase the second link capacity to 1000 Gb/s.

Intuitively, the capacity of some links can be reduced based on latency requirements so that delivery times are stretched to the maximum allowed. Therefore, any efficient capacity allocation algorithm must be latency aware. Hence, an efficient model for evaluating the delays of packets through the network is required. The static delay model derived in Section 3 will be used to direct our capacity allocation algorithm.

3. WORMHOLE DELAY MODEL

In a wormhole network, packets are divided into a sequence of flits which are transmitted over physical links one by one in pipeline fashion. A hop-to-hop credit mechanism assures that a flit is transmitted only when the receiving port has free space in its input buffer. Performance of such networks has been studied extensively in the past. While several analytical studies have evaluated the mean latency of packets in wormhole networks (e.g., [22–29]), no previous work analyzes wormhole networks with heterogeneous link capacities, which is a fundamental characteristic of NoC that makes it cost effective [4]. Moreover, existing models evaluate the mean latency over all flows in the system. Since each SoC flow must meet its delay requirement, a perflow analysis is necessary. In addition, no previous work analyzes networks with both virtual-channels and nonuniform source-destination communication patterns. For example, [22] presents an analytical model of a wormhole network with an arbitrary number of virtual channels per link. Though comprehensive, it assumes that messages destinations are uniformly distributed. In [23], the model does reflect nonuniform traffic but does not address networks with multiple virtual channels.

Consequently, a new wormhole delay analysis is called for, one that captures the above fundamental characteristic of SoC. In the following subsections, we present our wormhole analysis. Section 3.1 introduces the network model and notations, Section 3.2 presents the analysis and Section 3.3 evaluates the analysis.

3.1. The network model

The time to deliver a packet between a specific source-destination pair is composed of the source queuing time and the time it take the packet to traverse the network (hereafter referred to as *network time*). In a wormhole network, network time is composed of two components [29]: the time it takes the head flit to reach the destination module (*path acquisition time*) and the time it takes the rest of the packet to exit the network (*transfer time*); path acquisition time is affected by the complex interaction among different flows in the system and transfer time is affected by other flows sharing the same links (link capacity is time multiplexed among all virtual channels sharing the link).

Since NoC delay/cost tradeoffs are different from those of off-chip networks and since performance is a key issue, we assume that NoCs will be designed to operate under a relatively moderate load. Consequently, for the sake of simplicity and computational efficiency, our analysis addresses low to medium loads and does not attempt to achieve high accuracy under very high utilizations.

Previous research [21] has showed that adding virtual channels increases the maximal network throughput. We assume that physical links are split into an adequate number of virtual channels. In particular, we assume that a head flit can acquire a VC instantaneously on every link it traverses.

Our analysis focuses on the transfer of long packets, that is, packets which are composed of a number of flits significantly larger than the number of buffers along their path. From the simulations presented in [4], it is clear that such packets (termed the Block Transfer class of service) are the ones that place the most stringent demand on NoC resources and hence dominate the bandwidth requirements. A delay analysis capturing the latencies of short packet in wormhole NoCs which is valuable for the system architect is left for future work.

We consider a wormhole deadlock-free fixed routing network that is composed of N routers connected by unidirectional links. The packets that constitute the traffic of each source-destination pair identify a *flow*.

Our model uses the following assumptions.

(1) Each flow generates fixed length packets using a Poisson process (bursty traffic can be modeled by using artificially larger packet size).
(2) Sources have infinite queues, and sinks immediately consume flits arriving at their destination.
(3) Routers have a single flit input queue per virtual channel.
(4) The propagation delay of flits through links and routers is negligible.
(5) Back pressure credit signal is instantaneous.

We will use the following additional notation to characterize the network:

l = *flit size [bits]*,
π^i = *the set of links composing the fixed path of flow* f^i,
π^i_j = *the set of links that are subsequent to link* j *on, flow* i*'s path (a suffix of the path* π^i*)*.

The following notation is used in order to analyze the packets delay:

T^i = *the mean packet delivery time of packets of flow* f^i (*the average time elapsed since a packet is created until its last flit arrives at its destination*),
Q^i = *mean source queuing time of packet of flow* f^i (*the average time elapsed since the packet is created until it enters the network*),
T^i_{network} = *the mean network time of packets of flow* f^i (*the average time elapsed since the packet is inserted into the network until its last flit is received by the destination module*),
t^i_j = *the mean time to deliver a flit of flow i over link j (waiting for transmission and transmission times)*,
Λ^i_j = *the total flit injection rate of all flows sharing link* j, except flow f^i *[flits/second]*.

3.2. Wormhole delay analysis

We approximate the source queuing time using the M/D/1 model [30]:

$$Q^i = \frac{1}{2 \cdot (1/T^i_{\text{network}} - \lambda^i)} - \frac{T^i_{\text{network}}}{2} \quad . \tag{1}$$

Clearly, when a flow does not share any of its links with other flows, (1) is the exact mean queuing time, since the time required to deliver a packet through the network (T^i_{network}) is deterministic. When a packet might be interleaved with other flits within the network, the service time is not deterministic anymore, and the assumptions of the M/D/1 model do not hold. However, thorough simulation (presented in Sections 3.3 and 5) show that (1) is a good approximation for the queuing time even for flows that are frequently multiplexed.

The network time of a packet in a wormhole network resembles a pipeline traversal. When the number of parts composing the packet is considerably larger than the number of pipeline stages, the latency (the time it takes the first bits to exit the pipe) is insignificant compared with the total time, which in this case is mostly affected by the pipeline's throughput. Since packets are assumed to be considerably longer than the buffers along their path, and since each head flit is assumed to instantaneously acquire a virtual channel on every link it arrives at, we ignore path acquisition time and approximate the transmission time only.

As in a classic pipeline, the transfer time is dominated by the stage with the smallest service rate. Since flits of different flows are interleaved on links, the approximation of t^i_j should account for the transmission time of other flits on the same link. We use a modification of the basic M/M/1 modeling [30] as an approximation of the flit interleaving delay,

$$t^i_j = \frac{1}{1/l \cdot C_j - \Lambda^i_j} \quad , \tag{2}$$

where Λ^i_j is the bandwidth consumed by all flows other than flow i on link j. Formally,

$$\Lambda^i_j = \sum_{f \mid j \in \pi^f \wedge f \neq i} \lambda^f \cdot m^f \quad . \tag{3}$$

Equation (2) models the mean interleaving delay experienced by flits of flow i on link j as a simple queue, without accounting for the bandwidth consumed by packets of flow i itself. This modification is based on the observation that flits of a flow are interleaved on a physical link due to the delivery of packets that belong to other flows only.

By substituting (3) into (2) we get

$$t^i_j = \frac{l}{C_j - l \cdot \sum_{f \mid j \in \pi^f \wedge f \neq i} \lambda^f \cdot m^f} \quad . \tag{4}$$

The total network time, which is dominated by the hop with the longest delay, can then be written as

$$T^i_{\text{network}} \simeq m^i \cdot \max\left(t^i_j \mid j \in \pi^i\right) \quad . \tag{5}$$

The above approximation does not capture interlink dependencies and is generally too optimistic for medium and high loads. Wormhole link loads affect each other mainly by the back-pressure mechanism: a flit must wait for the arrival of a credit for its virtual channel from the following link. Therefore, flit delivery time over a link (t^i_j) is affected by the delivery time in subsequent (downstream) links on the flow's path. To reflect the effect of flit delay on other (upstream) links, we replace (4) by the following expression which accounts for these interlink dependencies. Our simulations (Sections 3.3 and 5) show that this simple expression successfully estimates the resulting link delay:

$$\tilde{t}^i_j = t^i_j + \sum_{k \mid k \in \pi^i_j} \frac{l \cdot \Lambda^i_k}{C_k} \cdot \frac{t^i_k}{\text{dist}^i(j,k)} \quad , \tag{6}$$

where $\text{dist}^i(k,j)$ is the distance (measured in number of hops) between link j and k on the path of flow i. Formally written as

$$\text{dist}^i(k,j) = \left| \frac{\pi^i_j}{\pi^i_k} \right| \quad . \tag{7}$$

Equation (6) approximates the delay experienced by flits of flow i on link j by adding to the basic flit delay (t^i_j) a term that takes into account the cumulative effect of the delays of subsequent links along the path. Each subsequent link delay is weighted by two factors: the links' distance from link $j(\text{dist}^i(k,j))$ and the factor by which the link is utilized by flows other than flow i itself ($l \cdot \Lambda^i_k / C_k$).

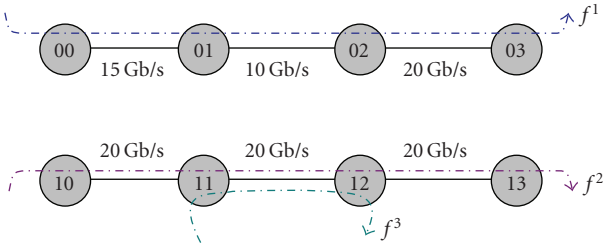


Figure 3: NoC simple example.

As explained above, the mean total network time of each flow is calculated using the longest interleaving delay on its path. Therefore, (5) is replaced by

$$T^i_{\text{network}} \simeq m^i \cdot \max\left(\tilde{t}^i_j \mid j \in \pi^i\right) \ . \tag{8}$$

Finally, the total packet delivery time is equal to

$$T^i = Q^i + T^i_{\text{network}} \ . \tag{9}$$

3.3. Model characterization

In this section, we evaluate the accuracy of the wormhole network delay model by comparison with detailed network simulation of example systems. First, Section 3.3.1 presents a simple example to demonstrate the analysis. Then, Section 3.3.2 considers a classic scenario of 4×4 homogeneous systems. We take QNoC (described in the appendix) as an example of an NoC architecture. The OPNET simulator was extended to include a full NoC simulation based on the QNoC architecture, including all complex dependencies between different flows in wormhole networks (due to particular virtual channel assignment schemes, finite router queues, etc.).

3.3.1. Simple example

To demonstrate the analysis we present a simple and intuitive example, depicted in Figure 3.

The example system consists of three flows: flow f^1 does not share any of its links with any other flow. Its path (from source 00 to destination 03) is composed of links with different capacities as shown. Flow f^2, running from source 10 to destination 13, shares one of its links (11→12) with flow f^3 (running from source 11 to destination 12). Consequently, while the service time for flow f^1 is constant, the transmission times of flow f^2 and f^3 are affected by the contention on their shared link. All of the links on their path have the same capacity of 20 Gb/s.

Figure 4 presents the normalized average packet delay of the three flows as a function of varying system load, created by changing the generation rate of each flow. As expected, the network delay (path acquisition time plus transfer time) of flow f^1 (Figure 4(a)) is unaffected by the varying utilization, because its packets are never interleaved on a link, and can always use the entire capacity of the links on its path. However, when utilization is high, the packet delay increases due to source queuing latency. Figure 4(a) also shows that the analytical terms for both the network delay (8) and queuing time (1) provide good prediction of the simulation results, as the delivery time is dictated by the path's lowest capacity link and the source queue follows M/D/1 assumptions.

Figures 4(b) and 4(c) show the delays of flows f^2 and f^3, respectively. Unlike flow f^1, the network delays of these flows grow as link utilization increases, since the available residual capacity on the common link decreases. As expected, both flows also experience source queuing delay, which becomes significant at high utilization. For all three flows, the analysis closely predicts the network and queuing delays measured by simulations.

3.3.2. Homogeneous all-to-all example

To demonstrate the accuracy of our analytical model we now consider a homogeneous system in which every module injects the same amount of bandwidth into the network and packet destinations are chosen randomly with uniform distribution. While this is not typical of SoC, this scenario is very often used to evaluate wormhole networks [21, 22, 25–29]. We compare the analytical delay model with simulation results for a varying utilization factor, by using a wide range of uniform capacity allocation vectors.

Our network, illustrated in Figure 5, is comprised of a regular four by four two dimensional mesh with symmetric XY routing [4]. All links have identical capacities; packets of each flow are generated by a Poisson process, with a mean rate $\lambda = 1/0.00048$ [packets/s]; packets consist of 500 flits, each flit is 16 bit long.

As can be seen in Figure 6, although all flows inject the same bandwidth, the different distance and different aggregated load of links along their paths result in a large variation in packet delays. Assuming that all flows have identical requirements, the mean packet delivery time of some flows is much lower than needed. This slack can be trimmed by a more efficient link capacity scheme.

Figure 7 compares the mean end-to-end packet delay predicted by the analytical model with simulation results, as a function of the utilization level of the most utilized link (i.e., for a wide range of uniform capacity allocations). The analytical model closely predicts the simulation results for a wide range of loads, way beyond the loads that are expected in a practical SoC (the mean absolute error reaches 8% when utilization is over 90%). The mean absolute error is presented in Figure 8.

Figure 9 zooms in on two specific flows, depicting the simulations and analysis end-to-end packet delays of flows 00→10 and 00→33, respectively. As can be seen from the figure, flow 00→10, which runs on a single link shared by only two other flows (according to the symmetric XY routing), suffers a moderate increase in its end-to-end delay as system load increases. On the other hand, the delay of flow 00→33 is much more sensitive to the system load, experiencing significant degradation in its delay as the load increases. This

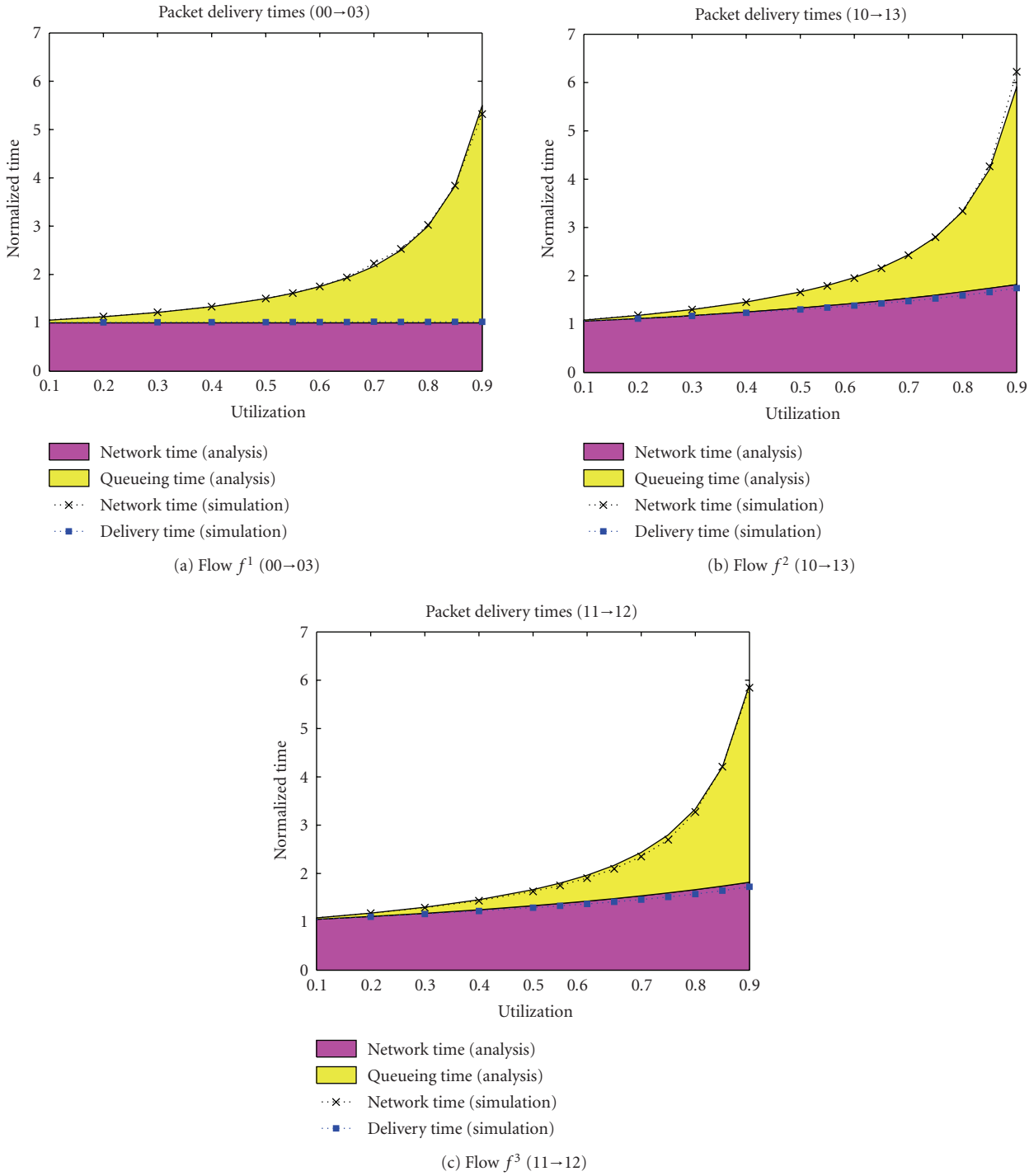


FIGURE 4: Simulation and analytical delay decomposition of flows f^1, f^2, and f^3. Horizontal axis marks the maximum link utilization on the flow's path; vertical axis is the delay normalized by delivery time in an unloaded network.

flow, running on multiple links, among them few of the most loaded links in the system, suffers from many contentions with other flows along its path. The higher loads result in many collisions, and hence significant increase in its total delay. Although these flows have different characteristics, the analysis closely predicts the resulting delays measured in simulations.

4. CAPACITY ALLOCATION

Similar to the TILOS sizing algorithm [31], which uses static timing analysis to optimize transistor sizes, our algorithm minimizes the resources allocated to network paths with relaxed timing requirement, and assigns extra network bandwidth to critical paths that need to be optimized.

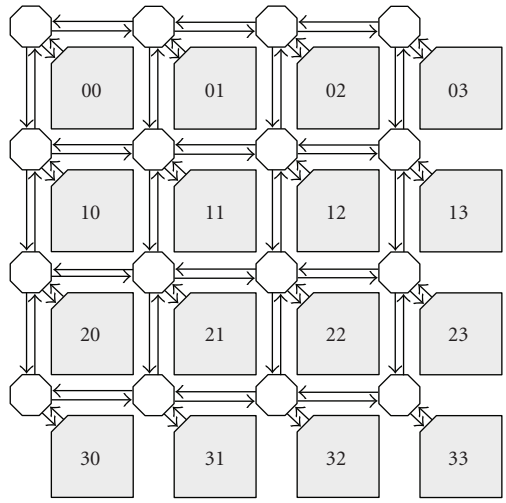


Figure 5: The homogeneous NoC example: a four-by-four mesh with 48 unidirectional interrouter links.

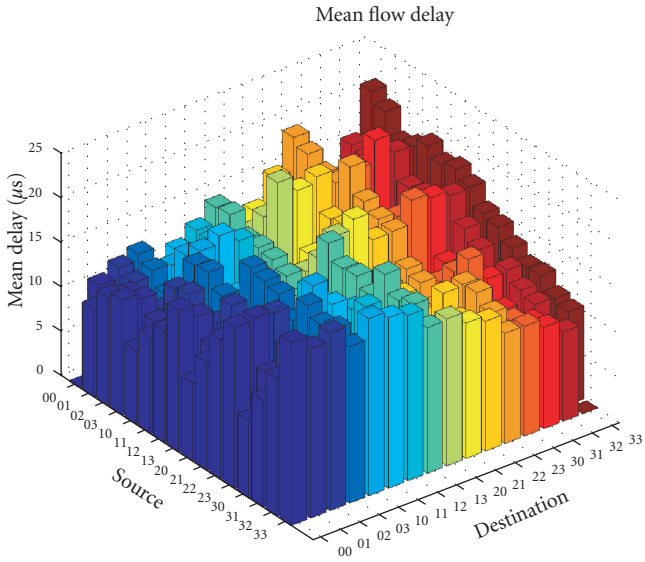


Figure 6: Simulated mean delay in the homogeneous system. Base plane axes mark the source and destinations module addresses as defined in Figure 5, respectively, and the vertical axis is the mean packet delivery time (longest delay is 2.5× longer than the shortest one).

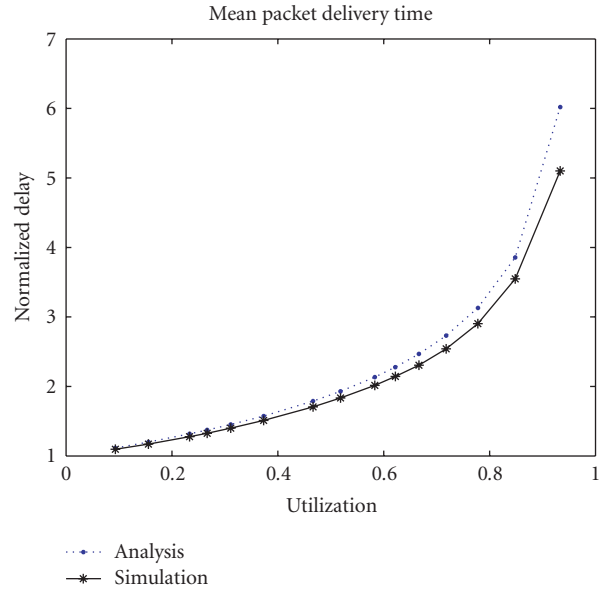


Figure 7: Model estimation and simulation results for the homogeneous system. Horizontal axis is the utilization of the most utilized link and the vertical axis represents the delay normalized by the delay in a zero-loaded system (i.e., where no other flows exist).

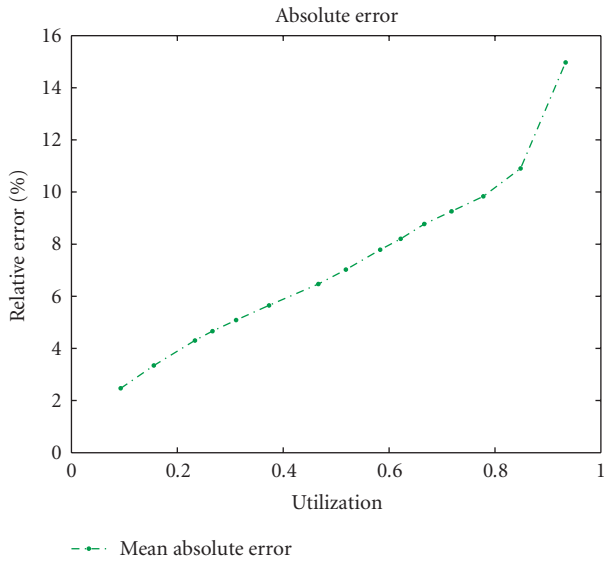


Figure 8: Mean absolute error in the homogeneous system.

However, unlike classical VLSI circuits in which each wire has a unique delay constraint, NoC links are shared by data flows with different timing requirements which must all be met.

Using the model presented in Section 3, all network delays can be computed directly, without resorting to simulation, for any set of network parameters. This computation can be performed in the inner loop of an optimization algorithm which searches for a low-cost capacity allocation for the NoC links such that all delay requirements are satisfied.

If the SoC is to handle multiple use-cases [3], the capacity allocation process should be performed for each case individually. If link capacities can be dynamically adjusted, the appropriate bandwidth should be selected at run-time. Else, the designer can statically set each link's capacity to the maximal bandwidth it requires in all use cases. This will assure that all timing requirements are met in all operating scenarios.

Our capacity allocation algorithm, specified in Algorithm 1, takes a greedy approach. The initialization phase

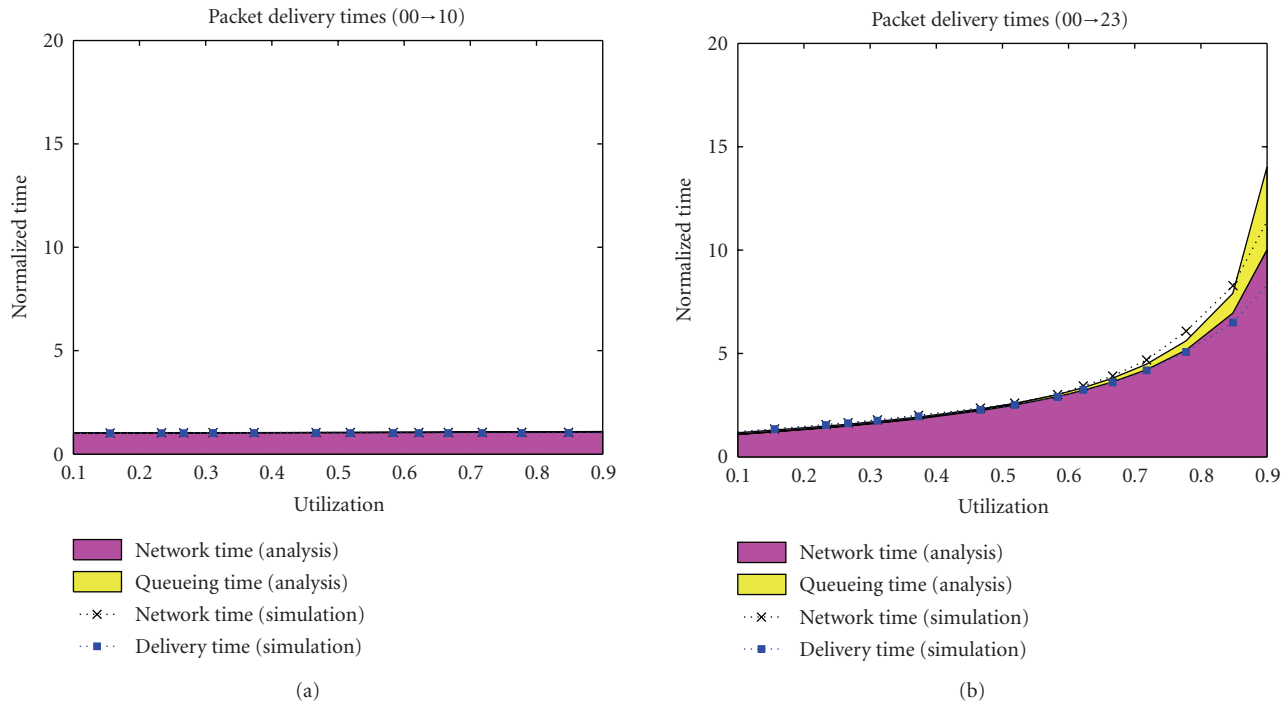


Figure 9: Model estimation and simulation results for the end-to-end delay of flow (a) 00→10 and (b) 00→33 in the homogenous system. The horizontal axis shows utilization of the most utilized link in the system and the vertical axis represents the delay normalized by the delay in an unloaded system.

```
    /* assign initial capacities */
(1) foreach link e:
(2)   C_e ← Σ_{f∈F:e∈π^f} λ^f · m^f · l
(3) end foreach
(4) foreach flow f ∈ F:
    /* evaluate current packet delivery time */
(5)   T^f ← Delay_Model (C, f)
(6)   while (T^f > T^f_REQ)
        /* look for most sensitive link */
(7)     foreach e ∈ π^f:
(8)       ∀ j ≠ e: C̃_j = C_j
(9)     C̃_e = C_e + δ
(10)    T̃^f_e ← Delay_Model (C̃, f)
(11)    end foreach
      /* get most sensitive link */
(12)  e' = argmin_e {T̃^f_e}
      /* increase its capacity */
(13)  C_e' = C_e' + δ
(14)  end while
(15) end foreach
```

Algorithm 1: Capacity allocation algorithm.

allocates a minimal preliminary capacity to each link in the network (lines 1–3). The main loop (lines 4–15) analyses each source-destination flow separately. It first uses the delay model (Section 3) to approximate the flow's delay given the current capacity allocation vector (line 5). If the delay is longer than required (line 6), the algorithm allocates a small, predefined amount (δ) of extra capacity to the flow's path. It first temporarily increases the capacity of all links along the path separately (lines 7–11) and approximates the resulting packet delay. It then searches for the link with the largest gain from bandwidth addition, that is, the link for which adding capacity results in the shortest delay overall (line 12). The extra capacity is added only to that link (line 13). When the algorithm terminates, all flows meet their required delay. Since the algorithm handles flow-by-flow, considering for every flow only the links along its path for optimization, and since every iteration on the flow's links reduces the delay of the weakest link along that flow's path, the algorithm is bound to terminate with locally minimal allocation as its output. Investigation of detailed convergence properties and complexity analysis are left as future work.

In practice, the analytical model does not capture all complex dependencies and effects in a virtual channeled wormhole network. As a result, the mean delay of a few flows may be underestimated and the capacity of some links might be too small. Therefore, the resulting assignment is verified using network simulation, and some extra capacity is added to these flows' path.

5. DESIGN EXAMPLES

Two examples are presented, to demonstrate the delay model and the benefit of using the capacity allocation algorithm. Both examples exhibit nonuniform communication patterns, which are more realistic in SoC than the homogeneous

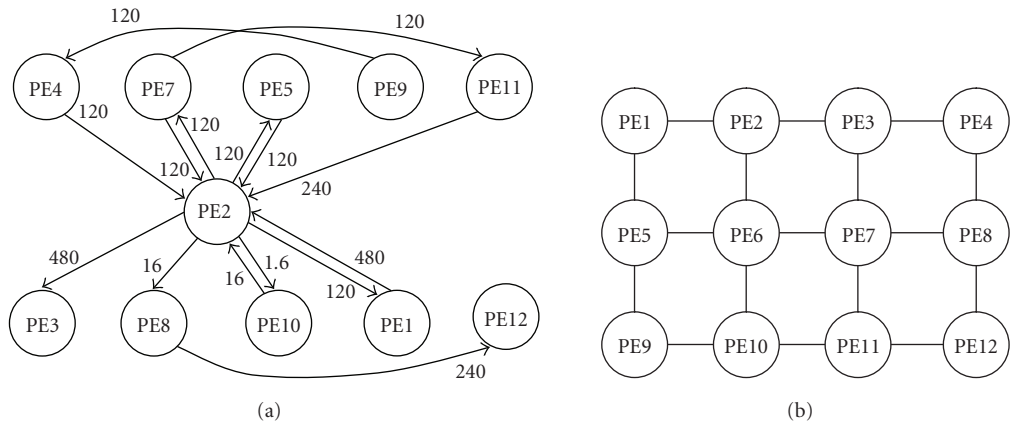


Figure 10: (a) Task graph and (b) placement for the DVD decoder system.

example used in Section 3. Modules communicate with only a subset of all possible destinations and different flows have different bandwidth and different delay requirements. In the first example, some modules send and receive packets from a single module (many-to-one and one-to-many patterns), emulating a SoC in which data is generated and destined at specific modules (e.g., SoCs in which there is a main CPU or a single cache memory, or SoCs that use an off-chip DRAM). The second example considers a video application, in which the main data path is pipelined, that is, data blocks are forwarded from module to module.

For each example, we apply the suggested capacity allocation algorithm and present its benefit over uniform link capacities assignment. We have implemented a tool that automatically assigns link capacities given the network topology, routing scheme, communication demands, and QoS requirements. This tool, which uses the aforementioned delay model and capacity allocation algorithm (described in Section 4), is to be used by the chip interconnect designer to minimize network resources. For simulation, we have used the OPNET-based NoC simulator described in Section 3.3.

Table 1: DVD decoder system.

Flow	Interarrival time (μs)	Packet length (flits)	Required delay (μs)
00→01	16.67	500	5
01→12	66.67	500	10
01→10	66.67	500	10
01→00	66.67	500	5
01→02	16.67	500	10
01→21	5 000.00	500	15
01→13	500.00	500	10
03→01	66.67	500	10
10→01	66.67	500	10
12→01	66.67	500	10
12→22	66.67	500	5
20→03	66.67	500	10
21→01	500.00	500	15
22→01	33.30	500	10
23→13	33.30	500	10

5.1. DVD video decoder example

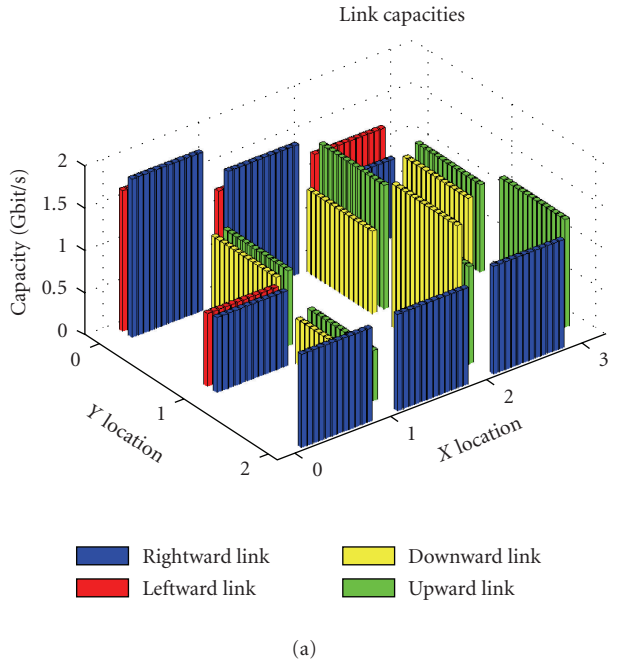
The first heterogeneous network comprises a regular three by four, two dimensional mesh with XY routing, implementing a DVD video decoder. Table 1 describes the different flow characteristics and delay requirements and Figure 10 presents the task graph and module placement.

We have compared the total capacity assigned by the capacity allocation algorithm with a uniform assignment that meets the same requirements. Figure 11 presents the capacities assigned to each link by the algorithm. While the uniform assignment requires a total of 41.8 Gb/s, the algorithm used only 25.2 Gb/s, achieving a 40% reduction of resources. Figure 12 shows that all flows meet their requirements and that the static analytical delay model adequately predicts the simulated latencies.

Figure 13 compares the packets delay slacks (i.e., the difference between the actual packet delay and the delay requirement) in the two allocations. Since in the uniform allocation all links are assigned identical capacity, the mean packet delays of some of the flows are much lower than required. On the other hand, the capacity allocation algorithm adjusts individual link capacities, thus reducing the slack and improving efficiency. Note that some flows may benefit from links with high capacity on their path that is needed to satisfy the requirements of other flows sharing the link. As a result slacks are still possible even in an optimal allocation.

5.2. Video processing application

The second example is a video processing application, based on the video object plane decoder (VOPD) system presented in [5]. The system, illustrated in Figure 14, has 15 data flows. The commutation characteristic (Table 2) and modules' placement (Figure 14(b)) were adapted from [5] and a reasonable set of delay requirements were added.



(a)

Link	Assigned capacity (Gb/s)	Link	Assigned capacity (Gb/s)
00→01	1.87	23→22	0
01→02	1.53	00→10	0
02→03	0.93	10→20	0
10→11	0.89	01→11	0.86
11→12	0	11→21	0.53
12→13	0	02→12	0.97
20→21	1.10	12→22	1.69
21→22	1.14	03→13	0.93
22→23	1.27	13→23	0
01→00	1.66	10→00	0
02→01	1.22	20→10	0
03→02	1.22	11→01	0.89
11→10	0.86	21→11	0.6
12→11	0	12→02	1.46
13→12	0	22→12	1.15
21→20	0	13→03	1.03
22→21	0	23→13	1.27

(b)

Figure 11: Links capacity assigned by the capacity allocation algorithm for the DVD system.

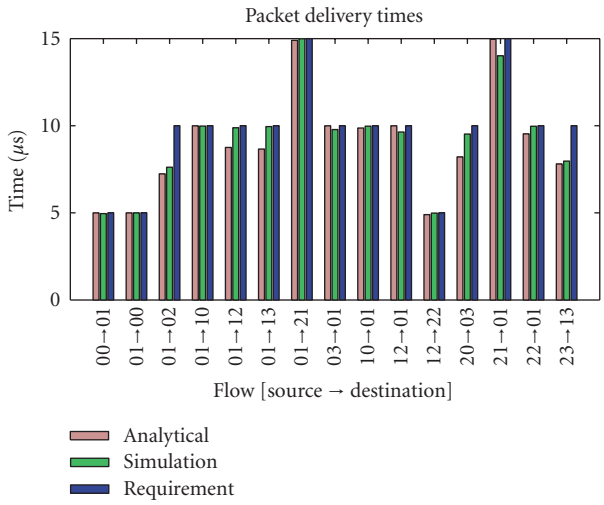


Figure 12: Flow delay (DVD decoder system).

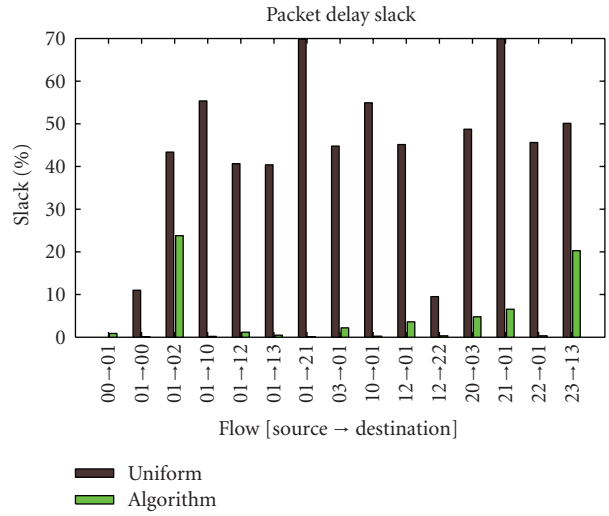
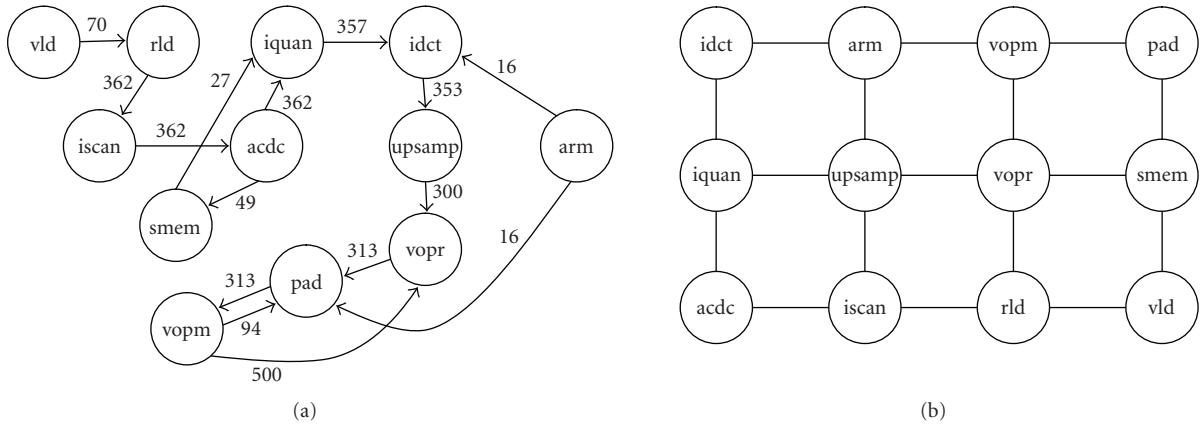


Figure 13: Mean packet delay slack as extracted from simulation, for uniform and algorithm-based allocation.

As in the previous example, we have compared the total capacity assigned by the capacity allocation algorithm with a uniform assignment that meets the same requirements.

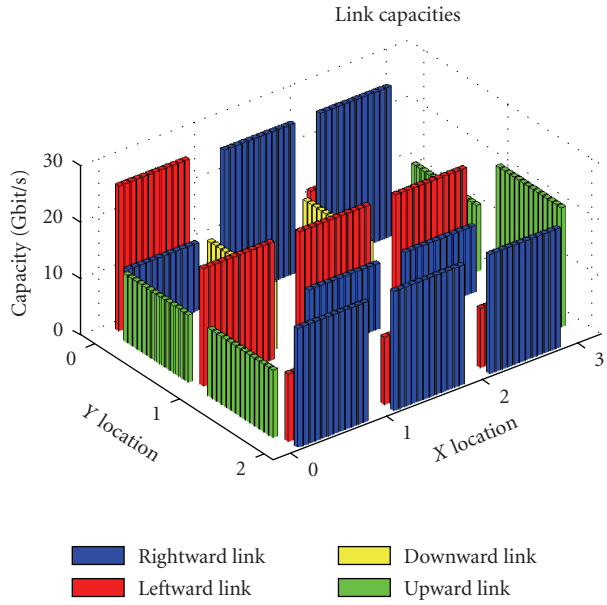
Figure 15 presents the capacities assigned by the algorithm to each link and Figure 16 shows the resulting delays. In this example, a uniform allocation that meets all latency requirements consumes a total of 640 Gb/s, while the algorithm used only 369 Gb/s, thus reducing total capacity by 40%. Unlike the DVD decoder example, here there are almost no slacks remaining. This is due to the fact that most links are not shared by multiple flows, and hence flows do not benefit from capacity allocated in order to satisfy the needs of other flows.

In both examples presented above the capacity allocation algorithm was able to achieve significant resource savings thanks to different bandwidth and requirements of the different flows, and thanks to the diversity in link loads. Since some flows have weaker requirements than others, it is possible to allocate relatively low capacity to some of the links, without causing any flow to violate its delay requirement. As



(a) (b)

Figure 14: VOPD (a) task graph and (b) placement.



(a)

Link	Assigned capacity (Gb/s)	Link	Assigned capacity (Gb/s)
00→01	12.03	23→22	10.54
01→02	26.59	00→10	0
02→03	26.59	10→20	0
10→11	0	01→11	12.03
11→12	11.66	11→21	0
12→13	11.78	02→12	12.75
20→21	21.17	12→22	0
21→22	21.17	03→13	0
22→23	21.17	13→23	0
01→00	25.67	10→00	11.95
02→01	0	20→10	12.01
03→02	11.73	11→01	0
11→10	20.88	21→11	0
12→11	20.88	12→02	0
13→12	20.88	22→12	0
21→20	11.98	13→03	11.78
22→21	11.98	23→13	21.17

(b)

Figure 15: Links capacity assigned by the capacity allocation algorithm for the VOPD system.

a result, heterogeneous systems are more likely for a substantial resource saving than homogeneous ones.

6. SUMMARY

Allocating different capacities to different network links is an important phase in the design process of application-specific NoC-based systems. A good assignment algorithm should allocate network resources efficiently so that QoS and performance requirements are met but total cost is minimized.

The paper made two novel contributions: first, a simple static timing analysis delay model was presented. The analysis captures virtual channeled wormhole networks with different link capacities and eliminates the reliance on simulations for timing estimation. The paper also introduced an allocation algorithm that greedily assigns link capacities using the analytical delay model, so that packets of each flow arrive within the required time. Using design examples, we showed the potential benefit of automated link capacity allocation in a typical NoC-based SoC design, where the traffic is heterogeneous and critical delay requirements vary significantly.

APPENDIX

QNoC ARCHITECTURE

We present QNoC (Quality-of-service NoC) [4] architecture as an example of network characteristics and NoC-based system design. Although we have used the QNoC architecture for evaluation of our analysis and capacity allocation

Table 2: VOPD application.

Flow	Interarrival time (μs)	Packet length (flits)	Required delay (μs)
00→11	0.6916	128	0.2
01→00	15.26	128	0.08
01→03	15.26	128	0.08
02→03	2.597	128	0.1
02→12	0.4883	128	0.2
03→02	0.78	128	0.2
10→00	0.6839	128	0.2
11→12	0.8138	128	0.2
12→03	0.78	128	0.2
13→10	9.042	128	0.1
20→10	0.6744	128	0.2
20→13	4.982	128	0.1
21→20	0.6744	128	0.2
22→21	0.6744	128	0.2
23→22	3.488	128	0.2

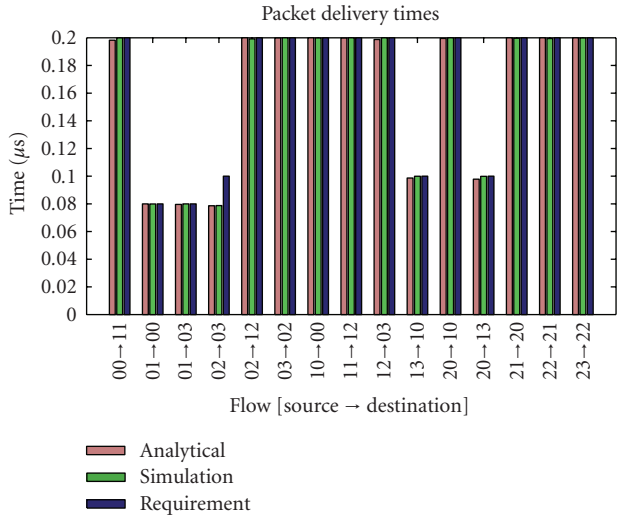


Figure 16: Flow delays (VOPD system).

technique, it should be noted that both the delay analysis (Section 3) and the allocation algorithm (Section 4) are applicable to other architectures, with different topologies and other fixed routing schemes.

Unlike traditional off-chip computer networks, which are built for future growth and compatibility with standards, on-chip networks can be designed and customized for an a priori known set of computing resources, given precharacterized traffic patterns and Quality-of-Service (QoS) requirements, while minimizing VLSI cost [4, 32, 33]. For example, topology generation, module placement, routing path selection and network link capacity allocation are performed at system design time, resulting in an efficient network implementation in terms of area and power.

The generic QNoC architecture is based on QoS wormhole packet routing and a planar irregular mesh topology

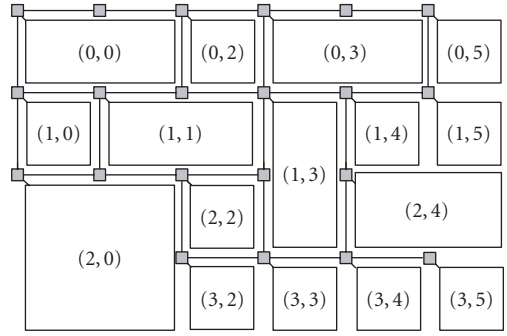


Figure 17: Example of an irregular mesh QNoC.

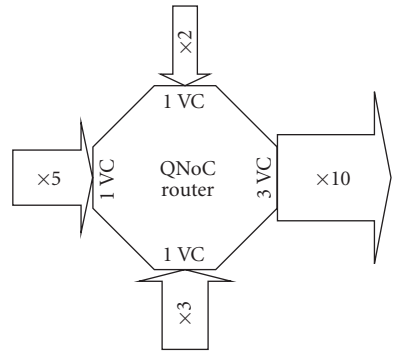


Figure 18: An example of an asymmetric network: multiple VCs are needed on the outgoing link.

(Figure 17). Our definition of an irregular mesh topology is identical to the full mesh including module addresses, except that some routers and links are missing. Wormhole routing [20] is an increasingly common interconnect architecture for NoC as it minimizes communication latencies and reduce buffer space. In QNoC, we enhance regular wormhole routing with the ability to provide different classes of QoS, related to end-to-end delay, throughput or round-trip delay.

The full design cycle of such a network consists of the following stages. First, the traffic and QoS requirements of the target SoC are identified. Next, the network is customized by appropriate module placement and by applying a least cost, shortest path routing function, thus minimizing power dissipation and maximizing network resource utilization [4, 33]. Finally, network load balancing is performed by link bandwidth allocation so that the predefined multiclass QoS requirements of each communication flow are satisfied while reducing the cost to a minimum (see Section 2). Note that this methodology is in contrast with off-chip networks, where the traffic requirements change over time and the routing mechanisms need to balance the load of this changing traffic over given topology and link capacities, which were designed for legacy or unknown loads. This important feature of QNoC allows constructing a heterogeneous wormhole network with interrouter links of different speeds.

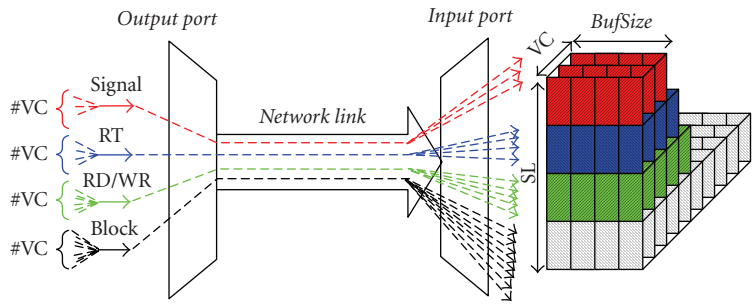


Figure 19: Multiplexing multiple SLs and VCs over a physical QNoC link.

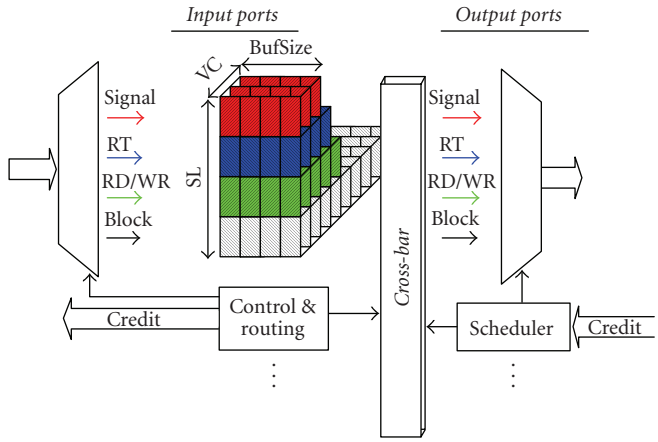


Figure 20: A QNoC router.

QNoC service levels

In order to support different classes of QoS for different kinds of on-chip traffic, we identify different types of communication requirements and define appropriate service levels (SL) to support them. For example, consider the following four different SLs: *Signaling* (urgent short packets that are given the highest priority), *Real-Time* (guaranteed bandwidth and latency to streamed audio and video), *Read/Write* (short memory and register accesses), and *Block-Transfer* (long messages such as DMA transfers).

In QNoC, a priority ranking among different SLs is established. For example, Signaling is given the highest priority and Block-Transfer the lowest. QNoC employs preemptive communication scheduling where data of a higher priority packet is always transmitted before that of a lower service level (a round-robin is employed within service levels). Additional service levels may be defined if desired.

Virtual channels

High performance wormhole-based interconnection networks are often equipped with virtual channels (VCs), which increase link utilization and overall network performance [21]. When links have different capacities, multiple VCs allow better utilization of high bandwidth links by multiplexing several slow flows over the link. Figure 18 depicts a simple example where the capacity of an outgoing link ($\times 10$) equals the overall capacity of the three incoming links. However, the high capacity outgoing link can be fully utilized only by simultaneous multiplexing of the incoming flows. This can be achieved by implementing multiple VCs on this high capacity outgoing link.

In QNoC every SL at each network link can be extended with its own number of VCs. The flits of different SLs/VCs that contend for the link bandwidth are time-multiplexed according to some arbitration policy over a single physical link (Figure 19). Previous research [21] has showed that adding VCs increases the maximal network throughput. We assume that physical links are split into an adequate number of VCs. In particular, we assume that a head flit can acquire a VC instantaneously on every link it traverses.

QNoC routers

QNoC router consists of input and output ports that are interconnected by a crossbar switch (Figure 20). Arriving flits are buffered at the input ports, awaiting transmission by the output ports. There are dedicated buffers for each SL and each VC. Relatively small buffers are allocated to each, capable of storing only a few flits. On the first flit of a packet, the router invokes a routing algorithm to determine to which

output port that packet is destined. The router then schedules the transmission for each flit on the appropriate output port.

Each output port of a router is connected to an input port of a next router via a communication link. The output port maintains the number of available flit slots per each SL and VC in the buffer of the next input port. The number is decremented upon transmitting a flit and incremented upon receiving a buffer-credit from the next router. When a space is available, the output port schedules transmission of flits that are buffered at the input ports and wait for transmission through that output port.

Router arbitration policy

Each output port schedules transmission of flits according to the availability of buffers in the next router and the service level priority of the pending flits. Once a higher priority packet appears on one of the input ports, transmission of the current packet is preempted and the higher priority packet gets through. Transmission of the lower priority packets is resumed only after all higher-priority packets have been serviced.

In QNoC, each SL is further divided into multiple VCs that are scheduled for transmission by the VC arbitration mechanism. VC arbitration consists of two phases. The first phase is VC allocation, in which available output VCs are allocated to the pending packets on the VCs from the input ports. When an output VC is allocated for a packet, this VC status becomes "active" and it can participate in the second phase of VC arbitration. The second phase is scheduling of these "active" VCs for transmission on each output link.

The QNoC architecture was modeled in detail in the OPNET environment [34], and flit-accurate simulations of every network instance can be performed.

ACKNOWLEDGMENTS

This work was partially supported by the Semiconductor Research Corporation (SRC), Intel Corporation, and the iSRC consortium.

REFERENCES

[1] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proceedings of Design, Automation and Test in Europe Conference ands Exhibition (DATE '00)*, pp. 250–256, Paris, France, March 2000.

[2] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 684–689, Las Vegas, Nev, USA, June 2001.

[3] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. de Micheli, "A methodology for mapping multiple use-cases onto networks on chips," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE '06)*, vol. 1, pp. 118–123, Munich, Germany, March 2006.

[4] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *Journal of Systems Architecture*, vol. 50, no. 2-3, pp. 105–128, 2004, special issue on network on chip.

[5] D. Bertozzi, A. Jalabert, S. Murali, et al., "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113–129, 2005.

[6] J. Henkel, W. Wolf, and S. Chakradhar, "On-chip networks: a scalable, communication-centric embedded system design paradigm," in *Proceedings of the 17th International Conference on VLSI Design (VLSID '04)*, vol. 17, pp. 845–851, Mumbai, India, January 2004.

[7] K. Srinivasan, K. S. Chatha, and G. Konjevod, "An automated technique for topology and route generation of application specific on-chip interconnection networks," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '05)*, pp. 231–237, San Jose, Calif, USA, November 2005.

[8] M. K.-F. Schäfer, T. Hollstein, H. Zimmer, and M. Glesner, "Deadlock-free routing and component placement for irregular mesh-based networks-on-chip," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '05)*, pp. 238–245, San Jose, Calif, USA, November 2005.

[9] M. Palesi, S. Kumar, and R. Holsmark, "A method for router table compression for application specific routing in mesh topology NoC architectures," in *Proceedings of the 6th International Workshop on Architectures, Modeling, and Simulation (SAMOS '06)*, pp. 373–384, Samos, Greece, July 2006.

[10] K. Goossens, J. Dielissen, O. P. Gangwal, S. G. Pestana, A. Rădulescu, and E. Rijpkema, "A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification," in *Proceedings of Design, Automation and Test in Europe (DATE '05)*, vol. 2, pp. 1182–1187, Munich, Germany, March 2005.

[11] J. Hu and R. Marculescu, "Application-specific buffer space allocation for networks-on-chip router design," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '04)*, pp. 354–361, San Jose, Calif, USA, November 2004.

[12] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "HERMES: an infrastructure for low area overhead packet-switching networks on chip," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 69–93, 2004.

[13] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini, "XPIPES: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs," in *Proceedings of the 21st International Conference on Computer Design (ICCD '03)*, pp. 536–539, San Jose, Calif, USA, October 2003.

[14] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch, "The Nostrum backbone-a communication protocol stack for networks on chip," in *Proceedings of the 17th International Conference on VLSI Design (VLSID '04)*, pp. 693–696, Mumbai, India, January 2004.

[15] M. Coenen, S. Murali, A. Ruadulescu, K. Goossens, and G. de Micheli, "A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control," in *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis*, pp. 130–135, Seoul, Korea, October 2006.

[16] G. Ascia, V. Catania, and M. Palesi, "Multi-objective mapping for mesh-based NoC architectures," in *Proceedings of the 2nd International Conference on Hardware/Software Codesign and Systems Synthesis*, pp. 182–187, Stockholm, Sweden, September 2004.

[17] S. Murali and G. de Micheli, "SUNMAP: a tool for automatic topology selection and generation for NoCs," in *Proceedings of the 41st Design Automation Conference*, pp. 914–919, San Diego, Calif, USA, June 2004.

[18] U. Y. Ogras and R. Marculescu, ""It's a small world after all": NoC performance optimization via long-range link insertion," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 7, pp. 693–706, 2006.

[19] N. Banerjee, P. Vellanki, and K. S. Chatha, "A power and performance model for network-on-chip architectures," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE '04)*, vol. 2, pp. 1250–1255, Paris, France, February 2004.

[20] W. J. Dally and C. J. Seitz, "The torus routing chip," *Distributed Computing*, vol. 1, no. 4, pp. 187–196, 1986.

[21] W. J. Dally, "Virtual-channel flow control," in *Proceedings of the 17th Annual International Symposium on Computer Architecture (ISCA '90)*, pp. 60–68, Seattle, Wash, USA, June 1990.

[22] H. Sarbazi-Azad, A. Khonsari, and M. Ould-khaoua, "Performance analysis of deterministic routing in wormhole k-ary n-cubes with virtual channels," *Journal of Interconnection Networks*, vol. 3, no. 1-2, pp. 67–73, 2002.

[23] S. Loucif and M. Ould-Khaoua, "Modeling latency in deterministic wormhole-routed hypercubes under hot-spot traffic," *Journal of Supercomputing*, vol. 27, no. 3, pp. 265–278, 2004.

[24] C. Roche, P Palnati, M. Gerla, F. Neri, and E. Leonardi, "Performance of congestion control mechanisms in wormhole routing networks," in *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies, Driving the Information Revolution (INFOCOM '97)*, vol. 3, pp. 1365–1372, Kobe, Japan, April 1997.

[25] J. Kim and C. R. Das, "Hypercube communication delay with wormhole routing," *IEEE Transactions on Computers*, vol. 43, no. 7, pp. 806–814, 1994.

[26] R. I. Greenberg and L. Guan, "Modeling and comparison of wormhole routed mesh and torus networks," in *Proceedings of the 9th IASTED Iasted International Conference on Parallel and Distributed Computing Systems*, Washington, DC, USA, October 1997.

[27] B. Ciciani, M. Colajanni, and C. Paolucci, "Performance evaluation of deterministic wormhole routing in k-ary n-cubes," *Parallel Computing*, vol. 24, no. 14, pp. 2053–2075, 1998.

[28] J. T. Draper and J. Ghosh, "A comprehensive analytical model for wormhole routing in multicomputer systems," *Journal of Parallel and Distributed Computing*, vol. 23, no. 2, pp. 202–214, 1994.

[29] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Transactions on Computers*, vol. 39, no. 6, pp. 775–785, 1990.

[30] L. Kleinrock, *Queuing Systems, Volume 1: Theory*, John Wiley & Sons, New York, NY, USA, 1975.

[31] J. P. Fishburn and A. E. Dunlop, "TILOS: a posynomial programming approach to transistor sizing," in *Proceedings of the IEEE International Conference on Computer Aided Design (ICCAD '85)*, pp. 326–328, Santa Clara, Calif, USA, November 1985.

[32] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Cost considerations in network on chip," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 19–42, 2004.

[33] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Routing table minimization for irregular mesh NoCs," in *Proceedings of Design Automation and Test in Europe (DATE '07)*, Nice, France, March 2007.

[34] OPNET modeler, http://www.opnet.com/.

Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 18372, 14 pages
doi:10.1155/2007/18372

Research Article

Comparison of a Ring On-Chip Network and a Code-Division Multiple-Access On-Chip Network

Xin Wang and Jari Nurmi

Institute of Digital and Computer Systems, Tampere University of Technology, 33101 Tampere, Finland

Received 5 October 2006; Accepted 6 February 2007

Recommended by Shashi Kumar

Two network-on-chip (NoC) designs are examined and compared in this paper. One design applies a bidirectional ring connection scheme, while the other design applies a code-division multiple-access (CDMA) connection scheme. Both of the designs apply globally asynchronous locally synchronous (GALS) scheme in order to deal with the issue of transferring data in a multiple-clock-domain environment of an on-chip system. The two NoC designs are compared with each other by their network structures, data transfer principles, network node structures, and their asynchronous designs. Both the synchronous and the asynchronous designs of the two on-chip networks are realized using a hardware-description language (HDL) in order to make the entire designs suit the commonly used synchronous design tools and flow. The performance estimation and comparison of the two NoC designs which are based on the HDL realizations are addressed. By comparing the two NoC designs, the advantages and disadvantages of applying direct connection and CDMA connection schemes in an on-chip communication network are discussed.

1. INTRODUCTION

As the technology feature size of integrated circuit fabrication is continuously shrinking down in the deep submicron regime, the number of components which can be integrated into an on-chip system is getting larger and larger. Therefore, the communications among the large number of components in a system-on-chip (SoC) are challenging tasks to deal with. Network-on-chip addresses this communication issue in an on-chip system by separating the concerns of communication from the concerns of computation. It means that the communication issue in an SoC is abstracted and handled by an on-chip communication network which hides the detailed information about how the communications are performed. Therefore, a system designer can pay more attention on the functions of system components and system integration by treating the NoC as a component of an on-chip system.

The NoC structures which have been proposed can be roughly classified into two categories, circuit-switched network and packet-switched network, in terms of the way of using the communication media. PROPHID architecture [1] is an example of a circuit-switched network which connects the terminals in the network by allocating them a set of time or space slices on the communication links. Examples in the packet-switched category are SPIN [2] and Proteo NoC [3]. SPIN network applies fat-tree topology and router blocks to transfer data packets from source node to destination node. In Proteo NoC, the components in the system are connected through network nodes and hubs. The network topology and connections in Proteo NoC can be customized and optimized for a specific application. Since an on-chip system can contain hundreds of functional intellectual property (IP) blocks in the near future, the circuit-switched network will face the problem of scalability and parallelism in that situation. Therefore, a packet-switched scheme is a better choice for future NoC designs because its structure is scalable and its data transfers are performed in parallel by sharing the communication media among multiple network nodes in a time-division manner.

As the number of IP blocks in an SoC is increasing, it is natural that different functional blocks work with different clock frequencies in an SoC. Hence, data transfer among multiple clock domains is another issue that needs to be handled by an on-chip network. A globally asynchronous locally synchronous (GALS) scheme [4] has been proposed to solve this problem of SoC designs. For an NoC, GALS means

that data transfers between each functional IP block and its attached network node are synchronous, whereas data transfers between network nodes are asynchronous.

From the analysis addressed in the previous two paragraphs, we can see that the GALS packet-switched scheme is a promising direction to explore the NoC designs for future on-chip systems. A common and natural way of composing a GALS packet-switched on-chip network is to connect two network nodes with a direct link. This way of connecting network nodes will be referred to as point-to-point (PTP) connection in this paper. Different patterns of the connection links in the network form different network topologies. For example, a mesh topology is applied in the NoC design presented in [5]. In a PTP connection NoC, the routing scheme and router architecture, such as the router presented in Æthereal NoC [6], are very important factors for supplying guaranteed services because the packet transfer latency may vary largely when data packets are transferred to different destinations or to the same destination via different routes in the network.

In order to eliminate the variance of the data transfer latency and complexity incurred by routing in a PTP connection NoC, a connection scheme which applies code-division multiple-access technique has been briefly introduced in [7].

By separating the different data from different users in the code domain, the data transfer latency in the CDMA NoC is stabilized by enabling multiple users to use the communication media parallel in time domain.

In order to examine the advantages and disadvantages of both the PTP connection scheme and the CDMA connection scheme, a bidirectional ring NoC design and a CDMA NoC design developed in our institute will be addressed and compared in terms of the network structure, data transfer principle, network node design, asynchronous design, and performance. For the sake of abbreviation, the bidirectional ring NoC design will be referred to as the PTP NoC in this paper.

The following sections of this paper will be arranged as follows. The network structures of the PTP NoC design and the CDMA NoC design will be presented and compared in Section 2. In Section 3, the data transfer principles of the two NoC designs will be studied and compared. Then the different network node structures in the two NoC designs will be addressed and compared in Section 4. Section 5 will present the asynchronous designs applied in both the PTP NoC and the CDMA NoC designs. In Section 6, two simulation networks of the two NoC designs built up for performance estimation will be presented. Then the performance comparison of the two NoC designs will be addressed upon the simulation results. Finally, the conclusion will be drawn in Section 7.

2. THE NETWORK STRUCTURES

2.1. The network structures of the two NoC designs

The PTP NoC design examined in this paper is based on a network node design [8] proposed for implementing GALS scheme in Proteo NoC. The network structure of the PTP NoC is illustrated in Figure 1. From Figure 1, we can see that

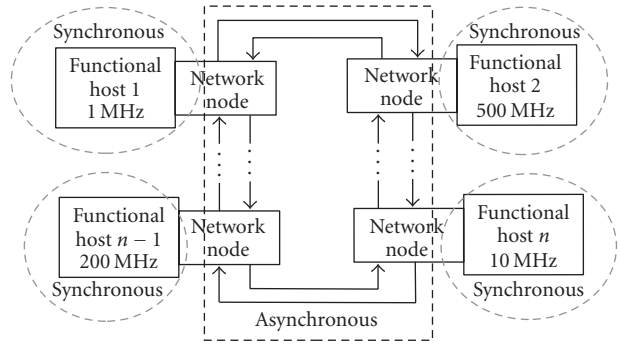


Figure 1: The bidirectional ring NoC structure.

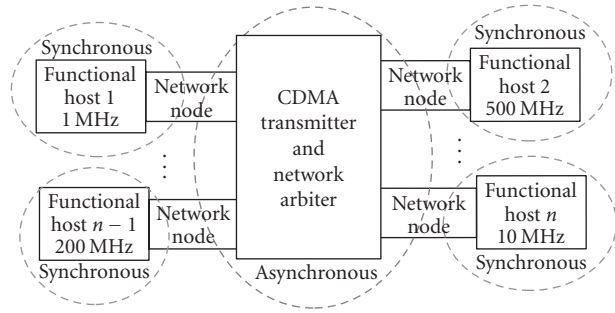


Figure 2: The CDMA NoC structure.

the communication between a "functional host" (functional IP block) and its network node is synchronous, while the data transfers among network nodes are preformed in asynchronous manner. For a large network, it may be necessary to break the entire network into sections and use some bridge nodes or hub nodes as addressed in [3] to connect the network sections together, whereas these bridges or hubs can be seen as one type of network nodes. Therefore, as illustrated in Figure 1, the PTP NoC can be composed simply by connecting the network nodes together with direct links.

The network structure of the CDMA NoC design introduced in [7] is illustrated in Figure 2. In the CDMA NoC, the GALS scheme is applied in the same way as in the PTP NoC; however, the network nodes in the CDMA NoC are no longer connected to each other with direct links. A "CDMA transmitter" and a "network arbiter" blocks are introduced in the CDMA NoC. All the network nodes need to communicate with the "CDMA transmitter" and "network arbiter" blocks directly. The functionality of the "CDMA transmitter" and "network arbiter" blocks will be addressed thoroughly in Section 3. With a direct comparison, we can see that the structure of the CDMA NoC is more complex than the PTP NoC since extra blocks are introduced in the CDMA NoC.

2.2. Distributed traffic versus centralized traffic

After a direct comparison of the two network structures in terms of simplicity, we can take a further analysis of the effects of the two different network structures on the features

of the networks. In the PTP NoC as illustrated in Figure 1, the data traffic load is distributed into the links among the network nodes. This distributed data traffic scheme has the merits of flexibility and scalability, whereas the main disadvantage of the PTP connection is that the data transfer latency between two network nodes can be largely different because the data may be transferred through different routes or because of data traffic congestions in the network. Therefore, the main concern of designing a PTP NoC is to find out the optimal topology and use all kinds of routing and flow-control methods to guarantee a high throughput and low transfer latency.

In the CDMA NoC illustrated in Figure 2, a centralized data transfer scheme is applied since all network nodes communicate with the "CDMA transmitter" and the "network arbiter" blocks directly. This centralized data transfer scheme is different from the conventional bus structures since it can supply parallel data transfers both in time and space domains by applying CDMA technique, whereas a bus structure supplies data transfer service among users in a time-division manner. The advantage of the centralized scheme applied in the CDMA NoC is that the data transfer latency between network nodes is a stable value. This stable transfer latency is contributed by the feature of parallel data transfer in time domain and the universal link distance among all network nodes. With the stable data transfer latency, the communication quality in the CDMA NoC will not vary.

3. THE DATA TRANSFER PRINCIPLES

The fundamental reason of the different network structures between the PTP NoC and the CDMA NoC is the different data transfer principles applied in the two NoC designs. Thus, the data transfer principles of the two NoC designs will be addressed and compared in this section.

3.1. Data transfer principle in the PTP NoC

As presented in Section 2.1, the PTP NoC is built by connecting the network nodes with direct links. The reason of this simplicity of connecting the network nodes is that the data are transferred in their original form in the PTP NoC. The only operation on the data is to encapsulate them into a packet format. This operation is done by a network node after getting the data from its attached functional host block. The packet format used in the PTP NoC is illustrated in Figure 3. After the data packet is formed, the PTP NoC will transfer the data bits with their original values to their destination through the links to the other nodes. Therefore, direct links between the network nodes in the PTP NoC are enough to handle the data transfers.

3.2. Data transfer principle in the CDMA NoC

As indicated by the name, the CDMA NoC applies CDMA technique to perform data transfers in the NoC. The basic principle of CDMA technique is illustrated in Figure 4. At the sending end, the data from different senders are encoded using a set of orthogonal spreading codes. Then the encoded data from different senders are added together for transmission without interfering with each other because of the orthogonal property of spreading codes. The orthogonal property means that the normalized autocorrelation of the spreading codes is 1, while the cross-correlation of the spreading codes is 0. Therefore, at the receiving end, the data can be decoded from the received sum signals by multiplying the received signals with the corresponding spreading code used for encoding. The data packet format applied in the CDMA NoC is the same format as illustrated in Figure 3. The issues related with the data encoding/decoding and transfer principles in the CDMA NoC will be addressed with details in the following subsections.

Destination node ID	Source node ID (optional for CDMA NoC)	(Other fields)
Data cell 1 Data cell 2(optional) Data cell 3(optional)		

Figure 3: Packet format specification.

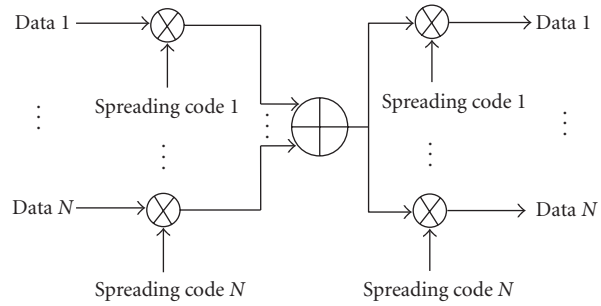


Figure 4: CDMA technique principle.

3.2.1. Data encoding and decoding schemes

Some CDMA encoding and decoding schemes for on-chip communication implemented by analog circuits have been proposed [9–11]. In those schemes, the encoded data are represented by the continuous voltage or capacitance value of the circuits. Therefore, the data transfers in the analog circuits are challenged by the coupling noise, clock skew, and the variations of capacitance and resistance caused by circuit implementation [11]. In order to avoid the challenges faced by the analog circuit implementation, digital encoding and decoding schemes are developed for the CDMA NoC and are illustrated in Figures 5 and 7, respectively. In the presented encoding scheme, data from different senders are fed into the encoding function bit by bit. Each data bit will be spread into S bits by multiplying it with a unique S-bit spreading code. The multiplications are performed by XOR logic gates as illustrated in Figure 5. Each bit of the S-bit encoded data generated by XOR operations is called a data chip. Then the data chips which come from different senders are added

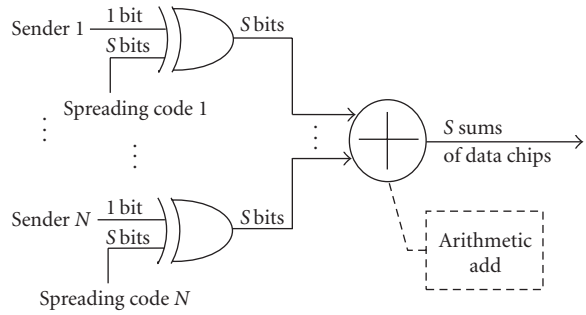


Figure 5: Digital CDMA encoding scheme.

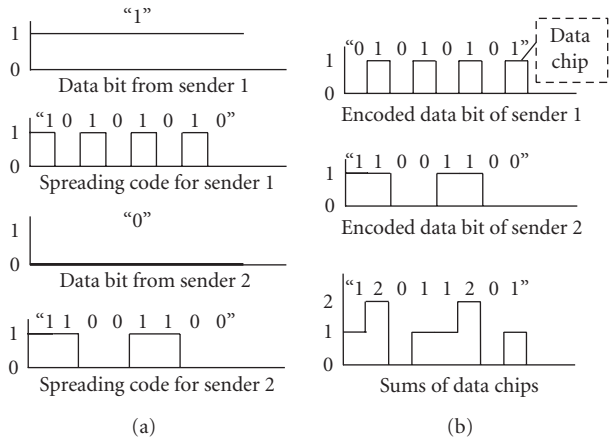


Figure 6: Data encoding example.

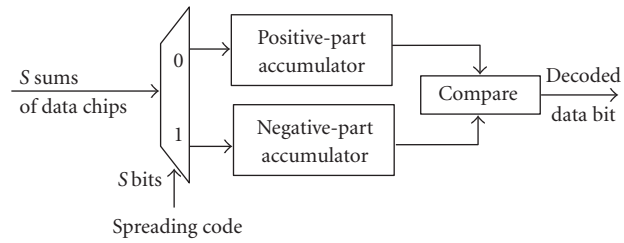


Figure 7: Digital CDMA decoding scheme.

together arithmetically according to their positions in the S-bit sequences. Namely, all the first data chips from different senders are added together, and all the second data chips from different senders are added together, and so on. Therefore, after the add operations, we will get S sum values of S-bit encoded data. Finally, as proposed in [12], binary equivalents of the S sum values are transferred to the receiving end one by one. An example of encoding two data bits from two senders is illustrated in Figure 6 in order to explain the proposed encoding scheme more specifically. Figure 6(a) shows two original data bits from different senders and two 8-bit spreading codes. The top two figures in Figure 6(b) illustrate the results after data encoding (XOR operations) for the original data bits. The bottom figure in Figure 6(b) presents the 8 sum values after adding operations. Then the binary equivalents of each sum value will be transferred to the receiving end. In this case, two binary bits are enough to represent the three possible decimal sum values, "0," "1," and "2." Hence, for example, if a decimal sum value "2" needs to be transferred, we need to transfer two binary digits "10."

The digital decoding scheme used in the CDMA NoC is illustrated in Figure 7. The decoding scheme accumulates the received sum values into two separated parts, a positive part and a negative part, according to the bit value of the spreading code used for decoding. For instance, as illustrated in Figure 7, the received first sum value will be put into the positive accumulator if the first bit of the spreading code for decoding is "0," otherwise, it will be put into the negative accumulator.

The same selection and accumulation operations are also performed on the other received sum values. The principle of this decoding scheme can be explained as follows. If the original data bit to be transferred is "1," after the XOR logic in the encoding scheme illustrated in Figure 5, it can only contribute a nonzero value to the sums of data chips when a bit of spreading code is "0." Similarly, the 0-value original data bit can only contribute a nonzero value to the sums of data chips when a bit of spreading code is "1." Therefore, after accumulating the sum values according to the bit values of the spreading code, either the positive part or negative part is larger than the other if the spreading codes have orthogonal and balance properties. Hence, the original data bit can be decoded by comparing the values between the two accumulators. Namely, if the positive accumulation value is larger than the negative accumulation value, the original data bit is "1"; otherwise, the original data bit is "0."

3.2.2. Spreading code selection

As discussed in Section 3.2.1, the presented encoding/decoding scheme requires the spreading codes used in the CDMA NoC to have both the orthogonal and balance properties. The orthogonal property was explained in the first paragraph of Section 3.2. The balance property means that the number of bit "1" and the number of bit "0" in a spreading code should be equal. Because Walsh code [13] has the required orthogonal and balance properties, it is chosen as the spreading code for the CDMA NoC. In an S-bit ($S = 2^N$, integer $N > 1$) length Walsh code set, there are S-1 sequences which have both the orthogonal and balance properties. Hence, the proposed CDMA NoC can have at most S-1 nodes connecting with one "CDMA transmitter" and one "network arbiter" block as illustrated in Figure 2.

3.2.3. Spreading code protocol

In a CDMA network, if multiple users simultaneously use the same spreading code to encode their data packets for transmission, the data to be transferred will interfere with each other because of the loss of orthogonal property among the spreading codes. This situation is called spreading code conflict, which should be avoided. Spreading code protocol is a

policy used to decide how to assign and use the spreading codes in a CDMA network in order to eliminate or reduce the possible spreading code conflicts. Several spreading code protocols have been proposed for CDMA packet radio network [14, 15]. Among these proposed spreading code protocols, only transmitter-based protocol (T protocol) and transmitter-receiver-based protocol (T-R protocol) are conflict-free if the users in the network send data to each other randomly. The principles of these two spreading code protocols will be shortly introduced in the following two paragraphs.

(1) *Transmitter-based protocol (T protocol)*: the unique spreading code allocated to each user will be used by the user himself to transfer data to others.

(2) *Transmitter-receiver-based protocol (T-R protocol)*: two unique spreading codes will be assigned to each user in the network, and then a user will generate a new spreading code from the assigned two unique codes for its data encoding.

Because the T-R protocol has the drawback of using a large amount of spreading codes and complicated decoding scheme, T protocol is preferred in the CDMA NoC. However, if T protocol is applied in the network, a receiver cannot choose the proper spreading code for decoding because it cannot know who is sending data to it. In order to solve this problem, an arbiter-based T protocol (A-T protocol) is proposed for the CDMA NoC. In a CDMA network which applies A-T protocol, each user is assigned a unique spreading code for data transfer. When a user wants to send data to another user, he will send the destination information of the data packet to the arbiter before starting data transmission. Then the arbiter will inform the requested receiver to prepare the corresponding spreading code for data decoding according to the sender. After the arbiter has got the acknowledge signal from the receiver, it will send an acknowledge signal back to the sender to grant its data transmission. If there are several users who want to send data to the same receiver, the arbiter will grant only one sender to send data at a time. Therefore, by applying the A-T protocol, spreading code conflicts in the CDMA NoC can be eliminated.

3.2.4. Parallel data transfer principle

The parallel data transfer principle of the CDMA NoC is based on the A-T spreading code protocol described in Section 3.2.3. By applying A-T spreading code protocol, every node in the CDMA NoC needs to send the destination address of the packets to the "network arbiter" as illustrated in Figure 2. After getting the grant signal from "network arbiter," the sender node will send data packets to the "CDMA transmitter" block. The data encoding operations and data transfers are performed in the "CDMA Transmitter" block. Finally, the data decoding operation will be carried out by the data receiving network node. Therefore, the data transfer process in the CDMA NoC can be clarified by describing the functions in the "network arbiter" and the "CDMA transmitter" block, respectively.

(1) *Network arbiter*. After receiving a data transfer request from a network node, "network arbiter" will inform the requested receiver node to prepare the proper spreading code for decoding and send a grant signal back to the sender node. In case that there are more than one sender nodes requesting to send data to the same receiver node simultaneously or at different times, the arbiter will apply "round-robin" arbitration scheme or the "first-come first-served" principle to guarantee that there is only one sender sending data to one specific receiver at a time. The reason for this limitation is that the "packet receiver" block in a network node can receive and decode data from only one sender at a time. However, if different sender nodes request to send data to different receiver nodes, these requests will not block each other and will be handled in parallel in the "network arbiter." The "network arbiter" in the CDMA NoC is different from the arbiter used in a conventional bus. This is because the "network arbiter" here is only used to set up spreading codes for receiving and it handles the requests in parallel in the time domain. In contrary, a conventional bus arbiter is used to allocate the usage of the common communication media among the users in the time-division manner.

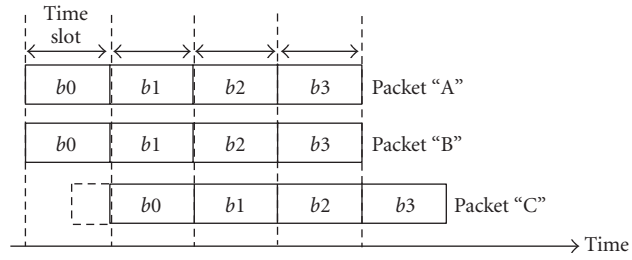


Figure 8: Bit-synchronous transfer scheme.

(2) *CDMA transmitter*. The sender node will start to send data packets to the "CDMA transmitter" after it gets the grant signal from the arbiter. Then the "CDMA transmitter" will encode the data to be transferred with the corresponding unique spreading code of the sender node. Although the "CDMA transmitter" block is implemented by asynchronous circuits, it applies the bit-synchronous transfer scheme. This means that the data from different nodes will be encoded and transmitted synchronously in terms of data bits rather than any clock signals. In Figure 8, the principle of the referred bit-synchronous transfer is illustrated by a situation in which network nodes "A" and "B" send data packets to "CDMA transmitter" simultaneously and node "C" sends a data packet later than "A" and "B." In this situation, the data packet from node "A" will be encoded and transmitted together with the data packet from node "B" synchronously in terms of each data bit. As the data packet from node "C" arrive at a later time point, the transmitter will handle the data bit from "packet C" together with the data bits from packets "A" and "B" at the next start point of the time slot for bit encoding and transmitting processes. The dotted-line frame at the head of the "packet C" in Figure 8 illustrates the waiting duration if the "packet C" arrived in the middle of the time slot for handling the previous data bit. The time slot for handling a data bit is formed by a four-phase

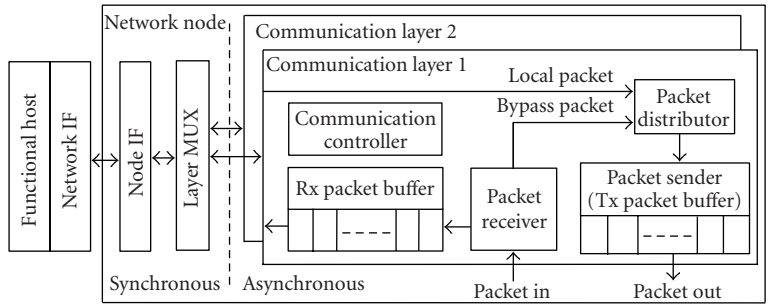


Figure 9: Network node structure of the PTP NoC.

handshake process. The bit-synchronous scheme can avoid the interferences caused by the phase offsets among the orthogonal spreading codes when the data bits from different nodes are encoded and transmitted asynchronously with each other. Because the nodes in the network can request data transfer randomly and independently of each other, "CDMA transmitter" applies the "first-come first-served" mechanism to ensure that the data encoding and transmission are performed as soon as there is a data transfer request.

3.3. Comparison of the data transfer principles

One advantage of the data transfer principle in the CDMA NoC is the feature of parallel data transfer. Although the data transfers in the PTP NoC can also be parallel if they take place in different links among the network nodes, the parallelism in the PTP NoC is largely limited by the possible traffic congestions in a link because the data are transferred through a link between two network nodes in a time-division manner. Another advantage of the CDMA data transfer principle is that no data routing is needed because of the centralized data transfer scheme. This feature can supply stable transfer latency in the CDMA NoC, which in turn facilitates the CDMA NoC to provide a guaranteed service for the on-chip system.

Another advantage of the CDMA NoC is that it can easily support multicast data transfers by requesting multiple receiver nodes to use the same spreading code for receiving. In the PTP NoC, the multicast transfer can be realized only by sending multiple copies of a data packet to its multiple destinations, unless extra logic is added in each network node to copy the multicast packet to both the functional host and the output link to the next node. This increases the traffic load in the network, or complicates the network implementation.

By comparing with the data transfer principle in the PTP NoC, one disadvantage of the CDMA data transfer principle is its complexity caused by the data encoding and decoding operations. Another drawback of the CDMA data transfer principle is that the data transfer efficiency obtained by parallel transfers in the time domain is compromised by the latency introduced by the data spreading scheme. As presented in Section 3.2.1, one data bit will be extended to S bits for the CDMA data transfers. The parameter S is the width of the spreading code applied in the CDMA NoC. As the number of nodes in the NoC increases, the width of the applied spreading code will also be increased. Then the data latency caused by the data spreading will be also increased.

4. THE NETWORK NODE STRUCTURES

"Network node" block is a common type of component needed in both of the PTP NoC and the CDMA NoC. However, different data transfer principles in the two NoC designs imply different structures in the network nodes. This section will present the network node structure in each of the two NoCs.

4.1. Network node structure in the PTP NoC

The network node structure of the PTP NoC is illustrated in Figure 9. The network node consists of "node if," "layer MUX," and "communication layer" blocks. The two blocks outside of the network node illustrate how a "functional host" block as presented in Figure 1 is connected with a network node through a network interface ("network if") block. In the PTP NoC, the applied interface standards include VCI [16] and OCP [17]. As illustrated in Figure 9, GALS scheme in the network is implemented by applying both synchronous and asynchronous designs in each network node. The synchronous blocks, "node if" and "layer MUX," are used to communicate with locally synchronous "functional host" in the system, while the asynchronous blocks are used to perform asynchronous communications among the network nodes. The arrows in Figure 9 demonstrate the data packet flow in a network node. The blocks in the network node illustrated in Figure 9 will be introduced in the following three paragraphs.

(1) *Node if*. This block is responsible for assembling the data from "functional host" into data packets or delivering the data packets from network node to "functional host" according to the interface standard applied in "network if" block.

(2) *Layer MUX*. As the name of this block indicates, this block behaves as a multiplexer used to connect "node if" block with a certain "communication layer" block during the data transfers between network node and "functional host" block.

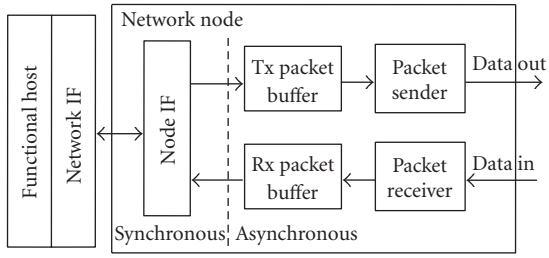


Figure 10: Network node structure of the CDMA NoC.

(3) *Communication layer*. The function of this block is to perform the globally asynchronous communication with other network nodes through a handshake protocol. As illustrated in Figure 9, the two "communication layer" blocks in a network node are used to connect with two other network nodes in the bidirectional ring NoC. More "communication layer" blocks can be used in a network node to implement other types of topology. There are five subblocks in a "communication layer" block. The "packet receiver" subblock is used to receive data packets from another network node. If the destination of the received packet is the current network node, the packet is called "incoming packet," and it will be stored in "Rx packet buffer." Otherwise, the received packet is called "bypass packet," and it will be dispatched into "packet sender" block via "packet distributor" for further transferring. The "communication controller" subblock in Figure 9 represents the controller which takes care of the necessary arbitrations and communication control.

4.2. Network node structure in the CDMA NoC

The block diagram of the network node structure of the CDMA NoC is shown in Figure 10 where the arrows represent the flows of data packets. In Figure 10, the "network if" block which belongs to the functional host is an interface block for connecting a functional host with a "network node." The GALS scheme is applied in "network node" block of the CDMA NoC by using synchronous design in the "node if" subblock and using asynchronous design in the other subblocks. The network interface standards supported in the CDMA NoC also include the VCI and OCP standards. The subblocks in the network node will be addressed in the following four paragraphs.

(1) *Node if*. This block is used to assemble the data from "functional host" into packets and send the packets to "Tx packet buffer" or disassemble the received packet from "Rx packet buffer" and send the extracted data to "functional host."

(2) *Tx/Rx packet buffer*. "Tx packet buffer" is used to store the data packets from "node if," and then deliver the packets to "packet sender." The "Rx packet buffer" stores and delivers the received packets from "packet receiver" to "node if."

(3) *Packet sender*. If "Tx packet buffer" is not empty, "packet sender" will fetch a data packet from the buffer by an asynchronous handshake protocol. Then it will extract the destination information from the fetched packet and send the destination address to "network arbiter." After "packet sender" gets the grant signal from the arbiter, it will start to send data packets to "CDMA transmitter."

(4) *Packet receiver*. After system reset, this subblock will wait for the sender information from "network arbiter" to select the proper spreading code for decoding. After the spreading code for decoding is ready, the receiver will send an acknowledge signal back to "network arbiter" and start to receive data from "CDMA transmitter," and then send the decoded data to "Rx packet buffer" in the packet format.

4.3. Comparison of the network node structures

By comparing with the presented network node in the PTP NoC, the network node in the CDMA NoC has less complexity. The main reason is that the network node of the CDMA NoC does not need to handle any bypass packets or the packet routing issues because of the centralized traffic scheme. Therefore, the "communication controller" block and the "packet distributor" block in the network node for the PTP NoC are not needed in the node for the CDMA NoC. When the data transfer parallelism needs to be increased in the PTP NoC, more "communication layer" blocks in a network node are needed in order to set up more links with other nodes, whereas the network node in the CDMA NoC does not need to change in this situation. One advantage of both of the network nodes is that they are both replicable because each network node structure in the network is the same. This advantage makes both the PTP NoC and the CDMA NoC designs modular.

5. ASYNCHRONOUS DESIGNS

As presented in Section 4, the GALS scheme is applied in the PTP NoC and in the CDMA NoC by implementing the global interconnect fabric with asynchronous designs. However, this is not the only way to implement GALS scheme in an on-chip network. For example, in "islands of synchronicity" (IoS) methodology presented in [18], the GALS scheme is implemented in SoC designs by localizing the clock in each of the functional IP blocks and connecting the isolated clock "islands," the functional IPs, with asynchronous communication links. If the IoS methodology is applied in the presented PTP NoC and the CDMA NoC, it means that all the blocks, including network nodes, "CDMA transmitter," and "network arbiter," in the designs need to be synchronous designs which work with different local clock frequencies. Then, the communications among the blocks in the NoC designs use asynchronous protocols. The advantage of applying the IoS methodology is that all the blocks in the design can be implemented by using standard synchronous design tools and flow. However, two disadvantages addressed in the following two paragraphs need to be noticed.

(1) *Synchronization cost*. The signals need to be synchronized with the local clocks when they cross different clock domains. If the IoS methodology is applied, two synchronization operations are needed when data enter into and leave from the global interconnect fabric during a data transfer process because the interconnect fabric works with its own clock rate. If the interconnect fabric is implemented by asynchronous designs, the synchronization step is not needed when data enter into the global interconnect fabric because a signal from a synchronous domain can enter into an asynchronous domain directly. Therefore, synchronization-related latency and area cost can be reduced 50% if the global interconnect fabric is implemented by asynchronous designs directly.

(2) *Area and power costs*. As presented in Section 4, "Tx/Rx packet buffer" composed by the asynchronous FIFO presented in [19] takes a large portion of the network node structure in both of the NoCs. As addressed in [19], the area and power costs of the asynchronous FIFO are 51.5% and 54.2% less than a synchronous implementation. Hence, if all the blocks related with global interconnection are implemented by synchronous designs, the area and power costs of the "Tx/Rx packet buffer" will be nearly doubled by comparing with the asynchronous implementation.

Therefore, in order to reduce the cost of synchronization, area, and power, asynchronous designs are applied to implement the blocks for global interconnections in the PTP NoC and the CDMA NoC. The asynchronous designs applied in the two NoC designs will be addressed in this section.

5.1. Asynchronous design in the PTP NoC

The basic component of the PTP NoC is the network node presented in Section 4.1. As illustrated in Figure 9, the blocks which apply asynchronous designs in the network node are the "communication controller," "packet receiver," "packet distributor/sender," and "packet Rx/Tx buffer" blocks. The four-phase dual-rail protocol is applied in the asynchronous designs in order to make the data transfers delay-insensitive. The control logic used in the asynchronous blocks of the PTP NoC will be presented in this subsection.

5.1.1. Control logic in the "communication controller"

The "communication controller" block is the main control block which takes care of data packet receiving, sending, and storing processes in the network node. Each of the mentioned packet handling processes is controlled by a control pipeline which can be seen as a finite state machine (FSM) in the "communication controller" block. The control processes of the FSMs are illustrated in Figure 11 and are explained in the following three paragraphs.

(1) *Packet receiving FSM*. As illustrated in Figure 11(a), there are six states in this FSM. The machine will move from its initial "rx_idle" state to "rx_pkt" state when "packet receiver" starts to receive a packet. After the packet receiving is completed, the FSM will move to "chk_addr" state to check the destination of the received packet. If the received packet is "incoming packet," the FSM will move to "chk_buf" state to check the status of "Rx packet buffer." If the buffer is not full, the "incoming packet" will be stored in the buffer during "st_pkt" state, otherwise, it will be held by "packet receiver" until there is room available in the buffer. If the received packet is "bypass packet," it will be sent to the network node connected to the output port of current node in "bp_pkt" state.

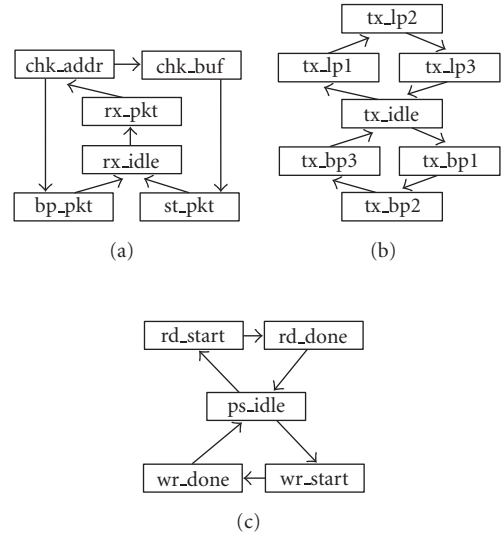


Figure 11: State transfer graphs of the FSMs.

(2) *Packet sending FSM*. This FSM illustrated in Figure 11(b) is responsible for sending two types of data packets into the "Tx packet buffer" in "packet sender" via "packet distributor" block. One type of packets is "local packet" which refers to the packet which comes from the "functional host" connected with the network node. Another type of packets is the "bypass packet" explained in Section 4.1. For sending "local packet," the FSM will be triggered by the signal from the "node if" block after a "local packet" is ready to be transferred. Then the FSM will go through "tx_lp1," "tx_lp2," and "tx_lp3" states for checking the status of the "Tx packet buffer," sending the packet into the buffer, and going back to "tx_idle" state, respectively. The process of sending a "bypass packet" into the transfer buffer is similar to the process of sending "local packet" except that the FSM will go through "tx_bp1," "tx_bp2," and "tx_bp3" states.

(3) *Packet storing FSM*. This FSM presented in Figure 11(c) is used to store or fetch an "incoming packet" to or from "Rx packet buffer" block. The process of storing an "incoming packet" will be triggered by packet receiving FSM during the "st_pkt" state as illustrated in Figure 11(a). The packet storing FSM will go through the "wr_start" and "wr_done" states to complete the storing task. The "rd_start" and "rd_done" states are for the process of fetching a stored "incoming packet" from "Rx packet buffer." The fetching process will be triggered by the "node if" block after getting flag signal from"communication controller" block.

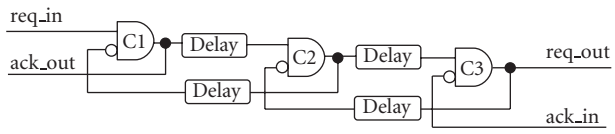


Figure 12: Micropipeline control logic.

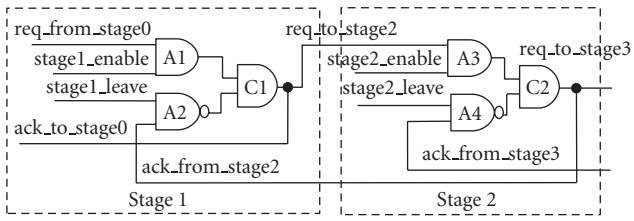


Figure 13: Control pipeline structure in the FSMs.

The presented control FSMs in the "communication controller" block are realized by applying the delay-insensitive micropipeline control logic presented in [20]. The structure of the micropipeline control logic is illustrated in Figure 12. The principle of micropipeline control logic is to use the output from the current stage to enable or disable the input of previous stage. Two stages of the control pipeline used in "communication controller" block for building the FSMs are illustrated in Figure 13. Each stage of the pipeline represents a state element of an FSM. In Figure 13, we can see that the FSM uses micropipeline control logic as the backbone and few AND gates as the delay components illustrated in Figure 12, hence it is also delay-insensitive. The state information of the FSM is passed through each stage in the pipeline by a four-phase handshake protocol. If we take the "stage 1" illustrated in Figure 13 as an example, when both the "req_from_stage0" and "stag1_enable" signals are "1," the output of "C1" will be set to logic "1" which indicates that the current state of the FSM is in "stage 1." Then the output of "C1" can be used as a request signal to trigger the control logic in the corresponding function blocks for a certain communication process.

5.1.2. Control logic in other blocks

Besides the FSMs in the "communication controller" block, control pipelines exist also in other asynchronous blocks used to perform the concrete task of moving the data packets in or out of the individual blocks through a four-phase dual-rail protocol. The FSMs in the "communication controller" block control the processes of receiving, sending, or storing data packets by triggering the control pipeline in the corresponding function blocks. The control pipeline structure used in the "packet receiver," "packet distributor," and "packet sender" blocks is illustrated in Figure 14. This structure is derived from the micropipeline control logic and is used to perform data transfers by interacting with the control signals coming from the "communication controller" block. For example, when the pipeline structure is used in "packet

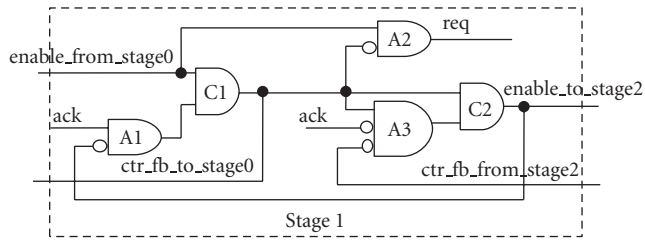


Figure 14: Block control pipeline structure.

receiver," it will receive the packet receiving enable signal from the "communication controller" block as the enable signal for the first stage of the pipeline structure. Then, a request signal will be generated by gate "A2" in the pipeline stage as illustrated in Figure 14. The generated request signal will start a handshake process to receive and store a packet. When the acknowledge signal appears at the input of "A1," it will turn the output of "C1" to "1," which will clear the request signal via "A2" and enable the "C2" to capture the falling edge of the acknowledge signal through "A3." The falling edge of the "ack" signal means that the required tasks of the current step have been done and the current handshake process is completed. Hence, when it appears at the input of "A3," the output of "C2" will be triggered to "1" to enable the next stage to take over the control process of the next operation, for example, receiving next data packet cell in the "packet receive" block.

The presented block control pipeline structure can only meet quasidelay-independent (QDI) model because the input "ack" signal is branched to "A1" and "A3," however, the timing requirement for distributing the "ack" input signal along the isochronic wire forks is quite loose since the logic delays in "A1" and "C1" are usually much larger than the logic delay of the inverter at the input of "A3."

The control pipeline structure illustrated in Figure 14 is also used in the "Tx/Rx packet buffer" blocks to control the process of accessing the asynchronous FIFOs presented in [19].

5.2. Asynchronous design in the CDMA NoC

As illustrated in Figure 2 and addressed in Section 4.2, the asynchronous blocks in the CDMA NoC include the "CDMA transmitter," "network arbiter," "Tx/Rx packet buffer," and "packet receiver/sender" blocks. Since the "CDMA transmitter" and "network arbiter" blocks are data-path centric blocks, the control logic used in these blocks can be composed by a straightforward C-element pipeline as illustrated in Figure 15. Each stage in the C-element pipeline is enabled by the enable signals generated by the data completion detection circuits. The control token will be passed from one stage to the next one through each C-element in the pipeline.

The control logic used in the "Tx/Rx packet buffer" and "packet receiver/sender" blocks of the network node for the CDMA NoC is similar to the control logic illustrated in Figure 14 except that the enable conditions of the C-element are different.

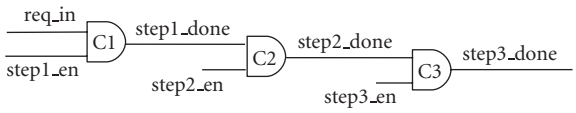


Figure 15: C-element control pipeline.

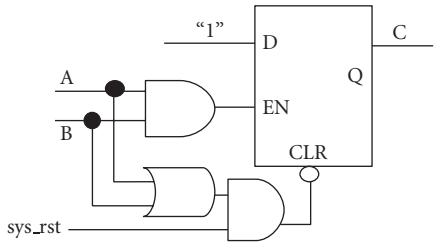


Figure 16: C-element structure.

5.3. Asynchronous design implementation

Since the synchronous designs in the presented PTP NoC and the CDMA NoC are done in register-transfer level (RTL) by using VHDL, it would be convenient for the implementation if the asynchronous designs apply the same design format. From the control logic structures described, we can see that the C-element is a basic component widely used in the asynchronous designs and a C-element is normally implemented in transistor level. Therefore, modeling the C-element in RTL is an important task for modeling the asynchronous designs using VHDL. Hence, an RTL two-input C-element structure is proposed in Figure 16. The proposed C-element structure is based on a D-flipflop (D-FF) which uses "A AND B" as the enable ("EN") signal and "A OR B" as the reset signal ("CLR"). The data input port ("D") of the D-FF is attached to logic "1" constantly. The idea of using a flipflop to build a C-element has been presented in [21] where an RS-FF is suggested to be used; whereas, the D-FF C-element structure in Figure 16 is more stable because it avoids data switching at the data input port.

The C-element structure illustrated in Figure 16 is hazard-free under one-input change assumption by applying the "AND" gate and "OR" gate at the "EN" port and "CLR" port, respectively. For certain two-input switch patterns, 00→11 and 11→00, the structure in Figure 16 is also hazard-free; whereas, the input switch patterns, 01→10 and 10→01, are not allowed because they may produce a logic error which depends on the wire delay. Because all the C-elements in the PTP and the CDMA NoC designs are used to follow a four-phase handshake protocol, there are no 01→10 or 10→01 input switch patterns for the C-elements in the designs. Thus, the proposed C-element structure can be safely used in the NoC designs.

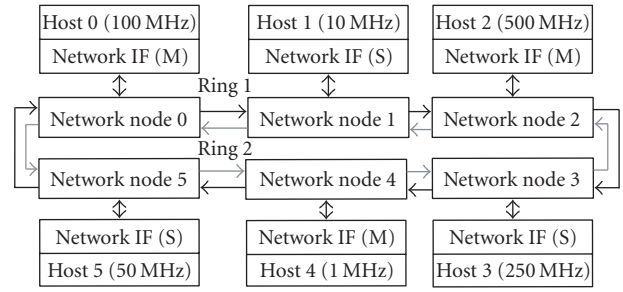


Figure 17: Six-node PTP NoC simulation network.

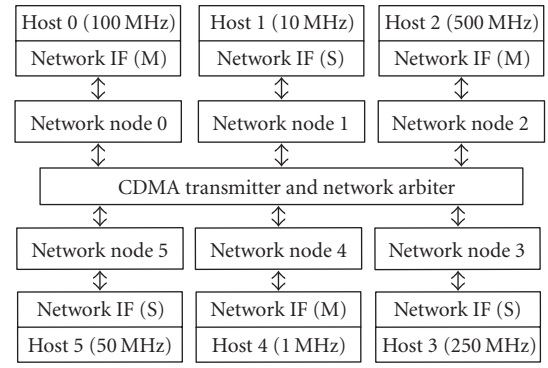


Figure 18: Six-node CDMA NoC simulation network.

By using the proposed RTL C-element structure, the asynchronous designs of the two NoCs are modeled in RTL using VHDL. Since the entire designs are in a uniform VHDL format, the commonly used synchronous design tools and flow can be used for implementing the NoC designs.

6. PERFORMANCE ESTIMATION

After the structures and designs of the PTP NoC and the CDMA NoC have been discussed, the performance of the two NoC designs will be addressed in this section. The two six-node simulation networks used for performance estimations and the estimation results will be presented and discussed in the following subsections.

6.1. The simulation network setup

In order to estimate the performance, two six-node networks have been set up for simulation purpose. The simulation network which applies the PTP NoC structure is illustrated in Figure 17, while the network which applies the CDMA NoC structure is illustrated in Figure 18. In each of the two simulation networks, six "functional host" blocks are connected into the network through six network nodes. The network nodes are connected to each other through the two different NoC structures, respectively, for the purpose of comparison. The interface standard applied in the simulation networks is BVCI standard [16]. Three hosts act as masters and the other three act as slaves, as denoted by the labels "M" and "S,"

Table 1: Area cost of the PTP NoC components.

Blocks of network node	Area (μm^2)
Node IF (BVCI slave type)	13430.8
Layer MUX	18346.0
Communication controller	7823.4
Packet distributor	6783.0
Packet sender (include Tx packet buffer)	44740.6
Packet receiver	6955.0
Rx packet buffer	40255.5
Total area of a network node (includes 2 "communication layer blocks")	**244891.8**

Table 2: Area cost of the CDMA NoC components.

Block name		Area (μm^2)
Network node	Node IF	18825.2
	Tx/Rx packet buffer	71778.3
	Packet sender	17707.0
	Packet receiver	23253.0
	Total area of a network node	**131563.5**
	CDMA transmitter	10338.3
	Network arbiter	17686.5

respectively, in the "network if" blocks. The master hosts can generate requests to any slave hosts, while the slave hosts can generate responses only for the received requests passively. The functional hosts in the two simulation networks are not implemented as any concrete designs. They are simulated by the stimulus which comes from the "network if" block to the network nodes. Hence, the configurations of the two simulation networks are the same except for the connection structures.

6.2. The area costs

By using the scheme described in Section 5.3, both the synchronous and asynchronous designs in the simulation networks are realized in RTL using VHDL. A 0.18 μm standard cell library is used for synthesizing the two simulation networks. The area costs of the two simulation networks are listed in Tables 1 and 2, respectively. As listed in Table 2, the area cost of the "network node" in the CDMA NoC is 53% smaller than the area of the "network node" block in the PTP NoC. The reason of this big difference of the area costs is that there are two "communication layer" blocks contained in each network node of the PTP NoC in order to set up bidirectional ring links. After including the area costs of "CDMA transmitter" and "network arbiter" blocks, the total area cost of the six-node CDMA NoC is 55% smaller than the area cost of the six-node PTP NoC.

6.3. The data transfer latencies

After the networks were synthesized, gate-level simulations of the two networks were performed using an event-driven simulator. Because the GALS scheme is applied in the two simulation networks, the data transfer latency of the networks can be separated into two parts which include synchronous transfer latency (STL) and asynchronous transfer latency (ATL). The STL refers to the data transfer latency between a functional host and the network node attached to it. STL depends on the local clock and the type of interface. Since the same "node if" block is applied in both networks, the STL values for the two simulation networks are the same. The measured STL values are listed in Table 3. The constant values in Table 3 are caused by the handshakes in the asynchronous domain. They are independent of the local clock rate but belong to the synchronous transfer processes. Therefore they are counted as a part of STL.

Table 3: Synchronous transfer latency.

Interface type	Latency of sending data to "network node"	Latency of receiving data from "network node"
BVCI master	8 local clock cycles + 2.5 ns	8 local clock cycles + 3.2 ns
BVCI slave	4 local clock cycles + 2.5 ns	4 local clock cycles + 3.1 ns

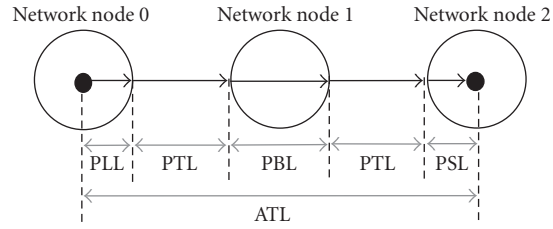


Figure 19: ATL parameters of the PTP NoC.

The ATL refers to the data transfer latency of transferring data packets from one network node to the other node through an NoC structure using asynchronous handshake protocols. The ATL values in the PTP and CDMA simulation networks consist of different parameters which will be discussed in the following subsections.

6.3.1. ATL in the PTP NoC

The ATL in the PTP NoC consists of four parameters: packet loading latency (PLL), packet transfer latency (PTL), packet bypass latency (PBL), and packet storing latency (PSL). These latency parameters are measured in a noncongested situation which means that no conflicts between "bypass packet" transfer and the "local packet" transfer are included in the simulation. The concept of the four latency parameters is illustrated in Figure 19 with an example that "network node 0" sends one packet to "network node 2" via "network node 1." The black arrows in Figure 19 represent the packet transfer direction. The portions of the transfer used to measure the different parameters of latency are marked by grey arrows in Figure 19 and are explained in the next four

Table 4: Measured ATL values of the PTP NoC.

Packet length	PLL (ns)	PTL (ns)	PBL (ns)	PSL (ns)
2 data cells	11.7	9.7	10.7	3.3
3 data cells	15.2	13.1	14.2	3.3
4 data cells	18.6	16.5	17.6	3.3

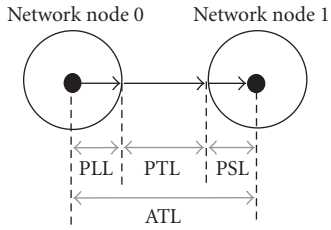


Figure 20: ATL parameters of the CDMA NoC.

paragraphs,

$$\text{ATL} = \text{PLL} + \text{PTL} \times (N + 1) + \text{PBL} \times N + \text{PSL}. \quad (1)$$

(1) *Packet load latency (PLL)*. It is the time used to load one "local packet" into "Tx packet buffer."

(2) *Packet transfer latency (PTL)*. This latency refers to the time used to transfer one data packet from the "packet sender" of a network node to the "packet receiver" of an adjacent node using a handshake protocol.

(3) *Packet bypass latency (PBL)*. After a network node receives a packet from another node, it will check its destination address. If it is a "bypass packet," it will be delivered into "Tx packet buffer." The time spent on this process is called PBL.

(4) *Packet storing latency (PSL)*. It is the time spent on storing one "incoming packet" into "Rx packet buffer."

The formula of calculating the ATL of transferring one packet in the PTP NoC is given in (1). It represents the situation in which the packet traverses several network nodes before reaching its destination. N refers to the number of intermediate nodes between the source node and destination node of a packet. If a packet is transferred between two adjacent network nodes, then N is 0. The values of ATL parameters measured in the simulation are listed in Table 4. The listed latency values only include the logic gate delay of the circuits, no wire delay is considered. More accurate latency values could be obtained by including the wire delay after layout.

6.3.2. ATL in the CDMA NoC

The ATL in the CDMA NoC consists of three parameters: packet loading latency (PLL), packet transfer latency (PTL), and packet storing latency (PSL). The concept of those ATL parameters is illustrated in Figure 20 with an example where "network node 0" sends one data packet to "network node 1." The black arrows in Figure 20 represent the packet transfer direction. The portions of the transfer used to measure the different parameters of ATL are marked by grey arrows in Figure 20 and are explained in the following three paragraphs.

Table 5: Measured ATL values of the CDMA NoC.

Packet length	PLL (ns)	PTL (ns)	PSL (ns)
2 data cells	5.7	384.6	5.5
3 data cells	5.7	768.9	5.5
4 data cells	5.7	1153.7	5.5

(1) *Packet load latency (PLL)*. This is the time used by the "packet sender" block in a "network node" to fetch one data packet from "Tx packet buffer" and prepare to send the data packet to "CDMA transmitter."

(2) *Packet transfer latency (PTL)*. This latency refers to the time used to transfer one data packet from the "packet sender" of the sender node to the "packet receiver" of the receiver node through the CDMA channel using a handshake protocol.

(3) *Packet storing latency (PSL)*. After the receiver node receives a data packet, it will spend a certain amount of time to store the received data packet into "Rx packet buffer." This time duration is measured as PSL.

The measured values of ATL parameters of the CDMA NoC are listed in Table 5. The listed latency values only include the logic gate delay of the circuits, no wire delay is considered. In Table 5, we can see that PTL increases as the packet length increases. This is because the data cells in a packet are sent in a serial manner in the CDMA NoC. Thus, more data cells need more transmission time. The PLL and PSL values are not affected by the packet length. The reason is that the data cells in a packet are loaded or stored in a parallel manner.

6.3.3. Comparing the ATL values

From the simulation results presented in Sections 6.3.1 and 6.3.2, we can see that the ATL value in the six-node CDMA NoC is a stable value for a certain data packet length, whereas the ATL value in the PTP NoC is a variable depending on the packet traffic route. The ATL parameter "PBL" of the PTP NoC does not exist in the ATL of the CDMA NoC because the data packets in the CDMA NoC are transferred directly from their source nodes to their destination nodes. The stable ATL value is an advantage of the CDMA NoC since it is very helpful for supplying guaranteed transfer latency service in the network. However, by comparing with the ATL value of the PTP NoC, the ATL value of the CDMA NoC is much larger. For example, according to values listed in Table 5, ATL of transferring a two-cell packet in the CDMA NoC is 395.8 nanoseconds. This value equals the ATL value of transferring the same size packet through 17 intermediate nodes in the PTP NoC according to (1) and Table 4. The reason for the large ATL value in the CDMA NoC is that each original data bit is extended into S bits by an S-bit spreading code during the obligatory data spreading process for CDMA transmission, and the encoded data are transferred bit by bit in the current realization of the CDMA NoC; whereas, in the

PTP NoC, the data bits are transferred cell by cell without any encoding, namely 32 original data bits are transferred at one time. Therefore, the ATL value of CDMA NoC can be reduced by transferring the encoded bits in parallel.

6.4. SystemC modeling for further estimation

The data transfer latency estimations made in Section 6.3 are based on two six-node simulation networks. However, in different applications, the number of nodes in an NoC can be different. Therefore, data transfer latency estimations under different numbers of network nodes of the two NoC designs would be helpful for further evaluation.

As discussed in Section 6.3.1, the data transfer latency of the PTP NoC is mainly affected by the number of intermediate network nodes which a packet passes through during the transfer. Therefore, by using the transfer latency values extracted from the six-node RTL simulation network, the ATL values of the PTP NoC with different numbers of network nodes can be estimated by using (1). For the CDMA NoC, the ATL values with different network node numbers are difficult to get from the six-node simulation network presented in Section 6.3 due to the lack of scalability in the CDMA NoC. Since the data transfer latency estimation presented in Section 6.3 is based on the RTL simulation network realized by using VHDL, any changes in the simulation network will incur a time-consuming synthesis and simulation design cycle. Therefore, a flexible and fast simulation model of the CDMA NoC is preferred for further ATL performance estimations of the CDMA NoC.

SystemC [22] is a C++ class library which can be used to model system-level designs. Since a SystemC model is totally described by a software programming language, the abstraction level of the system model can be very flexible and the simulation can run at a faster speed than an RTL model. Thus, a SystemC model of the CDMA NoC is built for the flexible and fast simulation purpose.

The SystemC model of the CDMA NoC is built in transaction level by modelling each block of the CDMA NoC as a channel [22]. The asynchronous communications among the blocks are modelled by calling each others' channel interface functions in the SystemC model. In order to estimate the ATL values of the CDMA NoC via the transaction-level SystemC model, a set of latency values listed in Table 6 is extracted from the gate-level simulation of the RTL six-node CDMA network presented in Section 6.3. By back-annotating the transfer latency values to the corresponding channels in the SystemC model, the ATL estimations of the CDMA NoC with different numbers of network nodes can be obtained through simulating the SystemC model in transaction level. The obtained ATL estimation values from the SystemC model simulations are listed in Table 7. From Table 7, we can see that the transfer latency of the CDMA NoC increases as the number of network nodes increases. The main reason of the latency increasing is that the data encoding latency in "CDMA transmitter" block is getting larger when the number of network nodes increased. Another reason is that the width of the orthogonal codes used for encoding increases as the number of nodes increased in the network. Thus, the spreading code loading latency in "packet receiver" block would be increased as a consequence. Because the back-annotated SystemC model of the CDMA NoC only uses a limited number of extracted latency values presented in Table 6, the ATL values listed in Table 7 only can give a quick glimpse on the latency situations when the number of network nodes in the CDMA NoC is changed. The accurate latency information needs to be obtained through real circuit implementations.

Table 6: Extracted transfer latency values.

Blocks	Processes	Latency
Tx/Rx packet buffer	Read	10.9 ns
	Write	11.5 ns
Packet sender	Send a 2-cell packet to "CDMA transmitter"	99.2 ns
Packet receiver	Load decoding PN code	1.2 ns
	Receive a 2-cell packet from "CDMA transmitter"	192.0 ns
Network arbiter	Arbitration	4.3 ns
CDMA transmitter	Data encoding	2.9 ns

Table 7: ATL estimation values of the CDMA NoC with different numbers of nodes.

Number of nodes	Spreading code length (bits)	ATL of sending a 2-cell packet
3	4	362.7 ns
6	8	411.8 ns
12	16	510.0 ns
24	32	706.4 ns

7. CONCLUSION

A PTP connection NoC and a CDMA connection NoC were examined and compared in this paper. Both of the presented NoC designs are packet-switched networks and support the GALS communication scheme. The two NoC designs are compared in terms of NoC structures, data transfer principles, network node designs, asynchronous designs, and the performances. The features of the two NoC structures are summarized in the following five paragraphs.

(1) *NoC structures.* The PTP NoC applies direct links among the network nodes and a distributed traffic scheme for the data communication. The CDMA NoC applies a centralized connection scheme and provides parallel data transfers in time domain.

(2) *Data transfer principles.* The PTP NoC transfers data packets in their original form among the links in the network. The CDMA NoC applies CDMA technique to share the centralized communication channel among all the network nodes both in time and space domains. The data streams from different network nodes in the CDMA NoC are separated from each other by encoding them with a set of orthogonal codes.

(3) *Network node designs.* The network node structure in the PTP NoC is more complex than the structure of the network node in the CDMA NoC. The communication control tasks in the CDMA NoC network node are less than the tasks in the PTP NoC network node. The complexity caused by packet routing processes in the network node of the PTP NoC is avoided in the network node of the CDMA NoC.

(4) *Asynchronous designs.* The asynchronous designs applied in the PTP and CDMA NoCs are similar to each other. The four-phase dual-rail protocol is applied in both NoC designs. The control logic used in the asynchronous designs of the two NoC designs is based on the micropipeline control logic. Both the synchronous and asynchronous designs in the two NoC designs are realized in RTL using VHDL.

(5) *Performance estimations.* Two simulation networks which apply the PTP NoC structure and the CDMA NoC structure, respectively, have been synthesized using a 0.18 μm standard cell library. The area cost of the CDMA simulation network is 55% smaller than the PTP simulation network. When the number of network nodes is certain, the ATL value of the CDMA NoC is a stable value for the same-size packets. However, the ATL value of the PTP NoC is smaller than the value of the CDMA NoC when no data transfer congestions are considered in the PTP NoC. One reason of the large ATL in the CDMA NoC is that the applied data spreading technique produces a large amount of encoded data bits for transmission. Another reason is that the encoded data are delivered bit by bit in the CDMA NoC, whereas the PTP NoC transfers 32 data bits at one time. Therefore, the ATL value of the CDMA NoC can be improved largely by increasing the number of data bits delivered at one time.

REFERENCES

[1] J. A. J. Leijten, J. L. van Meerbergen, A. H. Timmer, and J. A. G. Jess, "PROPHID: a data-driven multi-processor architecture for high performance DSP," in *Proceedings of European Design and Test Conference*, p. 611, Paris, France, March 1997.

[2] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proceedings of the Design, Automation and Test in Europe Conference (DATE '00)*, pp. 250–256, Paris, France, March 2000.

[3] D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi, "Issues in the development of a practical NoC: the Proteo concept," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 95–105, 2004.

[4] J. Muttersbach, T. Villiger, H. Kaeslin, N. Felber, and W. Fichtner, "Globally-asynchronous locally-synchronous architectures to simplify the design of on-chip systems," in *Proceedings of the 12th Annual IEEE International ASIC/SOC Conference*, pp. 317–321, Washington, DC, USA, September 1999.

[5] C. A. Zeferino, F. G. M. E. Santo, and A. A. Susin, "ParlS: a parameterizable interconnect switch for networks-on-chip," in *Proceedings of the 17th Symposium on Integrated Cicuits and Systems Design (SBCCI '04)*, pp. 204–209, Pernambuco, Brazil, September 2004.

[6] K. Goossens, J. Dielissen, and A. Rădulescu, "Æthereal network on chip: concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, 2005.

[7] X. Wang and J. Nurmi, "An on-chip CDMA communication network," in *Proceedings of International Symposium on System-on-Chip*, pp. 155–160, Tampere, Finland, November 2005.

[8] X. Wang, D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi, "Asynchronous network node design for Network-on-Chip," in *Proceedings of International Symposium on Signals, Circuits and Systems (ISSCS '05)*, vol. 1, pp. 55–58, Iasi, Romania, July 2005.

[9] B.-K. Tan, R. Yoshimura, T. Matsuoka, and K. Taniguchi, "A novel dynamically programmable arithmetic array using code division multiple access bus," in *Proceedings of the 8th IEEE International Conference on Electronics, Circuits and Systems (ICECS '01)*, vol. 2, pp. 913–916, Malta, September 2001.

[10] S. Shimizu, T. Matsuoka, and K. Taniguchi, "Parallel bus systems using code-division multiple access technique," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 240–243, Bangkok, Thailand, May 2003.

[11] M. Takahashi, B.-K. Tan, H. Iwamura, T. Matsuoka, and K. Taniguchi, "A study of robustness and coupling-noise immunity on simultaneous data transfer CDMA bus interface," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 611–614, Phoenix, Ariz, USA, May 2002.

[12] R. H. Bell Jr., C. Y. Kang, L. John, and E. E. Swartzlander Jr., "CDMA as a multiprocessor interconnect strategy," in *Proceedings of the 35th Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1246–1250, Pacific Grove, Calif, USA, November 2001.

[13] E. H. Dinan and B. Jabbari, "Spreading codes for direct sequence CDMA and wideband CDMA cellular networks," *IEEE Communications Magazine*, vol. 36, no. 9, pp. 48–54, 1998.

[14] E. S. Sousa and J. A. Silvester, "Spreading code protocols for distributed spread-spectrum packet radio networks," *IEEE Transactions on Communications*, vol. 36, no. 3, pp. 272–281, 1988.

[15] D. D. Lin and T. J. Lim, "Subspace-based active user identification for a collision-free slotted ad hoc network," *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 612–621, 2004.

[16] VSI Alliance, "Virtual Component Interface Standard version 2," April 2001, http://www.vsi.org/.

[17] OCP-IP Association, "Open Core Protocol Specification," 2001, http://www.ocpip.org/.

[18] A. P. Niranjan and P. Wiscombe, "Islands of synchronicity, a design methodology for SoC design," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 3, pp. 64–69, Paris, France, February 2004.

[19] X. Wang and J. Nurmi, "A RTL asynchronous FIFO design using modified micropipeline," in *Proceedings of the 10th Biennial Baltic Electronics Conference (BEC '06)*, Tallinn, Estonia, October 2006.

[20] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, 1989.

[21] Q. T. Ho, J. B. Rigaud, L. Fesquet, M. Renaudin, and R. Rolland, "Implementing asynchronous circuits on LUT based FPGAs," in *Proceedings of the 12th International Conference on Field-Programmable Logic and Applications (FPL '02)*, vol. 2438 of *Lecture Notes in Computer Science*, pp. 36–46, Montpellier, France, September 2002.

[22] *IEEE Standard SystemC Language Reference Manual*, http://www.systemc.org/.

Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 95348, 17 pages
doi:10.1155/2007/95348

Research Article

Stochastic Communication: A New Paradigm for Fault-Tolerant Networks-on-Chip

Paul Bogdan, Tudor Dumitraş, and Radu Marculescu

Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA

Received 12 December 2006; Accepted 6 February 2007

Recommended by Maurizio Palesi

As CMOS technology scales down into the deep-submicron (DSM) domain, the costs of design and verification for Systems-on-Chip (SoCs) are rapidly increasing. Relaxing the requirement of 100% correctness for devices and interconnects drastically reduces the costs of design but, at the same time, requires SoCs to be designed with some degree of system-level fault-tolerance. Towards this end, this paper introduces a novel communication paradigm for SoCs, called *stochastic communication*. This scheme separates communication from computation by allowing the on-chip interconnect to be designed as a reusable IP and also provides a built-in tolerance to DSM failures, without a significant performance penalty. By using this communication scheme, a large percentage of data upsets, packet losses due to buffers overflow, and severe levels of synchronization failures can be tolerated, while providing high levels of performance.

1. INTRODUCTION AND NOVEL CONTRIBUTION

Nowadays, the application-specific integrated circuits (ASICs) have evolved into complicated systems-on-chip (SoCs), where dozens, and soon hundreds, of predesigned IP cores are assembled together to form large chips with complex functionality. Extensive research on how to integrate and connect these IPs is currently being conducted, but there are many open issues that are difficult to address within the framework of existing CAD tools and design methodologies.

Indeed, shrinking transistor dimensions, smaller interconnect features, and higher operating frequencies lead to a higher sensitivity of deep-submicron (DSM) circuits to neutron and alpha radiation, significantly higher soft-error rates, and an increasing number of timing violations [1]. These new types of failures are impossible to characterize using deterministic measurements and, thus, *probabilistic metrics*, such as average values and variances, are likely to be needed to quantify the critical design objectives, such as performance and power. It has become clear that, in order to reduce the cost of design and verification, the "100% correctness" requirement for VLSI circuits has to be relaxed [2, 3]. This means that, in the near future, circuits will be designed with some degree of architectural and system-level fault-tolerance [4–6].

Furthermore, as emphasized in the ITRS 2001 [3] it is very important, especially at the system-level, to separate the computation from communication, as they are orthogonal issues which should remain separate. A novel communication paradigm has to be developed in order to enable the separation between the design of the on-chip communication architecture and the design of the SoC main functionality.

Traditionally, the IP cores on a typical SoC are connected with a shared bus or a hierarchy of buses. When a bus needs to connect a large number of modules, its performance decreases drastically because of the contention for access to the shared medium. Therefore, this communication architecture is not well suited to future SoCs which may incorporate hundreds of communicating IPs. On-chip buses need to be supplemented or even replaced with a more scalable on-chip communication infrastructure [7].

A recently proposed design platform for the on-chip interconnect is the network-on-chip (NoC) architecture [8, 9], where the IPs are placed on a grid (see Figure 1) and the communication between tiles is implemented by a stack of networking protocols. Such regular structures are very attractive because they can offer well-controlled electrical parameters, which enable high-performance circuits by reducing latency and increasing bandwidth. From this perspective, the SoC design will resemble more the creation of large-scale

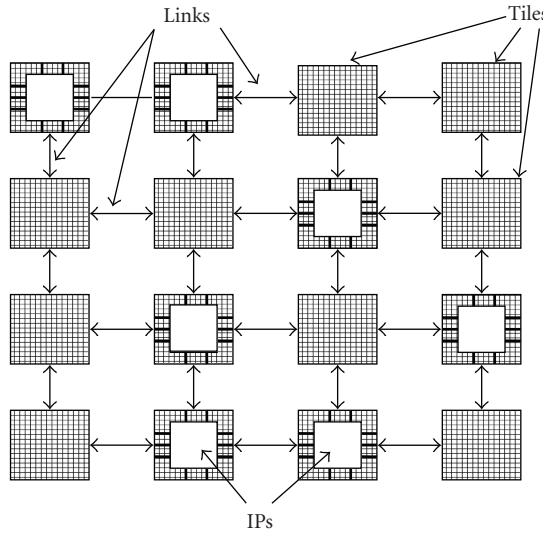


Figure 1: Network-on-chip (NoC).

communication networks rather than traditional IC design practice.

However, providing communication via NoCs is not an easy matter, as mitigating the effects of on-chip failures on the network communication remains largely an open question. Furthermore, the resources used in traditional networks in order to achieve fault-tolerance are not easily available in VLSI chips. For instance, the static routing approach transmitting messages along a fixed path would fail if even a single tile or link on the path becomes faulty. Generally speaking, the deterministic algorithms do not behave well in the presence of random failures [10, 11]. On the other hand, implementing adaptive dynamic routing for on-chip networks is prohibitive because of the need for very large buffers, lookup tables, and complex shortest-path algorithms [12].

Perhaps the greatest challenge introduced by the advent of new technologies is the shift from design determinism to *design uncertainty* [13, 14]. Failures that occur in the DSM technologies can only be characterized by stochastic models, as they are either nondeterministic in nature or too complex to be described by simplistic models. Therefore, the on-chip communication has to be aware of this inherent nondeterminism induced by the DSM technologies.

1.1. Contributions of this paper

The research presented in this paper addresses the issue of on-chip fault-tolerant communication. Towards this end, we introduce a novel communication paradigm, called *on-chip stochastic communication*. In an NoC such as the one in Figure 1, the IPs can communicate using a probabilistic broadcast scheme, very similar to the *randomized gossip* protocols used in databases or sensor networks [15, 16]. More precisely, if a tile has a message that needs to be transmitted, this will be forwarded to a randomly chosen subset of the tiles in the neighborhood. Hence, the messages are *diffused* through the network. At the same time, every IP selects from the set of received messages, only the messages whose destination field is identical to the ID of the tile. The behavior of this communication scheme is similar, but *not* identical, to the proliferation of an epidemic in a large population[1] [17].

This approach achieves many desired features of future SoCs. As shown later in the paper, the algorithm provides the following.

(i) *Separation between computation and communication*, as the communication scheme is implemented in the network logic and is transparent to the IPs.
(ii) *Fault-tolerance* since a message can still reach its destination despite severe levels of DSM failures.
(iii) *Extremely low latency* since this communication scheme does not require retransmission of corrupted data.
(iv) *Low production costs* because the fault-tolerant nature of the algorithm eliminates the need for detailed testing/verification.
(v) *Design flexibility* since it provides a mechanism to tune the tradeoff between performance and energy consumption.

We also note that the proposed on-chip stochastic communication is accompanied with a theoretical justification. The analytical model allows us to explain the essence of stochastic communication and determine nodes coverage which is later validated via simulation.

1.2. Structure of this paper

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 introduces a fault model for NoCs which captures the typical errors that may appear in a DSM circuit. Section 4 describes the on-chip stochastic communication algorithm meant to work in a failure-prone environment. Section 5 presents the analytical treatment of the novel communication paradigm. Section 6 presents the experimental results obtained for a simple two-dimensional FFT application working on a flat NoC. Finally, in Section 7 we outline the limitations of the proposed model. We then conclude in Section 8 by summarizing our main contribution.

2. RELATED WORK

The starting point of on-chip stochastic communication is the epidemics theory and gossip algorithms. The epidemics theory has its origin in the Bernoulli's work who tried to prove that the exposure of healthy people to smallpox may contribute to their immunization and decrease the mortality rate [18]. Generally speaking, the *simple* epidemic models characterize the infection process in a finite population of susceptible (S) and infected (I) individuals. The velocity of infection process is given by the product between the num-

[1] This analogy explains intuitively the high performance and robustness of this protocol in the presence of failures.

ber of susceptibles and infected individuals [17, 18]. Besides these two types of individuals, the *general* epidemics model, also called *SIR* model, introduces the removal of individuals (R) which designates an infected person that is removed either by immunization or by death.

In contrast to epidemics, the rumor spreading theory takes into account an additional type of interaction that may happen between a spreader (similar to an infected individual) and a stifler (viewed as a removal). While in epidemics the stifler corresponds to an isolated individual, in rumor spreading it influences directly the dissemination process.

The first complete stochastic model for rumor spreading was introduced by Daley and Kendall in [19]. They divided the entire population of individuals in three categories: spreaders (S), ignorants (I), and stiflers (R). The *spreader* designates an individual (or node) who disseminates the rumor or forwards the message according to a fixed probabilistic rule. An *ignorant* represents an individual who is not aware of the rumor contents yet. A *stifler* refers to an individual who is aware of the rumor, but decides to stop its dissemination.

According to the epidemics modeling, the interaction between a spreader and an ignorant leads to a state with two spreader individuals at a rate proportional to the number of the spreader and ignorant populations (i.e., SI). Similarly, the interaction between a spreader and a stifler may lead to a new stifler at a rate proportional to the number of the spreader and stifler subpopulations (i.e., SR). Moreover, in the case of an interaction between two spreaders, it may happen that one of them becomes a stifler at a rate proportional with the frequency of interactions between individuals belonging to the same subpopulation (i.e., $0.5S(S-1)$).

An alternative approach, called the Maki-Thompson rumor model, is presented in [18]. The main difference between the Daley-Kendall and Maki-Thompson models is that in the later, the interaction between two spreaders results in a new stifler at a rate double than the one in the first approach (i.e., $S(S-1)$). A unified treatment of the rumor spreading process was recently proposed by Pearce [20].

All these approaches inspired the so-called *gossip protocols* for information dissemination in computer systems or sensor networks [15, 16, 21]. They are very attractive for applications that require *localized* communication. A distant ancestor is the USENET news protocol, NNTP [22], developed in early 80s. The news servers running the NNTP protocol exchange updates with the neighboring servers without knowing the entire set of hosts that run NNTP worldwide. Thus, a new message that has been sent to a newsgroup will propagate from server to server until it is known by all of them. An interesting property of this protocol is that the servers are *not* required to know about all other servers, and yet they are able to broadcast the updates to the entire group. This reduces drastically the bandwidth required for the broadcast, which makes this protocol more scalable than most traditional distributed algorithms.

Demers et al. in [15] propose randomized gossip protocols for the lazy update of data objects in databases replicated at many sites. In that paper, the authors show how the gossip-based communication is related to the propagation of epidemics and develop a family of gossip-based multicast protocols. It can be shown that, the updates spread *exponentially* fast among the replicated instances of the database, and the broadcast is accomplished with only a few retransmission rounds.

Several networking protocols, such as the Internet Muse protocol [23], the Scalable Reliable Multicast [24], and the Xpress Transfer Protocol [25] were based on the same principles. Birman et al. in [21] have shown that for the gossip-based multicast protocols there is a high probability that *almost all or almost none* of the players will receive the broadcast, as opposed to the stronger *all or none* guarantee of the classical distributed algorithms. Therefore, these protocols are best suited for applications that can tolerate a small percentage of message losses, but need to be scalable and have a steady throughput.

More recently, these types of algorithms have been applied to the networks of sensors [16]. Their ability to limit the communication to local regions and support light-weight protocols is appealing to applications where power, complexity, and size constraints are critical. Although promising from the practical standpoint, the claims about the practicality of the proposed algorithm are not supported by an analytical framework.

We argue that this communication paradigm can be successfully applied to the SoC design as well, especially for the NoC type of architecture as in Figure 1. Preliminary results investigating the basic paradigm and hardware implementation were presented in [5, 26]. Recently, Manolache et al. in [27] proposed a method to reduce the number of broadcast messages and improve the application response time. In terms of the analytical modeling, while the on-chip stochastic communication resembles epidemics and rumor spreading, it is highly dependent on the topology on which the message diffusion takes place [28]. These topological considerations dictate the calculations of the transition probabilities in our analytical model and help explain the nature of on-chip stochastic communication and interactions that take place. Finally, the analytical model validation is done by evaluating nodes coverage.

3. FAULT MODELING FOR NoCs

Moore's law has been valid for the past three decades. Presently, the advances in wiring and manufacturing technology, as well as the device scaling below the 100 nm threshold, seem to allow Moore's law to continue only for a few more years. Indeed, shrinking transistor dimensions, smaller interconnect features, and higher operating frequencies lead to a higher sensitivity of DSM circuits to neutron and alpha radiation, significantly higher soft error rates, and an increasing number of timing violations [1]. Dependability modeling shows that complex VLSI circuits can be seriously impacted by transient faults and silent data corruption [29, 30]. Furthermore, in order to reduce the costs of design, manufacturing, and verification and make such technologies affordable not only for the highest volume products, the "100% correctness" requirement for VLSI circuits has to be relaxed.

The faults that may appear in NoCs are either transient or permanent. The *transient faults*, also known as *data upsets* or soft errors, are caused by fluxes of neutron and alpha particles, power supply and interconnect noise, electromagnetic interference, or electrostatic discharge. They represent the most common problem for future VLSI circuits. Simply stated, if noise in the interconnect causes a message to be scrambled, a data upset will occur; these faults are subsequently characterized by a probability P_{upset}. As predicted in [1], the rate of occurrence increases as technology scales down into the DSM domain.

Permanent faults reflect irreversible physical changes in the structure of the circuit. They make recovery very hard or even impossible. However, while these errors occur infrequently [31] and do not pose a serious threat to the mass production of VLSI chips, this may change for future nanotechnologies [32].

Another common failure is when a message is lost because of buffer overflow. These faults are known as send/receive omissions [33] or *buffer overflows*. The occurrence rate of these faults is modeled by the probability P_{overflow}.

For complex SoCs, there are additional, more subtle, error modes that can appear. The high coupling capacities of the interconnect and tighter integration favor the Miller effect. As such, it becomes very difficult to achieve predictable delays. Furthermore, as modern circuits span multiple clock domains (e.g., GALS architectures [34]), and can function at different voltages and frequencies (as for "voltage/frequency island"-based architectures [35]), the communication between domains has to go through a special interface with mixed clocks [36]. Due to the special handshake needed before transferring data, the latency in communication may increase and *synchronization errors* appear. In our experiments, every tile has its own clock domain. In this architecture, synchronization errors are normally distributed with a standard deviation σ_{synchr}.

In summary, our fault model depends on the following parameters:

(i) P_{tiles} and P_{links}: probability that a tile/link is affected by a *permanent failure*,
(ii) P_{upset}: probability that a packet is scrambled because of a *data upset*,
(iii) P_{overflow}: probability that a packet is dropped because of *buffer overflows*,
(iv) σ_{synchr}: standard deviation error of the duration of a round (T_R) which indicates the magnitude of *synchronization errors*.

4. THE IDEA OF ON-CHIP STOCHASTIC COMMUNICATION

Traditionally, data networks dealt with fault-tolerance by using complex algorithms, like the Internet Protocol or the ATM Layer [23]. However, these algorithms require many resources that are not available on chip. In addition, they are not always able to guarantee a low latency which is vital for SoCs. For example, in these protocols, the packets are protected by a *cyclic redundancy code* (*CRC*) which is able to detect if a packet contains correct or corrupted data. If an error is detected, the receiver asks for the retransmission of the scrambled messages. This method is known as the *automatic retransmission request (ARQ)*; it has the disadvantage that it increases the communication latency. Another approach is the *forward error correction* (*FEC*), where the errors are corrected directly by the receiver by using an error correction scheme, like the Reed-Solomon code. *FEC* is appropriate when a return channel is not available, as in deep-space communications or in audio CD recordings. *FEC*, however, is less reliable than *ARQ* and incurs significant additional processing complexity.

Based on these considerations, we propose a fast and computationally lightweight scheme for the on-chip communication, based on an error-detection/multiple-transmissions scheme. The key observation behind our strategy is that, at chip level, the bandwidth is less expensive than in traditional networks; this is due to the existing high-speed buses and interconnect fabrics which can be used to implement NoCs. In the following subsections, we illustrate the idea of stochastic communication and then describe the mathematical model of on-chip stochastic communication which is built on this very principle.

4.1. Example of a Producer-Consumer application

In Figure 2, we give the example of a generic Producer-Consumer application. On an NoC with 16 tiles, the Producer is placed on tile 6 and the Consumer on tile 12. Suppose the Producer needs to send a message to the Consumer. Initially the Producer sends the message to a randomly chosen subset of its neighbors (e.g., tiles 2 and 7 in Figure 2(a)). At the second gossip round, tiles 6, 2, and 7 forward it in the same manner. After this round, eight tiles (i.e., 6, 2, 7, 1, 3, 8, 10, and 11 in Figure 2(b)) become aware of the message and are ready to send it to the rest of the network. At the third gossip round, the Consumer finally receives the packet from tiles 8 and 11 (see Figure 2(c)). Note that

(i) the Producer needs *not* know the location of the Consumer and the message will arrive at the destination *with high probability (w.h.p.)*;[2]
(ii) the message reaches the Consumer *before* the full broadcast is completed. For instance, by the time the Consumer gets the message, tiles 13–16 have not received the message yet.

The most appealing feature of this algorithm is its excellent performance in presence of failures. For instance, suppose the packet transmitted by tile 8 is affected by a data upset. The Consumer will discard it as it receives (from tile 11) another copy of the same packet anyway. The presence of such upsets is detected by implementing an error-detection scheme on each tile. Bandwidth is less expensive compared to traditional networks, so we can afford more packet transmissions to simplify the communication scheme and guarantee low latencies.

[2] This means with probability at least $1 - O(n^{-\alpha})$ for any constant $\alpha > 0$.

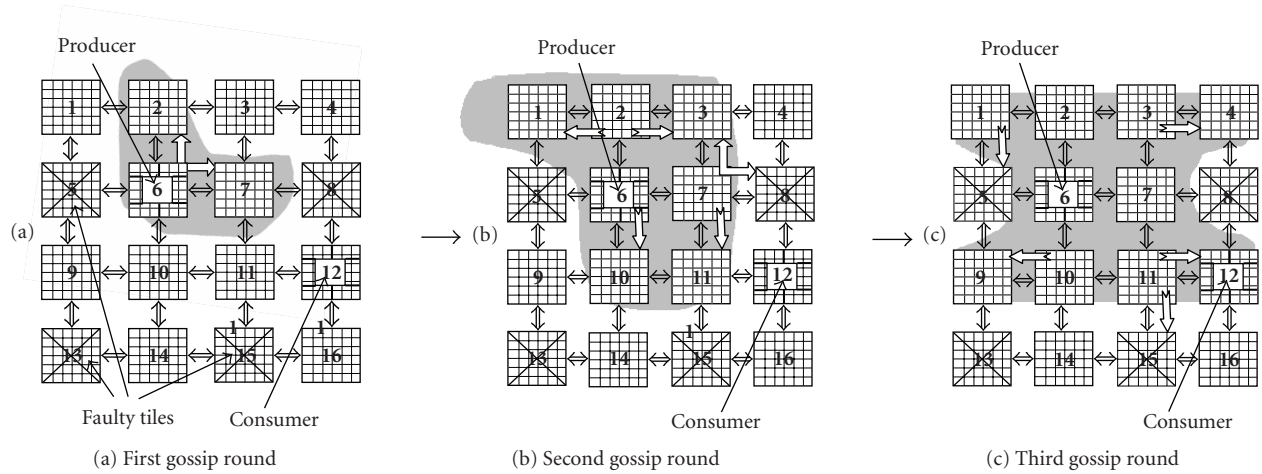


Figure 2: Producer-Consumer application in a stochastically communicating NoC.

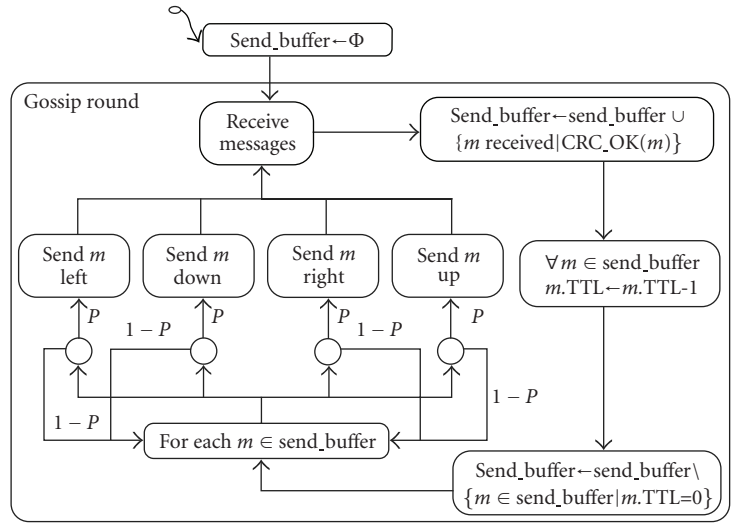


Figure 3: The basic algorithm for stochastic communication.

4.2. A generic algorithm for stochastic communication

The above example assumes that individual tiles can detect if data transmissions are affected by upsets. This is possible by protecting the packets with a CRC code. If an error is discovered, then the packet is simply discarded. Because a packet is retransmitted many times in the network, the receiver does *not* need to ask for retransmission, as it will receive the packet again anyway. This is why stochastic communication can sustain low latencies even under severe levels of failures.

Our proposed algorithm is presented in Figure 3 (with standard set theory notations) and it is executed concurrently by all nodes in the network. A tile forwards the packet that is available for sending to all four output ports, and then a decision with probability P is made whether or not to transmit the message to the next tile. In practice, a gossip round takes several clock cycles. We also note that, since a message can reach its destination before the broadcast is completed, the spreading can be terminated even earlier in order to reduce the number of messages transmitted in the network. This is important because it is directly related to the bandwidth used and energy dissipated (see Section 5.2). To this effect, we assign a *time-to-live* (*TTL*) to every message upon creation and decrement it at every hop until it reaches value 0 and so the message is garbage-collected. In Section 6, we show how the probability P and the *TTL* can be used to tune the trade-off between performance and energy consumption.

4.3. The hardware interface

A typical tile of such an NoC is shown in Figure 4. The IP core is placed in the center of the tile. On the four edges of the tile, there exist buffers to hold the messages that are sent and received by the IP. A *CRC* decoding circuit checks all the

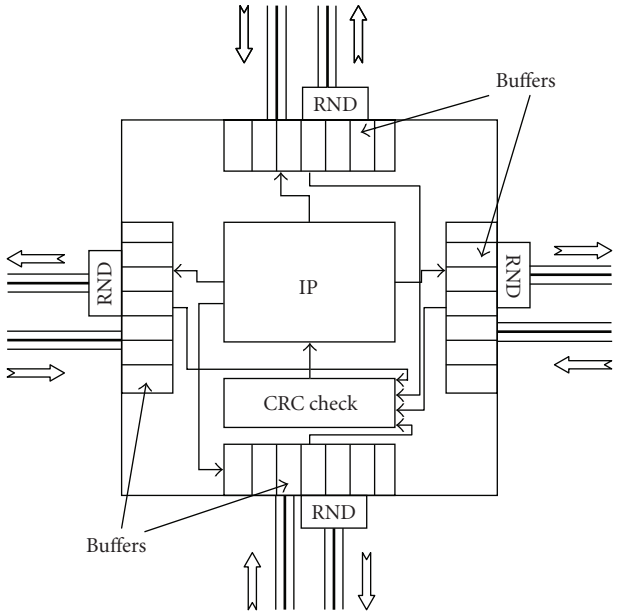


Figure 4: A typical tile of a stochastically communicating NoC.

received packets and when an error is discovered, the message is discarded before being fed into the IP. The tile keeps a list of messages that have to be sent in an output buffer. The messages received during the last round and the new messages generated by the IP core are constantly added to the list. However, if a message is already present, a duplicate message will *not* be inserted. Consequently, even if the message is received a second time from one of the tiles in the neighborhood, only *one copy* is kept in the send buffer and sent to the neighbors during the next round. However, before being actually transmitted over a link, some messages are randomly dropped by a specialized circuit. The selected threshold voltage determines the probability P that a message is forwarded over a link. This is how the aforementioned random subset of neighbors is actually selected.

4.4. Relationship between stochastic communication and rumor spreading

With this technique, the NoC communication becomes similar to the dissemination of a rumor within a large group of friends [37]. Assume that, initially, only one person in the group knows the rumor (see Figure 5(a)). Upon learning the rumor, this person (initiator) passes it to a group of friends (considered confidants if they spread the rumor) randomly chosen. At the next round, both the initiator and the confidants, if there is at least one, independently of each other, select someone else to pass the rumor to. The process continues in the same fashion until everyone is informed or there are no spreaders of the rumor in the entire population.

By analogy with rumor spreading, the NoC tiles are the "gossiping friends," while packets transmitted between them become the "rumors" (see Figure 5(c)). Since in the social network any "friend" is able to communicate with anyone else in the same group (see Figure 5(b)), communication in this setup may lead to the small world phenomenon [38]. From a silicon implementation perspective, however, the topology in Figure 5(b) does *not* represent a viable solution due to its huge wiring demands. Therefore, for NoCs, we consider a grid-based topology (Figure 5(c)), as this is easier and cheaper to implement in silicon.

To analyze the square grid in Figure 5(c)), we clearly need a new method which takes into account the network topology. This is achieved by considering a stochastic model where transition probabilities are dictated by the network topology.[3] This aspect distinguishes the small world approach from the NoC stochastic communication as it will be detailed next. Besides topology, we need to also consider buffer overflows, links and/or nodes failures. Although deriving a good theoretical model for rumor spreading across a grid is an open research question, to the best of our knowledge, our effort represents the first approach that describes both qualitatively and quantitatively the on-chip stochastic communication.

5. ON-CHIP STOCHASTIC COMMUNICATION THEORY

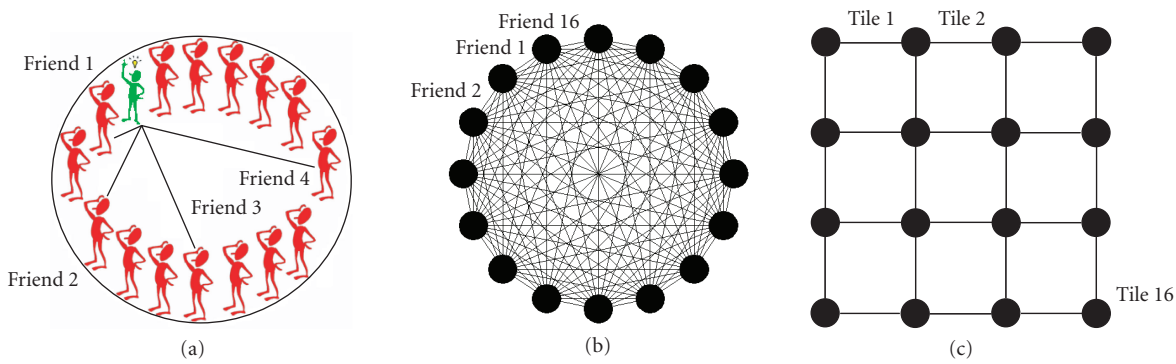
5.1. Basic node interactions and problem formulation

Starting from theory of epidemic and rumor dissemination modelling [17, 19, 37], we consider a grid network of N^2 nodes able to communicate as in Figure 1. Following the model of epidemics theory, we classify the entire population into spreader, ignorant, and stifler nodes. The *spreader* represents any node that is aware of the message and decides to disseminate it. We denote by $S(t)$ the number of spreader nodes at time t. The *ignorant* defines a node that is not included in the communication area yet and it is denoted by $I(t)$. The *stifler* node refers to the case in which a node aware of the message chooses to cease its dissemination (either the link is faulty or the node is dropping the packet). The initial number of spreader and ignorant nodes is denoted by $S(0) = N_1$ and, respectively, $I(0) = N_2$. The number of stifler nodes is obtained by subtracting the spreader and ignorant populations from the total number of nodes: $R(0) = N^2 - N_1 - N_2$. Since we are dealing with a closed population of N^2 nodes, we can write the following conservation law, for time $t \geq 0$:

$$S(t) + I(t) + R(t) = N^2. \tag{1}$$

The process $\{(S, I, R)(t) : t > 0\}$ evolves over a finite state space so that it is completely described by a continuous time Markov chain, for which the initial state and the infinitesimal transition probabilities are specified. While the number of stiflers is simply $R(t) = N^2 - S(t) - I(t)$, it is sufficient to deal with stochastic processes $S(t)$ and $I(t)$ in order to ensure the system evolution. Since we deal with three types of

[3] As we can see in Figure 5(c), each spreader node may communicate directly with at most four neighbors.



(a) Social network (b) Fully connected network (c) Square grid network

FIGURE 5: Different topologies illustrating social and technological networks.

populations, using the decomposition of a pathway into elementary reactions or interactions [39], we can distinguish the following types of interactions.

(a) Spreader-ignorant interaction

In a mesh topology, each spreader node can be surrounded by one, two, three, or four ignorant nodes. This can contribute to an increase with one, two, three, or four new spreaders. Using the law of mass action as in [17], we can write the associated transition probability in the infinitesimal time interval $(t, t+h)$ as follows:

$$\begin{aligned} P\{S(t+h) = s+k,\ I(t+h) = i-k \mid S(t) = s,\ I(t) = i\} \\ = \alpha_k(t)sih + O(h), \quad k = 1,2,3,4, \end{aligned} \tag{2}$$

where $\alpha_1(t)$, $\alpha_2(t)$, $\alpha_3(t)$, $\alpha_4(t)$ are *time varying* rates that characterize the interaction between any spreader and one, two, three, and four ignorant neighbors. More precisely, starting with $s-1$ spreaders and $i+1$ ignorants (i.e., initial state $(s-1, i+1)$), we end up with s spreaders and i ignorants (i.e., final state (s,i)) at a rate proportional with $\alpha_1(t)(s-1)(i+1)$ (i.e., $k=1$).

We note that the α_k parameters in (2) are directly related to the forwarding probability P in the algorithm in Figure 3. More precisely, $\alpha_k = \binom{4}{k}P^k(1-P)^{4-k}$. This illustrates the direct connection between the formalism developed in this section and the "knobs" that control the stochastic communication in Figure 3.

(b) Spreader-spreader interaction

Another type of reaction is the interaction between two spreaders that are neighboring nodes. In this situation, it can happen that either both spreader nodes stop the packet diffusion process with a rate $\beta_2(t)$, or one node ceases the dissemination while the second node continues to forward packets in the network with a rate $\beta_1(t)$. The situations can describe the behavior of a link between two congested nodes which is characterized by P_{overflow}. We do not consider the situation when *both* nodes continue the packet dissemination since this can be viewed as a consequence of the spreader-ignorant interaction mentioned above. The corresponding transition probability for these two cases is

$$\begin{aligned} P\{S(t+h) = s-m,\ I(t+h) = i \mid S(t) = s,\ I(t) = i\} \\ = \beta_m(t)sh + O(h), \quad m = 1,2. \end{aligned} \tag{3}$$

(c) Spreader-stifler interaction

The third type of interaction we consider is between a spreader and a stifler. The result of this interaction consists in altering the state of the spreader node with a rate $\alpha_5(t)$. More precisely, a faulty node, also called stifler, may corrupt the received packet and disseminate it, affecting the state of other nodes. The associated transition probability is given by

$$\begin{aligned} P\{S(t+h) = s-1,\ I(t+h) = i \mid S(t) = s,\ I(t) = i\} \\ = \alpha_5(t)s(N^2 - s - i)h + O(h). \end{aligned} \tag{4}$$

(d) No interaction

If there is no change in the continuous time Markov chain, the corresponding transition probability has the form:

$$\begin{aligned} P\{S(t+h) = s,\ I(t+h) = i \mid S(t) = s,\ I(t) = i\} \\ = 1 - \left\{ \sum_{k=1}^{4} \alpha_k(t)i + \sum_{m=1}^{2} \beta_m(t) + \alpha_5(t)(N^2 - s - i) \right\} sh \\ - O(h). \end{aligned} \tag{5}$$

The parameters that characterize the above interactions were introduced for capturing both permanent and transient errors. For $s+i \le N_1+N_2$ and $i \le N_2$ we define

$$P\{S(t+h) = s,\ I(t+h) = i \mid S(t) = N_1,\ I(t) = N_2\} = P(s,i,t) \tag{6}$$

which stands for the probability that, at time t, we have s spreader nodes and i ignorant nodes. If needed, the number of stifler nodes can be easily obtained by subtracting

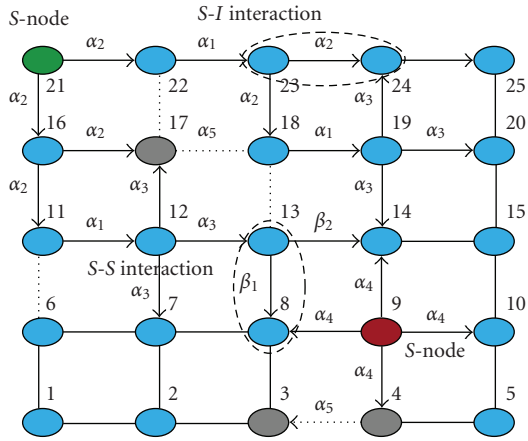


FIGURE 6: Stochastic dissemination in a 5×5 mesh network. The dotted lines show damaged links. The lines with arrows represent packet communication.

from the entire population the number of spreader and ignorant nodes. The packet dissemination process can be then described by the following forward Kolmogorov equation:

$$\begin{aligned}\frac{dP(s,i,t)}{dt} &= \sum_{k=1}^{4}\alpha_k(t)(s-k)(i+k)P(s-k,i+k,t)\\ &+\sum_{k=1}^{2}\beta_k(t)(s+k)P(s+k,i,t)\\ &+\alpha_5(t)(s+1)\left(N^2-s-i-1\right)P(s+1,i,t)\\ &-s\left[\sum_{k=1}^{4}\alpha_k(t)i+\sum_{k=1}^{2}\beta_k+\alpha_5(t)\left(N^2-s-i\right)\right]P(s,i,t).\end{aligned} \tag{7}$$

Next, we give some intuition about the stochastic packet dissemination using a 5×5 mesh network (see Figure 6). We set the nodes 9 and 21 to be the sources for packet dissemination. Source 21 can follow the spreader-ignorant interaction and send packets to its neighbors (i.e., nodes 16 and 22) with rate α_2. Further, nodes 16 and 22 can forward their packets with rates α_2 and α_1, respectively. Node 22 sends packets with rate α_1 (instead of α_2) because the dotted link between nodes 22 and 17 is assumed to be damaged. We observe that node 17 does not send any packet and becomes a stifler with rate α_5. Subsequently, nodes 11, 12, and 13 behave similarly as they describe new spreader-ignorant interactions.

By the same token, the links between the neighboring nodes 6, 11 and 13, 18 are considered damaged. Source 9 can send a packet to its neighbors with probability α_4. Later on, nodes 8, 13, and 14 try to disseminate their packets. If the network is overloaded, we may notice new interactions. For instance, it can happen that only one of the nodes 8 or 13 disseminates its packets; this transition probability (β_1) is shown on the link between them (spreader-spreader interaction). Moreover, if nodes 13 and 14 are blocked due to congestion, then we end up with two nodes blocking the dissemination process (β_2).

Next, we illustrate the derivation of the closed-form solution for the packet diffusion process [28]. First, we use the probability generating functions

$$f_i(x,t)=\sum_{s=0}^{\infty}P(s,i,t)x^s,\quad |x|\le 1,\ 0\le i\le N_2 \tag{8}$$

to rewrite (7) as a transport equation,

$$\begin{aligned}\frac{\partial f_i(x,t)}{\partial t} &= \sum_{k=1}^{4}\alpha_k(t)(i+k)\frac{\partial f_{i+k}(x,t)}{\partial x}+\alpha_5(t)x(x-1)\frac{\partial^2 f_i(x,t)}{\partial x^2}\\ &+\left\{\sum_{k=1}^{2}\beta_k(t)\left[\frac{1}{x^{k-1}}-x\right]+\alpha_5(t)\left(N^2-i-1\right)(1-x)\right.\\ &\left.-x\sum_{k=1}^{4}\alpha_k(t)i\right\}\frac{\partial f_i(x,t)}{\partial x}.\end{aligned} \tag{9}$$

Furthermore, we can construct a vector

$$F(x,t)=\left[f_{N_2}(x,t),f_{N_2-1}(x,t),\ldots,f_1(x,t),f_0(x,t)\right]^T \tag{10}$$

from all the probability generating functions and obtain the following nonlinear differential equation:

$$\frac{\partial F(x,t)}{\partial t}=A(x,t)\frac{\partial F(x,t)}{\partial x}+B(x,t)\frac{\partial^2 F(x,t)}{\partial x^2}, \tag{11}$$

where the functions A and B are given by

$$A(x,t)=\begin{bmatrix} a(N_2) & 0 & \cdots & 0\\ b(N_2-1) & a(N_2-1) & \cdots & 0\\ c(N_2-2) & b(N_2-2) & \cdots & 0\\ d(N_2-3) & c(N_2-3) & \cdots & 0\\ e(N_2-4) & d(N_2-4) & \cdots & 0\\ \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & \cdots & a(0)\end{bmatrix} \tag{12}$$

$$B(x,t)=\alpha_5(t)x(x-1) \tag{13}$$

with the following components:

$$\begin{aligned}a(i) &= \sum_{k=1}^{2}\beta_k(t)\left(\frac{1}{x^{k-1}}-x\right)\\ &+\alpha_5(t)\left[\left(N^2-i\right)(1-x)-1\right]-\sum_{k=1}^{4}xi\alpha_k(t)\end{aligned} \tag{14}$$

$$b(i)=\alpha_1(t)(i+1)x^2 \qquad c(i)=\alpha_2(t)(i+2)x^3$$
$$d(i)=\alpha_3(t)(i+3)x^4 \qquad e(i)=\alpha_4(t)(i+4)x^5.$$

Generally speaking, it is difficult to find out the solution of the above-mentioned differential equation in matrix form. We briefly summarize the methods for finding the solution in time and frequency domains for the case in which A and B are time-independent. This leads to the following equation:

$$\frac{\partial F(x,t)}{\partial t}=A(x)\frac{\partial F(x,t)}{\partial x}+B(x)\frac{\partial^2 F(x,t)}{\partial x^2}. \tag{15}$$

5.1.1. Time domain investigation

To solve (15) in time domain, we use the method of separating variables. Consequently, the base solution will have the following form $F(x,t) = X(x)T(t)$, where X is a real function and T is a temporal scalar function. Consequently, the parabolic differential equation can be written as follows:

$$LX(x) = A(x)D^2X(x) + B(x)DX(x) = \frac{dT(t)}{T(t)}\frac{X(x)}{T(x)}. \quad (16)$$

Hence, by introducing the eigenvalue λ in the following equation $dT(t)/dt = -\lambda T(t)$, we obtain $T(t) = e^{-\lambda t}$. The general solution will be given by the following formula:

$$F(x,t) = \sum_{\lambda} c_\lambda X_\lambda(x) e^{-\lambda t}, \quad (17)$$

where X_λ is the eigenvector associated with the eigenvalue λ of the Sturm-Liouville differential operator L (i.e., $L = A(x)D^2 + B(x)D$).

5.1.2. Frequency domain investigation

To find out the solution of the forward Kolmogorov equation (7), we use the method described in [18, 37]. Using the Laplace transform for the probability generating function associated with the forward Kolmogorov equation, we reach the following relation for $0 \le i \le N_2$:

$$\begin{aligned}\theta f_i(x,s) - \delta_{N_2} x^{N_2} = {} & \left\{\sum_{k=1}^{2}\frac{\beta_k}{x^{k-1}} + \alpha_5\left(N^2 - i - 1\right)\right\}\frac{\partial f_i(x,s)}{\partial x} \\ & - x\left[\sum_{k=1}^{4} i\alpha_k + \sum_{k=1}^{2}\beta_k\alpha_5\left(N^2 - i\right)\right]\frac{\partial f_i(x,s)}{\partial x} \\ & + \sum_{k=1}^{4}\alpha_k(i+k)x^{k+1}\frac{\partial f_{i+k}(x,s)}{\partial x} \\ & + \alpha_5 x(x-1)\frac{\partial^2 f_i(x,s)}{\partial x^2}.\end{aligned} \quad (18)$$

In order to solve the differential equation, we apply the Laplace transform to the column vector introduced in (10) which leads to:

$$B(x)\frac{\partial^2 F(x,s)}{\partial x^2} + A(x)\frac{\partial F(x,s)}{\partial x} - sF(x,s) = -x^{N_2}E_{N_2+1}, \quad (19)$$

where $A(x)$ is given by (12), and $B(x) = \alpha_5 x(x-1)I_{N_2+1}$. The symbol I_{N_2+1} represents the identity matrix of dimension $N_2 + 1$. If we differentiate the above equation with respect to variable x, and use the notation

$$F_d = F_d(x,s) = \frac{\partial^d F(x,s)}{\partial x^d}, \quad (20)$$

then (19) takes the form

$$\begin{aligned}B(x)F_{d+2} + \left[z(d)B^{(1)}(x) + A(x)\right]F_{d+1} & \\ + \left[w(d)B^{(2)}(x) + z(d)A^{(1)}(x) - sI_{N_2+1}\right]F_d & \\ + w(d)A^{(2)}(x)F_d = -\frac{N_2!x^{N_2-d}}{(N_2-d)!}E_{N_2+1} &\end{aligned} \quad (21)$$

with $z(d) = d$ and $w(d) = d(d-1)/2$, for $d \ge 0$. Considering that the probability generating functions $f_i(x,s)$ are polynomial functions of variable x of degree N_1+N_2-i and using the Taylor expansion of $F(x,s)$, we get the following relations:

$$F(x,s) = F_0 + \sum_{d=1}^{N_1+N_2+2}\frac{(x-x_0)^d}{d!}\cdot\frac{\partial^d F(x,s)}{\partial x^d}, \quad (22)$$

where F_0 represents the initial condition, and the intermediate Taylor factors are given by

$$\frac{\partial^d F(x,s)}{\partial x^d} = \left[\prod_{d=0}^{N_1+N_2} A_d \begin{bmatrix} F_1 \\ F_0 \\ 0 \end{bmatrix}\right]_{N_2+1} + \sum_{d=0}^{N_1+N_2}\prod_{k=d+1}^{N_1+N_2} A_k B_d \quad (23)$$

with the following coefficients:

$$\begin{aligned}B_d &= \begin{bmatrix} -\dfrac{N_2!x^{N_2-d}}{(N_2-d)!}E_{N_2+1} \\ 0 \\ 0 \end{bmatrix}, \\ A_d &= \begin{bmatrix} A_d^1 & A_d^2 & A_d^3 \\ I_{N_2+1} & O_{N_2+1} & O_{N_2+1} \\ O_{N_2+1} & I_{N_2+1} & O_{N_2+1} \end{bmatrix}, \\ A_d^1 &= B^{-1}(x)\left[z(d)B^{(1)}(x) + A(x)\right]A_d^3 = w(d)B^{-1}(x)A^{(2)}(x), \\ A_d^2 &= B^{-1}(x)\left[w(d)B^{(2)}(x) + z(d)A^{(1)}(x) - \theta I_{N_2+1}\right].\end{aligned} \quad (24)$$

The symbols I_{N_2+1} and O_{N_2+1} stand for the identity and zero matrices, respectively, of dimension $N_2 + 1$.

In summary, (22) governs the evolution of packet dissemination on a grid. This equation lies at the heart of on-chip stochastic communication. More precisely, it captures the transition probabilities of the Markov process associated with the SIR interactions, as well as the effects induced by the network topology during the stochastic dissemination process.

5.2. Performance evaluation

The behavior of the stochastic communication scheme can be characterized based on a small number of parameters which are detailed in this section.

5.2.1. Nodes coverage

An interesting metric that can be derived from the proposed analytical framework is the *coverage* metric (i.e., the number of reached nodes as a function of time). To analyze the coverage metric for the mesh network, we get inspiration from systems biology [39]. As such, we construct a set of rate equations that describe the rumor dissemination process on a mesh network. The ordinary differential equation modeling starts from the following set of possible reactions and their

rate coefficients:

$$\begin{aligned}
S+I &\longrightarrow S+S \text{ with rate } \alpha_1 \quad (1,-1,0) \text{ with rate } \alpha_1,\\
S+2I &\longrightarrow S+2S \text{ with rate } \alpha_2 \quad (2,-2,0) \text{ with rate } \alpha_2,\\
S+3I &\longrightarrow S+3S \text{ with rate } \alpha_3 \quad (3,-3,0) \text{ with rate } \alpha_3,\\
S+4I &\longrightarrow S+4S \text{ with rate } \alpha_4 \quad (4,-4,0) \text{ with rate } \alpha_4,\\
S+S &\longrightarrow S+R \text{ with rate } \beta_1 \quad (-1,0,1) \text{ with rate } \beta_1,\\
S+S &\longrightarrow R+R \text{ with rate } \beta_2 \quad (-2,0,2) \text{ with rate } \beta_2,\\
S+R &\longrightarrow R+R \text{ with rate } \alpha_5 \quad (-1,0,1) \text{ with rate } \alpha_5,
\end{aligned} \tag{25}$$

where S denotes the number of spreaders, I designates the number of ignorants, and R represents the number of stiflers. Whenever a reaction takes place, the number of individuals in each population evolves as shown in the right-hand side of (25). For example, if the third reaction takes place with rate α_3, then the expression $(3,-3,0)$ shows that the number of spreader nodes (S) increases by 3 individuals, the number of ignorants (I) decreases by the same amount and the number of stiflers (R) remains unchanged. More precisely, if we start from (S,I,R) configuration and we follow the third reaction, then we end up in the $(S+3, I-3, R)$ state.

The reaction velocity of each equation can be obtained by using the following rationale: given the reaction $mX + nY \rightarrow qZ$ that takes place at rate k, the reaction velocity is $(1/m)(d[X]/dt) = (1/n)(d[Y]/dt) = k[X]^m[Y]^n$. Further, we have the following relations:

$$\begin{aligned}
\frac{[dS/dt]}{1} &= \frac{[dI/dt]}{-1} = \alpha_1[S][I],\\
\frac{[dS/dt]}{1} &= \frac{[dI/dt]}{-2} = \alpha_2[S][I]^2,\\
\frac{[dS/dt]}{1} &= \frac{[dI/dt]}{-3} = \alpha_3[S][I]^3,\\
\frac{[dS/dt]}{1} &= \frac{[dI/dt]}{-4} = \alpha_4[S][I]^4,\\
\frac{[dS/dt]}{-1} &= \frac{[dR/dt]}{1} = \beta_1[S]^2[R],\\
\frac{[dS/dt]}{-2} &= \frac{[dR/dt]}{2} = \beta_2[S]^2[R],\\
\frac{[dS/dt]}{-1} &= \frac{[dR/dt]}{1} = \alpha_5[S][R].
\end{aligned} \tag{26}$$

This is a compact way of defining the velocity of each reaction. For instance, the first relation in (26) states that the small variation in the number of spreaders over a short time interval (i.e., dS/dt) is proportional with $\alpha_1 SI$; this needs to be accounted for since one ignorant becomes a spreader. At the same time, the variation in the number of ignorants over a short time interval (i.e., dI/dt) is proportional to $-\alpha_1 SI$ since the ignorant population losses one member. It is important to note that all the interactions in (26) are *independent* and only *one* such interaction occurs in any given time interval dt. Based on these relations, we can write the following rate equations to describe how the ensemble changes at any point in time:

$$\begin{aligned}
\frac{ds}{dt} &= \sum_{k=1}^{4} k\alpha_k s i^k - \sum_{k=1}^{2} k\beta_k s^2 - \alpha_5 sr,\\
\frac{di}{dt} &= -\sum_{k=1}^{4} k\alpha_k s i^k,\\
\frac{dr}{dt} &= \sum_{k=1}^{2} k\beta_k s^2 + \alpha_5 sr.
\end{aligned} \tag{27}$$

The variations in number of spreaders, ignorants, and stiflers over time are obtained by taking into account the *superposition* of all the above-mentioned individual effects. Thus, if the third reaction in (25) takes place, for instance, then the number of spreaders increases by three units (therefore, $k = 3$ in (27) and the first term becomes $3\alpha_3 SI^3$) since three ignorants are turned into spreaders. At the same time, the same quantity needs to be subtracted from the number of ignorants variation in (27) (i.e., $-3\alpha_3 SI^3$). We note that the semantics of these relations is similar to that used in process algebra of concurrent communicating systems [40].

Considering, for instance, (27) and the following initial configuration on a 10×10 grid: one spreader, one stifler, $N^2 - 2$ ignorants (i.e., 98 ignorants), and setting the parameters $\beta_1 = \beta_2 = \alpha_5 = 0$ (i.e., assuming neither damaged links, nor nodes), we obtain the time evolution in Figure 7, as a function of P. The conclusion from this plot is twofold: on one hand, there exists a region of exponential growth in the number of spreader nodes; this is also later shown experimentally by simulation in Section 6.1.2. On the other hand, in absence of faults or upsets, the number of spreader nodes reaches a maximum point after which it does not change significantly with time.

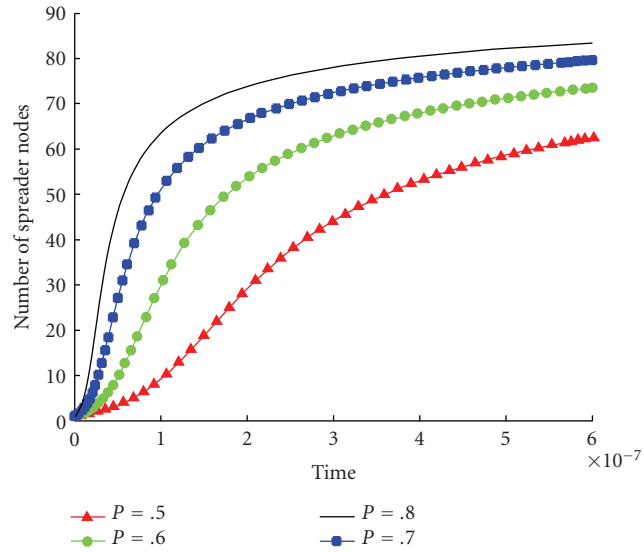


Figure 7: Time evolution of spreader nodes with and without faults.

In order to investigate the impact of faults or upsets on the coverage metric, we increase the values of the fault parameters (i.e., β_1, β_2, α_5) and use the analytical approximation, in (27). We compare these results with the case in which there are neither damaged links, nor nodes. As we can see in Figure 8 ($P = .7$), the number of nodes that become aware of the packet dissemination reaches a maximum point in both cases. However, in presence of faults, the number of reached nodes is significantly smaller. This can be explained with the spreader-spreader and spreader-stifler interactions introduced in Section 5.1 (i.e., (3) and (4)).

5.2.2. Broadcast rounds

A *broadcast round* is the time interval in which a tile has to finish sending all its messages to the next hops; this usually takes several clock cycles. The optimal duration of a round (T_R) can be determined using (28)

$$T_R = \frac{S \cdot N_{\text{packets/round}}}{f}, \tag{28}$$

where f is the maximum frequency of any link, $N_{\text{packets/round}}$ is the average number of packets a link sends during one round (this is application dependent), and S is the average packet size.

As shown in Section 6, the fast dissemination of rumors makes it possible to achieve very low latencies. While XY routing generates traffic along a single path, stochastic communication spreads the traffic uniformly across all links in the network, thereby reducing the chances that packets are delayed due to local congestion. Under the proposed protocol, congestion occurs only when the entire network becomes saturated. This is especially important in multimedia applications, for instance, where a sustainable constant bit-rate is highly desirable. Furthermore, the fact that we do not store or compute the shortest paths (as in dynamic routing) makes this algorithm computationally lightweight, simpler and easier to customize for every application and interconnection network. We also note that since the stochastic communication algorithm uniformly spreads the traffic across the entire network, the proposed algorithm does not suffer from deadlock. Indeed, the packets can reach their destinations even when using finite buffers if the injection rates are sufficiently small.

5.2.3. Energy metrics

To estimate the energy consumption of this algorithm, we use (29)

$$\begin{aligned} E_{\text{total}} &= E_{\text{computation}} + E_{\text{communication}} \\ &= E_{\text{computation}} + N_{\text{packets}} \cdot S \cdot E_{\text{bit}}, \end{aligned} \tag{29}$$

where N_{packets} is the total number of messages generated in the network, S is the average size of one packet (in *bits*), and E_{bit} is the energy consumed per bit per link. The parameter N_{packets} can be estimated by simulation, S is application-dependent, and E_{bit} can be derived from the technology library.

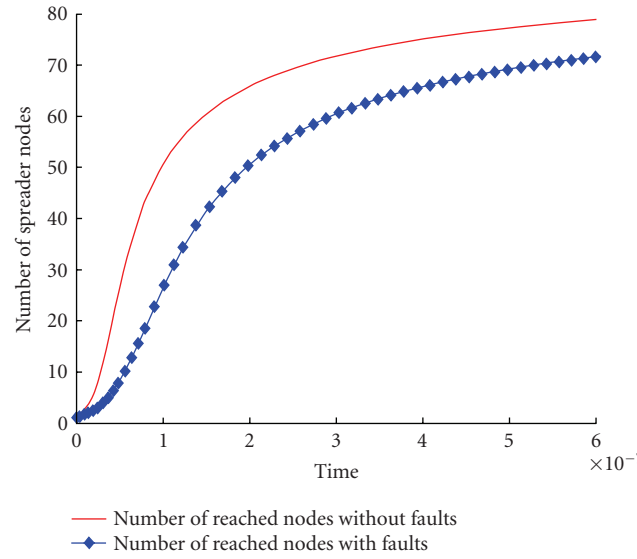


Figure 8: Time evolution of spreader nodes with and without faults.

As shown in (29), the total energy consumed is also influenced by the computational cores ($E_{\text{computation}}$). Since our focus is on the performance of the *communication scheme*, it is not necessary to estimate precisely the energy required by computation. However, this can be added, to our estimations in Section 6 in order to get the combined energy values.

In summary, the parameters relevant to our analysis are:

(i) *nodes coverage*, which is a measure of message dissemination in the network (similarly to molecules diffusion in gases),
(ii) *number of broadcast rounds* needed, which is a direct measure of the inter-IP communication latency,
(iii) *total number of packets* sent in the network, which indicates the bandwidth required by the algorithm; it can be controlled by varying the message TTL value,
(iv) *fault-tolerance*, which evaluates the algorithm resilience to abnormal conditions in the network,
(v) *energy consumption*, which is computed with (29) (as detailed in Section 6).

6. PRACTICAL CONSIDERATIONS AND EXPERIMENTAL RESULTS

Stochastic communication can be beneficial to many application areas ranging from parallel SAT solvers and multimedia applications to periodic data acquisition from noncritical sensors. Starting from a computational task graph, the fault-tolerant scheme distributes the independent tasks and operations to different nodes (i.e., information dissemination towards slave nodes) similar to a network of sensors. Later on, the results are collected at the master node through multiple paths. This idea is better emphasized by our first example, the two-dimensional Fast Fourier Transform (FFT) (see Figure 9). The inherent parallelism of the FFT algorithm is exploited by computing the partial results at the slave nodes.

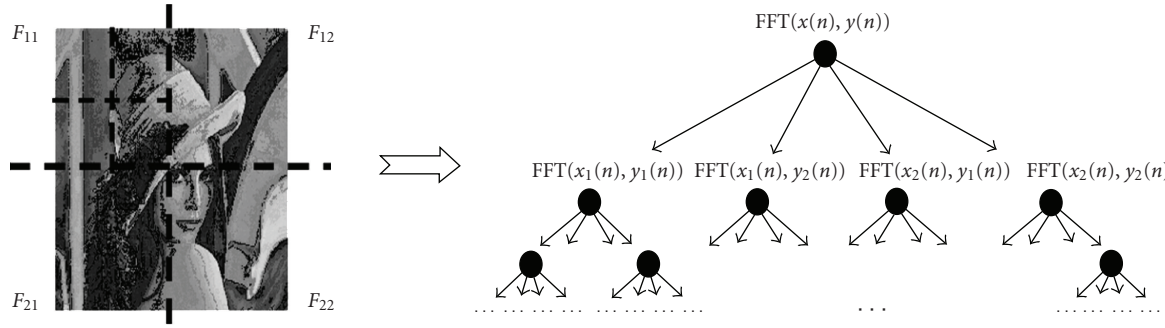


Figure 9: Parallel scheme for computing FFT2.

The stochastic communication scheme ensures that the master node ultimately receives these results and is able to return the desired FFT value even for high data error rates.

Since realistic data about failure patterns in regular SoCs are not easily available, we explore the entire parameter space of our fault model (i.e., $P_{\text{upset}} \in [0, 1]$). Another important parameter we vary is P, the probability that a packet is forwarded over a link (see Section 4.1). Stochastic communication allows us to tune the trade-off between energy and performance by varying the probability of transmission (P) between 0 and 1.

6.1. Case study: the 2D Fast Fourier Transform

Given the structure of the Discrete Fourier Transform of N samples, a recursive divide-and-conquer algorithm can be used to compute the FFT of N samples (see Figure 9). This reduces the number of operations to $O(N \log_2 N)$. Because of its widespread use in engineering (especially in image and multimedia processing), we have focused on the *two-dimensional FFT* algorithm (FFT2) applied to both XY dimensions. Every node in the tree from Figure 9 represents a parallel process. The leaves compute the FFT on a small number of samples and send the intermediate results back to the root which assembles the final result.

We map this application onto a 4×4 NoC running the stochastic communication algorithm in Section 4. To increase the tolerance to permanent tile failures, the IPs can be duplicated (as for the Master-Slave computations). Our models simulate grids of 16–25 tiles, which is relevant to nowadays designs, but the gossip algorithms scale extremely well so stochastic communication can be applied to much larger designs.

For this case study we have considered a restricted version of the fault model from Section 3, which includes only the most cited failure modes in the current literature [33] permanent failures of tiles and links (with probabilities P_{tiles} and P_{links}) and *data upsets* (with probability P_{upset}). The random failures are distributed uniformly in time during the length of the simulation (see Figure 10 where we consider a 4×4 NoC).

In our framework, the *packet* (not the flit) represents the basic unit of information and as such, packet reordering is not a real issue. More precisely, in our experiments, while packets may arrive at destination in an out-of-order fashion, they can be easily reordered based on the information contained in their header. As such, the experiments reported in this paper do not suffer from the out-of-order arrival problem.

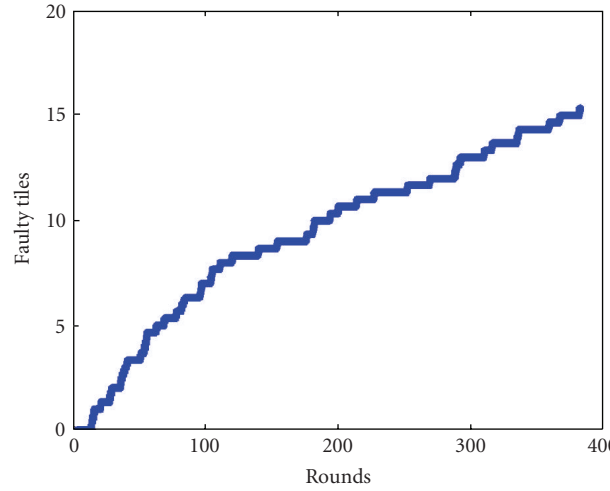


Figure 10: Distribution of failures during simulation.

We evaluate the *latency*, *the energy dissipation*, and the *fault-tolerance* of stochastic communication.For consistency reasons, all the results presented in this section are averages obtained after several simulations. We compare here four versions of the stochastic communication, obtained for different values of the parameter P:

(i) the network *flooding*, which is a deterministic algorithm where the tiles send the messages to *all* their neighbors all the time ($P = 1$);
(ii) three versions of the stochastic communication algorithm in Section 4, which uses different probabilities to transmit the message across the links; namely, we use $P = .75$, $P = .5$, and $P = .25$.

The reason for comparing our protocol against the flooding algorithm is because the latter is the simplest possible example of a deterministic broadcast protocol. The flooding

Table 1: Stochastic communication in an NoC with 6.25% defective tiles and 8.33% defective links.

	Initial broadcast (no. of rounds)				FFT2 finished (no. of rounds)				Total number of packets				Average energy ($\times 10^{-8}$ J/bit)			
	Flood	Stoch.comm.(P)			Flood	Stoch.comm.(P)			Flood	Stoch.comm.(P)			Flood	Stoch.comm.(P)		
	($P = 1$)	.75	.5	.25	($P = 1$)	.75	.5	.25	($P = 1$)	.75	.5	.25	($P = 1$)	.75	.5	.25
$P_{\text{upset}} = 0$	5	7	—	16	5	6	9	16	471	481	619	558	2.261	1.924	1.651	0.836
$P_{\text{upset}} = 0.2$	5	9	12	30	5	7	11	16	444	559	811	508	2.129	1.917	1.768	0.761
$P_{\text{upset}} = 0.4$	9	11	—	—	5	13	16	20	387	1417	1289	486	1.858	2.616	1.934	0.583
$P_{\text{upset}} = 0.6$	19	11	17	38	10	16	18	39	1226	1997	1172	1700	2.942	2.995	1.562	1.046
$P_{\text{upset}} = 0.8$	22	18	46	85	23	25	34	84	4158	2734	2080	3562	4.339	2.624	1.468	1.018

Table 2: Impact of failures on latency of stochastic communication with $P = .5$ (number of rounds to finish: initial broadcast/FFT2).

	$P_{\text{tiles}} = 0$				$P_{\text{tiles}} = .125$				$P_{\text{tiles}} = .25$			
	$P_{\text{links}} =$				$P_{\text{links}} =$				$P_{\text{links}} =$			
	0	.083	.166	.25	0	.083	.166	.25	0	.083	.166	.25
$P_{\text{upset}} = 0$	8/6	9/9	15/8	—/9	8/10	7/7	13/13	—/11	6/8	—/—	—/14	—/—
$P_{\text{upset}} = 0.2$	11/10	10/8	—/12	—/12	9/12	—/10	23/19	—/19	8/—	—/20	—/17	—/—
$P_{\text{upset}} = 0.4$	9/11	16/14	—/14	—/25	15/10	15/16	—/16	—/—	—/11	—/37	—/9	—/22
$P_{\text{upset}} = 0.6$	15/18	21/13	—/22	—/25	43/15	—/36	21/12	—/—	24/16	—/16	—/17	—/—
$P_{\text{upset}} = 0.8$	33/27	46/33	83/37	—/69	24/53	—/49	35/46	—/—	—/—	42/—	—/39	—/35

algorithm is also optimal with respect to latency; that is, the number of intermediate hops between source and destination is always equal to the Manhattan distance between the two tiles. We show that our protocol has a latency close to this optimum. The flooding is, however, extremely inefficient to implement with respect to the bandwidth and energy consumed. The stochastic communication enables various trade-offs between energy and performance figures by varying the values of the transmission probability P.

A summary of our results is given in Tables 1 and 2. For example, the second row in Table 1 shows that, when the probability of an unsuccessful transmission is $P_{\text{upset}} = .2$, depending on the algorithm used, the broadcast of the first message takes between 5–30 rounds to reach all the tiles, FFT2 is computed after 5–16 rounds, there are 444–811 packets transmitted in the network, with an average energy consumption of 7.61–21.29 nJ per message bit.[4] The second row of Table 2 shows the number of rounds needed to finish the initial broadcast/the computation of FFT2, for different levels of tile and link failures, when $P_{\text{upset}} = .2$.

To give an idea about the average communication energy consumption of the stochastic communication algorithm, we compare the energy consumption of the deterministic XY routing, against the energy of the proposed approach when there are no defective tiles or links in the network. We consider a random mapping which is not optimal for either stochastic or deterministic cases. Moreover, we assume that for a nonzero probability of data upsets (i.e., $P_{\text{upset}} > 0$) in the deterministic case, a number of ($P_{\text{upset}} \times N_{\text{packets}}$) packets need to be retransmitted during the next communication round.[5] The packet retransmission is done in n communication rounds, where n is determined from the following condition $(P_{\text{upset}})^{\mathbf{n}} \leq \varepsilon$ and ε depends on the allowed tolerance to missing FFT2 values. In absence of data upsets (i.e., $P_{\text{upset}} = 0$), the deterministic XY routing scheme consumes 7.2 nJ energy which is smaller than the energy consumption needed in flooding (i.e., 17.22 nJ) and stochastic communication (i.e., 18.34 nJ for $P = .75$, 11.94 nJ for $P = .5$ and 7.83 nJ for $P = .25$). In presence of data upsets, the deterministic routing scheme requires several retransmissions and the energy consumption is given by $E_0/(1 - P_{\text{upset}})$. For instance, for a $P_{\text{upset}} = .4$, the communication energy consumption of the deterministic case is 12 nJ, while the energy consumption of the stochastic communication varies from 22.29 nJ for $P = .75$ to 16.64 nJ for $P = .5$ and 10.48 nJ for $P = .25$. Interestingly enough, for higher error rates (e.g., $P_{\text{upset}} = .8$), the communication energy consumption in the deterministic case is 36 nJ, which is higher than the stochastic communication energy consumption which varies from 33.55 nJ for $P = .75$ to 19.85 nJ for $P = .5$ and 7.72 nJ for $P = .25$. Thus, the stochastic communication algorithm proves to work well and consume less energy at higher data upset rates, while the deterministic case seems to imply a nonlinear increases with P_{upset}.

[4] The total energy dissipation can be obtained by multiplying these figures with the packet size S. In our experiments, we have used $S = 364$ bits.

[5] A communication round refers to the number of cycles needed until the first FFT2 value is computed.

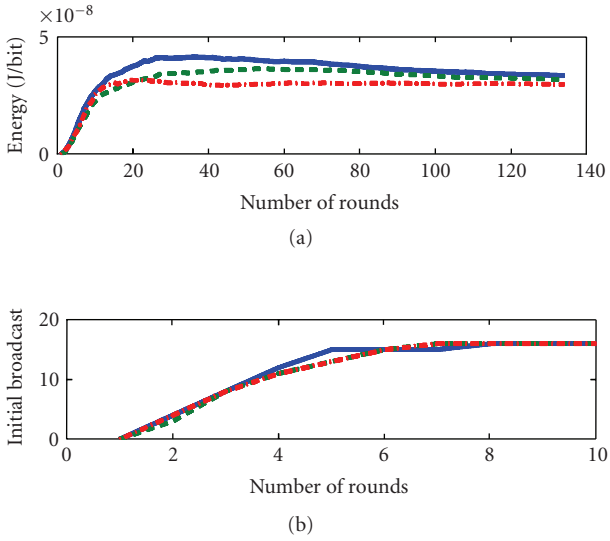


FIGURE 11: Three different runs of the algorithm.

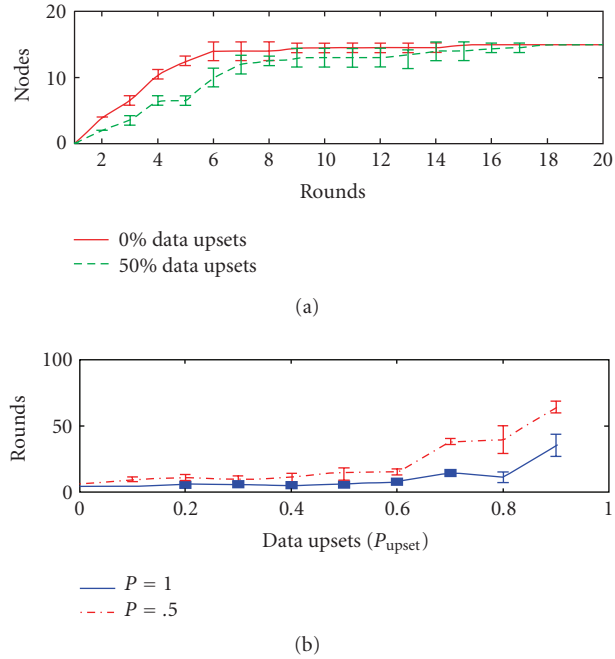


FIGURE 12: Coverage and latency of the stochastic communication.

6.1.1. *Repetability of results*

Our experiments show that, for several runs of the protocol, the parameter estimations are consistent and do not fluctuate much around the mean value. To support this claim, we present in Figure 11 a comparison for the evolution of the energy dissipation across the entire simulation, and the number of rounds needed to complete the application.

6.1.2. *Latency*

From the communication point of view, the FFT2 application has two phases.

(i) First, the initial message, containing the raw image samples, has to reach all of the leaf nodes,
(ii) Second, the computed results have to come back to the root node, which will assemble the final result.

The evolution of the first phase, namely the spread of the initial broadcast, is shown in the upper part of Figure 12. As we can see, the spread is relatively slow in the beginning, but soon reaches a stage of explosive growth and so the entire network becomes aware of the message after a small number of rounds. This behavior is reasonably close to the one predicted by the theoretical model in Section 5.2.1 (see Figure 8). In both graphs, the data upsets influence the number of nodes reached. The presence of transmission failures slows down the spreading (dashed line in Figure 12 and the blue line in Figure 8), but the message still reaches most of the network tiles.

The end of the second phase, namely the time when all the partial results reach the root node, marks the completion of the application and is illustrated in the lower part of Figure 12. We note that stochastic communication with probability of transmission $P = .5$ (when up to 50% of the sent packets are corrupted because of data upsets) has a latency very close to the optimal one which corresponds to the flooding algorithm. The standard deviation (shown by the error bars in Figure 12) is small which proves that the stochastic communication has indeed a *stable behavior* and the results can be reproduced across several runs of the algorithm.

Last but not least, these numbers depend on the mapping of IPs to tiles. In some cases, we notice that the replies may come back *before* the full broadcast of the original message reaches all the tiles of the network. This indicates that the application mapping phase has to take into account the communication performance in order to obtain an efficient design [41].

6.1.3. *Energy dissipation*

We have estimated the energy dissipation of our algorithm using (29). We do not include the energy consumed during computation and so, all the results presented here reflect solely the performance of the NoC stochastic communication.

In the top part of Figure 13 we compare the energy consumption of flooding ($P = 1$) and stochastic communication (with $P = .5$) algorithms. While stochastic communication has a latency close to optimal (as explained above), its energy dissipation is about half of the one of the flooding algorithm. This is because the energy is proportional to the total number of messages generated in the network (see Section 5.2.3), which is controlled by the probability of transmission P. This observation indicates that, by modifying parameter P, the designer can tune the trade-off between the performance and the energy dissipation of the communication architecture.

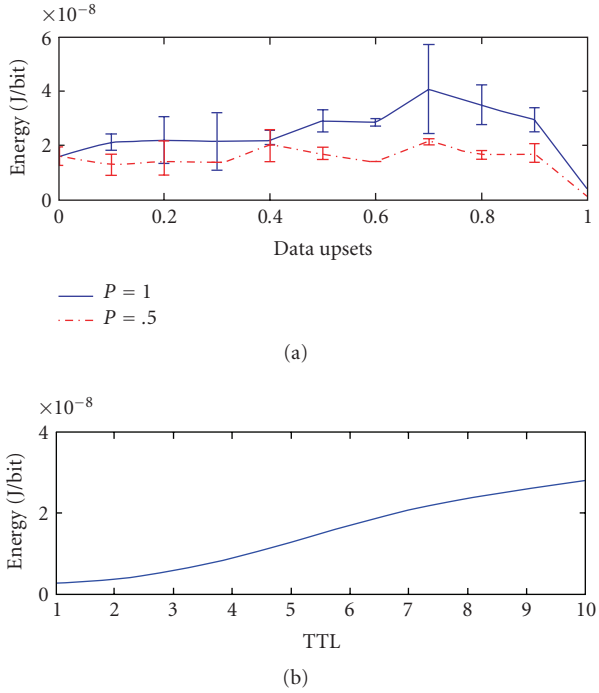


Figure 13: Energy dissipation of the stochastic communication.

The energy dissipation drops to 0 in the extreme case when $P_{\text{upset}} \approx 1$ because all the packets are corrupted and therefore, they are not retransmitted anymore. We also note that the energy dissipation of the stochastic communication also has a smaller variance across several runs of the protocol.

Further energy savings can be achieved by stopping the spreading of messages after a fixed number of rounds. This can be done by assigning messages a finite TTL. For a small TTL, neither the broadcast of the initial message nor the application itself can complete. However, after a certain threshold, the latency does not improve if the TTL is increased. The lower half of Figure 13 shows that the energy consumption increases almost linearly with the TTL. This suggests that the TTL can be set to the smallest value which guarantees tasks completion, in order to keep the energy consumption at minimum.

6.1.4. Fault-tolerance

Different types of failures have different effects on the performance of stochastic communication (see Table 2). The levels of defective links and tiles do not seem to have a big impact on latency. However, if a significant number of permanent tile and link failures occurs during the early stages of the algorithm (fortunately, this is actually not likely with modern manufacturing technologies, as explained in Section 3), the applications would fail because too many important modules are not working or entire regions of the NoC may become isolated.

The computation of FFT2 succeeds in many cases, for instance even with 12.5% faulty tiles and 16.67% faulty links,

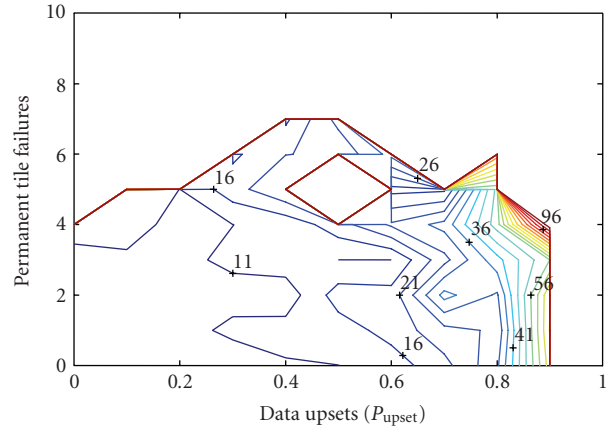


Figure 14: Impact of permanent failures and data upsets on latency.

FFT2 succeeds in almost all runs (see Table 2). The initial broadcast is especially affected by the number of defective links, which can disconnect a region of the chip from the network and prevent those tiles from sending/receiving messages. However, due to resource duplication, even in these cases the computation of FFT2 may succeed, if there are enough resources left that can communicate with each other.

On the other hand, data upsets seem to have little influence on the probability that the applications will be able to terminate; however, upsets do have a serious impact on latency, especially if $P_{\text{upset}} > .5$. Figure 14 illustrates how the defective tiles and data upsets influence the latency of stochastic communication, again on a 4×4 NoC. As shown, FFT2 cannot finish with more than 8 permanently failed tiles. Generally speaking, data upsets will not cause the communication to fail completely. As shown in Figure 14, for less than 4 stopped tiles (this represents 25% faulty tiles), the algorithm eventually terminates with levels of data upsets as high as 90%, even if it requires close to 100 rounds to do so.

6.2. More complex applications

Another application of stochastic communication that was extensively studied in [26] is the MP3 encoder implemented in a "voltage/frequency island" architecture [35]. For this application, some regions of the NoC run at different voltages and even at different frequencies without a significant performance penalty. Indeed, certain parts of the chip, like memories or control logic, do not require as high a voltage as the processor; by placing them on different voltage islands, the total power consumed by the design can be significantly reduced.

As shown in [26], the level of buffer overflows does not seem to have a big impact on latency. Moreover, the synchronization errors do not prevent the application from terminating; however, they do cause a great variability in the latency. Data upsets also seem to have little influence on the chances to finish encoding, but have a significant impact on the latency, especially if $P_{\text{upset}} > .7$.

7. LIMITATIONS AND EXTENSIONS

The on-chip stochastic communication approach is inspired by epidemic models where communication is seen as a consequence of interactions between individuals from various subpopulations (i.e., spreaders, ignorants, and stiflers). These interactions are defined in Section 5.1, in accordance to the restrictions imposed by the topology. Also, the mathematical model captures the situation in which due to buffer overflow, link failures or nodes malfunctions, the dissemination process can be stopped. Further, the mathematical model in Section 5.2.1 retrieves the evolution of coverage of spreader nodes which is also illustrated in the experiments (Section 6.1.1).

A similar behavior can be obtained if the Euler-Maruyama stochastic method is used for evaluating the evolution of the number of spreader nodes over time [42]. Although it seems attractive for its simplicity, this method may not serve as a prediction tool for latency and energy values because it does not capture the spatial relationship between the network nodes. Future analysis should take into account the variation of stochastic communication in space; this can provide insight on the covered surface by the packet dissemination process.

We plan to extend the present work in several directions. One possibility is to address the issue of task mapping in the context of stochastic communication. The main challenge here comes from the difficulty involved in precise TTL evaluation. The TTL value is not only dependent on the distance between source and destination, but also on the network diameter and node buffering resources. This remains an open problem.

Other possible extensions include investigating the tradeoff between TTL and packet transmission probability for different network topologies. Another important extension is to accommodate the proposed approach with a wormhole switching technique and consider the problem of out-of-order packet delivery.

Finally, the problem of CRC code design should be also considered in the NoC context. In our framework, a packet is treated as the basic unit of information. The packet header does not include the routing path information since it is retransmitted many times over different paths. Instead, we assume that each packet is protected by a CRC code. While there are several criteria for characterizing the optimality of the CRC codes, two properties are of particular interest, namely, the Hamming distance and the burst error detection capability. A CRC code of degree d can detect a burst error whose length is smaller than its degree. For instance, Koopman and Chakravarty report in [43] that the USB-5 (i.e., Universal Serial Bus protocol) is optimum for data words of 11-bit long and BERs of 1e-6. Nevertheless, the literature (e.g., [9, 44]) reports that the routing tag consists of at least 16 bits and the required number of bits increases with network size. As such, the use of CRC codes may bring some benefits in reducing the number of header bits. However, the detailed analysis that would help us determine the optimal CRC codes is beyond the purpose of the present work and it is left for future work.

8. CONCLUSIONS

This paper proposed stochastic communication as a novel SoC communication paradigm. This approach takes advantage of the large bandwidth available on chip in order to provide the needed system-level tolerance to synchronization errors, data upsets, and buffer overflows. A theoretical framework which accounts for network topology was proposed to analyze the packet dissemination and interactions between nodes.

Experimental results show that this approach is very scalable, robust, and has a low latency, while keeping the energy consumption at reasonable levels. At the same time, this method simplifies the design process by offering a low-cost and easy to customize solution for on-chip communication. Last but not least, stochastic communication enables various hybrid architectures that may be used to enable on-chip diversity.

ACKNOWLEDGMENTS

The authors thank Nick Zamora and Umit Ogras of System Level Design group at CMU for their insightful comments on the stochastic communication approach. Also, the authors would like to thank Sam Kerner for help with the experimental part in the initial stages of this project. This Research was supported by SRC 2004-HJ-1189 and Marco GSRC.

REFERENCES

[1] C. Constantinescu, "Impact of deep submicron technology on dependability of VLSI circuits," in *Proceedings of the International Conference on Dependable Systems and Networks (DNS '02)*, pp. 205–209, Washington, DC, USA, June 2002.

[2] W. Maly, "IC design in high-cost nanometer-technologies era," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 9–14, Las Vegas, Nev, USA, June 2001.

[3] Semiconductor Association, "The International Technology Roadmap for Semiconductors (ITRS)," 2001.

[4] D. Bertozzi, L. Benini, and G. De Micheli, "Low power error resilient encoding for on-chip data buses," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '02)*, pp. 102–109, Paris, France, March 2002.

[5] T. Dumitraş, S. Kerner, and R. Marculescu, "Towards on-chip fault-tolerant communication," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '03)*, pp. 225–232, Kitakyushu, Japan, January 2003.

[6] T. Valtonen, T. Nurmi, J. Isoaho, and H. Tenhunen, "Interconnection of autonomous error-tolerant cells," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 4, pp. 473–476, Phoenix, Ariz, USA, May 2002.

[7] H. G. Lee, U. Y. Ogras, R. Marculescu, and N. Chang, "Design space exploration and prototyping for on-chip multimedia applications," in *Proceedings of the ACM/IEEE 43rd Design Automation Conference (DAC '06)*, pp. 137–142, San Francisco, Calif, USA, July 2006.

[8] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 684–689, Las Vegas, Nev, USA, June 2001.

[9] A. Jantsch and H. Tenhunen, *Networks on Chip*, Kluwer Academic Publishers, Norwell, Mass, USA, 2003.

[10] L. Gasieniec and A. Pelc, "Adaptive broadcasting with faulty nodes," *Parallel Computing*, vol. 22, no. 6, pp. 903–912, 1996.

[11] T. Leighton, B. Maggs, and R. Sitaraman, "On the fault tolerance of some popular bounded-degree networks," in *Proceedings of the 33rd IEEE Annual Symposium on Foundations of Computer Science*, pp. 542–552, Pittsburgh, Pa, USA, October 1992.

[12] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *Computer*, vol. 26, no. 2, pp. 62–76, 1993.

[13] G. De Micheli, "Robust system design with uncertain information," in *The Asia and South Pacific Design Automation Conference (ASP-DAC '03) Keynote Speech*, Kitakyushu, Japan, January 2003.

[14] T. Karnik, S. Borkar, and V. De, "Sub-90nm technologies—challenges and opportunities for CAD," in *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD '02)*, pp. 203–206, San Jose, Calif, USA, November 2002.

[15] A. Demers, D. Greene, C. Hauser, et al., "Epidemic algorithms for replicated database maintenance," in *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, Vancouver, British Columbia, Canada, August 1987.

[16] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '99)*, pp. 263–270, Seattle, Wash, USA, August 1999.

[17] N. Bailey, *The Mathematical Theory of Infectious Diseases*, Charles Griffin and Company, London, UK, 2nd edition, 1975.

[18] D. J. Daley and J. Gani, *Epidemics Modelling: An Introduction*, Cambridge University Press, Cambridge, UK, 1999.

[19] D. J. Daley and D. G. Kendall, "Stochastic rumours," *IMA Journal of Applied Mathematics*, vol. 1, no. 1, pp. 42–55, 1965.

[20] C. E. M. Pearce, "The exact solution of the general stochastic rumour," *Mathematical and Computer Modelling*, vol. 31, no. 10, pp. 289–298, 2000.

[21] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal multicast," *ACM Transactions on Computer Systems*, vol. 17, no. 2, pp. 41–88, 1999.

[22] B. Kantor and P. Lapsley, "Network News Transfer Protocol," RFC 977, February 1986. http://www.w3.org/Protocols/rfc977/rfc977.

[23] K. Lidl, J. Osborne, and J. Malcolm, "Drinking from the firehose: multicast USENET news," in *Proceedings of the USENIX Winter Technical Conference*, San Francisco, Calif, USA, January 1994.

[24] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 784–803, 1997.

[25] XTP Forum, "Xpress Transfer Protocol Specification Revision 4.0," March 1995.

[26] T. Dumitraș and R. Marculescu, "On-chip stochastic communication," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '03)*, pp. 790–795, Munich, Germany, March 2003.

[27] S. Manolache, P. Eles, and Z. Peng, "Fault and energy-aware communication mapping with guaranteed latency for applications implemented on NoC," in *Proceedings of the 42nd Design Automation Conference (DAC '05)*, pp. 266–269, ACM Press, Anaheim, Calif, USA, June 2005.

[28] P. Bogdan and R. Marculescu, "A theoretical framework for on-chip stochastic communication analysis," in *Proceedings of the 1st International Conference on Nano-Networks (NANONETS '06)*, Lausanne, Switzerland, September 2006.

[29] C. Constantinescu, "Dependability analysis of a fault-tolerant processor," in *Proceedings of Pacific Rim International Symposium on Dependable Computing*, pp. 63–67, Seoul, South Korea, December 2001.

[30] R. Horst, D. Jewett, and D. Lenoski, "The risk of data corruption in microprocessor-based systems," in *Proceedings of the 23rd International Symposium on Fault-Tolerant Computing (FTCS-23 '93)*, pp. 576–585, Toulouse, France, June 1993.

[31] T.-T. Y. Lin and D. P. Siewiorek, "Error log analysis: statistical modeling and heuristic trend analysis," *IEEE Transactions on Reliability*, vol. 39, no. 4, pp. 419–432, 1990.

[32] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *IEEE Micro*, vol. 23, no. 4, pp. 14–19, 2003.

[33] V. Hadzilacos and S. Toueg, "A modular approach to fault-tolerant broadcasts and related problems," Tech. Rep. TR94-1425, Department of Computer Science, Cornell University, Ithaca, NY, USA, May 1994.

[34] D. M. Chapiro, *Globally-asynchronous locally-synchronous systems*, Ph.D. thesis, Stanford University, Stanford, Calif, USA, 1984.

[35] D. E. Lackey, P. S. Zuchowski, T. R. Bednar, D. W. Stout, S. W. Gould, and J. M. Cohn, "Managing power and performance for system-on-chip designs using voltage islands," in *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD '02)*, pp. 195–202, San Jose, Calif, USA, November 2002.

[36] T. Chelcea and S. M. Nowick, "Robust interfaces for mixed-timing systems with application to latency-insensitive protocols," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 21–26, Las Vegas, Nev, USA, June 2001.

[37] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Stochastic Models for Social Processes," in *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing, Vancouver*, John Wiley & Sons, British Columbia, Canada, August 1987.

[38] D. J. Watts, *Small Worlds, the Dynamics of Networks between Order and Randomness*, Princeton University Press, Princeton, NJ, USA, 1999.

[39] D. J. Wilkinson, *Stochastic Modelling for Systems Biology*, Chapman & Hall Press, London, UK, 2006.

[40] R. Milner, *Communication and Concurrency*, Prentice-Hall, Upper Saddle River, NJ, USA, 1989.

[41] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 551–562, 2005.

[42] P. E. Protter, *Stochastic Integration and Differential Equations*, Springer, Berlin, Germany, 2004.

[43] P. Koopman and T. Chakravarty, "Cyclic Redundancy Code (CRC) polynomial selection for embedded networks," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN '04)*, pp. 145–154, Florence, Italy, June-July 2004.

[44] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection Networks: An Engineering Approach*, Morgan Kaufmann Publishers, San Francisco, Calif, USA, 2002.