Journal of Electrical and Computer Engineering

Networks-on-Chip: Architectures, Design Methodologies, and Case Studies

Guest Editors: Sao-Jie Chen, An-Yeu Andy Wu, and Jiang Xu

Networks-on-Chip: Architectures, Design Methodologies, and Case Studies

Journal of Electrical and Computer Engineering

Networks-on-Chip: Architectures, Design Methodologies, and Case Studies

Guest Editors: Sao-Jie Chen, An-Yeu Andy Wu, and Jiang Xu

Editorial Board

The editorial board of the journal is organized into sections that correspond to the subject areas covered by the journal.

Circuits and Systems

Communications

Signal Processing

Contents

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 634930, 1 page
doi:10.1155/2012/634930

Editorial

Networks-on-Chip: Architectures, Design Methodologies, and Case Studies

Sao-Jie Chen,[1] An-Yeu Andy Wu,[2] and Jiang Xu[3]

[1] *Department of Electrical Engineering and Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan*
[2] *Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan*
[3] *Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Kowloon, Hong Kong*

Correspondence should be addressed to Sao-Jie Chen, csj@cc.ee.ntu.edu.tw

Received 26 December 2011; Accepted 26 December 2011

As the density of VLSI design increases, more processors or cores can be placed on a single chip. Therefore, the design of a Multi-Processor System-on-Chip (MP-SoC) architecture, which demands high throughput, low latency, and reliable global communication services, cannot be done by just using current bus-based on-chip communication infrastructures. Networks-on-Chip (NoC) has been proposed in recent years as a promising solution of on-chip interconnection network to provide better scalability, performance, and modularity for current and future MP-SoC architectures.

The paper entitled "Networks on chips: structure and design methodologies" introduces several NoC architectures and discusses the design issues of communication performance, power consumption, signal integrity, and system scalability in an NoC. Then, a novel Bidirectional NoC (BiNoC) architecture with a dynamically self-reconfigurable bidirectional channel is presented, which can break the performance bottleneck caused by bandwidth restriction in conventional NoCs.

Since buffers in on-chip networks constitute a significant proportion of the power consumption and the area of interconnects, reducing the buffer size is an important problem. The paper entitled "A buffer sizing algorithm for network on chips with multiple voltage-frequency islands" describes a two-phase algorithm to size the switch buffers in NoC in considering the support of multiple-frequency islands.

The paper entitled "Self-calibrated energy-efficient and reliable channels for on-chip interconnection networks" depicts the design of an energy-efficient and reliable channel for on-chip interconnection networks (OCINs) using a self-calibrated voltage scaling technique with self-corrected green (SCG) coding scheme.

Among the NoC components, links that connect the NoC routers are the most power-hungry components. The paper entitled "Intelligent on/off dynamic link management for on-chip networks" presents an intelligent dynamic power management policy for NoCs with improved predictive abilities based on supervised online learning of the system status, where links are turned off and on via the use of a small and scalable neural network.

Monitoring and diagnostic systems are required in modern NoC implementations to assure high performance and reliability. In the paper entitled "Status data and communication aspects in dynamically clustered network-on-chip monitoring," the design of a dynamically clustered NoC monitoring structure for traffic and fault monitoring is illustrated.

Since biological organisms have better adaptability than computer systems in dealing with environmental changes or noise. A case study on the design of an evolvable neuromolecular hardware motivated from some biological evidence, which integrates inter- and intra-neuronal information processing, is depicted in the paper entitled "A hardware design of neuromolecular network with enhanced resolvability: a bio-inspired approach."

Sao-Jie Chen
An-Yeu Andy Wu
Jiang Xu

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 107821, 12 pages
doi:10.1155/2012/107821

Research Article

Intelligent On/Off Dynamic Link Management for On-Chip Networks

Andreas G. Savva,[1] Theocharis Theocharides,[1] and Vassos Soteriou[2]

[1] *Department of Electrical and Computer Engineering, University of Cyprus, 1678 Nicosia, Cyprus*
[2] *Department of Electrical Engineering and Information Technology, Cyprus University of Technology, 3036 Limassol, Cyprus*

Correspondence should be addressed to Andreas G. Savva, eep7sa4@ucy.ac.cy

Received 15 August 2011; Accepted 3 December 2011

Academic Editor: Sao-Jie Chen

Networks-on-chips (NoCs) provide scalable on-chip communication and are expected to be the dominant interconnection architectures in multicore and manycore systems. Power consumption, however, is a major limitation in NoCs today, and researchers have been constantly working on reducing both dynamic and static power. Among the NoC components, links that connect the NoC routers are the most power-hungry components. Several attempts have been made to reduce the link power consumption at both the circuit level and the system level. Most past research efforts have proposed selective on/off link state switching based on system-level information based on link utilization levels. Most of these proposed algorithms focus on a pessimistic and simple static threshold mechanism which determines whether or not a link should be turned on/off. This paper presents an intelligent dynamic power management policy for NoCs with improved predictive abilities based on supervised online learning of the system status (i.e., expected future utilization link levels), where links are turned off and on via the use of a small and scalable neural network. Simulation results with various synthetic traffic models over various network topologies show that the proposed work can reach up to 13% power savings when compared to a trivial threshold computation, at very low (<4%) hardware overheads.

1. Introduction

Power management is a crucial element in modern-day on-chip interconnects. Significant efforts have been made in order to address power consumption in networks-on-chips (NoCs) [1–6]. One of the most power-hungry NoC components are the links connecting the routers to each other and the processing elements (PEs) of the on-chip interconnection network (NoC). Recent data from Intel's Teraflop NoC prototype [7] suggests that link power consumption could be as high as 17% of the network power and could be even more given the types of links used as well as the size and pipelining involved in designing the link structure. These links, which can be designed with differential signals and low-voltage swing hardware using level converters as circuit-based optimizations for low power consumption, are almost active all the time, even when not transmitting useful data thus spending energy when no inter-router communication exits. While such traditional hardware design techniques have contributed towards reducing the power of these links, a system-level technique becomes necessary for more efficient power reduction, as the number of links increases with the scaling and increasing sizes of NoCs, and as application-specific knowledge becomes available. For example, power-aware-encoding techniques [8] such as Gray coding cannot be efficiently used, as the hardware cost in the encoder/decoder increases drastically as the system scales to a higher number of network components. As such, recent research focuses on turning links off and on in order to reduce power consumption, and has been adopted by several works [1, 6, 9–11], as certain links in the system are severely underutilized during a specific operational time frame [1]. Techniques such as DVFS (dynamic voltage and frequency scaling) applied to the link hardware, [9, 11] have been used to vary the link frequency and power according to link utilization; however, even when not data is sent across a link, static power is still being consumed, especially

in multipipelined links with pipeline buffers in place. In addition, CMOS technology scaling is pointing towards an increased portion of the allocated power budget being consumed as static energy instead of dynamic energy; hence, switching on/off links instead of just selectively reducing their frequency/voltage levels offers better power saving advantages as links still do burn power even at lower (i.e., nonzero) voltage-frequency settings [12]. The majority of these on/off link dynamic power-management works employ traditionally a statically-computed threshold value on the link utilization, and based on that threshold value, the link is turned off for an amount of time and then is turned back on when the algorithm decides so. This of course is a pessimistic approach by nature, and imposes harder performance constraints. Recently, the use of control theory for managing candidate links for turning off has been proposed as an idea in [10], with promising results when compared to the statically-based approaches.

Motivated by the findings in [10], this paper proposes the use of artificial neural networks (ANNs) as a dynamic link power consumption management mechanism, by utilizing application traffic information. Based on their ability to dynamically be trained by variable scenarios, ANNs can offer flexibility and high prediction capabilities [13]. An ANN-based mechanism can be used to intelligently compute dynamically the threshold value used to determine which links can be turned off and on during discrete time intervals. The ANN receives link utilization data in discrete time intervals, and predicts the links that should be turned off or on based on the computed threshold. ANNs can be dynamically trained to new application information, and have been proven that they can offer accurate prediction results in similar scenarios [14]. ANNs can be efficiently designed in hardware provided they remained relatively small, through efficient resource sharing and pipelining. Furthermore, by partitioning the NoC, individual small ANNs can be assigned to monitor each partition independently, and in parallel monitor the entire network. This work also introduces topology-based directed training as a pretraining scheme, using guided simulation, which helps to minimize the large training set and the ANN complexity. This work extends our initial idea presented in [15] by several new contributions: (a) the ANN architecture has been redesigned, making it flexible and smaller through trade-off simulations involving the size and structure of the ANN and the offered power savings, (b) extended discussion on the architecture and its hardware implementation, (c) extended discussion on the simulation platform, synthetic traffic benchmarks and power modeling, and (d) extended results related to the power savings versus the performance penalty and the associated hardware overheads.

The rest of this paper is organized as follows. Section 2 discusses background and related work. In Section 3, we introduce the ANN-based approach for managing link power in NoCs. Section 4 presents the simulation framework and simulation results and analysis through various topologies and synthetic traffic, and Section 5 concludes the paper giving brief future research directives.

2. Background and Related Work

2.1. ANN Background and Motivation. An ANN is an information-processing paradigm that is inspired by the way biological neurons systems process information. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs learn by training and are typically trained for specific applications such as pattern recognition or data classification. ANNs have been successfully used as prediction and forecasting mechanisms in several application areas, as they are able to determine hidden and strongly nonlinear dependencies, even when there is a significant noise in the data set [14]. ANNs have been used as branch prediction mechanisms in computer architecture, as forecasting mechanisms in stocks [14], and in several other prediction applications. The ANN operates in two stages: the training stage and the computational stage. A neuron takes a set of inputs and multiplies each input by a weight value, which is determined in training stage, accumulating the result until all the inputs are received. A threshold value is then substracted from the accumulated result, and this result is then used to compute the output of the neuron based on an activation function. The neuron output is then propagated to the neurons of the next layer which perform the same operation with the newly set of inputs and their own weights. This is repeated for all the layers of an ANN. A neural network can be realized in hardware by using interconnected neuron hardware models, each of which is composed by multiplier accumulator (MAC) units, and a look-up Table (LUT) as a shared activation function. Some memory is also required to hold the training weights.

2.2. Related Work in Link Dynamic Power Management. Recently published research practice surveys such as [16] which outline the design challenges and lay the roadmap in future NoC design have emphasized the critical need to conduct research in NoC power management due to concerns of battery life, cooling, environmental issues, and thermal management, as a means to safeguard the scalability of general-purpose multicore systems that employ NoCs as their communication backbone. Link dynamic power management has been given significant attention by NoC researchers, as circuit-based techniques such as differential signals and low-voltage swing hardware using level converters do not seem to adequately address the power management problem [17, 18]. As such, there is a significant shift towards high-level techniques such as selective turning of links on and off. The challenge involved in those techniques includes the computation of the decision on whether a certain link is to be turned off, and when it will be turned back on. These decisions typically rely on information from the system concerning link utilization, and, so far, have been taken using a threshold-based approach. There have been attempts in dynamic link frequency and dynamic link voltage (DVFS) management with most using these thresholds as well.

Among the proposed techniques, some approaches use software-based management techniques such as the one in

[17], which proposes the use of reducing energy consumption through compiler-directed channel voltage scaling. This technique uses proactive power management, where application code is analyzed during static compilation time to identify periods of network inactivity; power management calls are then inserted into the compiled application code to direct on/off link transitions to save link power. A similar approach was also taken in [19] for communication power management using dynamic voltage-scalable links. Both of these techniques, however, have been applied to highly predictive array-intensive applications, where precise idle and active periods can be extracted. Hence, run-time variability, applicable to NoCs found in general-purposed multicore chips, has not been examined. Further the work in [20] proposes software-hardware hybrid techniques that extend the flow of a parallelizing compiler in order to direct run-time network power reduction. In this paper, the parallelizing compiler orchestrates dynamic-voltage scaling of communication links, while the hardware part handles unpredicted online traffic variability in the underlying NoC to handle unexpected swings in link utilization that could not be captured by the compiler for improved power savings and performance attainability.

Low-level, hardware-based techniques that determine on/off periods and manage the voltage and frequency, exhibit however better energy savings as they can shorten the processing time required for a decision whether to turn a link off or on to be made. The most commonly used power management policies deal with adjusting processing frequency and voltage (dynamic voltage scaling—DVS). The works in [5, 18] present DVS techniques that feature a utilization threshold to adjust the voltage to the minimum value while maintaining the worst case execution time. In [21], the authors propose that the dynamic voltage scaling is performed based on the information concerning execution time variation within multimedia streams. The work in [22] proposes a power consumption scheme, in which variable-frequency links can track and adjust their voltage level to the minimum supply voltage as the link frequency is changed. Furthermore, [11] introduces a history-based DVS policy which adjusts the operating voltage and clock frequency of a link according to the utilization of the link/input buffer. Link and buffer utilization information are also used in [9], which proposes a DVS policy scheme that dynamically adapts its voltage scaling to achieve power savings with minimal impact on performance. Given the task graph of a periodic real-time application, the proposed algorithm in [9] assigns an appropriate communication speed to each link, which minimizes the energy consumption of the NoC while guaranteeing the timing constraints of real applications. Moreover, this algorithm turns off links statically when no communications are scheduled because the leakage power of an interconnection network is significant. In general on/off links have, in most cases, been more efficient than DVFS techniques, as links, even if operating at a lower voltage, still consume leakage and dynamic power [1, 6]. These works therefore present a threshold-based technique that turns links off when there is low utilization, using a statically computed threshold. Given that static computation by nature is pessimistic, dynamic policies have been proposed. Research work in [23] proposes a mechanism to reduce interconnect power consumption that combines dynamic on/off network link switching as a function of traffic while maintaining network connectivity, and dynamically reducing the available network bandwidth when traffic becomes low. This technique is also based on a threshold-based on/off decision policy. Next, the work in [24] considers a 3D torus network in a cluster design (off-chip interconnection network) to explore opportunities for link shutdown during collective communication operations. The scheme in [25] introduces the Skip-link architecture that dynamically reconfigures NoC topologies, in order to reduce the overall switching activity and hence associated energy consumption. The technique allows the creation of long-range Skip-links at run-time to reduce the logical distance between frequently communicating nodes. However, this is based on application communication behavior in order to extract such opportunities to save energy. Finally the related work in [26] explores how the power consumed by such on-chip networks may be reduced through the application of clock and signal-gating optimizations, shutting power to routers when they are inactive. This is applied at two levels: (1) at a granular level applied to individual router components and (2) globally at the entire router.

Run-time link power management has recently gained ground in research to address the leakage issues as well. As links become heavily pipelined to satisfy performance constraints, link buffers and pipeline buffers contribute significantly in leakage power consumption. As such, the problem becomes significant with the increased on-chip NoC sizes, impacting both the power consumption as well as the thermal stability of the chip. Dynamic link management techniques have therefore been proposed; the work in [2] proposes an adaptive low-power transmission scheme, where the energy required for reliable communications is minimized while satisfying a QoS constraint by varying dynamically the voltage on the links. The work in [27] introduces ideas of dynamic routing in the context of NoCs and focuses on how to deal with links or/and routers that become unavailable either temporarily or permanently. Such techniques are a little more complicated than a threshold-based approach, and inhere performance overheads during each dynamic computation. As such, the work in [10] introduces the idea of an intelligent method for dynamic (run-time) power management policy, utilizing control theory. A preliminary idea of a closed-loop power management system for NoCs is presented, where the estimator tracks changes in the NoC and estimates changes in service times, in arrival traffic patterns and other NoC parameters. The estimator then feeds any changes into the system model, and the controller sets the voltage and frequency of the processor for the newly estimated frequency rate. Motivated by the promising results presented in [10], and the potential performance benefits of dynamic threshold computation techniques, this work proposes a dynamic, intelligent, and flexible scheme based on ANNs for dynamic computation of the threshold that determines which links can be turned off or on.

3. ANN-Based Threshold Computation Methodology

3.1. Static Threshold Computation for On/Off Links. The first step in realizing the proposed ANN methodology is to establish a framework for comparing whether an intelligent management is comparable to the nonintelligent case, not only in terms of energy savings, but also in terms of throughput and hardware overheads. As such, a trivial case, where a simple threshold mechanism was used to determine whether or not a link would turn off or back on, was first implemented using an NoC simulation framework and the Orion power models [28] (explained later in Section 4.1). The mechanism chooses an appropriate threshold based on which the links turn on and off. This trivial algorithm takes as input the link utilizations of all the links in the experimental NoC system, and outputs control signals based on a statically defined threshold; based on this threshold, the algorithm then decides which links are turned off and then back on. The statically-defined threshold was computed based on simulation observations from different synthetic traffic models and based on the observed power savings and throughput reduction when compared to a system without the mechanism. Figure 1 shows the real-time power savings for four synthetic traffic models, observed over a 4×4 NoC.

This method was introduced in [1], and the results presented therein as well as the experiments with our framework indicate that such mechanisms can be quite effective. However, a run-time mechanism, which can potentially benefit from real-time information stemming from the network, can potentially outperform this method. Such mechanism is described next. Furthermore, [1] uses an open-loop mechanism, prone to oscillations that potentially can limit both the attainable performance and also the power savings, as power is still used during the transition [10].

3.2. Mechanism Overview. The ANN-based mechanism can be integrated as an independent processing element in the NoC (PE), potentially located in a central point in the network for easy access by the rest of the PEs, and each base ANN mechanism can be assigned to monitor an NoC partition. Such cases are shown in Figure 2(a). Each base ANN mechanism monitors all the average link utilization rates within its region. These values are processed by the ANN, which computes the threshold utilization value for each link within its region, during each interval. The threshold value is then used to turn off any links in the region that exhibit lower utilization. Links which have been turned off remain off for a certain period of time. Experiments in related work [1, 10] indicate that such time should be within a few hundred cycles, as longer periods tend to create a vast performance drop-off (as the network congestion increases due to lack of available paths), whereas shorter periods do not incur worthy power savings. The proposed ANN mechanism uses a 100-cycle interval, during which all new utilization rates are received. This interval was chosen based on existing experiments in [1], which shows that a 100-cycle interval incurs better performance to power savings. The interval, however, is a system parameter, which can also be taken into consideration by the system training, and involves future work. During the interval span, the ANN computes and outputs the new threshold, which is then used by the link control mechanisms in each router to turn off underutilized links. The links remain off for another 100 cycles, and turn back on when a new threshold is computed. During the 100-cycle interval, links which are off, do not participate in the computation of the next threshold; instead, they are encoded with a sentinel value that represents them being fully utilized, so they are not kept off in two subsequent intervals. This reserves fair path allocation within the network.

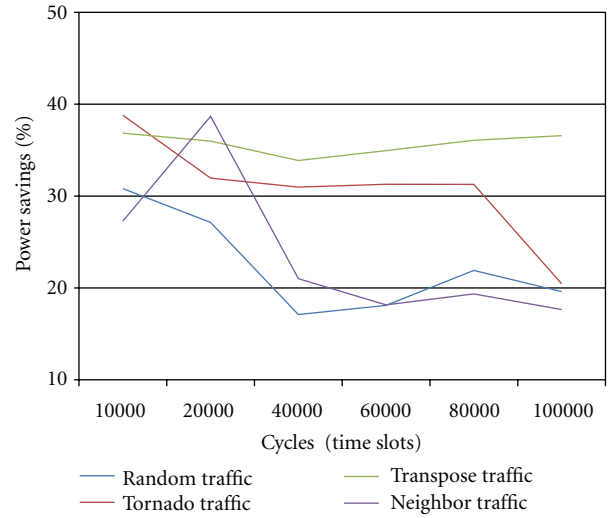


Figure 1: Power savings of a trivial threshold case compared to no on/off links case.

Each ANN-based mechanism follows a fully connected multilayer perceptron model [13, 14], consisting of one hidden layer of internal neurons/nodes and a single output-layer neuron. The activation function used in this work is the hyperbolic tangent function, which is symmetric and asymptotic, henceforth easy to implement in hardware as a LUT [29]. Furthermore, the specific function has been extensively used in several ANNs and its accuracy has been very good [13]. The ANN system is shown in Figure 2(b). The number of internal neurons was chosen to be the half of the summation of the input and output neurons [13]. The input neurons depend on the number of links that the system receives as feedback. As such, the size of the ANN depends on the number of inputs to the system. The output neuron chooses the corresponding threshold that best matches the pattern observed through the hidden layer neurons and outputs the threshold value to the link controller.

The neuron computation involves computing the weighted sum of the link utilization inputs. An activation function is then applied to the weighted sum of the inputs of the neuron in order to produce the neuron output (i.e., *activate* the neuron). Equation (1) shows how the output of a neuron is calculated.

Function which calculates the output of a neuron:

$$f(x) = K\left(\sum_i w_i g_i(x)\right). \quad (1)$$

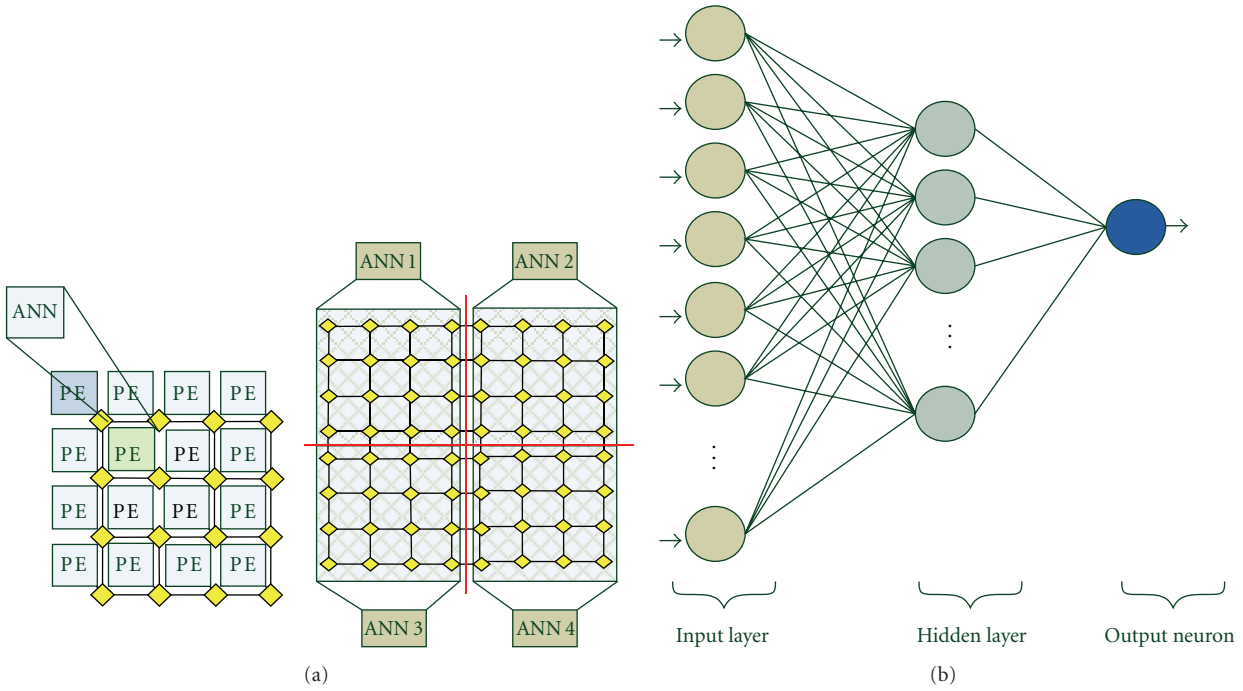


Figure 2: (a) ANN predictor with NoCs and an 8×8 network partition into four 4×4 networks with their ANNs; (b) Structure of the neural network.

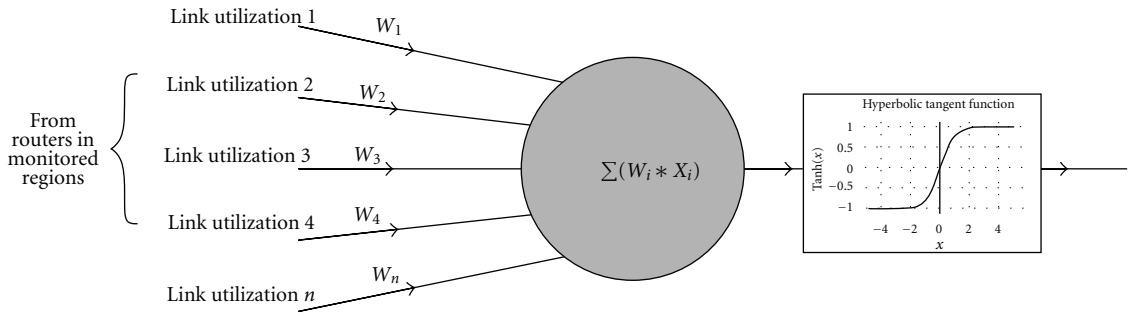


Figure 3: Neuron computations.

where K represents the activation function which is the hyperbolic tangent, w represents the weights which apply to the link utilization inputs which are represented by $g(x)$ input function. The overall procedure is shown in Figure 3.

3.3. ANN Training and Operation. The training stage can be performed off-line, that is, when the NoC is not used, and the training weights can be stored in SRAM-based LUTs for fast and on-line reconfiguration of the network. The network is trained using application traffic patterns, off-line, using the back-propagation ANN training algorithm [14]. In our experiments, we used synthetic traffic patterns and the Matlab ANN toolbox; the weight values were then fed to the simulator as inputs, where the actual prediction was then implemented and simulated. The operation of the ANN can be potentially improved, by categorizing the applications that a system will practically run. As such, for each application category (and subsequently traffic patterns with certain common characteristics), the ANN can be trained with the corresponding weights. Each training set can then be dynamically loaded during long operation intervals, where the system migrates to a new application behavior.

3.4. Intelligent Threshold Computation—ANN Size and NoC Scalability Issues. While ANNs are heavily efficient in predicting scenarios based on learning algorithms, they require careful hardware design considerations, as their size and complexity depend on the number of inputs received as well as the number of different output predictions (classes) that they have to do. NoCs consist of a large number of links which grow exponentially as the size of the NoC grows. Therefore, receiving link utilization and having to determine the threshold that controls which links are candidates for turning off and on would require an exponentially scalable ANN. As such, we devise a preprocessing technique, which identifies, based on simulation and observations, the set of

candidate links for turning off and on, eliminating links which are almost always utilized. This depends obviously on the chosen network topology (e.g., in a 2D mesh topology, links that are likely to be less busy include links which are at the edges of the mesh, whereas central links are usually more active and can be left on all the time), so that the ANN mechanism can handle the output decision in a more manageable way. This can be aided by intelligent floor planning and placement of processing elements inside the NoC as well, but is beyond the scope of this paper. Through various synthetic traffic simulations, for each given NoC topology, the average utilization values for each link through various phases in the simulation are computed, and the links with the highest utilization values are always assumed that they will be on. Obviously this step reduces a little the effectiveness of the ANN, but it is necessary to minimize the size and overheads of the ANN both in terms of performance and in terms of hardware resources. This step has to be done for a given topology, prior to the ANN training. However, both steps (determining the links that the ANN will use, as well as the ANN training) can be done off-line, during the NoC design stage. The ANN training can also be done repeatedly whenever new application knowledge becomes available that might alter the on-chip network traffic behavior. This particular property of ANNs provides a comparative advantage against a statically computed threshold, making the NoC flexible under any application that it is required to facilitate. It must be stated that the number of links that will be considered as likely candidates for on/off activity (i.e., the ones which do tend to have low utilization during the pretraining stage) impact both the size of the ANN itself and the overall size of the mechanism (which involves logic that sends the appropriate control signals). Through the two steps, pretraining and training, each ANN can be trained and configured independently to satisfy its targeted NoC structure (topology and number of monitored links).

Furthermore, large NoCs can be partitioned into smaller regions. As such, a base ANN architecture can be assigned to monitor each region, and all the link utilizations of the routers of the NoC partition arrive at the ANN which is responsible for that region. The size of this NoC region, however, depends on two major factors; the incurred power savings that the corresponding base ANN offers, which depend on its ability to process and evaluate the input information, and the resulting ANN size and hardware overheads (and subsequently power consumed within the ANN) which grow exponentially as the size of the NoC region grows. Choosing a small NoC region will likely result in a small ANN, but will result in smaller savings since the ANN will not have enough information to compute a good threshold value. On the other hand, a large NoC region will provide the ANN with much more information and potentially result in a much better threshold value, but its size and overheads would reduce the power savings making the ANN ineffective. As such, we experimented with several NoC regions and base ANNs, comparing their hardware overheads (a product of the ANN power consumption and the gate count required to implement each ANN in hardware) and responding savings incurred with the computed threshold.

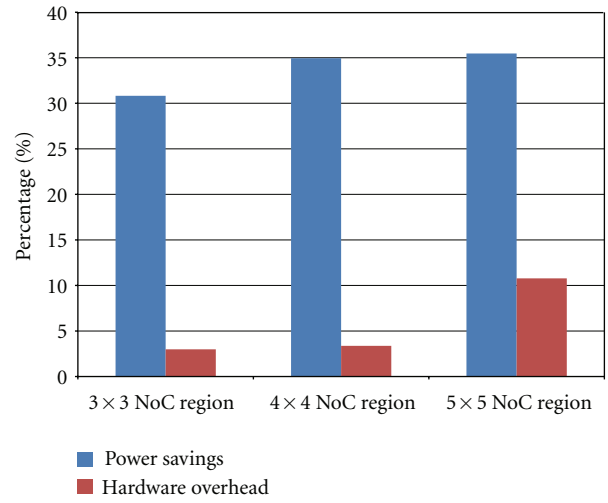


Figure 4: Power savings versus CMOS hardware overheads corresponding to various sizes of ANN monitoring regions in an NoC.

Figure 4 shows a comparison between hardware overheads (power × gate count) and power savings in the cases of 3×3, 4×4, and 5×5 ANN sizes for monitoring regions in an NoC. Results show that computation over a 4×4 NoC region offers satisfactory power savings and significantly less ANN overheads when compared to a 5×5 NoC region. A 3×3 NoC region does not provide enough information to the ANN in order to make accurate predictions. Based on these observations, we designed the base ANN system to monitor 4×4 NoC regions.

3.5. Base (4 × 4) Artificial Neural Network Operation. The ANN mechanism is responsible to compute for all the link utilizations the minimum values during each interval. Based on these values, the ANN calculates an optimal threshold. Figure 5 shows the procedure of the ANN mechanism for a 4×4 NoC partition. The ANN mechanism receives all the average link utilizations from all the links of the 4×4 NoC partition. These values are fed to the ANN in order to calculate an optimal threshold. Each router contains a control hardware monitor that measures the average link utilization for each of the four links in each router, and this value is sent to the ANN every n cycle (where n is the size of the time interval). If a router fails to transmit the values at a single interval, its value is set to sentinel value, which shows that its buffers are fully utilized. This mechanism acts also as a congestion information mechanism because links which are heavily active are not candidates to be turned off. The ANN uses the utilization values to find the threshold which will determine if a link is going to be turned off or on for the next n-cycle interval. As said earlier, we used 100-cycle intervals [1] (i.e., $n = 100$) in our simulations.

3.6. Base (4 × 4) Artificial Neural Network Hardware Architecture. One of the main advantages of ANNs is their simple hardware implementation when the number of neurons remains small and the activation function remains simple

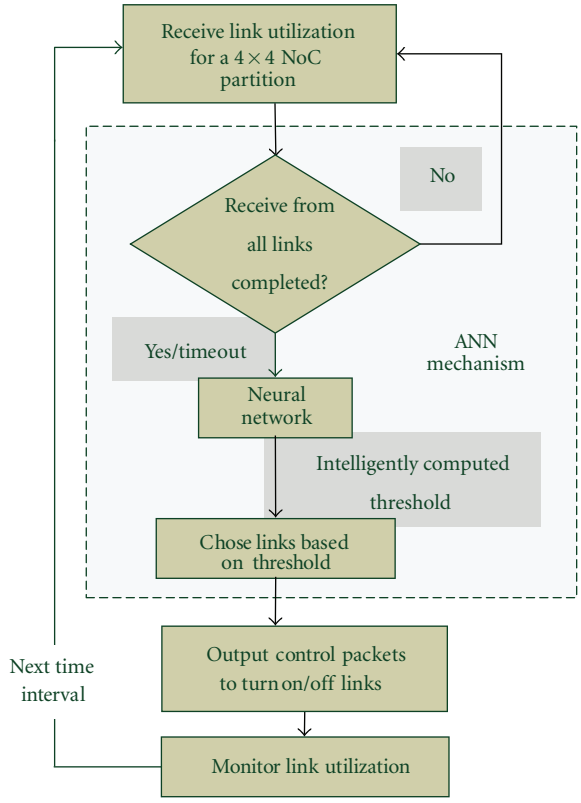


Figure 5: Main steps of a 4 × 4 ANN predictor.

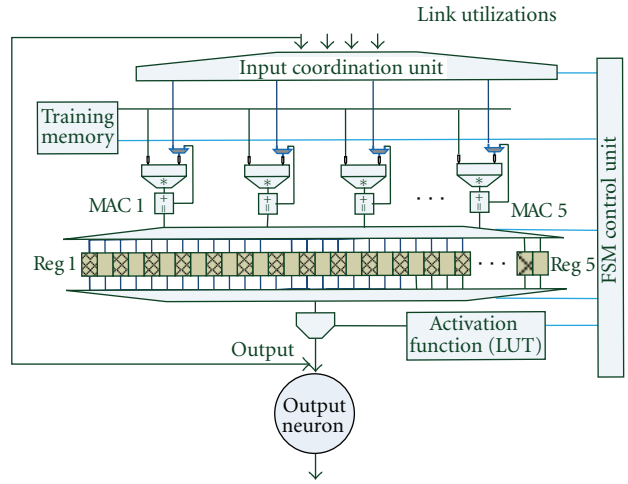


Figure 6: ANN hardware architecture and its hardware realizations.

[13]. The neuron operation can be designed efficiently in hardware since it can be modeled as a multiply-accumulate operation. The ANN hardware implementation depends on the number of hidden layer neurons; each neuron is implemented as a multiplier-accumulator (MAC) unit, with the accumulators being multiplexed for each neuron, so that the number of multipliers is minimized. The base ANN hardware architecture is shown in Figure 6. Utilization values for each link arrive and sorted through an input coordination unit, which distributes the values to each of the appropriate multipliers. The multipliers receive these values and through a shared weights memory, the corresponding weight. The weights and inputs product is then accumulated in the corresponding accumulator, with the entire process controlled via a finite-state machine controller. Each neuron has an assigned storage register, to enable data reuse; when one layer of neurons is computed, their outputs are stored inside a corresponding register. As such, the same hardware is reused for computing the next layer (i.e., from input layer to hidden layer and from hidden layer to output layer). When each neuron finishes its MAC computation, the result is then computed through the activation function LUT and propagates to the output neuron.

An ANN monitoring a 4×4 region in a torus topology, for example, receives 64 different inputs; if we are to assume that each router transmits a packet with its own link utilization during each interval, and if we also assume one packet per cycle delivered to the ANN during each interval, then, during each cycle, the ANN will receive at most 4 input values. Hence, if we use pipelined multipliers, we need only 4 multipliers for each ANN to achieve maximum throughput. The ANN therefore remains small and flexible, regardless of the size of the network it monitors. Furthermore, an ANN monitoring a 4 × 4 NoC partition receives 16 packets (one for each router); as such, it requires $16m$ cycles (where m is the cycle delay of each multiplier), plus 16 cycles for each accumulator, plus one cycle for the activation function plus one cycle for the output neuron, to output the new threshold (total of $16m + 18$ cycles). The overall data flow and architecture is shown in Figure 6.

3.7. ANN Hardware Optimization and Trade-Offs. In order to make the ANN architecture simpler and smaller we studied how the number of neurons of the hidden layer affect the total power savings of the system. Given that the 4 × 4 ANN monitors 16 routers, we need at least 8 input neurons [14]. Having eight neurons at the input layer of the ANN means that the hidden layer should have five neurons (based on the rule of thumb that a satisfactory number of the hidden layer neurons equals to half the number of input neurons plus one neuron) [14]. Three different ANNs were developed with five, four, and three neurons at the hidden layer, respectively. Figure 7 shows the power savings for these ANNs under the use of four different traffic patterns (Random, Tornado, Transpose, and Neighbor). Using four neurons therefore (instead of five), in the hidden layer exhibits the best power savings for all the traffic patterns. In addition, we studied how the bit representation of the training weights affects the threshold computation and subsequently the total power savings. Figure 8 shows how the bits used in representing the training weights influence the power savings of the system. As we can see 24, 16, 8, and 6 bits show similar power savings, but these savings are significantly reduced when 4 bits are used, due to reduced training accuracy. Based on the above, we selected the weight bit representation from 6 bits, which made the multiplier-accumulation hardware very small, requiring a 6-bit port for each weight and a 5-bit port for the utilization values.

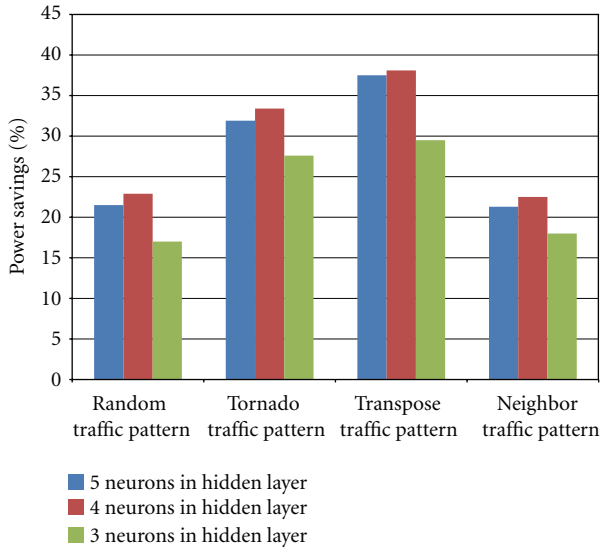


Figure 7: Power savings for five, four and three neurons in the hidden layer of the ANN.

4. Simulation and Results

4.1. Experimental Setup. In order to evaluate the ANN-based on/off link prediction mechanism, we developed a simulation framework based on the Java-based cycle-accurate *gpNoCsim* simulator (general-purpose simulator for network-on-chip architectures) [30]. The framework enables simulation of multiple topologies, utilizing dimension-ordered XY routing algorithm with virtual channel support and 4-stage pipelined router operation. The simulated router supports 64-bits flit width, 2 virtual channels per link and two buffers per virtual channel. The routers used are all the same, and we assume wormhole flow control. The framework supports various synthetic and user-defined traffic models. We experimented with a 4×4 mesh topology, and mesh and torus 8×8 topologies. Simulations are done over a range of 200,000 cycles, with a warm-up period of 100,000 cycles. In the 8×8 topologies, we partitioned the NoC into four regions of 4×4 routers/links, where each ANN-based model was assigned as responsible for monitoring. The ANN-based models monitored all links in their corresponding partition, all links were candidates for off/on, and all ANN results related to the size and operation of the ANN are given based on these architectural details.

Time was divided into 100-cycle intervals [1]; at the end of each interval, all routers in the NoC partition transmit their average utilization data for that span (computed via a counter and LUT-based multiplication with the reciprocal of the interval). A time-out mechanism equal to the expected delay of each router towards the ANN mechanism is imposed to maintain reasonable delays. The ANN receives one packet from each router with four utilization values, one for each port. The ANN then proceeds to compute the new threshold which is transmitted to each router through a control packet. Each router then turns off each link, depending whether or not its utilization value is above or below the new threshold. The router continues operation until the end of the new interval. It must be repeated that when a link is turned off or on, an extra 100-cycle penalty is inserted into the simulation to indicate the impact on the network throughput.

While the savings could significantly be improved using a more intelligent routing algorithm than the trivial XY dimension-ordered algorithm (DOR), we experimented with the XY algorithm and induced blocking when a link was off and the output buffer of that link fully utilized. In the case that a packet arrives in a router, and its destination link is off, the packet resides in the buffer until the link is turned back on. While this incurs a performance penalty, it is necessary to maintain correctness; this is a pessimistic approach obviously—a better routing algorithm would likely yield much better throughput and power savings, but is beyond the scope of this paper.

In order to study the power savings and the throughput of the dynamic ANN-based prediction algorithm for turning links on/off, we compare this to a static threshold-based algorithm and to a system without any on/off mechanism. Prior to discussing the simulation results, we first explain the power modeling followed in the experiments.

4.2. Power Modeling. We adopted the Orion power models for the dynamic power consumption of each router [28]. Router and link hardware were designed and synthesized in Verilog and Synopsys Design Compiler in order to obtain the leakage power values. We used a commercial CMOS 65 nm library, and a sequence of random input vectors, for several thousand cycles, and measured the leakage power of each router and link, through all computation cycles and combinations of events. The leakage values are then fed into the simulator, along with the Orion models for active power, and the overall power is computed. In addition, we take into consideration the start-up power consumed when we turn a link back on.

4.3. Simulation Results and Discussion. Using synthetic traffic patterns with varied injection rates (Random, Tornado, Transpose, and Neighbor) [31, 32], we first evaluated the power savings of the ANN-based mechanism when compared to the same system without any on/off link capability, and when compared to a system that employs a statically determined threshold. The traffic patterns, for which we experimented, are a superset of the patterns used to train the ANN; we measured power savings and the impact of the throughput on all the traffic patterns. However, in order to compute the power savings in the torus network, we follow the guided-training approach as described in Section 3.3, and we measure link utilizations in all possible partitions of the torus network to compensate for the toroidal links. The link utilizations with the least values (from all the link utilizations, from all the partitions of the torus network) are then passed through the ANNs. Figure 9 shows the comparison when targeting 8×8 mesh and torus NoCs. The power savings of the ANN-based mechanism are better than the savings in the cases of statically determined threshold

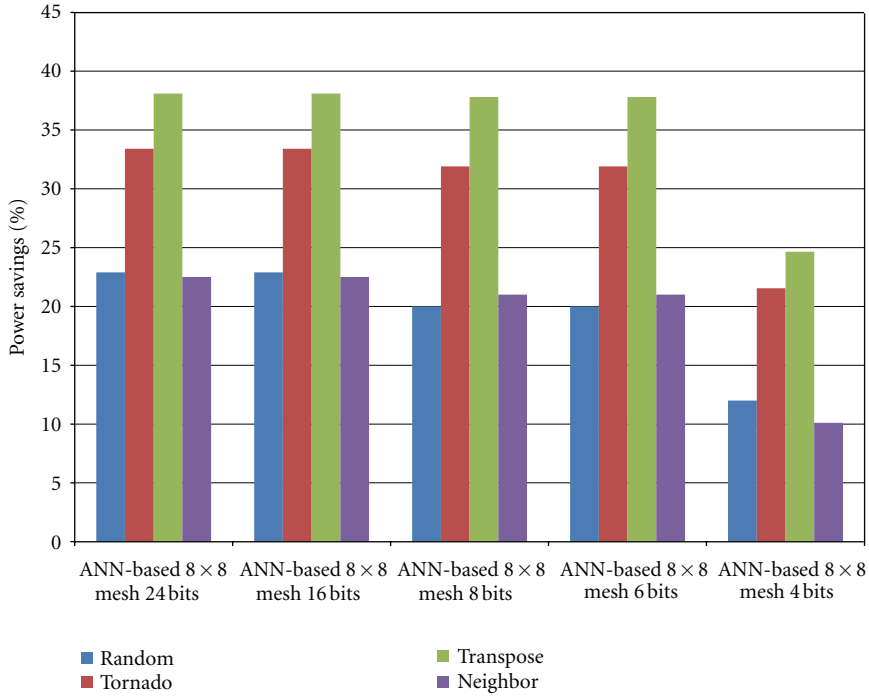


Figure 8: Power savings for different training weight bit representations.

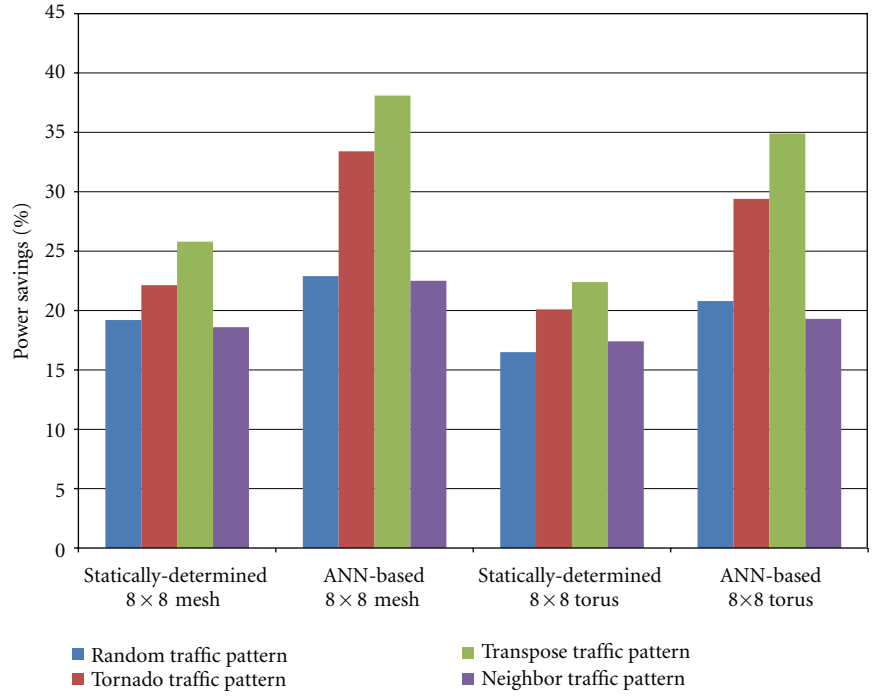


Figure 9: Power Savings for 8 × 8 mesh and 8 × 8 torus networks for the ANN-based technique, static threshold technique and no on/off technique.

and the case without any on/off links. The ANN-based mechanism can identify a significant amount of future behavior in the observed traffic patterns; therefore, it can intelligently select the threshold necessary for the next timing interval.

Next, we measure the impact of the throughput in each mechanism; while having no on/off mechanism obviously yields a higher throughput, the ANN-based technique shows better throughput results compared to statically determined threshold techniques. Figure 10 shows the throughput

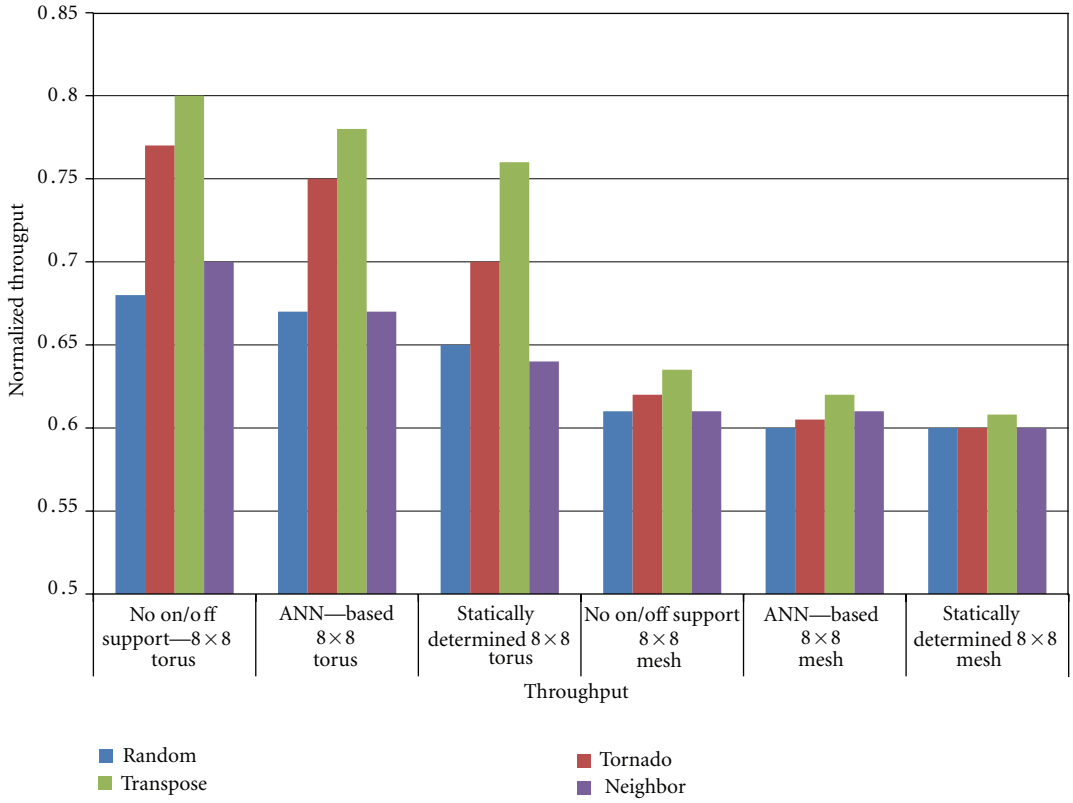


Figure 10: Average network throughput comparisons for 8 × 8 mesh and torus networks.

Table 1: Power savings/hardware overhead comparisons.

Related work	Characteristics	Power savings comparing to no algorithm	Hardware overhead
[1]	8 × 8 2D mesh topology, Uniform traffic	~37,5%—turning on/off 2 links per router	N/A
[11]	8 × 8 2D mesh topology, Pareto distribution—0.5 packet injection rate	~30%	500 equivalent logic gates per router port Delay ignored
Proposed ANN-based technique	8 × 8 2D mesh topology and 8 × 8 torus topology,uniform traffic	Up to ~40%—turning on/off links based on ANN prediction	4% of the NoC hardware for a complete 4 × 4 Mesh NoC

comparisons for an 8 × 8 mesh and an 8 × 8 torus network. The throughput values are normalized based on the number of the simulation cycles.

Figure 11 represents the normalized energy consumed in a 8 × 8 mesh network. We observe that the energy consumed using the ANN mechanism is less than the cases of statically-computed threshold and without on/off link management algorithm. The ANN exhibits a reduction in the overall energy, because of a balanced performance-to-power savings ratio, when compared to not having on/off links or when compared to static threshold computation.

Figure 12 presents the average packet delay in packets per cycle for the 8 × 8 mesh, when the ANN-based mechanism is used compared to the cases where no on/off mechanism is used and the statically computed threshold case. The ANN-based mechanism incurs more delay, but we believe that the delay penalty is acceptable when compared to the associated power savings.

4.4. ANN Hardware Overheads: Synthesis Results. To compute the hardware overheads of the proposed scheme, the ANN-based mechanism for one 4 × 4 NoC region was synthesized and implemented targeting a commercial 65 nm CMOS technology. The ensuing synthesized ANN-based controller and the associated hardware overheads in each router consume approximately 4 K logic gates (for comparison purposes, an NoC router similar to the one used in our simulation [29] consumes roughly 21 K gates),

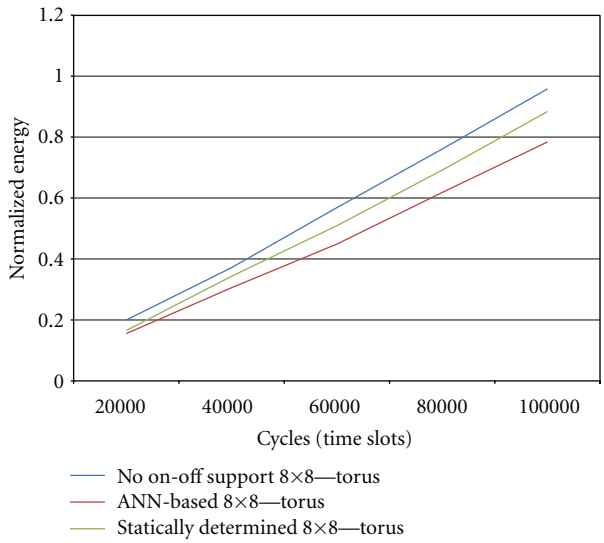


Figure 11: Energy consumption for a 8 × 8 mesh network.

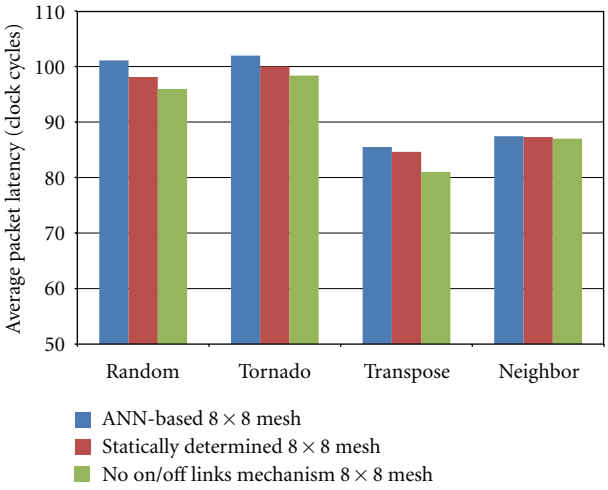


Figure 12: Average packet latency for the cases where ANN-based mechanism is used, when trivial case is used and when there is no on/off mechanism.

bringing the estimated hardware overhead for an 4 × 4 mesh network to roughly 4% of the NoC hardware.

4.5. Comparison with Related Works. Lastly, we briefly give a comparison with relevant related works that follow dynamic threshold techniques in Table 1. When compared to both [1, 11], the ANN-based prediction yields better power savings than having no prediction mechanism, while still maintaining lower hardware overheads. We must note that while [10] was the motivating idea behind our paper, it presented only a preliminary implementation of the idea, without enough information about hardware overheads and power savings in order to make an informed comparison.

5. Conclusions

This paper presented how an ANN-based mechanism can be used to dynamically compute a utilization threshold, that can be in turn used to select candidate links for turning on or off, in an effort to achieve power savings in an NoC. The ANN-based model utilizes very low hardware resources, and can be integrated in large mesh and torus NoCs, exhibiting significant power savings. Simulation results indicate approximately 13% additional power savings when compared to a statically determined threshold methodology under synthetic traffic models. We hope to expand the results of this paper to further explore dynamic reduction of power consumption in NoCs using ANNs and other intelligent methods.

References

[1] V. Soteriou and L.-S. Peh, "Dynamic power management for power optimization of interconnection networks using on/off links," in *Proceedings of the 11th Symposium on High Performance Interconnects*, pp. 15–20, 2003.

[2] 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*, pp. 92–100, October 2002.

[3] S. Kumar, A. Jantsch, J.-P Soininen et al., "A network on chip architecture and design methodology," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pp. 117–124, 2002.

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

[5] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power CPU," in *Proceedings of the 1st Annual International Conference on Mobile Computing and Networking*, pp. 13–25, November 1995.

[6] V. Soteriou and L. S. Peh, "Exploring the design space of self-regulating power-aware on/off interconnection networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 3, pp. 393–408, 2007.

[7] Y. Hoskote, S. Vangal, S. Dighe et al., "Teraflops Prototype Processor with 80 Cores," Microprocessor Technology Labs, Intel Corporation.

[8] W.-C. Cheng and M. Pedram, "Low power techniques for address encoding and memory allocation," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '01)*, pp. 245–250, 2001.

[9] D. Shin and J. Kim, "Power-aware communication optimization for networks-on-chips with voltage scalable links," in *Proceedings of the 2nd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '04)*, pp. 170–175, September 2004.

[10] T. Simunic and S. Boyd, "Managing power consumption in networks on chips," in *Proceedings of the Design, Automation and Test in Europe*, pp. 110–116, 2002.

[11] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in *Proceedings of the International Symposium on High Performance Computer Architecture*, pp. 91–102, 2003.

[12] Semiconductor Industry Association, "International Technology Roadmap for Semiconductors," 2009, http://www.itrs.net/Links/2009ITRS/Home2009.htm.

[13] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.
[14] R. Schalkoff, *Artificial Neural Networks*, McGrow-Hill, 1997.
[15] A. Savva, T. Theocharides, and V. Soteriou, "Intelligent on/off link management for on-chipnetworks," in *Proceedings of the IEEE Annual Symposium on VLSI*, pp. 343–344, 2011.
[16] R. Marculescu, U. Y. Ogras, L. S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3–21, 2009.
[17] G. Chen, F. Li, M. Kandemir, and M. J. Irwin, "Reducing NoC energy consumption through compiler-directed channel voltage scaling," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '06)*, pp. 193–203, June 2006.
[18] T. Pering, T. Burd, and R. Brodersen, "Voltage scheduling in the IpARM microprocessor system," in *Proceedings of the 2000 Symposium on Low Power Electronics and Design (ISLPED '00)*, pp. 96–101, July 2000.
[19] F. Li, G. Chen, and M. Kandemir, "Compiler-directed voltage scaling on communication links for reducing power consumption," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '05)*, pp. 455–459, November 2005.
[20] V. Soteriou, N. Eisley, and L.-S. Peh, "Software-directed power-aware interconnection networks," *ACM Transactions on Architecture and Code Optimization*, vol. 4, no. 1, pp. 274–285, 2007.
[21] E. Y. Chung, L. Benini, and G. De Micheli, "Contents provider-assisted dynamic voltage scaling for low energy multimedia applications," in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 42–47, August 2002.
[22] J. Kim and M. Horowitz, "Adaptive supply serial links with sub—1V operation and per-pin clock recovery," in *Proceedings of the International Solid State Circuits Conference (ISSCC '02)*, pp. 1403–1413, 2002.
[23] M. Alonso, S. Coll, J. M. Martínez, V. Santonja, P. López, and J. Duato, "Power saving in regular interconnection networks," *Parallel Computing*, vol. 36, no. 12, pp. 696–712, 2010.
[24] S. Conner, S. Akioka, M. J. Irwin, and P. Raghavan, "Link shutdown opportunities during collective communications in 3-D torus nets," in *Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS '07)*, pp. 1–8, March 2007.
[25] C. Jackson and S. J. Hollis, "Skip-links: a dynamically reconfiguring topology for energy-efficient NoCs," in *Proceedings of the 12th International Symposium on System-on-Chip 2010 (SoC '10)*, pp. 49–54, September 2010.
[26] R. Mullins, "Minimising dynamic power consumption in on-chip networks," in *Proceedings of the International Symposium on System-on-Chip (SoC '06)*, pp. 1–4, November 2006.
[27] M. Ali, M. Welzl, and S. Hellebrand, "A dynamic routing mechanism for network on chip," in *Proceedings of the 23rd NORCHIP Conference*, pp. 70–73, November 2005.
[28] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *Proceedings of the International Symposium on Microarchitecture*, pp. 294–305, 2002.
[29] E. Kakoullit, V. Soteriou, and T. Theocharides, "An artificial neural network-based hotspot prediction mechanism for NoCs," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI '10)*, pp. 339–344, July 2010.
[30] H. Hossain, M. Ahmed, A. Al-Nayeem, T. Z. Islam, and M. M. Akbar, "gpNoCsim—a general purpose simulator for network-on-chip," in *Proceedings of the International Conference on Information and Communication Technology (ICICT '07)*, pp. 254–257, March 2007.
[31] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, San Francisco, Calif, USA, 2004.
[32] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection Networks: An Engineering Approach*, Morgan Kaufmann, San Francisco, Calif, USA, 2003.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 537286, 12 pages
doi:10.1155/2012/537286

Research Article

A Buffer-Sizing Algorithm for Network-on-Chips with Multiple Voltage-Frequency Islands

Anish S. Kumar,[1] M. Pawan Kumar,[1] Srinivasan Murali,[2] V. Kamakoti,[1] Luca Benini,[3] and Giovanni De Micheli[4]

[1] *Indian Institute of Technology Madras, Chennai 600036, India*
[2] *iNoCs, 1007 Lausanne, Switzerland*
[3] *University of Bologna, 40138 Bologna, Italy*
[4] *EPFL, 1015 Lausanne, Switzerland*

Correspondence should be addressed to Anish S. Kumar, ask.anish@gmail.com

Received 17 July 2011; Accepted 1 November 2011

Academic Editor: An-Yeu Andy Wu

Buffers in on-chip networks constitute a significant proportion of the power consumption and area of the interconnect, and hence reducing them is an important problem. Application-specific designs have nonuniform network utilization, thereby requiring a buffer-sizing approach that tackles the nonuniformity. Also, congestion effects that occur during network operation need to be captured when sizing the buffers. Many NoCs are designed to operate in multiple voltage/frequency islands, with interisland communication taking place through frequency converters. To this end, we propose a two-phase algorithm to size the switch buffers in network-on-chips (NoCs) considering support for multiple-frequency islands. Our algorithm considers both the static and dynamic effects when sizing buffers. We analyze the impact of placing frequency converters (FCs) on a link, as well as pack and send units that effectively utilize network bandwidth. Experiments on many realistic system-on-Chip (SoC) benchmark show that our algorithm results in 42% reduction in amount of buffering when compared to a standard buffering approach.

1. Introduction

In modern SoC designs, power consumption is a critical design constraint as they are targeted as low-power devices. To achieve this, SoC designs employ power gating, where the cores are shutdown when they are unused. Instead of shutting down each core, certain techniques cluster cores into *voltage and frequency* (VF) *islands,* and when all the cores in an island are unused, the entire VI is shut down. The cores in a single VI have same operating voltage but can operate at different frequencies. Running cores at different frequencies is an effective method to trade off performance and power consumption.

Scalable on-chip networks, network-on-chips (NoCs), have evolved as the communication medium to connect the increasing number of cores and to handle the communication complexity [1–3]. With designs having multiple VF islands, the interconnect can reside in a separate island. By clustering the NoC into a single island, routing the VDD and ground lines across the chip becomes difficult. Instead, the NoC is spread across the entire chip with different components of the network operating at different voltage/frequency. If the core in an island is operating in a different frequency than the switch to which it is connected, the NI does the frequency conversion, and when a switch from one island is connected to another switch in a different island, *frequency converters* (FCs), such as the ones in [4, 5], are used to do the frequency conversion. Even if the two switches are operating at same frequencies, there might be clock skew for which synchronization is required.

In an NoC, a packet may be broken down into multiple flow control units called flits, and NoC architectures have the ability to buffer flits inside the network to handle contention among packets for the same resource link or switch port. The buffers at the source network interfaces (NIs) are used to queue up flits when the network-operating frequency is different from that of the cores or when there is congestion inside the network that reaches the source. NoCs also employ

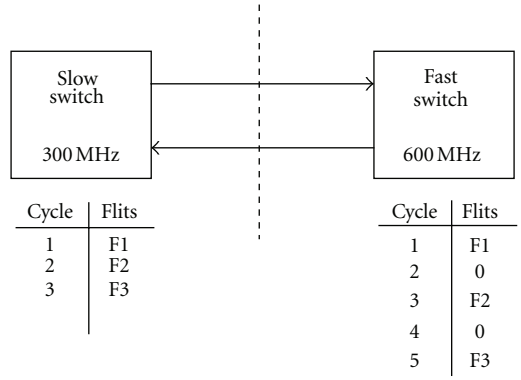


Figure 1: Bubbles generated moving from slow to fast clock.

some flow control strategy that ensures flits are sent from the switch (NI) to another switch (NI) only when there are enough buffers available to store them in the downstream component.

In many NoCs, a credit-based flow control mechanism is used to manage transfer of flits at full throughput. In this scheme the upstream router keeps a count of the number of free buffers in the downstream router. Each time a flit is communicated from an upstream router and is consumed by the downstream router, the credit counter is decremented. Once the downstream router forwards the flit and frees a buffer, a credit is sent to the upstream router and hence incrementing the credit count.

The network buffers account for a major part of the power and area overhead of the NoC in many architectures. For example, in [6], the buffers account for more than 50% of the dynamic power consumption of the switches. A major application domain for NoCs is in mobile and wireless devices, where having a low-power consumption is essential. Thus, reducing the buffering overhead of the NoC is an important problem.

As such NoCs are targeted for specific applications, the buffers and other network resources can be tuned to meet the application bandwidth and latency constraints. Several earlier works have dealt with application-specific customization of various NoC parameters, such as the topology, frequency of operation, and network paths for traffic flows [7–9]. In fact, several works have also addressed the customization of NoC buffers to meet application constraints [10, 11]. Many of the existing works utilize methods such as queuing theory and network calculus to account for dynamic queuing effects. While such methods could be used to compute the buffer sizes quickly, they have several limitations in practice. Most queuing theory-based works require the input traffic injection to follow certain probabilistic distributions, such as the Poisson arrival process. Other schemes require regulation of traffic from the cores, which may not be possible in many applications (details given in the next section).

Although these methods can be used for fast design space exploration, for example, during topology synthesis, final buffer allocation needs to consider simulation effects to accurately capture the congestion effects. In this paper, we present a simulation-based algorithm for sizing NoC buffers for application traffic patterns. We present a two-phase approach. In the first phase, we use mathematical models based on static bandwidth and latency constraints of the application traffic flows to minimize the buffers used in the different components based on utilization. In the second phase, we use an iterative simulation-based strategy, where the buffers are increased from the ideal minimal values in the different components, until the bandwidth and latency constraints of all the traffic flows are met during simulations. While in some application domains, such as in *chip multiprocessors* (CMPs), it is difficult to characterize the actual traffic patterns that occur during operation at design time, there are several application domains (such as mobile, wireless) where the traffic pattern is well behaved [12]. Our work targets such domains where the traffic patterns can be precharacterized at design time and a simulation-based mechanism can be effective.

With the communication subsystem running on different operating frequencies, the effective bandwidth and utilization on the links change. For example, when a switch operating at a slower clock frequency communicates to a switch at a higher operating frequency, bubbles may be introduced between flits. This will lead to overutilization of resources at the faster switch. As an example, consider a setup as illustrated in Figure 1. Here the destination is operating twice as fast as the source. Assume flits are forwarded at every cycle of the source switch. Since the destination is faster, the forwarded flits are consumed every other cycle (of the faster clock). This results in empty flits being generated in between the flits of a packet. The destination buffer is held by the packet till the tail flit leaves. And hence this leads to overutilization of the destination buffer, which otherwise would have been half this utilization. One way to effectively handle bubbles in the network is by employing *pack and send* (PS) units (discussed later in the paper).

Moreover, when switches of different frequencies communicate with each other, the number of buffers required varies depending where the *frequency converters* are placed. When the converters are placed to the slower clock, the link operates at the faster clock domain, thereby incurring smaller delay in transferring flits and credits. Thus, fewer buffers are required as the number of in-flight flits that need to be stored at the buffers is fewer. However, placing the converters near

the slower clock leads to higher power consumption on the links as they are operating at a higher frequency. This effect also needs to be considered during sizing of buffers.

In this paper, we present a buffer-sizing algorithm for application-specific NoCs having multiple VF islands. We consider a complex mobile benchmark to validate the buffer-sizing algorithm presented. We also analyze the effect of placement of frequency converters on the buffer size. Our results show that there is 42% reduction in the buffer budgets for the switches, on an average. Based on the models from [6], this translates to around 35% reduction in the overall power consumption of the NoC switches. We also apply the approach on a variety of *system-on-chip* (SoC) benchmarks, which show a significant 38% reduction in buffer budget. Also, we study the impact of pack and send units on the buffer utilization. Results show that the PS units have a better utilization of network resources.

2. Related Work

A lot of work has gone into proposing techniques for scaling the voltage and frequencies of different IP cores on a chip. The authors of [13] propose techniques to identify optimal voltage and frequencies levels for a dynamic voltage and frequency scaling (DVFS) chip design. In [14] the authors propose methods of clustering cores into islands, and DVFS is applied for these islands. In our work, we assume such clustering of cores and NoC components as a part of the architecture specifications. In [15] the authors identify the theoretical bounds on the performance of DVFS based on the technology parameters.

With such partitioning of cores into VF islands becoming prevalent, globally asynchronous- and locally synchronous-(GALS-) based NoC designs have become the de facto interconnect paradigm. In [16] the authors propose an algorithm to synthesize an NoC topology that supports VF islands. This is one of the first approachs to design an NoC considering the support for shutting down of VF islands. The output of this algorithm can serve as input to our approach. In [17], the authors propose an reconfigurable NoC architecture to minimize latency and energy overhead under a DVFS technique. In [18], the authors propose asynchronous bypass channels to improve the performance of DVFS enabled NoC.

In this work we extend the proposed buffer-sizing algorithm to designs with VF islands to optimize NoC power and area while meeting the design requirements. Sizing buffers is critical for reducing the power and area footprint of an NoC.

In [10], the authors proposed an iterative algorithm to allocate more buffers for the input ports of bottleneck channels found using analytical techniques and also proposed a model to verify the allocation. The model assumes the Poisson arrival of packets. In [19], buffer sizing for wormhole-based routing is presented, also assuming the Poisson arrival of packets. The problem of minimizing the number of buffers by reducing the number of virtual channels has been addressed in [20] assuming that input traffic follows certain probabilistic distributions. In [21], a queuing theory-based model to size the number of virtual channels is proposed by performing a chromosome encoding of the problem and solving it using standard genetic algorithm, again assuming the Poisson arrival of packets. The authors of [22] proposed an analytical model to evaluate the performance of adaptively routed NoCs. This work again assumes the Poisson arrivals for the flows. In [23] the authors proposed a probabilistic model to find the average buffer utilization of a flow accounting for the presence of other flows across all ports. The authors of [24] used an approach to minimize buffer demand by regulating traffic through a delayed release mechanism and hence achieving the goal of appropriate buffer sizing. Unlike all these earlier works, we make no assumption on the burstiness of input traffic and the arrival pattern for packets.

In [25], the authors propose an algorithm to size the buffers, at the NIs, using TDMA and credit-based flow control. This work is complimentary to ours, as the authors target designing NI buffers to match the different rate of operation of cores and the network. A trace-driven approach to determine the number of virtual channels is presented in [26]. While the notion of simulation-driven design method is utilized in the work, the authors do not address the sizing of buffers. Our buffer-sizing methods are significantly different from methods for virtual channel reduction, as we need a much more fine grained control on buffer assignment. Towards this end, we present an iterative approach to buffer sizing that utilizes multiple simulation runs.

3. Design Approach

In this section, we give a detailed explanation of the design approach used for buffer sizing. The approach is presented in Figure 2. We use a two-phase method: static sizing, involving constraint solving, followed by a simulation-based approach.

We obtain two sets of inputs for the buffer-sizing approach: application and architecture specifications. The application specifications include the bandwidth and latency constraints for the different flows. The architecture specifications include the NoC topology designed for the application, routes for the traffic flows, the number of voltage/frequency islands, and flit width of the NoC.

We have the following assumptions about the architecture and application.

(i) For illustrative purposes, we consider input-queued switches for buffer sizing. In fact, the algorithm presented in generic and can be easily extended to output-queued (and hybrid) switches as well.

(ii) We define the term *number of buffers* used at a port to be the number of flits that the buffers can store at that port.

(iii) A wormhole, credit-based flow control is assumed in the NoC, which is widely used.

(iv) We do not explicitly consider the use of virtual channels. The algorithm, in fact, can be easily applied to architectures that support multiple virtual channels as well.

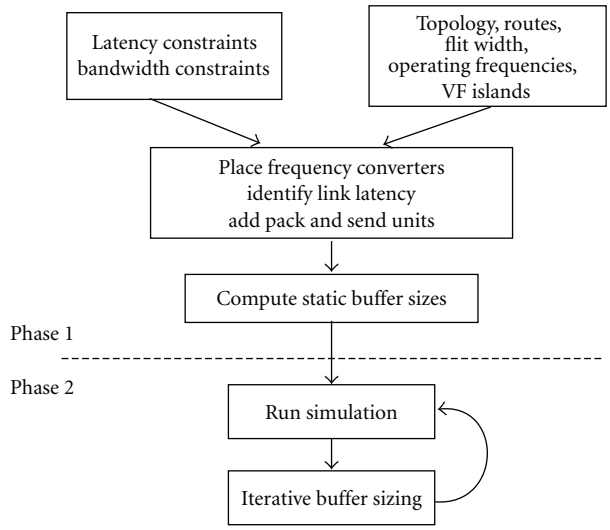


Figure 2: Buffer sizing design approach.

(v) We assume a uniform flit width across the whole network, which is again commonly observed in many NoCs.

(vi) We apply the buffer-sizing algorithm for a single application. In many designs, multiple applications can be run on the same device. The extension of the buffer-sizing method to support multiple application scenarios is similar to the extension of topology synthesis methods to consider multiple applications and has been thoroughly addressed by several researchers before [27]. Hence, we only show the core method applicable for a single application here.

The output of the buffer-sizing algorithm is the number of flit buffers at each input port of the different switches. We only perform the sizing of the switch buffers, and we refer the reader to earlier works on sizing NI buffers that consider the different rates of the cores and the network [25].

The algorithm phases are as follows.

Phase 1 (*Static Sizing*). To achieve full throughput and utilization on all the switches and links, the credit-based flow control mechanism requires a minimum number of buffers that depends on the number of cycles to traverse the links. In an application-specific topology, many parts of the network can have much less than 100% utilization. In this first phase, we formulate mathematical models relating the buffering at a port with the bandwidth utilization of the port and capturing latency constraints of traffic flows. We build a *linear program-* (LP-) based model to minimize the number of buffers at a port based on the utilization at the port and to respect the latency constraints of all flows across the different paths.

Phase 2 (*Simulation-Based Sizing*). In the second phase, we perform simulation of the NoC with the buffering values obtained from Phase 1. There are three important parameters of a traffic flow that significantly affect the congestion behavior: the bandwidth of the flow, the burstiness, and the number of flows overlapping at each link/switch port. While the static sizing mechanism considers the bandwidth of flows to compute utilization, the effects of burstiness and overlapping flows are considered during this second phase. We run simulations and utilize methods to iteratively increase buffers at ports until the bandwidth and latency constraints of all flows are met.

4. Buffer Sizing

4.1. Basic Architecture. In this section, we formulate the problem of buffer sizing.

We represent the communication constraints between the cores using a core graph.

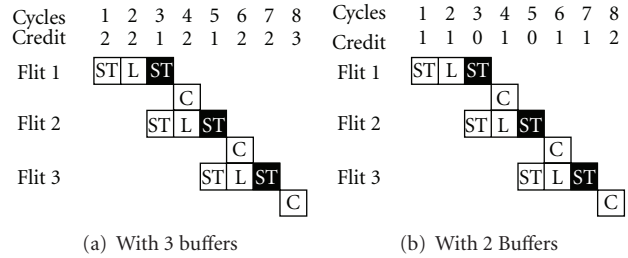
Definition 1. The core graph is a directed graph, $G(V, E)$ with vertex $v_i \in V$ representing the core and the directed edge $e_{i,j} \in E$ connecting vertices v_i and v_j, representing the communication link between the cores. The edge weight, $\text{comm}_{i,j}$, denotes the communication bandwidth between v_i and v_j. The set F represents the set of flows between the cores.

An NoC graph denotes the NoC topology and the capacity of the links in the topology.

Definition 2. The NoC graph is a directed graph, $T(P, Q)$ with vertex $p_i \in P$ representing the switch and the directed edge $q_{i,j} \in Q$ connecting the vertices p_i and p_j representing the link connecting the switches. The edge weight, $\text{bw}_{i,j}$, denotes the link bandwidth or capacity available across p_i and p_j.

Definition 3. Let the set of operating frequencies of various domains (in GHz) be denoted by the set D. Let freqency (i) be a mapping function that maps a chip component to the frequency at which the domain is operating at:

$$\text{frequency (i)} : \{V, P\} \longrightarrow D, \quad i \in V, P \tag{1}$$



(a) With 3 buffers (b) With 2 Buffers

Figure 3: Timing diagram of two different buffer configurations. ST—switch traversal delay, L—link latency, C— credit latency.

When crossing VF islands, converters are required to do the frequency conversion. For this purpose, we use FC units that are basically dual-clocked FIFOs. The size of the FCs is uniform throughout the design. Most FCs incur a delay penalty for traversal, which is typically few cycles of the slow clock [28]. We denote this latency by FC_lat.

The latency of a link is the sum of the latency to traverse the FC and link traversal latency. The link traversal latency is defined by the frequency at which the link is operated. Let unit_len denote the distance in mm a signal can traverse in 1 *ns*. This can be determined based on the design's technology node. Then the latency of a link is given by

$$N_{i,j} = \text{FC\_lat} + \frac{1}{\text{freq}} \times \frac{\text{length}_{i,j}}{\text{unit\_len}} \tag{2}$$

$$\text{freq} = \begin{cases} \text{frequency } (i) & \text{FC near destination,} \\ \text{frequency } (j) & \text{FC near source,} \end{cases} \tag{3}$$

where freq denotes the operating frequency of the link and it depends on where the FC is placed on the link, and $\text{length}_{i,j}$ denotes the length of the link in mm.

The bandwidth or capacity of a link is given by the product of link width and frequency of operation of the link:

$$\text{bw}_{i,j} = \text{freq} \times \text{link\_width}_{i,j}. \tag{4}$$

Definition 4. The links traversed by a flow, f_k, $\forall k \in F$, connecting source s_k and destination d_k is represented by the set path_k.

The utilization $U_{i,j}$ of a link $q_{i,j}$ is the sum of the bandwidths of all the flows using the link divided by the capacity:

$$U_{i,j} = \frac{\sum_{l,m} \text{comm}_{l,m}}{\text{bw}_{i,j}}, \quad \forall l, m, k, \tag{5}$$

$$\text{s.t. } q_{i,j} \in \text{Path}_k,\ s_k = v_l,\ d_k = v_m.$$

We assume a *pack and send* (PS) unit that can be used to better utilize the network resource. A PS unit is a 1-packet-long buffer that holds an entire packet before forwarding it to the downstream buffer. Employing PS units changes the above link utilization $U_{i,j}$.

The NoC architecture assumes a credit-based flow control mechanism. The following is a lemma for the number of buffers required in the downstream switch for credit-based flow control mechanism to support full throughput and utilization [29].

Lemma 5. *For a link with delay N cycles, in credit-based flow control, the number of flit buffers required at the downstream router in order to get 100% throughput is at least* $(2N + 1)$.

The intuitive reasoning for this buffering value is as follows. A flit takes N cycles to reach the next switch and, when it leaves the downstream buffer, the credit takes another N cycles to reach back and it takes one more cycle to process the credit. Thus, the overall time delay between sending a flit and processing the corresponding credit is $(2N + 1)$. During that time, under full utilization, the same number of flits could be sent from the sender switch which needs buffering at the downstream switch.

When the link utilization is less than 100%, the downstream router needs not have $(2N + 1)$ buffers and can be sized according to the utilization. The illustration in Figure 3 shows that buffers in the downstream router can be less than $(2N+1)$. In the example, two setups with downstream router having 3 and 2 buffers and link latency of 1 cycle are shown. The flow is assumed to have a 50% link utilization with the packet comprising of 1 flit. The packets are generated every other cycle, hence having utilization of 50%. In Figure 3(a), the timing diagram for a setup with 3 buffers and the credit counter value (available buffers at downstream router) at each cycle are shown. The same throughput (50%) can be achieved with 2 (lesser than $(2N + 1)$) buffers (Figure 3(b)). However, when the number of buffers is reduced from the ideal values, the packet latencies increase. For example, consider a 4-flit packet in the above scenario with 2 buffers. Since, the buffers are reduced from the ideal, the flits can be sent only every other cycle, and hence the packets have a latency of 7 cycles to be sent from the upstream to the downstream switch. Thus, when reducing the buffer size, we should also consider whether the latency constraints for the flows are met.

Table 1 summarizes the different parameters of the network.

5. Static Buffer Sizing

The latency of a link in the network is defined by the rate at which the link is clocked. Without loss of generality, let us assume $N_{i,j}$ to be the number of cycles needed to traverse the link $q_{i,j}$. Then the minimum buffering required at a port, based on the utilization at the port, is given by

$$\beta_{i,j} \geq \left(2N_{i,j} + 1\right) * U_{i,j}, \tag{6}$$

Table 1: Network parameters.

Parameter	Description
V	Set of IP cores
P	Set of NoC switches
D	Set of frequencies of VF islands
F	Set of flows in the network
FC_lat	Latency of frequency converter
$\text{length}_{i,j}$	length of link in mm
$N_{i,j}$	Link latency in cycles
$U_{i,j}$	Link utilization
$\beta_{i,j}$	Buffer size at link $q_{i,j}$
ps_k	Packet size of flow k
LC_k	Latency constraint of flow k

where $\beta_{i,j}$ represents the buffers statically assigned at the port connecting switches p_i & $p_j \in P$.

Let the latency constraint that needs to be met by a flow f_k, which is obtained as part of the input application specifications, be LC_k. The latency of a flow depends on the size of the buffers along its path and the size of packet (in flits). For the first flit of the packet, the latency is given by the length of the path (hops), and for the consequent flits it is determined by the buffering available at the downstream router. When buffering is less than the ideal value at an input port that is connected to the link $q_{i,j}$, the body (and tail) flits may need to wait at the upstream switch for credits to reach back. This delay for the flits at a link $q_{i,j}$ is given by

$$\frac{\left(2N_{i,j}+1\right)}{\beta_{i,j}} \times (\text{ps}_k - 1), \tag{7}$$

where ps_k denotes the number of flits in a packet.

A packet encounters this delay at that part of the path where the amount of buffering, when compared to the ideal value, is lowest.

Under zero load conditions, the latency constraint on a flow is met if the following constraint is satisfied:

$$\max_{\forall i,j,\ \text{s.t. } q_{i,j} \in \text{Path}_k} \left\{ \frac{\left(2N_{i,j}+1\right)}{\beta_{i,j}} \times (\text{ps}_k - 1) \right\} + H_k \le \text{LC}_k, \tag{8}$$

where H_k denotes the hop count of the flow f_k. The first term on the left-hand side accounts for the maximum delay across the entire path due to the reduced buffering.

The problem of computing the minimum number of buffers required at the different ports to meet the bandwidth and latency constraints can be formulated as a *linear program* (LP) as follows:

$$\begin{aligned} &\min : \sum_{i=1}^{|P|}\sum_{j=1}^{|P|}\beta_{i,j}, \quad i \ne j, \\ &\text{s.t.} \quad \beta_{i,j} \ge \left(2N_{i,j}+1\right) * U_{i,j}, \\ &\max_{\forall i,j,\ \text{s.t. } q_{i,j} \in \text{Path}_k} \left\{ \frac{\left(2N_{i,j}+1\right)}{\beta_{i,j}} \times (\text{ps}_k - 1) \right\} + H_k \le \text{LC}_k \\ &\beta_{i,j} \le \left(2N_{i,j}+1\right), \quad \beta_{i,j} \ge 0, \quad \forall i,j \in P. \end{aligned} \tag{9}$$

The objective function to be minimized is the total buffering used in the switches. The bandwidth and latency constraints, obtained from (6) and (8), form the constraints of the LP. The formulation can be solved quickly and efficiently by any linear/convex program solver, such as the lp_solve [30]. Since the resulting buffer values by solving the LP can be fractional, we round up the value to the next integer. In fact, we could have formulated the above equations as an integer linear program (ILP), where we can force the buffer values to be integers. However, as solving the ILP formulation has exponential time complexity, it will be unfeasible to apply in practice. Hence, we use the heuristic of LP formulation with the rounding scheme.

6. Simulation-Based Buffer Sizing

After Phase 1, we perform simulation of the NoC using the computed buffer sizes and injecting packets to model the application communication patterns that are taken as inputs. The simulation-oriented approach is iterative, where buffers are added and simulations performed iteratively, until all the flows meet the bandwidth and latency requirements. To perform the sizing, we propose two strategies called as *uniform increment* and *flow-based increment*.

In the first strategy, buffers at all the ports are incremented iteratively by a small step. The buffer increment at a port depends on the burstiness of the flows and the number of flows contending for the same port. During simulations, the burst sizes of the flows at each port are tracked and the average burstiness of a flow is identified. The burstiness of flow f_k is denoted as B_k.

We use the following term to increment the buffers at a port connected to the link $q_{i,j}$ at each iteration:

$$\frac{\sum_{\forall f_k \text{s.t.} q_{i,j} \in \text{Path}_k} \alpha * B_k}{\max_{\forall f_k} B_k * |F|}, \tag{10}$$

where α is the parameter that is increased from 0 in small steps with each simulation iteration. Intuitively, the increment captures the burstiness of flows, scaled by the maximum burstiness of any flow of the application and the summation captures the number of contending flows, normalized to the total number of flows in the application. Thus, ports that have many contending flows, or flows with

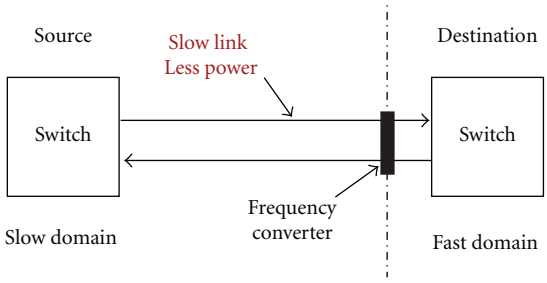


Figure 4: Placed closer to the fast clock domain.

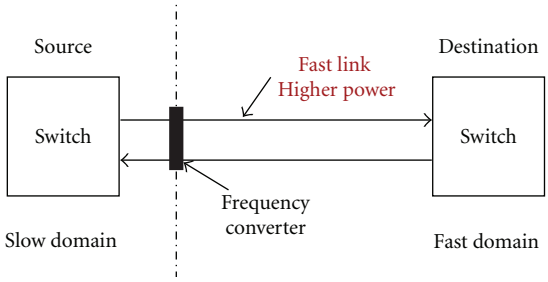


Figure 5: Placed closer to the slow clock domain.

large burst sizes get a larger increment at each iteration. The value of α is set experimentally. A very low value would result in lot more solutions being explored, while requiring more numbers of simulations.

In the second strategy, we track the flows that violate the bandwidth or latency constraints, and only the buffers along the path of such flows are incremented. This approach is an optimization of the previous strategy, giving finer control to access individual flows. For faster results, the *uniform increment* scheme can be used, while for better results, the *flow-based increment* scheme can be used. Thus, the two schemes allow a tradeoff between having fine control over the buffer allocation and minimizing the number of simulation runs required. In Section 9, the results of the two proposed strategies are discussed.

7. Placement of Frequency Converters

In this section the proposed buffer sizing scheme is extended to designs with multiple VF islands. We assume that the architecture characteristics include the VF island information.

FCs are used when crossing clock domains. Several kinds of FCs are proposed in the literature, but the most commonly used one is a dual clock FIFO (DCFIFO). A DCFIFO consists of a series of shift registers connected together that shift the input data from one domain to the output to the other domain. The input is clocked by one clock and output is clocked by the other, and the data is stored in the intermediate registers till the output is ready to consume it.

From the input architectural specification, the clock domains are identified and DCFIFOs are added along the links that cross the domains. Placement of the DCFIFOs is a critical design problem as they affect the power consumption and buffering required, hence, the performance of the interconnect.

7.1. Frequency Converters near Fast Domain. In this setup, the frequency converters are placed close to the fast clock domain, as shown in Figure 4. By placing the converters near the fast clock, the link is clocked by the slower clock. From (2), it is evident that when the link operates at a lesser frequency, the latency increases. But, since the operating frequency is lesser, the power consumed by the link is lesser, as $P \propto f$. The increased latency of the link will demand the downstream router to have more buffering (according to Lemma 5).

7.2. Frequency Converters near Slow Domain. In this setup, the FCs are placed closer to the slow clock domain, thereby clocking the link by the faster clock, as shown in Figure 5. This makes the link to operate at higher speed, and hence the latency is lesser (2), thereby reducing the buffering needed at the downstream router. But the higher operating frequency makes the link consumes more power.

This tradeoff between power consumed by the link and power consumed because of extra buffering can be explored to choose a specific design point that meets the system requirements. The effects of placement of FCs is analyzed, and the results are discussed in Section 9.3.

8. Handling Bubbles

In multiclock designs, inherently lot of empty flits are generated because of difference in the operating speed of the network components. Network flows traversing from a faster to slower frequency domain will incur higher latency, and enough buffering must be provided to meet the design constraints.

On the other hand, network flows traversing from a slower to a faster clock domain create *bubbles* in the network. Since destination is faster than the source, empty flits (bubbles) are generated in the network which underutilize the network resources. These bubbles must be reduced in order to better utilize the resources, and employing *pack and send* (PS) units can help in reducing them. Bubbles in a flow hold the buffers along the path for a long period unnecessarily, and hence other flows contending for the links are delayed. Waiting for the entire packet to arrive before the flow requests for the downstream buffer allows other flows contending for the link to proceed. Pack and send unit holds the flits temporarily till the entire packet is formed, and then it is forwarded to the downstream router. Hence a PS unit contains buffers to hold an entire packet. Section 9.4 discusses the results obtained when employing PS units.

9. Results

We consider a 26-core multimedia benchmark for a detailed study of the buffer-sizing algorithm. In Section 9.6, we show the application of the algorithm to a variety of benchmarks. The system includes ARM, DSPcores, memory banks, DMA engine, and several peripheral devices [31].

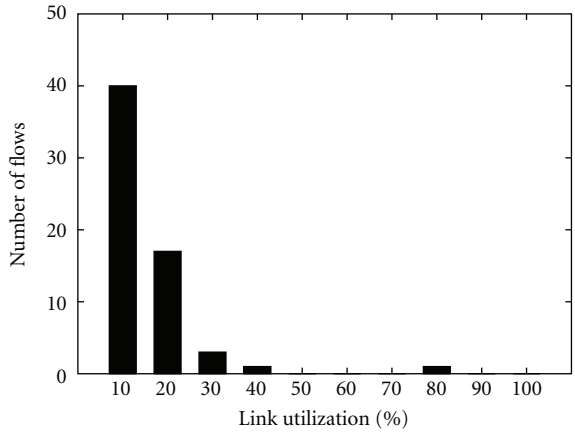


Figure 6: Histogram of link utilization.

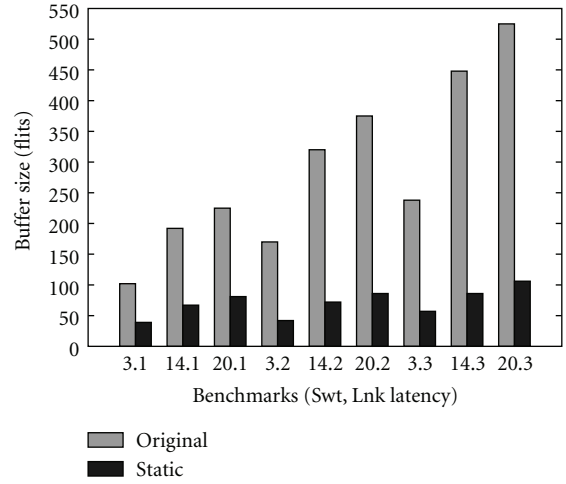


Figure 7: Comparison of buffering schemes.

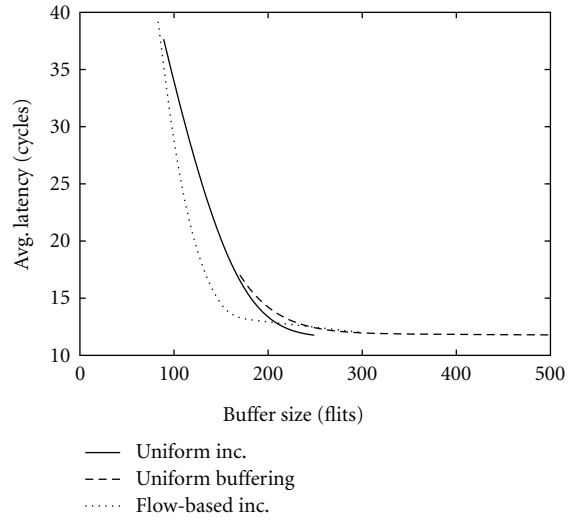


Figure 8: Latency versus buffering.

We consider benchmarks with custom topologies, but the algorithm is extendable to regular topologies too. Some of the benchmarks involve topologies with large number of switches (26 cores and 20 switches) which is similar to regular topologies. For the initial study to validate the buffer-sizing algorithm we consider no voltage and frequency domains for the network. The entire interconnect operates at a single frequency.

We use existing tools to synthesize NoC topologies with different number of switches [9]. The flit width of the NoC is set to 32 bits. For each topology synthesized, we perform simulations with very large (40 buffers at each port) buffer sizes and set the frequency of operation to a value where the application constraints are met during simulations. This ensures a working solution with very large buffering values, and the objective is to reduce it to much smaller values. We chose 3 different topologies with few to large number of switches (3, 14, and 20 switches) for the study. The number of cycles needed to traverse the links between the switches depend on the floor plan of the design, the technology library characteristics, and the operating frequencies of the components. For this study, we consider 3 different configurations for the link delay: 1, 2, and 3 cycles across all the links. This allows us to show the effect of link delays on the efficiency of the proposed methods. We denote the topology points by *the number of switches, link delay*.

9.1. Effects of Static Buffer Sizing. In the static sizing, we reduce the buffering at any port when the utilization at the port is less than 100%. We observed that in this and most other embedded benchmarks, the link utilization is very nonuniform, and only some bottleneck links have high utilization while others have much lower values. For example, in Figure 6, we show the utilization of the different links for a 20-switch benchmark. It can be seen that there are a lot of flows that have a very low utilization of less than 10%. Hence, utilization-based sizing can lead to a large reduction in buffering. Please note that, due to protocol overhead, we could not achieve 100% utilization on any link and needed to operate the network at a slightly higher frequency than the minimum needed to meet the bandwidth constraints.

We show the effect of static sizing in Figure 7. We compare the results with an original uniform buffering, where the minimum number of buffers for full utilization is used at all the ports. We can see that the proposed scheme results in large reduction in buffering requirements.

9.2. Effects of Simulation-Based Buffer Sizing. In this subsection, we compare the application of the two strategies, *uniform increment* and *flow-based increment*. For comparisons we also developed a standard *uniform buffering* strategy, where all ports have the same number of buffers. This is set to the minimum value at which the latency and bandwidth constraints of all the flows are met during simulations. We consider 3 different burst sizes for all the traffic flows: 4, 8, and 16 byte bursts.

Figure 8 shows the average latency across all the flows for a spectrum of buffer budgets for a benchmark with 3, 3 design and a burst size of 16 for all the flows. Depending on

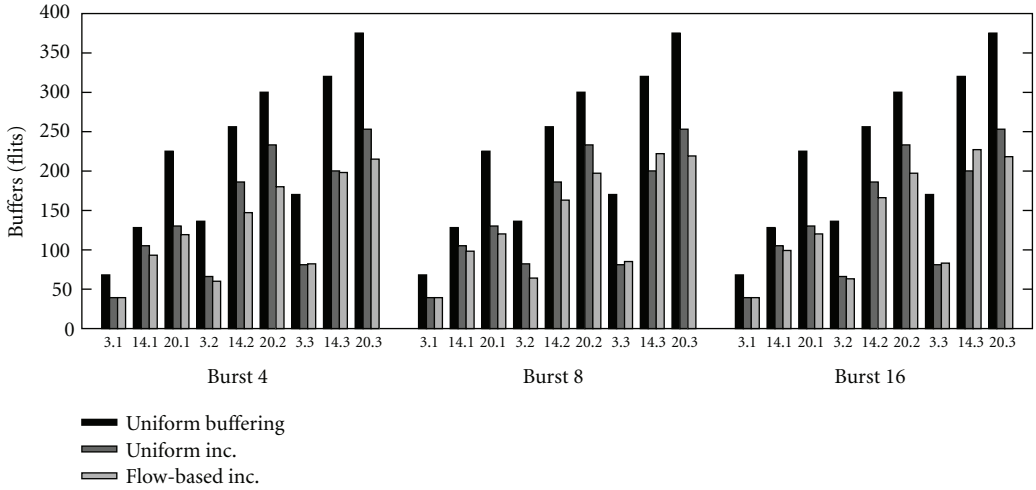


Figure 9: Comparison of different buffer-sizing strategies.

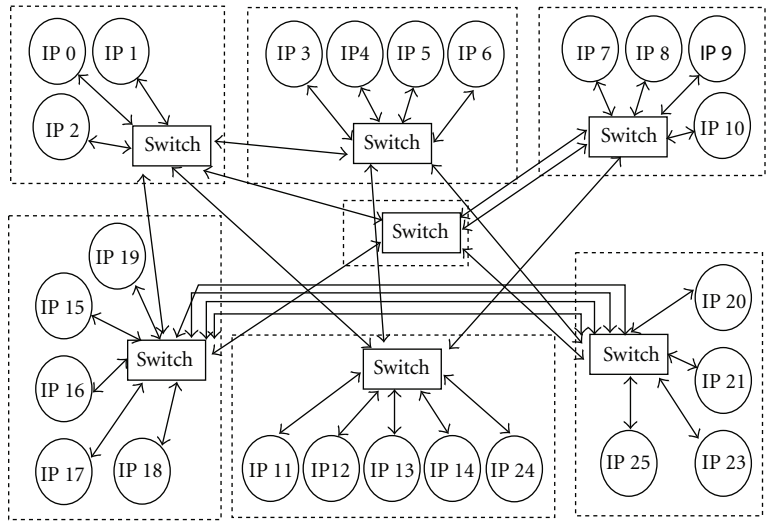


Figure 10: Example topology.

the tightness of the average latency constraint, the amount of buffering achieved by the different schemes varies. When loose latency constraint is used, the proposed strategies can provide large reduction in buffering requirements. For a very tight average latency constraint, all schemes perform similarly and the simulation-based sizing methods do not give any savings. Depending on the constraint, the algorithm can give the best buffer setting.

We set the average latency constraint to 50 cycles and perform buffer sizing. The buffer budget in the graphs (Figure 9) denotes the minimum buffer budget at which the simulation is stable and all constraints are met. We find that both *uniform increment* and *flow-based increment* strategies perform significantly better when compared to the standard *uniform buffering* strategy. The minimum buffer budget in the case of *uniform increment* is higher than *flow-based increment* in most cases, because the increment to the buffers is uniform on all the ports, and thus the addition of buffers is at a coarse level. Moreover, as expected, the savings are more pronounced when the traffic is more bursty and/or the link delays are larger. The results show that there is a 42% reduction in the buffer budgets for the switches, on an average. Based on the models from [6], this translates to around 35% reduction in the overall power consumption of the NoC switches.

9.3. Effect of Placement of FCs. We implemented the above buffer-sizing algorithm to a benchmark with multiple clock domains. The benchmark consisted of the entire topology clustered into different VF islands with operating frequencies ranging from 200 to 600 MHz. An example input topology is illustrated in Figure 10. One of the above proposed strategies, *uniform increment,* was used in this study. The main goal was to analyze the impact of placement of the FCs. Figure 11 shows the reduction in the buffer budgets compared to the standard uniform buffering scheme. Also the results show

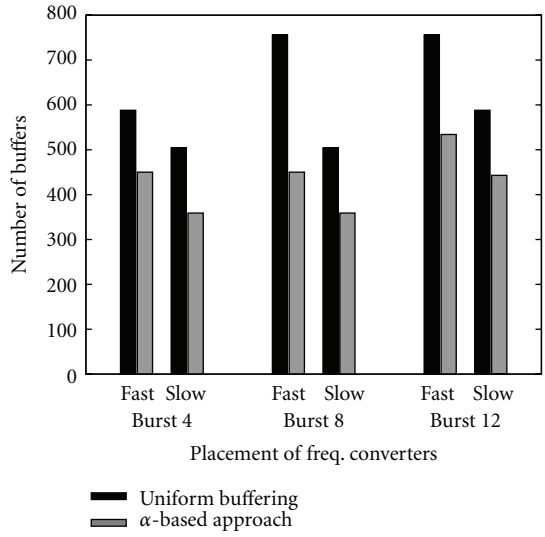


Figure 11: Placements of FCs.

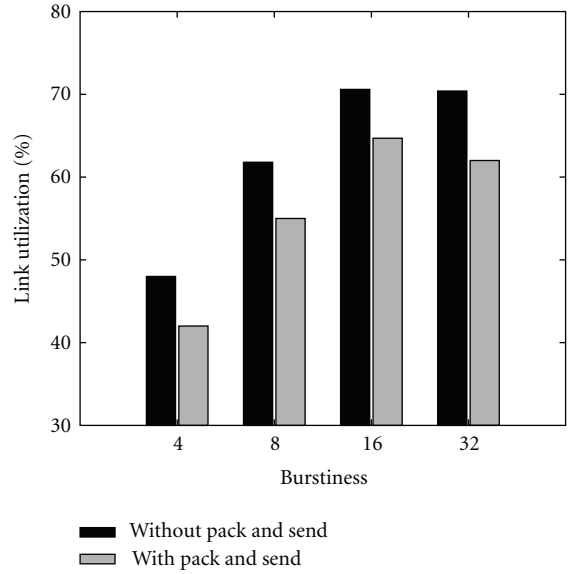


Figure 12: Pack and send.

that the buffering required when placing the FCs closer to the slow domain is lesser than the buffering required when placing it close to fast domain. This however affects the link power, as the links are clocked higher and hence the link power is more. With smaller transistor sizes, the link power is as significant as the logic power, and hence the power consumed by the links must also be taken into account.

9.4. Impact of Pack and Send. For studying the effect of pack and send unit, the benchmark with multiple clock domains was considered. Since the pack and send unit has an effect only across links where the utilization is high, we scaled the frequencies to a range of 500–900 MHz and the burstiness was increased. For the study the proposed buffer-sizing algorithm was used with PS units placed along links going from slow to fast clock domain. The effect on buffering with and without pack and send unit was analyzed. Though the PS units require extra buffers to hold a packet, results showed that the total buffering required (including the PS buffers) remains the same. However the link utilization reduces when using PS units. Since there is lesser contention among the flows while using PS units, there is a direct impact in terms of reduction in the buffering required at the switches. However, the total buffering required remains the same as the decrease in buffering at the switches is compensated by the extra buffers required in the PS units. Figure 12 shows the overall average link utilization with and without PS unit. This shows that the PS unit helps in better utilizing the resources, and this reduction in link utilization can be directly converted to power savings at lower technology nodes.

9.5. Run Time of the Methods. Since Phase 1 uses LP formulation and not ILP, the solution is tractable and is obtained quickly. Phase 1 of the algorithm finished in few seconds for all the benchmarks on a 2.66 GHz Linux Machine. Among the two strategies presented in Phase 2, there is a tradeoff between the running time of simulations and the granularity of controlling buffer sizes. Previous sections show that the fine control of buffer sizes for *flow based increment* approach helps in achieving a lower budget. But the running time to converge to a budget is more. Each simulation run can take from 20 minutes to few hours, depending on how long a trace needs to be simulated. The *uniform increment* approach took 20–60 simulation runs, while the *flow-based increment* approach took 50–100 runs. Thus, the design time required could be significantly higher for the *flow-based increment* strategy. The designer has the choice of the strategy and the step used for the α values and can make a tradeoff between buffer reduction and design time.

9.6. Experiments on Other Benchmarks. To show the generality of the method, we consider 4 other SoC benchmarks: *D_36_4, D_36_6, D_36_8, and D_35*. The first 3 benchmarks have 36 cores, with each communicating to 4, 6, and 8 other cores respectively. The last benchmark has 35 cores and models bottleneck traffic communication, such as memory controller traffic. On average, the proposed schemes result in 38% reduction in buffer sizes when compared to the standard uniform sizing schemes.

9.7. Comparison with Theoretical Models. To show that the proposed buffer-sizing methodology works for well-behaved traffic also, we compare our results with queuing theory-based model proposed in [10]. Figure 13 shows that the proposed model is able to achieve buffer budgets close to the theoretical limit proposed in the paper.

10. Conclusion

As buffers account for a large area, power overhead in NoCs, reducing the amount of buffering is an important problem. Buffer sizing is closely tied to dynamic congestion effects that can be observed only during simulations. Towards this end,

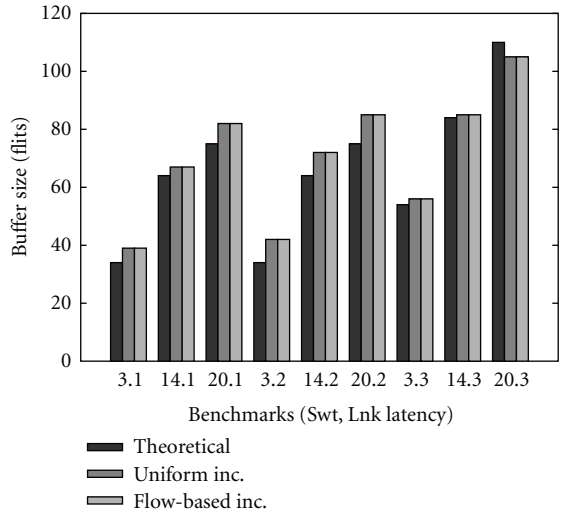


Figure 13: Comparison with theoretical models.

in this paper, we present a two-phase algorithm for buffer sizing. Our approach considers the nonuniformity in the utilization of the different parts of the network and uses a simulation-based iterative mechanism. Our results show a large reduction in buffering required (42%) when compared to standard buffering approach. The proposed buffer-sizing algorithm was extended to designs with multiple clock domains. Results showed a significant reduction in buffer budget. We also analyzed the effect of placement of FCs on the overall buffer budget. Also the use of PS units to handle bubbles in the network was studied, and results showed that employing them utilize resources better.

Future Work

In the proposed buffer-sizing approach, the second phase that involves simulations is the step that consumes a significant portion of the total run time. We can use intuitive approaches to reduce the simulation time for that step. But this is beyond the scope of this work and can be a part of an extension to this work.

Acknowledgments

The authors would like to acknowledge the ARTIST-DESIGN Network of Excellence. This work has also been supported by the project *NaNoC* (project label 248972) which is (partly) funded by the European Commission within the Research Programme FP7.

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] G. D. Micheli and L. Benini, *Networks on Chips: Technology and Tools*, Morgan Kaufmann, 2006.

[3] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet switched interconnections," in *Proceedings of the Design, Automation and Test in Europe*, pp. 250–256, 2000.

[4] R. W. Apperson, Z. Yu, M. J. Meeuwsen, T. Mohsenin, and B. M. Baas, "A scalable dual-clock FIFO for data transfers between arbitrary and haltable clock domains," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 15, no. 10, pp. 1125–1134, 2007.

[5] A. Strano, D. Ludovici, and D. Bertozzi, "A library of dualclock fifos for cost-effective and flexible mpsoc design," in *Proceedings of the International Conference on Embedded Computer Systems (SAMOS '10)*, pp. 20–27, 2010.

[6] 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.

[7] J. Hu and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," in *Proceedings of the Design, Automation and Test in Europe*, 2004.

[8] A. Hansson et al., "A unified approach to mapping and routing in a combined guaranteed service and best-effort Network-on-Chip architecture," in *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, 2005.

[9] S. Murali, P. Meloni, F. Angiolini et al., "Designing application-specific networks on chips with floorplan information," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD '06)*, pp. 355–362, November 2006.

[10] J. Hu and R. Marculescu, "Application-specific buffer space allocation for networks-on-chip router design," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers (ICCAD '04)*, pp. 354–361, November 2004.

[11] Y. Yin and S. Chen, "An application-specific buffer allocation algorithm for network-on-chip," in *Proceedings of the 8th IEEE International Conference on ASIC (ASICON '09)*, pp. 439–442, October 2009.

[12] W. H. Ho and T. M. Pinkston, "A methodology for designing efficient On-Chip interconnects on well-behaved communication patterns," in *Proceedings of the International Symposium on High Performance Computer Architecture*, 2003.

[13] J. Kong, J. Choi, L. Choi, and S. W. Chung, "Low-cost application-aware DVFS for multi-core architecture," in *Proceedings of the 3rd International Conference on Convergence and Hybrid Information Technology (ICCIT '08)*, pp. 106–111, November 2008.

[14] T. Kolpe, A. Zhai, and S. Sapatnekar, "Enabling improved power management in multicore processors through clustered dvfs," in *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE '11)*, pp. 1–6, March 2011.

[15] S. Garg, D. Marculescu, R. Marculescu, and U. Ogras, "Technology-driven limits on DVFS controllability of multiple voltage-frequency island designs: a system-level perspective," in *Proceedings of the 46th ACM/IEEE Design Automation Conference (DAC '09)*, pp. 818–821, July 2009.

[16] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli, "NoC topology synthesis for supporting shutdown of voltage islands in SoCs," in *Proceedings of the 46th ACM/IEEE Design Automation Conference (DAC '09)*, pp. 822–825, July 2009.

[17] L. Guang, E. Nigussie, and H. Tenhunen, "Run-time communication bypassing for energy-efficient, low-latency per-core dvfs on network-on-chip," in *Proceedings of the SOC Conference*, pp. 481–486, September 2010.

[18] T. Jain, P. Gratz, A. Sprintson, and G. Choi, "Asynchronous bypass channels: improving performance for multisynchronous nocs," in *Proceedings of the 4th ACM/IEEE International Symposium on Networks-on-Chip (NOCS '10)*, pp. 51–58, May 2010.

[19] W. Liwei, C. Yang, L. Xiaohui, and Z. Xiaohu, "Application specific buffer allocation for wormhole routing Networkson-Chip," *Network on Chip Architectures*, pp. 37–42, 2008.

[20] M. A. Al Faruque and J. Henkel, "Minimizing virtual channel buffer for routers in on-chip communication architectures," in *Proceedings of the Design, Automation and Test in Europe (DATE '08)*, pp. 1238–1243, March 2008.

[21] S. Yin, L. Liu, and S. Wei, "Optimizing buffer usage for networks-on-chip design," in *Proceedings of the International Conference on Communications, Circuits and Systems (ICCCAS '09)*, pp. 981–985, July 2009.

[22] N. Alzeidi, M. Ould-Khaoua, L. M. Mackenzie, and A. Khonsari, "Performance analysis of adaptively-routed wormhole-switched networks with finite buffers," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 38–43, June 2007.

[23] S. Foroutan, Y. Thonnart, R. Hersemeule, and A. Jerraya, "An analytical method for evaluating network-on-chip performance," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '10)*, pp. 1629–1632, March 2010.

[24] S. Manolache, P. Eles, and Z. Peng, "Buffer space optimisation with communication synthesis and traffic shaping for NoCs," in *Proceedings of the Design, Automation and Test in Europe (DATE '06)*, pp. 718–723, March 2006.

[25] 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, October 2006.

[26] A. B. Kahng, B. Lin, K. Samadi, and R. S. Ramanujam, "Trace-driven optimization of networks-on-chip configurations," in *Proceedings of the 47th Design Automation Conference (DAC '10)*, pp. 437–442, June 2010.

[27] 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 Design, Automation and Test in Europe (DATE '06)*, pp. 118–123, March 2006.

[28] E. Beigne and P. Vivet, "Design of On-chip and Off-chip interfaces for a GALS NoC architecture," in *Proceedings of the International Symposium on Asynchronous Circuits and Systems (ASYNC '06)*, pp. 172–183, IEEE Computer Society, Washington, DC, USA, 2006.

[29] W. J. Dally and B. Towels, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2003.

[30] Lp solve, http://lpsolve.sourceforge.net/.

[31] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli, "SunFloor 3D: a tool for networks on chip topology synthesis for 3D systems on chips," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '09)*, pp. 9–14, April 2009.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 728191, 14 pages
doi:10.1155/2012/728191

Research Article

Status Data and Communication Aspects in Dynamically Clustered Network-on-Chip Monitoring

Ville Rantala,[1,2] Pasi Liljeberg,[2] and Juha Plosila[2]

[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*

Correspondence should be addressed to Ville Rantala, vttran@utu.fi

Received 1 July 2011; Revised 28 October 2011; Accepted 1 November 2011

Academic Editor: Sao-Jie Chen

Monitoring and diagnostic systems are required in modern Network-on-Chip implementations to assure high performance and reliability. A dynamically clustered NoC monitoring structure for traffic and fault monitoring is presented. It is a distributed monitoring approach which does not require any centralized control. Essential issues concerning status data diffusion, processing, and format are simulated and analyzed. The monitor communication and placement are also discussed. The results show that the presented monitoring structure can be used to improve the performance of an NoC. Even a small adjustment of parameters, for example, considering monitoring data format or monitoring placing, can have significant influence to the overall performance of the NoC. The analysis shows that the monitoring system should be carefully designed in terms of data diffusion and routing and monitoring algorithms to obtain the potential performance improvement.

1. Introduction

Network-on-Chip (NoC) [1] based systems can be complex structures of tens or hundreds of processors, IP cores (Intellectual Property), and memory modules. These systems require versatile monitoring systems to handle the functionality and maintain their performance. While interconnect starts to increasingly dominate the overall performance, the monitoring systems become even more significant part of modern NoC systems. An advantage of the NoC paradigm is its scalability [2]. A fully scalable NoC architecture should have a scalable monitoring system which can be easily tailored to different NoC implementations and whose performance does not degrade when the size of the system increases.

NoC monitoring systems are typically designed for two purposes: system diagnostics and traffic management. The former aims to improve the reliability and performance of the computational parts while the latter concentrates to the same issues in the communication resources. The traffic management should take into account the status of the network resources including their load as well as their possible faultiness.

A technology-independent framework of the dynamically clustered monitoring structure for NoC is presented and its features are discussed in this paper. A SystemC-based NoC simulation model is also presented. The dynamically clustered monitoring structure is fully scalable monitoring system which is primarily aimed for traffic management purposes. This paper is organized as follows. NoC traffic management and different monitoring structures are discussed in Section 2. Section 3 gives a brief description about related works and writers' contributions. The SystemC-based NoC simulation model is presented in Section 4. The dynamically clustered monitoring structure is presented in Section 5, and its features are discussed and analyzed in Sections 6, 7, 8, and 9. Potential modifications to the monitoring structure are presented in Section 10. General discussion, future works, and conclusions are gathered to Section 11.

2. Monitoring in NoC

Traffic management is implemented into NoC to maintain network performance and functionality in the case of faults

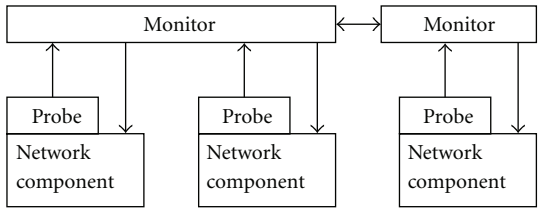


Figure 1: Network components.

and under high traffic load. Typically there is a monitoring system to collect traffic information from the network and an adaptive routing algorithm which adapts its operation when the conditions in the network change.

Two types of information are required in traffic management: traffic status in the network and locations of faults in the network. Traffic status can be observed from different network components: router activity, router FIFO occupancy, or link utilization, for instance. Fault information can cover the faultiness of different network components: routers or links, for instance. A network component is considered as faulty when it does not work as it should by its specification. The network components have to have mechanisms to detect these faults [3]. There are several methods to detect faults. For instance, faulty links can be detected using methods which are based on usage of spare resources or error control coding [4, 5].

2.1. NoC Monitoring Structures. The components of a monitoring system are monitors and probes. The probes are attached to a network components (e.g., routers, links, or network interfaces) to observe the functionality of a network component (see Figure 1). The observed data is delivered from probes to a monitor which can collect statistics or process the data to a format which can be utilized in different reconfiguration tasks. The processed monitoring data is finally delivered to the components which use it to reconfigure their operation. A monitoring structure can have dedicated resources for communication between probes and monitors, or it can share the resources of the data network. In our research we focus on shared-resource structures which require less additional resources than dedicated structures. We have also paid attention to dedicated resources for serial monitoring communication between monitors. In shared-resource structures nonintrusive operation of the monitoring system is a significant issue while in the serial monitoring communication the delays are crucial in terms of usefulness of the monitoring data.

Monitoring structure defines the number and type of monitors and probes, their placing, connections, and tasks. A centralized monitoring structure has one central monitor and several probes that observe the data and deliver it to the monitor. In centralized structure the central monitor has complete overall knowledge of the network but it causes significant amount of monitoring-related traffic in the network. A clustered monitoring structure has a few cluster monitors and several probes. The network is divided into subnetworks, clusters, each of them having a cluster monitor and several probes. The complete network knowledge can be reached using intercluster communication but most of the tasks can be executed inside a cluster. However, a clustered structure still causes a considerable amount of monitoring traffic [6].

In an NoC the data is typically transferred as packets which have a destination address. Routers forward these packets based on this address and the applied routing algorithm [7]. The NoC monitoring systems which use shared communication resources transfer the network status data using monitoring packets. When centralized or clustered monitoring structures are used, these packets have to be routed from probes to a monitor and from the monitor to the routers. Centralized control has its strengths and it is required for several tasks. However to optimize performance some of the traffic management tasks could be executed with simpler distributed, or dynamically clustered, monitoring structure to decrease the load of the centralized control system [6].

3. Related Work

NoC monitoring systems have been presented in several papers. A dedicated control network is used in a centralized operating system controlled NoC [8]. Dedicated resources are also used in [9] where dedicated embedded processors are used to monitor FIFO occupancies and transfer latencies. A monitoring system for Æthereal [10] is presented in [11]. This system monitors transactions in a network and can be configured to shared or dedicated communication resources. A congestion control system, which is based on monitoring the link utilization, is presented in [12]. All these systems include a centralized monitoring unit which collects the observed data and controls the system operation.

Clustered monitoring structures have been discussed in several papers. A monitoring system to collect error, run-time, and functional information from routers and network interfaces (NI) is presented in [13]. A traffic shaping mechanism with router monitoring is presented in [14]. In these two implementations there can be multiple central monitoring units, cluster monitors, so that the system is clustered.

These NoC monitoring structures are nonscalable or partly scalable. Our goal is to develop clustered monitoring towards yet finer granularity and better scalability. A scalable monitoring with regional congestion awareness is represented in [15]. It is aimed to balance the workload in the network based on the amount of congestion.

3.1. Contribution. Our research focuses on a scalable NoC monitoring structures where the knowledge about network conditions is spread widely enough over the network. There are two main factors taken into account while designing our NoC architecture. First, the structure should be not only aware of traffic but also aware of network faults so that network-level fault tolerance can be actively maintained during routing. Second, the structure should also be fully scalable to any size of mesh. All the probes and monitors

are identical, and they work autonomously without any centralized control. The presented ideas can be adapted to different kind of NoC topologies but, due to its popularity, we have decided to concentrate on the mesh topology.

Our dynamically clustered Network-on-Chip is previously discussed and analyzed in [6, 16]. The study in [6] includes extensive introduction and related work sections and presents the proposed dynamically clustered Network-on-Chip architecture. The architecture is simulated and analyzed on theoretical concept level including analysis of monitoring traffic overhead, status data diffusion, and cost of monitoring system. Brief performance analysis on transaction level is also presented in that paper.

Our in-house NoC simulation model has been presented and different status update intervals in the dynamically clustered NoC has been discussed and analyzed in [16].

This paper includes broader and more in-detail presentation of the in-house NoC simulation model (see Section 4). The status data diffusion analysis, originally presented in [6], is extended from theoretical concept level to transaction level using the presented NoC simulation model. In addition, issues concerning network status data format, monitoring communication protocols, and monitor mapping are studied in this paper. The NoC architecture is studied as a technology-independent framework, and therefore the analysis is mostly based on comparison of different features, not on pure analysis of absolute performance values.

4. Simulation Environment

The proposed DCM structure and its features are simulated and analyzed using a *SystemC*- and *TLM 1.0*-based NoC simulation model. The cycle-accurate NoC simulation model is designed for the analysis of different mesh-shaped NoCs. The NoC simulation model includes implementations of a router, a link, a monitor, and a general core. In real implementations the cores include processors and memories but in our NoC simulation model they operate as the senders and receivers of data following some traffic pattern. The NoC simulation model also includes structures to tie the components together and to model data packets. These packets can be considered as flits. In this paper each packet consists of a flit.

The analysis, presented in this paper, is performed using NoC with 64 cores and routers arranged to rows and columns of eight. Two-level traffic pattern has been used. The NoC simulation model is widely customizable which enables the analysis of several different design aspects.

4.1. Simulation. A simulation mechanism of the NoC simulation model is able to execute transient analysis where the cores are sending packets following to a specific traffic pattern and key figures are documented during the simulation. These figures include the numbers of sent and received packets (including the data packets and the monitoring packets, see Section 6), the transfer delay, and the number of dropped packets. The simulation durations can be customized, and it is possible to run simulation in multiple NoCs simultaneously with identical simulation parameters. In this case, the results are represented as averages from the simultaneously running NoCs. The amount of traffic during the simulation can be adjusted with the traffic pattern. There is also an option to put link faults in the network. This feature enables the fault tolerance analysis of NoC. All the network faults are modeled using only the link faults; for example, a faulty router can be illustrated with a bunch of faulty links. Faults can be put in the system randomly or manually so that modeling of larger uniform fault areas is also possible.

4.2. Router. The router model is designed for mesh networks and so it has five ports, four for traffic to and from the neighboring routers and one for traffic to and from the local core. There is a small FIFO buffer in each input port and a centralized FIFO for packets which cannot be routed at the first attempt but can be rerouted later. If the routing fails constantly, the packet is dropped and the router should notify the sender when dropping a packet. The reporting feature is not implemented at this point. The packet dropping typically happens in severe situations where routing is inhibited permanently due to permanently faulty network resources making destination unreachable. Sizes of the FIFO buffers are customizable. The router model includes several different routing algorithms. The used algorithm (see Section 5.1) can be chosen with the simulation parameters.

4.3. Monitor. The monitors are used to observe the functionality and state of the system. The monitor component in our NoC simulation model includes both a probe to collect the monitoring data as well as the monitor to process the collected data. The monitor component can be also configured to act only as a probe or a monitor. This is useful, for instance, when analyzing centralized monitoring structures [6]. Monitoring algorithms are discussed in Section 6. The monitors communicate with each other using shared or dedicated resources. The NoC simulation model includes both implementations. When shared resources are used, the monitors send packets in the data NoC. However, when dedicated resources are used, the amount of resources is limited and therefore the serial communication protocol is utilized.

4.4. Link. The model of a link in the NoC simulation model is unidirectional so that they are used in bunches of two in between two routers, one in each direction. The link is a simple component which forwards the incoming packets. A link can be set on usable or unusable state to model faults in the network.

4.5. Traffic Patterns and Fault Injection. The NoC simulation model has three traffic patterns to be used in analysis. A traffic pattern has two parameters, one defines the amount of sent data during a unit time while another defines how the destination cores are determined. The simplest pattern is a fully random traffic pattern which randomizes the packet destinations among all the cores in the network. A weighted

random traffic pattern is adjusted so that one-third of traffic is between neighboring cores, another third between neighbor's neighbors, and the last part between all the other cores in the network. This pattern roughly imitates a traffic pattern in a real NoC implementation.

The third implemented traffic pattern is a two-level pattern which includes uniform random traffic and varying hot spots each of which sends a relatively large number of packets to a single receiver during a certain time interval. A relatively small number of cores operate as hot spots simultaneously and send packets to a statically chosen receiver cores. At the same time, other cores are sending relatively smaller amount of traffic to random destinations. This two-level traffic pattern imitates real applications where most of the traffic takes place between certain cores at a time. It is aimed for even more realistic performance simulations. The simulations, presented in this paper, are done using this two-level traffic pattern.

During simulation the network links can be set faulty. In Network-on-Chip simulations the fault information can be simplified by using only the information on faulty links and representing other faulty components by marking the links around these components to be faulty. In our simulation framework the number of faults is defined by the user and the simulator places the faults randomly in the network. The simulation is executed several times with different fault patterns and the results are averaged from the original simulation results. This procedure gives overall insight of the system's operation when parts of the network are faulty.

5. Dynamically Clustered Monitoring Structure

Dynamically clustered monitoring (*DCM*) can be considered distributed monitoring because it does not require any centralized control. There is a simple monitor and a probe attached to each router in the network. The used mesh topology is illustrated in Figure 2. Centralized control is not required but the monitors exchange information autonomously with each other.

Each router has a dynamic cluster around itself from where the router receives the data it needs for traffic management. There are no fixed cluster borders as there is in traditional clustered networks and a router can belong to several different dynamic clusters. A dynamic cluster is the area around a single router of which the router has knowledge and to where the router shares its own status. Router's own status is delivered to all the routers in this cluster area. The delivery of router status is called as status data diffusion. The dynamic clusters of different routers overlap with each other. The simplest dynamic cluster includes 4 closest neighbors of a router but it can be expanded to neighbors' neighbors and so on. A system which uses DCM for traffic management could have, for instance, operating system level control for tasks that need complete knowledge of the system. When traffic management is implemented with a DCM structure, the load of the network can be optimized.

There are several issues which affect the functionality of a monitoring system. The used simulation environment

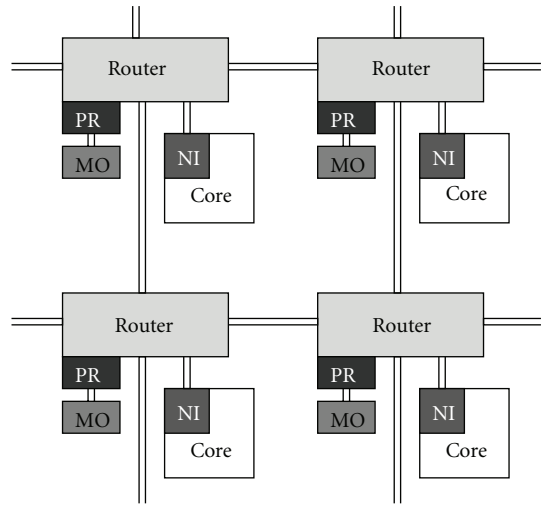


Figure 2: Network topology showing the connections between routers, networks interfaces (NI), monitors (MO), probes (PR), and cores in a part of mesh shaped Network-on-Chip.

is presented in Section 4. The monitoring communication is discussed in Section 6 while Section 7 concentrates on the diffusion and calculation of network status data. The different formats of network status data are presented and analyzed in Section 8.

5.1. Routing Algorithms in Dynamically Clustered NoCs. The NoC simulation model utilizes an adaptive routing algorithm [7]. The algorithm determines the routing direction among the eight candidates which include four main directions (north, south, east, and west) and the intermediate directions (e.g., northwest and southeast). These directions are illustrated in Figure 3. The algorithm chooses an output port to be used among the actual routing direction and its nearest neighbor directions. The decision is based on the traffic status values and the link statuses in potential directions. A packet which cannot be delivered is put back in the router's memory and rerouted. A packet lifetime is also utilized to prevent undeliverable packets from blocking the network. To prevent congestion the packets are not sent in directions where receivers' buffers are fully occupied.

We also propose an experimental routing algorithm where the destination's distances to the core in different routing directions are better taken into account. In this algorithm the destinations are classified to 24 different routing directions which differ in varying distances in different routing dimensions (X and Y dimension). These routing directions are illustrated in Figure 4. The idea behind this algorithm is that a packet should be routed always in a dimension where the distance to the destination is longer. This way the possibility to change the dimension remains making it possible to evade problematic areas without extending the routing path. The distance resolution in this experimental routing algorithm has three levels. The algorithm distinguishes the routing directions based on the following criteria: the destination core is (1) in the current

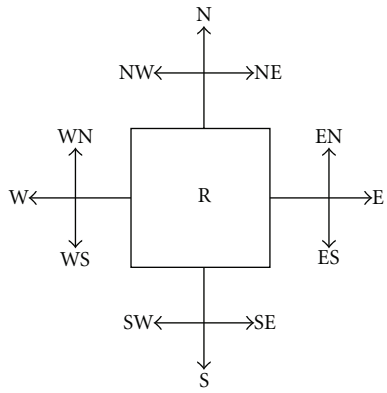


Figure 3: Routing directions. N: North, S: South, E: East, W: West, and R: Router.

Distance between current router and destination in *X*-direction

Distance between current router and destination in *Y*-direction	<−1	−1	0	1	>1
<−1	1	2	3	4	5
−1	6	7	8	9	10
0	11	12	Current router	13	14
1	15	16	17	18	19
>1	20	21	22	23	24

Figure 4: Routing directions in our experimental routing algorithm.

row/column, (2) in the next row/column in some direction, or (3) further away. Hence, there are altogether 24 different routing directions. These routing directions enable extensive classification of different routing cases and that way each case can be handled optimally.

In each of these 24 routing directions we have ranked the possible output ports based on the destination and network conditions. Every time a packet is routed the algorithm identifies the routing direction and uses available traffic status and fault information to select the appropriate output port.

6. Monitoring Algorithms and Communication

In the basic DCM structure the monitoring data is transferred in packets using the actual data network. These packets are called monitoring packets. The monitoring packets have a higher priority in the routers so that they can be transferred even when there is congestion in the network. The monitoring packets are sent from a monitor to a monitor, but because the monitors are not directly connected to each other, the packets are transferred via routers and links.

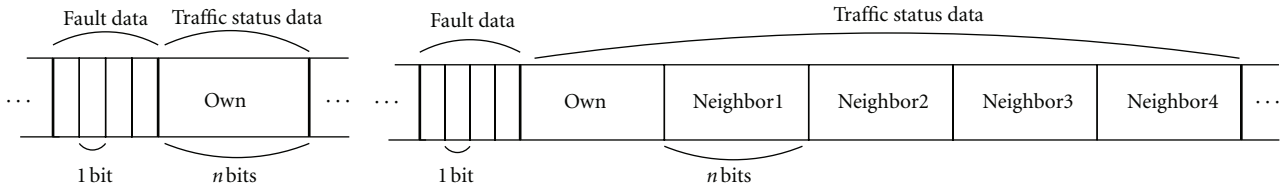
The router statuses in the DCM structure are represented with two binary numbers, one for traffic status and another for fault information. The status of a router is based on the occupancy of the FIFO buffer where packets are waiting to be routed forward. The faultiness of a single component can be represented using a single bit while number of bits in the traffic status values is related to the size of the FIFO buffer, required accuracy as well as the used additional status data processing (see Section 7.1). The resolution of the traffic status data is defined with status data granularity. The granularity defines the number of different values which can be used to illustrate the level of traffic load. For instance, when the status granularity of a router is 4 there is 4 different levels of traffic (1: no or just a little traffic, 4: highly loaded and cannot receive new packets, 2-3: scaled linearly between the edge values). The finer the granularity the better the accuracy of the status values is.

In the DCM structure the monitors exchange their own and their neighbors' statuses with each other. Typically monitoring packets include fault statuses of nearby links and one or more traffic statuses of routers depending on the size of the monitoring cluster. The structure of a monitoring packet payload in systems with monitoring cluster sizes 5 and 13 is presented in Figure 5. The contents of a monitoring packet payload are discussed in Section 7.

In centralized and clustered monitoring structures the monitoring packets are transferred in a network in the same way as the data packets (see Section 2.1). The dynamically clustered approach simplifies the monitoring communication because the routing of the monitoring packets is not needed but substituted with a packet-type recognition. Every monitor sends its status data and the neighbor status data it is forwarding to all its neighbors. The receiver recognizes these packets as monitoring packets and does not send them forward. The transfer distance of a monitoring packet is always one hop, from a router to its neighbor router. This simplicity of monitoring packet transferring combined with the simple routing procedure of monitoring packets makes it possible to keep the latency overhead on tolerable level for most applications. The presented DCM structure is targeted for applications without strict real-time constraints because the in-time delivery of packets cannot always be guaranteed. This is a trade-off of the improved fault tolerance.

A monitor stores the status data from received monitoring packets to its memory and provides this information forward to its own neighbors. This way the routers are able to receive information not only from their neighbors but also from the neighbors of their neighbors. In dynamically clustered monitoring structure the network status data spreads over the network without centralized control and without routing related processing.

The update interval of a monitoring data denotes the conditions when a monitor sends an up-to-date monitoring packet to its neighbors. In [16] we analyzed different update intervals including static and dynamic intervals as well as a hybrid interval combining both of the previous ones. The static interval sends a new packet after a certain time interval



(a) $C_{\text{Size}} = 5$ (b) $C_{\text{Size}} = 13$

Figure 5: Structure of monitoring packet payload with cluster sizes 5 and 13. n is the number of bits required to represent a traffic status value with a certain granularity.

while the dynamic interval has a status value threshold and sends a new monitoring packet when the difference between the previously sent data and the current one reaches the threshold. The third proposed interval is a hybrid of static and dynamic interval in a way that it is basically a dynamic interval but it also sends a monitoring packet after a certain time interval if the dynamic threshold has not been met during that time.

Our research has shown that, while being the most complex at the implementation level, the hybrid status update interval leads to the highest network performance in terms of throughput [16]. The difference to the static interval is significant while the dynamic interval is almost as good as the hybrid interval. We also found that the complexity difference between the implementations is negligible on large systems so we decided to use the hybrid interval in our works. The designer can have impact on the monitoring data overhead by adjusting these interval parameters. In our previous works we chose the parameters in a way that the monitoring traffic overheads were equal to enable the fair comparison of different interval methodologies. The used time interval and threshold values have straight influence on the feasibility of the status values. If these parameters are too high the status values could became badly outdated before next update. This is more crucial in networks where the traffic pattern is changing quickly. The designer should find a compromise between enough up-to-date status values and monitoring traffic overhead.

7. Status Data Diffusion

The network status data diffusion defines how far the status of a network component (router) spreads in the network. A wider spread area makes it possible to react to problems early and avoid routing packets to the worst hot spots or faulty areas.

There are two factors which affect the status data diffusion: the size of a dynamic cluster and status data processing. The cluster size defines how far the status data diffuses from a router and the amount of neighbor status data a router has. The cluster size has an effect on the quantity of neighbor status data to be transferred in a monitoring packet. If the size of a dynamic cluster is 5, it contains the neighbors of a router and the router itself. In this case it is enough to send only the router's own status to the neighbors. If the size of a cluster is 13, it also includes the neighbors of the neighbors and then four neighbor statuses should be included in every monitoring packet. In larger dynamic clusters the amount of monitoring data which has to be included in a monitoring packet increases to a level which is not practical. Therefore, we have limited our analysis to dynamic cluster sizes of 5 and 13.

7.1. Additional Status Data Processing. In DCM structure the network traffic status values are based on the load of the routers. When additional status data processing is used, each router status value is based on the state of the router itself and the state of its neighbors. When the neighbor routers' status is defined using also its own neighbors, the status data diffuses over the network.

A processed status of a router (S), which represents the traffic load in a router and its surroundings, is calculated using (1), where $S_x, x \in \{\text{L}, \text{N}, \text{S}, \text{E}, \text{W}\}$ is the status of a neighboring router in a specified direction (Local router, or a neighbor in North, South, East or, West direction):

$$S = \alpha S_{\text{L}} + \frac{1-\alpha}{4}(S_{\text{N}} + S_{\text{S}} + S_{\text{E}} + S_{\text{W}}), \tag{1}$$

where α is a factor which defines which part of the router status is related to its own utilization level. When status data of twelve neighbors is used, the statuses of different routing directions are calculated using, for example, (2), where $D_x, x \in \{\text{N}, \text{S}, \text{E}, \text{W}, \text{NW}, \text{WN}, \text{NE}, \text{EN}, \text{SW}, \text{WS}, \text{SE}, \text{ES}\}$ represents the status of a routing direction and N_i, where $1 \leq i \leq C_{\text{Size}} - 1$ is the status of a neighboring router in a specific index. Other ten statuses are calculated using corresponding equations. The routing directions are represented in Figure 3 and the used indexes of the neighbor routers in Figure 6:

$$\begin{aligned} D_{\text{N}} &= \beta N_2 + (1-\beta) N_7, \\ D_{\text{ES}} &= \gamma N_3 + \frac{1-\gamma}{2}(N_9 + N_{10}), \end{aligned} \tag{2}$$

where β and γ are factors which define at which intensity the status of a neighboring router affects the status of a direction. Coefficients α, β, and γ ($0 \leq \alpha, \beta, \gamma \leq 1$) are in here defined experimentally, but can be individually adjusted to avoid or favor routing packets to certain routing directions. $\beta, \gamma = 1$ when the size of a dynamic cluster is four. Therefore, (2) is simplified in that case and the complexity of the system is reduced.

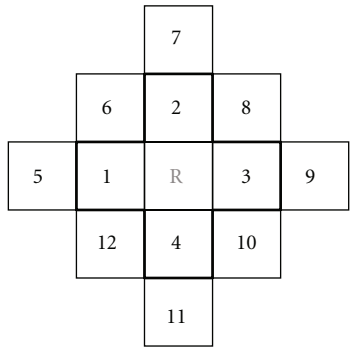


Figure 6: Indexes of the neighbor routers.

7.2. Diffusion Analysis. The analysis of network status diffusion is presented in Figures 7 and 8. The same simulation was executed with two different dynamic cluster sizes without faults and with 10% of network links being faulty. The analysis was not done with larger cluster sizes because the amount of transferred monitoring data then increases to intolerable level. The additional status data processing was used in Figures 7(a) and 8(a). The processing coefficients were experimentally chosen to obtain maximal throughput so that $\alpha = 0.5$, $\beta = 0.6$, and $\gamma = 0.6$. When the additional processing was not utilized (Figures 7(b) and 8(b)) the raw monitored status data was used and the statuses of neighbor routers did not have influence on the router status values.

The differences in network performance appear when the throughput has been fully or nearly saturated. The figures show that in a faultless network the performance differences are notable. The proportional differences were measured at the point where 60% (0.6 on the X-axle) of the maximum capacity of packets were sent during a routing cycle. The 60%point was chosen because it is clearly after the saturation point but still far from the maximum load.

All the following performance increment percentages are in proportion to the performance of an NoC with similar fault pattern, deterministic routing algorithm, and without network monitoring. In a faultless network (see Figure 7) the performance increase is 19% when status data processing is used, regardless of the monitoring cluster size. Surprisingly, when status data processing is turned off, the throughput increases 23% and 21% in systems with monitoring cluster sizes 5 and 13, respectively. Simulations were also executed in faulty networks where 10% of the links are set to unusable state (see Figure 8). These links were randomly chosen and simulations were run with several different random fault patterns. When status data processing is used, the performance increases are 78% and 74% in networks with cluster sizes 5 and 13, respectively. Without status data processing the corresponding values are 78% and 72%.

The analysis shows that DCM with small cluster size improves network performance significantly. An especially notable feature is its ability to maintain the network throughput in a faulty network. Without network monitoring the throughput decreases 41% when 10% of links become faulty. However, if the presented monitoring is used the decrement is only 11%. Furthermore, the throughput in a faulty network with monitoring is 6% higher than it of a faultless network without monitoring.

A noteworthy observation is that a larger cluster size does not have positive impact on the performance but actually reduces it. This phenomenon can have multiple reasons. One reason for the inefficiency can be that too much data processing leads to inaccurate status data and dissolves the differences between the statuses. Another reason could be the latency in the status data propagation which makes it outdated before it is utilized.

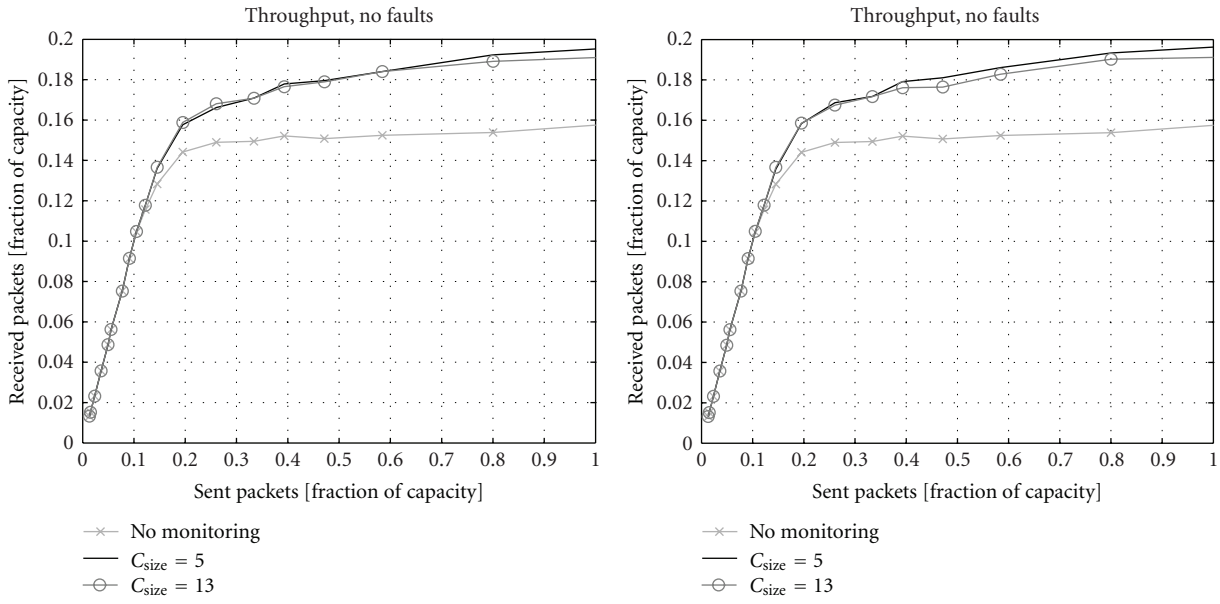
The influence of the additional status data processing is small or even nonexistent. In a faulty network there is a very small increase in the throughput. However, in faultless network the impact is even negative. For example, Figure 9 shows the difference in throughput in faulty network with $C_{\text{Size}} = 5$. As can be seen in this specific comparison the difference is negligible.

The inefficiency of the status data processing may stem from the same factors as that of the large cluster size. The differences between status values dissolve and are not on display so that the routing algorithm could make right decisions.

8. Format of Network Status Data

The network status data is used to deliver the information of the state of the network and it can be used to different purposes. When the main application is traffic management the data typically includes information concerning network load and faults. Faults are simply denoted using binary values which indicate if a component of the network is usable or faulty. In more complex systems multilevel fault indicators could be considered. The network load is denoted using a scale where different values represent different amounts of load on a network component. In our simulations the network load representation is linear. Different scales can be considered in some specific applications.

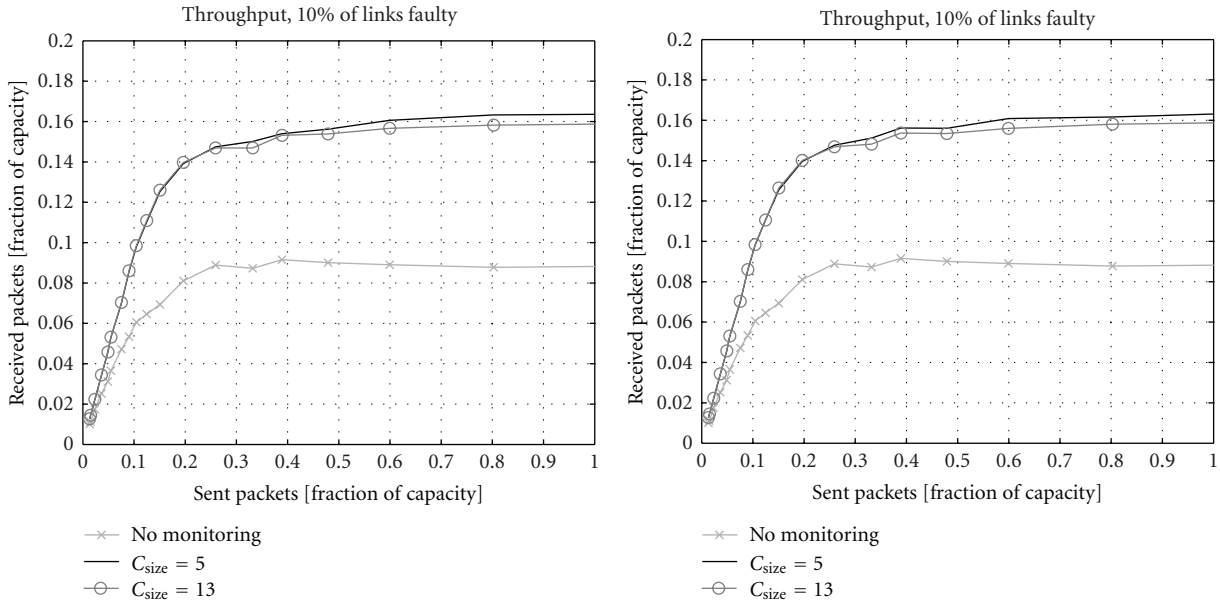
8.1. Granularity of the Router Status Values. The status data granularity defines the resolution of the status data values or how many different values there are on the scale which is used to represent the load on a network component. The smallest used value indicates that the load of a network component is very low and the highest value represents high load of the component. Rest of the values indicate component load linearly between the extreme values. An example of the status data granularity was given in Section 6. In physical implementations the status data values are represented as binary numbers which means that finer granularity requires more bits and that way increases the size of monitoring packet payload. The status data granularity impacts on the amount of the monitoring data to be transferred as well as to the required computational resources in the monitoring components. The granularity should be chosen so that the required data transfer and status data processing resources are adequate in the framework of the current NoC implementation.



(a) With additional status data processing

(b) Without additional status data processing

Figure 7: Throughput with different sized clusters (C_{size}) without network faults.



(a) With additional status data processing

(b) Without additional status data processing

Figure 8: Throughput with different sized clusters (C_{size}). 10% of links are faulty.

The 64-core NoC has been simulated with different granularity alternatives. 32 was defined to the maximum possible granularity because of the limited size of payload in the monitoring packets. A status value with 32-level granularity can be indicated with 5 bits. When monitoring cluster size is 13, a monitoring packet should include information on router's status and statuses of its four neighbors. With granularity of 32, this takes 25 bits which can be considered a realistic amount of data in a monitoring packet. The same data granularity is also used between probes and monitors.

The throughput of a 64-core NoC with diverse status granularity is presented in Figures 10 and 11. The simulations were carried out in a faultless network and in a network where 10% of links were faulty. The results show that in a fully functional network the performance is only slightly improved when granularity is larger than eight. It

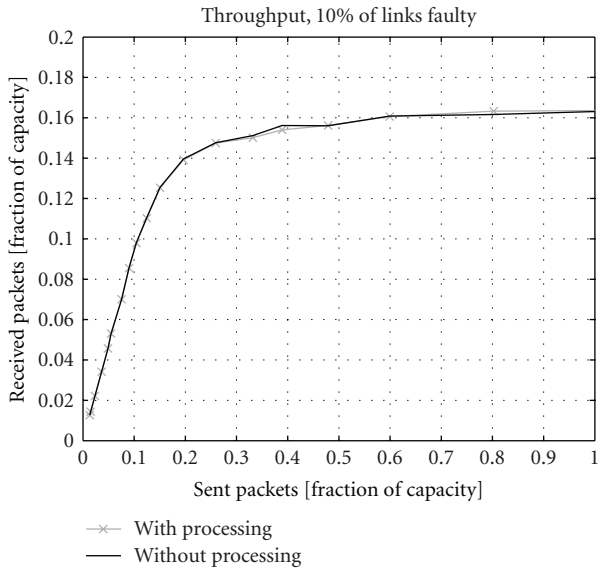


Figure 9: Throughput with and without processing. $C_{size} = 5$, 10% of links are faulty.

can be also noted that in faulty network the granularity should be at least 16. In both cases 4-level granularity leads to significantly lower performance which means 7% decrement in the faultless and 4% in the faulty network compared with the 16-level granularity. This supports the use of at least 16-level granularity. The performance of system without monitoring is illustrated as a reference.

8.2. Combining Router and Link Statuses. A method to simplify monitoring status data is to combine traffic and fault information. In the original status data format (see Figure 5) there is a binary number to represent the traffic load and a bit to indicate the resource faultiness. To decrease the monitoring complexity and the size of monitoring data payload, we analyzed two approaches where the monitoring data is combined to hybrid forms. These two hybrid data formats are presented below.

8.2.1. Hybrid Status Data Using Traffic Status Values. Traffic status values can be used to indicate faults by defining that the maximum status value does not only indicate high traffic load but also faulty resources. If there is a faulty component in some direction, the traffic status value of that direction is set to its maximum value. In this case the payload of a monitoring packet (see Figure 5) is reduced by 4 bits because the fault values are not included. When this format is utilized the routing algorithm has to be configured to totally avoid the routing directions with maximum traffic values.

8.2.2. Hybrid Status Data Using Fault Indicator Values. The monitoring data is simplified even further when all the status data is combined to the boolean fault indicator values. In this approach, a routing direction is marked as faulty when there is high traffic load. The status can be restored when the traffic load decreases. This way packets are not routed in highly loaded directions. A drawback in this approach is the loss of knowledge about differences between routing directions with low and medium traffic load. Because the traffic status values are not used, the reduction of the monitoring packet payload is n bits if $C_{Size} = 5$ and $5n$ bits if $C_{Size} = 13$.

The monitoring data combination approaches were simulated with the NoC model, and the results are presented in Figure 12. Obviously, in faultless network the traffic-status-based combination works similarly as separate data. When the traffic data is integrated into the fault statuses the decrease in throughput is 8%. However, the performance is still 12% better than without monitoring. In faulty network separated status data is notably the best solution. The traffic-data-based hybrid format causes 24% performance loss which is even larger with the fault-data-based format, 40%. Nevertheless, these hybrid formats increase the performance by 36% and 8%, correspondingly, compared to the system without traffic monitoring.

The presented analysis leads to a resolution that both monitoring data classes are necessary in a system where faults are a realistic threat. In less vital applications the hybrid formats could be a good compromise.

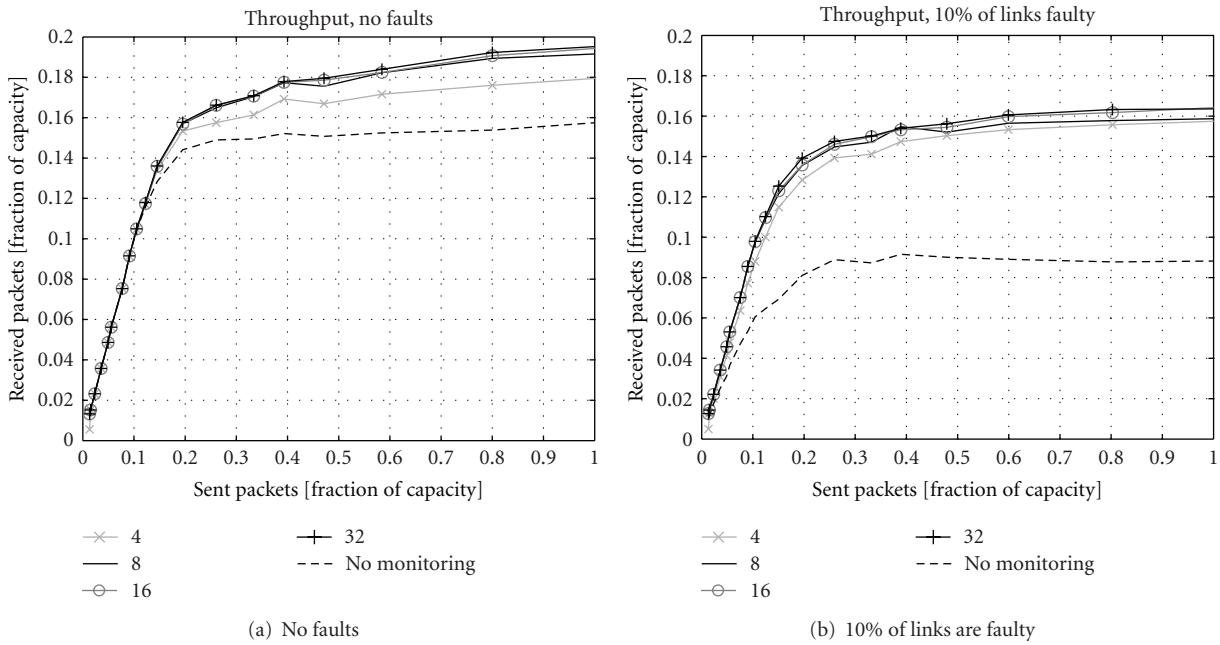
9. Serial Monitor Communication

In the DCM structure the monitoring data is transferred in the same network which is used by the original data packets. It is a straightforward solution which minimizes the requirement of additional resources. However, a shared-resource structure is always at least somewhat intrusive and it consumes the network resources which otherwise could be used by the actual data packets.

An alternative solution to the intermonitor communication is serial communication which is implemented with dedicated channels. It can be realized with relatively small amount of additional resources. A drawback in serial communication is the increased transfer delay. However, because the serial communication resources are dedicated to the monitoring communication there can be a nonstop status update without paying attention to update intervals [16].

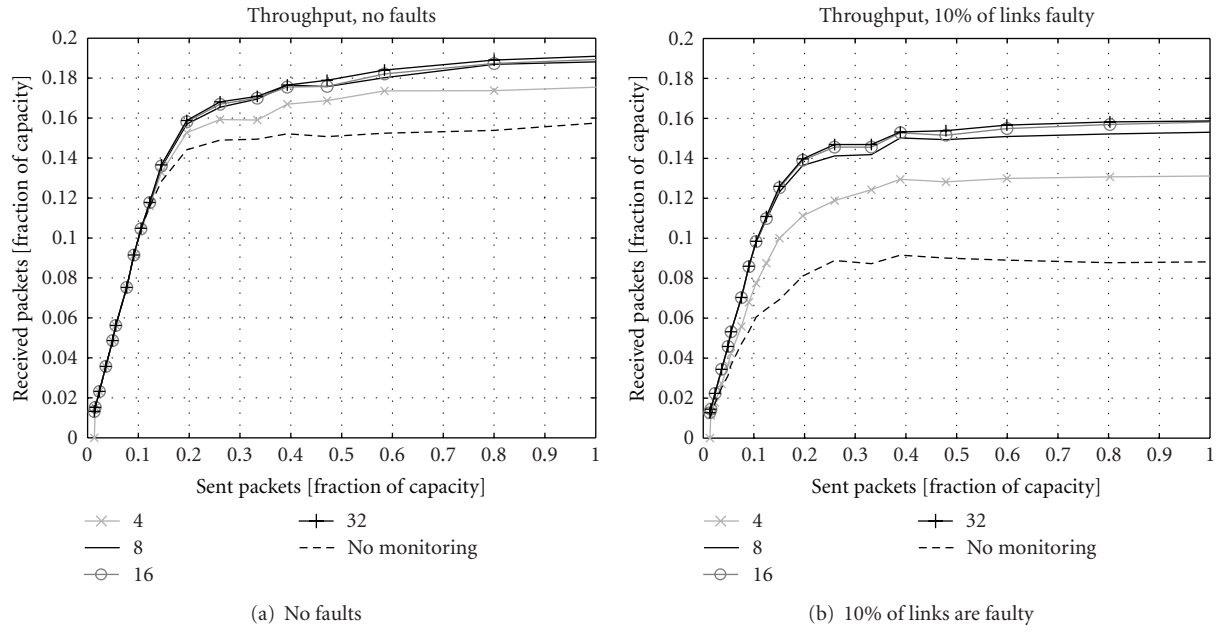
Serial monitor communication was simulated with the SystemC-based NoC simulation model. Throughput with different status data granularities and serial communication is presented in Figure 13. The serial transmitter operates at the same clock frequency as the maximum frequency of the monitoring packet transmitter. However, the latter rarely works on its maximum frequency because of the status update interval conditions.

Essentially serial communication is slower than the earlier discussed parallel, packet-based communication, and the theoretical delays of the serial communication are even more increased when there is large amount of data to be transferred, for example, in systems with relatively large monitoring clusters. However, in contrast the serial communication is operating in dedicated communication resources which can be used only to this purpose all the time. This way the status values can be updated actually more often than when the monitoring packets are transferred in



(a) No faults (b) 10% of links are faulty

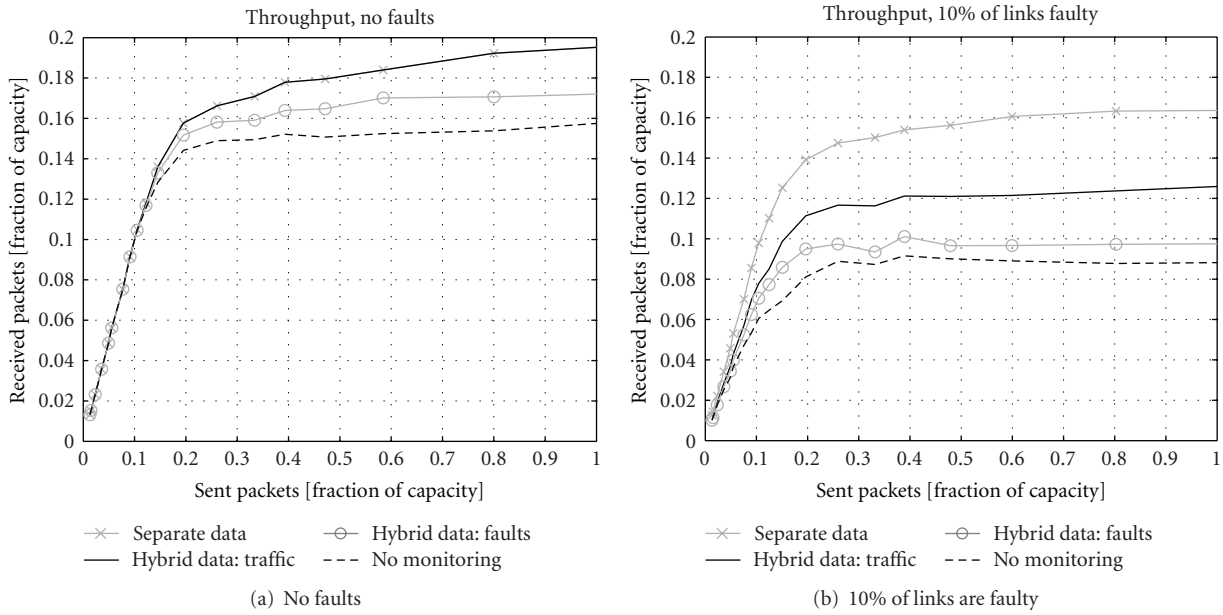
Figure 10: Throughput with different traffic status granularity alternatives. $C_{\text{Size}} = 5$ and data processing is enabled.



(a) No faults (b) 10% of links are faulty

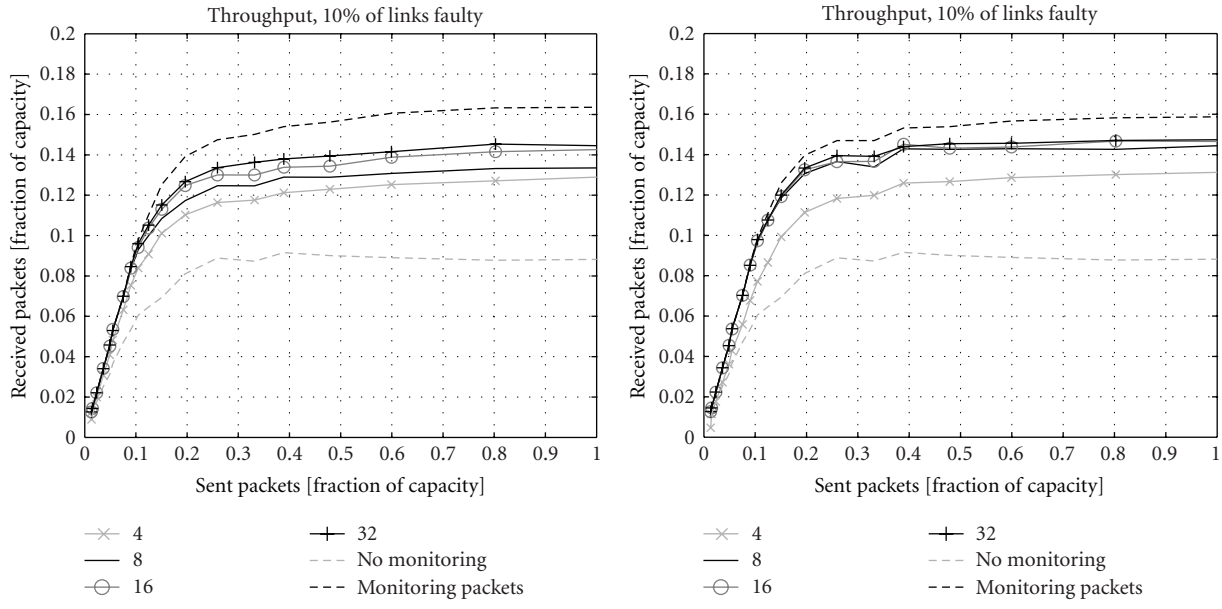
Figure 11: Throughput with different traffic status granularity alternatives. $C_{\text{Size}} = 13$ and data processing is enabled.

the shared resources. Somewhat surprisingly the system with $C_{\text{Size}} = 13$ works well also for coarser status granularities when serial communication is utilized. This is a result of shorter traffic status update interval even though in this case the amount of serially transferred data is quite large. In this case the granularity of 4 clearly stands out. Possibly the granularity of 4 is simply too rough to be used with the data amount of a system with large monitoring clusters. Figure 13(a) shows that when serial communication and small cluster size are used the performance differences are more notable also between granularities 8, 16, and 32. When serial communication is used in system with $C_{\text{Size}} = 5$ the performance with granularity of 32 is 11% less than with corresponding system using monitoring packets. Respectively, the performance of a system with serial communication and granularity of 4 is 39% better than the performance of system without monitoring. The corresponding percentages for system with $C_{\text{Size}} = 13$ are 6% and 42%.



(a) No faults (b) 10% of links are faulty

Figure 12: Throughput with separate traffic and fault data as well as with the hybrid formats. $C_{\text{Size}} = 5$ and data processing is enabled.



(a) $C_{\text{Size}} = 5$ (b) $C_{\text{Size}} = 13$

Figure 13: Throughput with different granularity alternatives using serial communication. 10% of links are faulty and data processing is enabled.

The serial communication could be a useful option when a designer wants to keep the communication resources of different applications separately. The serial approach guarantees that the monitoring communication does not disturb the actual data which is transferred in the network. It may be possible to increase the clock frequency of the serial transmitter from what was used in the presented analysis. In this case the performance differences should shrink.

10. Using Fewer Monitors

The DCM system is based on a structure where there is an identical monitor attached to each router. These monitors include both monitoring and probing components. One potential way to reduce monitoring structure complexity is to decrease the number of monitors systematically by removing every nth monitor. In this approach there is a probe

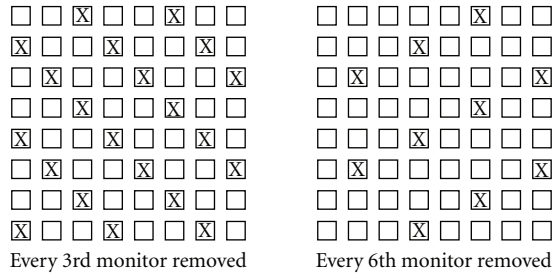
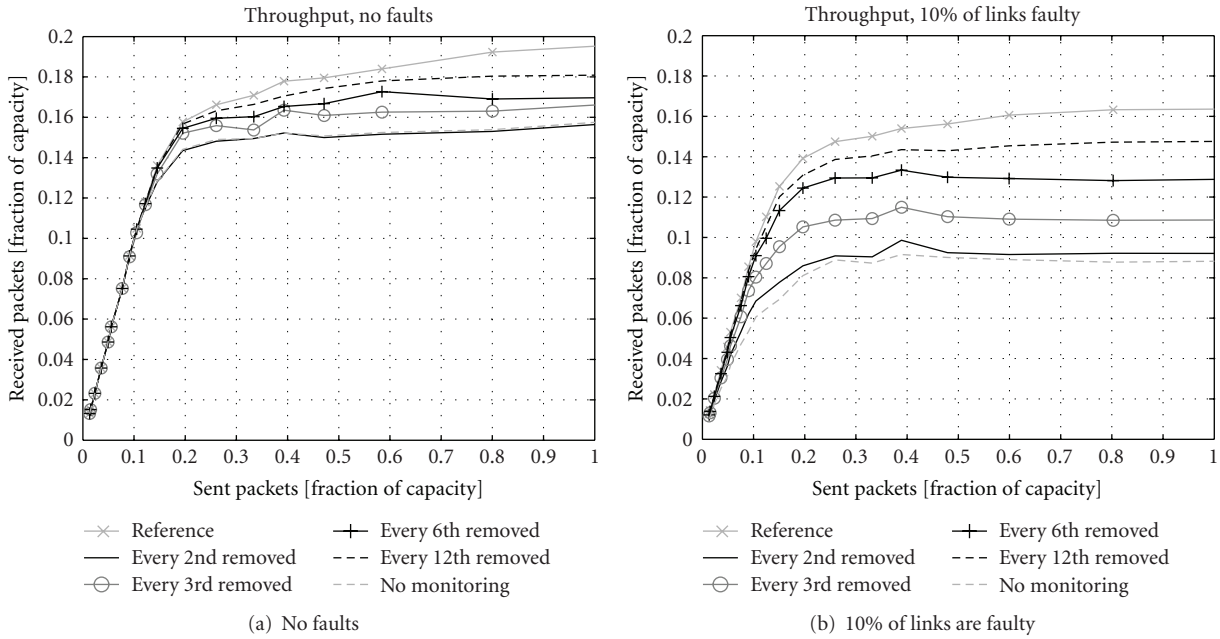


Figure 14: Two patterns of removed monitors. Removed monitors are marked with X.



(a) No faults

(b) 10% of links are faulty

Figure 15: Throughput with fewer monitors. $C_{\text{size}} = 5$ and data processing is enabled.

attached to every router but a monitor is attached only to a limited number of routers. This means that there is still complete knowledge of the network state in the monitors because there is probes attached to every router. Two monitor removal patterns are illustrated in Figure 14. The monitors receive probed data, process it, and deliver it to the local router. The routers that do not have their own monitor should utilize a deterministic routing algorithm because they do not have access to the probed status data from the neighboring routers. An adaptive algorithm is utilized in the other routers [7]. The simplified deterministic routers forward packets based on a deterministic routing algorithm. In problematic traffic or fault cases, a simplified router could route packets randomly just directing them to some of its neighbors. In any case the neighbors of a deterministic router are adaptive routers which can route the packet forward adaptively.

In addition to performance, the reduction of monitors affects the complexity of the NoC implementation. Routers which do not have their own monitoring component could have less complex routing logic which decreases the router area. This is due to the deterministic routing algorithm which substitutes the adaptive routing algorithm in the routers which do not have a monitoring component. However, the probing components cannot be simplified because they have still have to offer status data to other monitors.

This approach was analyzed using our SystemC-based NoC simulation model and the results are presented in Figure 15. The simulation cases were chosen so that the unmonitored routers are placed as evenly as possible in the network. This way we defined four simulation cases where every second, every third, every sixth, and every twelfth monitor was removed from the network.

The figure shows that the removal of a monitor, even if it is just every 12th, has notable influence on network throughput and the influence is even more remarkable when there are faults in the network. Removal of every second monitor causes 18% performance decrement in faultless network and 43% in the network with 10% of faulty links. In faultless network the performance is equal with the performance of a system without traffic monitoring. In faulty network there is 2% performance increase compared

to the unmonitored system. If just every 12th monitor is removed, the performance decreases by 3% and 10%, respectively.

The removal of monitors has positive impact to area and traffic overheads caused by the monitoring system. However, the total area of the monitoring system is almost negligible compared to the area of a 64-core NoC. This way the removal of monitors cannot be justified with the reduced complexity when the performance decrement is as large as presented here. In application-specific NoCs it could be reasonable to remove monitors from areas where traffic is predictable so that the resources can be sized properly during the design phase and adaptivity is not necessary. However, in our work the focus is on homogeneous general-purpose NoCs so the monitors are placed evenly over the network.

11. Discussion and Conclusions

The dynamically clustered monitoring structure for fault-tolerant Networks-on-Chip has been presented and analyzed in this paper. Dynamically clustered monitoring does not require any centralized control. There is a simple monitor and a probe attached to each router in the network. Centralized control is not required but the monitors exchange information with each other. Each router has a dynamic cluster around itself from where a router collects the data it needs for traffic management. The different features of the dynamically clustered monitoring structure were analyzed and their influence on the overall performance of the system were studied. Most of these presented features can be utilized in different NoC implementations with various requirements and limitations. However, due to nature of adaptive, shared-resource system, the presented DCM structure could not be the best solution to systems with strict real-time requirements.

In future works the analysis of individual cores and routers will be improved. In this phase, the NoC simulation model does not enable the analysis of specific senders and receivers but concentrates only on overall performance. This makes it possible to analyze how different monitoring methods and parameters affect performance from a component's point of view.

Performance of the DCM structure could be adjusted by using different sized monitoring clusters in different areas in the network. Areas with low traffic load may work at reasonable performance using very simple deterministic routing algorithms. At the same time in the same system there could be performance critical areas with high traffic loads and tight quality of service requirements. It could be necessary to use larger monitoring clusters and adaptive routing on these areas. Another useful feature could be on-fly reconfiguration of cluster size and the routing algorithm. Performance and energy consumption of the communication resources could be optimized by using more complex mechanisms in the critical areas of the network.

Our SystemC-based NoC simulation model has been proved to be an efficient tool to analyze and simulate different aspects in Networks-on-Chip. The model is reasonably easy to configure for the analysis of different features. The NoC simulation model was used to analyze monitoring algorithms, monitoring data diffusion areas, format of monitoring data and communication as well as number of monitors. The presented research shows that in most cases simple monitoring algorithms and small monitoring cluster areas perform at least as well as more complex implementations. In this paper the most complex structures did not cause significant improvements to performance. However, these structures and algorithms may be developed further in the future works. Another observation is that even small adjustments in the system parameters can have significant influence to the overall performance. Therefore the parameters should be chosen carefully while designing a complex DCM structure.

Acknowledgments

The authors would like to thank the Academy of Finland, the Nokia Foundation, and the Finnish Foundation for Technology Promotion for financial support.

References

[1] 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, June 2001.

[2] L. Benini and G. De Micheli, "Networks on Chip: a new paradigm for systems on chip design," in *Proceedings of the Design, Automation and Test in Europe (DATE '02)*, pp. 418–419, 2002.

[3] C. Grecu, A. Ivanov, R. Saleh, and P. P. Pande, "Testing network-on-chip communication fabrics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 12, pp. 2201–2213, 2007.

[4] 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.

[5] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu, "Self-adaptive system for addressing permanent errors in on-chip interconnects," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 18, no. 4, pp. 527–540, 2010.

[6] V. Rantala, T. Lehtonen, P. Liljeberg, and J. Plosila, "Analysis of monitoring structures for network-on-chip—a distributed approach," *IGI International Journal of Embedded and Real-Time Communication Systems*, vol. 2, no. 1, pp. 49–67, 2011.

[7] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2004.

[8] V. Nollet, T. Marescaux, and D. Verkest, "Operating-system controlled Network on Chip," in *Proceedings of the 41st Design Automation Conference*, pp. 256–259, 2004.

[9] R. Mouhoub and O. Hammami, "NoC monitoring hardware support for fast NoC design space exploration and potential NoC partial dynamic reconfiguration," in *Proceedings of the International Symposium on Industrial Embedded Systems (IES '06)*, pp. 1–10, October 2006.

[10] 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.

[11] C. Ciordas, K. Goossens, T. Basten, A. Radulescu, and A. Boon, "Transaction monitoring in networks on chip: the on-chip run-time perspective," in *Proceedings of the International Symposium on Industrial Embedded Systems (IES '06)*, pp. 1–10, October 2006.

[12] J. van den Brand, C. Ciordas, K. Goossens, and T. Basten, "Congestion-controlled best-effort communication for Networks-on-Chip," in *Proceedings of the Design, Automation and Test in Europe (DATE '07)*, pp. 1–6, April 2007.

[13] M. Al Faruque, T. Ebi, and J. Henkel, "ROAdNoC: runtime observability for an adaptive Network on Chip architecture," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '08)*, pp. 543–548, November 2008.

[14] T. Marescaux, A. Rångevall, V. Nollet, A. Bartic, and H. Corporaal, "Distributed congestion control for packet switched Networks on Chip," in *Proceedings of the International Conference ParCo*, pp. 761–768, 2005.

[15] P. Gratz, B. Grot, and S. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *Proceedings of the IEEE 14th International Symposium on High Performance Computer Architecture (HPCA '08)*, pp. 203–214, February 2008.

[16] V. Rantala, T. Lehtonen, P. Liljeberg, and J. Plosila, "Analysis of status data update in dynamically clustered network-on-chip monitoring," in *Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS '11)*, March 2011.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 278735, 11 pages
doi:10.1155/2012/278735

Research Article

A Hardware Design of Neuromolecular Network with Enhanced Evolvability: A Bioinspired Approach

Yo-Hsien Lin[1] and Jong-Chen Chen[2]

[1] *Department of Information Management, Yuanpei University, 306 Yuanpei Street, Hsinchu 30015, Taiwan*
[2] *Department of Information Management, National Yunlin University of Science and Technology, 123 University Road, Section 3, Douliou, Yunlin 64002, Taiwan*

Correspondence should be addressed to Jong-Chen Chen, jcchen@yuntech.edu.tw

Received 14 July 2011; Revised 30 August 2011; Accepted 30 August 2011

Academic Editor: Jiang Xu

Silicon-based computer systems have powerful computational capability. However, they are easy to malfunction because of a slight program error. Organisms have better adaptability than computer systems in dealing with environmental changes or noise. A close structure-function relation inherent in biological structures is an important feature for providing great malleability to environmental changes. An evolvable neuromolecular hardware motivated by some biological evidence, which integrates inter- and intraneuronal information processing, was proposed. The hardware was further applied to the pattern-recognition domain. The circuit was tested with Quartus II system, a digital circuit simulation tool. The experimental result showed that the artificial neuromolecularware exhibited a close structure-function relationship, possessed several evolvability-enhancing features combined to facilitate evolutionary learning, and was capable of functioning continuously in the face of noise.

1. Introduction

Effective programmability is an important feature inherent in computer systems, including software and hardware, which allows us to explore various problem domains. However, most of computer systems are brittle in the sense that a slight modification of a system's structure can inadvertently change its functions or cause it to malfunction [1]. This is because computer systems possess a mapping structure with fitness landscapes consisting of isolated peaks that are separated by wide, deep valleys. By contrast, organisms possess a mapping structure with fitness landscapes holding some degree of smoothness that a slight change in an organism's gene structure generally will not significantly alter its functions. Finding feasible solutions within a reasonable time may become much easier in a smooth landscape than in a rugged landscape [2]. In biological systems, the smoothness (gradualism) property is naturally represented in the close structure-function relationship.

In the early 1990s, some other researchers concentrated on applying evolutionary techniques to hardware design. They attempted to use a reconfigurable hardware to continually change the internal circuit structure until the desired structure appears. This field was called evolvable hardware (EHW). EHW brought an interdisciplinary integration. One such idea is to combine the merits of biological sys-tems and computer systems together and hopefully create hardware with better adaptability. For example, Sipper and Ronald [3] proposed an FPGA circuit to simulate the glob-al behaviour of a swarm of fireflies. Mange et al. [4] successfully applied evolutionary techniques into the design of a timer (stopwatch) and a full watch (biowatch) with digital circuits. Higuchi and his colleagues [5, 6] worked on the development of a number of evolvable hardware chips for various applications, including an analog chip for cellular phones, a clock-timing chip for Gigahertz systems, a chip for autonomous reconfiguration control, a data compression chip, and a chip for controlling robotic hands. Murakawa et al. [7] applied evolutionary techniques to reconfigure neural network topology, de Garis [8, 9] developed an artificial brain that assembled a group of cellular automata-based neural net modules to control a robot, and Torresen

[10] designed an evolutionary digital circuit to control prosthetic hands.

Our goal is to provide the digital machine with a representation of the internal structure-function relations of biological systems, to capture some of the dynamic modes of the processing of these systems, and to incorporate learning algorithms of the type used in natural systems. Redundancy, weak interactions, and compartmentalization are three important features inherent in biological structures that facilitate evolutionary learning [1]. The proposed system (artificial neuro molecular system, ANM) is a plastic architecture with rich dynamics that combines these three features into the system, in particular into the subneuronal level of processing. We note that redundancy allows an organism to absorb genetic changes and yet wait for other mutations to join together to make a significant change in its phenotypic traits. By virtue of redundancy, several genetic mutations do not have to occur simultaneously. Weak interaction is the other indispensable feature for facilitating evolution. When the interactions among the constituted components of a system are small, adding a component into (or removing it from) a system will not significantly alter its outputs (or functions). This allows a system to stabilize its current state in responding to structural changes or environmental changes. Compartmentalization is another evolution-friendliness feature. It allows a system to block off disturbance or noise in the environment.

2. Architecture of ANM

The ANM model was motivated from the molecular mechanisms inside real neurons. The model consists of two types of neurons: cytoskeletal neurons and reference neurons. Cytoskeletal neurons have significant intraneuronal information processing that might directly or indirectly relate to their firing behavior. They combine, or integrate, input signals in space and time to yield temporally patterned output signals. Reference neurons serve as pointers to other neurons in a way that allows for interneuronal memory manipulation.

In this section, we introduce the intraneuronal architecture that plays the role of integrating spatiotemporal signals inside a neuron and the interneuronal architecture that orchestrates groups of neurons for performing coherent tasks. We then explain the evolutionary learning algorithm used in this model.

2.1. Operation Hypotheses. The model is based on two hypotheses.

H1. There are some brain neurons in charge of the time-space information transition. This kind of neuron is called cytoskeletal neuron. Cytoskeletal neurons are based on the operation hypothesis between the nerve cell cytoskeletons and molecules, producing a time-space input signal and transducing it into a series of time outputs [1, 11].

H2. There are some brain neurons in charge of memory control and neuron group organization. This kind of neuron is called reference neurons. The purpose of reference neurons is to form a common-goal information processing group from cytoskeletal neurons. By the memory screening of reference neurons, each workgroup would have neurons of different internal structures, thus being able to finish the group task [1, 11].

2.2. Intraneuronal Architecture. It has been firmly established by now that information processing inside a neuron is significant. The objective of the present study is not to identify the precise nature of these mechanisms, but rather to capture the working hypothesis that the cytoskeleton serves as a signal integration system. Our model is restricted to the membrane components. In the present implementation, the membrane of the cytoskeleton is abstracted as a macromolecular network (a cytoskeletal network) comprising a number of components capable of initiating, transmitting, and integrating cytoskeletal signals. Our assumption is that an inter-neuronal signal impinging on the membrane of a neuron is converted to an intraneuronal signal (a cytoskeletal signal) transmitting on the cytoskeleton. This process was called "transduction"; therefore, a cytoskeletal neuron could be considered a transducer with a specific structure. Cytoskeletal neurons are platforms of message processing, and they are inspired by the signal integration and memory function of the cytoskeleton.

This research utilized 2D cellular automata (CA) [11–13] to conduct the experiment of cytoskeletal neurons, and the wraparound fashion links were adopted for the CA arrangement. A cytoskeleton has multiple molecule networks of microtubules, microfilaments, and neurofilaments. In order to simulate these networks, we defined three kinds of fibers to make a cytoskeleton type (C-type), and the fibers were named C1, C2, and C3. Each of the cytoskeletal elements will have its own shape, thus forming the cytoskeletal molecule networks. The conformation of each cytoskeletal element is variable; therefore, molecule-mass-like groups may possibly be formed. Different types of cytoskeletal elements have different signal transmission features. C1 has the strongest signal bearing capacity, but it has the slowest transmission speed. C3 has the weakest signal bearing capacity, but it has the fastest speed. C2's performance is between C1 and C3. The illustration of cytoskeletal neurons structure is shown in Figure 1. Each cytoskeletal neuron has its unique cytoskeletal fiber structure. The types of signal flows depend on the different structures and different transmission characteristics. Some signal flow would execute the transduction tasks with a diffusion-like method, sometimes fast and sometimes slow.

When an external stimulus hits a cytoskeletal neuron membrane, it will activate the readin enzyme at that location. The activation will cause a signal flow to transmit along the route of the same cytoskeletal elements. For example, after the on-location (3, 2) readin received the external input, it will transmit the signals to its eight neighbors that have the same cytoskeletal element locations. The illustration shows that it can transmit the signal to C2 at locations (2, 2) and (4, 2). Any cytoskeletal element that receives this kind of signal will do the same, thus forming the phenomenon of

a signal flow. In order to ensure it is a one-way transmission, meaning there will not be any signal backflow or loop formed, the cytoskeletal element will enter a temporal resting state after the transmission. This is called a refractory state. The additional remark is that after a signal was transmitted by a cytoskeletal element, the signal did not disappear immediately within the element. Instead, the signal would decrease progressively until it finally disappeared. The decreasing signal and the new-coming signals would cause a time-space integration reaction, and that is a very important mechanism that decides when a firing will occur.

There could be some interactions among different cytoskeletal fibers. Microtubule-associated proteins (MAPs) have the ability to connect different cytoskeletal fibers, thus causing cross-fiber signal flow channels. This will help the flow of microsubstances within neurons. For instance, when the input signal originated from location (3, 2) goes along the C2 elements of the second column, it will meet an MAP-linked C1 element at location (5, 2). The C2 signal will be transmitted to C1 through MAP, and another signal flow will be formed in C1. However, due to different types of cytoskeletal fibers and different transmission features, there might be some energy transition problems when signals going through different mediums. Hence, regarding the cross-fiber signals, this research defined the signal bearing capacity of C1, C2, and C3 as S, I, and W, meaning strong, intermediate, and weak. Because the linking function provided by MAP allows the signals to flow among different molecule elements, there exist information processing behaviors within the neurons.

When a time-space integrated cytoskeletal signal arrives at a location of a readout enzyme, the activation will lead to a neuron firing. For example, the signal flows started at locations (1, 5) and (8, 7) may be integrated at location (5, 5), and the readout enzyme at that location would be activated, thus causing a neuron firing. Because the integrated cytoskeletal signals may continuously appear, the firing outputs become a series of signals that happened in different time points. This research collected these signals to serve as the reference for transduction efficiency assessments.

2.3. Digital Hardware Design of Cytoskeletal Neuron. In the process of digitalization, each grid in cytoskeletal neuron is called processing unit (PU). Figure 2 shows the conceptual architecture of a PU, including four control parts and four signal processing blocks. The input department controler is responsible for controlling the conversions of the signals arriving at the input department into signals for the process department. The output department controler is responsible for controlling the layout of the information for signals sent out from output department to its neighboring cells. The processor department controller has two purposes. Firstly, it will control the countdown of an accumulator so that its value will degrade at a certain speed. Secondly, it will control the timing of the signals sent from the accumulator to its corresponding bounder, which in turn determines the transmitting speed of a cell. The following explains how to implement signal initiation, transmission, and integration on the cytoskeleton of a neuron.

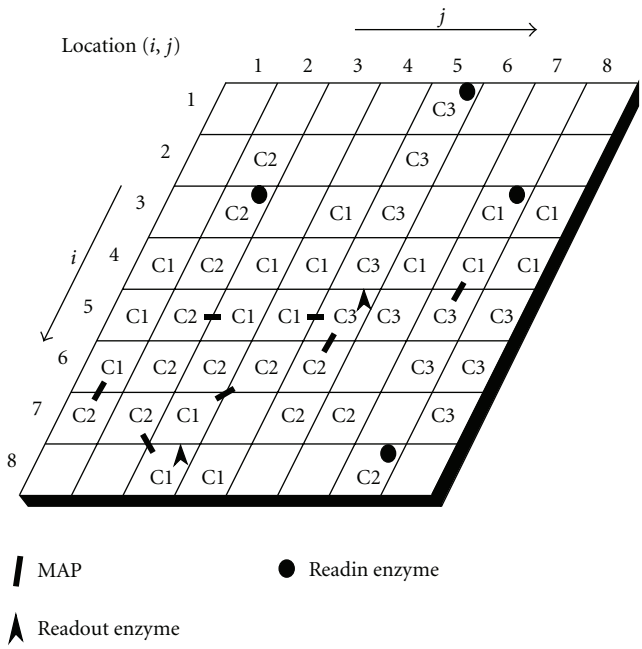


Figure 1: Structure of cytoskeletal neuron.

2.3.1. Signal Initiation. In the present implementation, there are two possible mechanisms to initiate a cytoskeletal signal. One is directly initiated by an external stimulus. When a PU receives an external stimulus and there is a readin enzyme sitting on it, a new cytoskeletal signal is initiated. The other mechanism is combining some specific combinations of cytoskeletal signals in space and time to turn a PU into a highly activated state, which in turn initiates a new signal. Each PU processes the signals sent from its eight neighboring PUs through the input block (Figure 3). We note that a PU will change its state when it receives a signal from its neighboring PU (through MAP). Different types of cytoskeletal signals are initiated and transmitted by different types of PUs. We assume that there are four possible PU types: C1, C2, C3, and none. The first three represent different types of cytoskeletal components for transmitting signals (i.e., different signal flows) whereas the last one represents the lack of a component. In Figure 3, a signal from a C1-type, C3-type, and C2-type neighboring PU is labeled as S, W, and I, respectively.

2.3.2. Signal Transmission. The following explains how to implement signal transmission on an 8×8 grid of PUs. We assume that signal transmission occurs through the neighboring PUs of the same type. An activated PU will activate its neighboring PUs of the same type at its next time step, which in turn activates its neighboring PUs of the same type at the following next time step. This process continues as long as there is a neighboring PU belonging to the same type. To assure unidirectional signal transmission, an activated PU will enter a refractory state. The refractory period depends on the update time of each PU type (to be described in the next section). It will then go back to the quiescent state after the refractory period is over. A PU in the refractory state

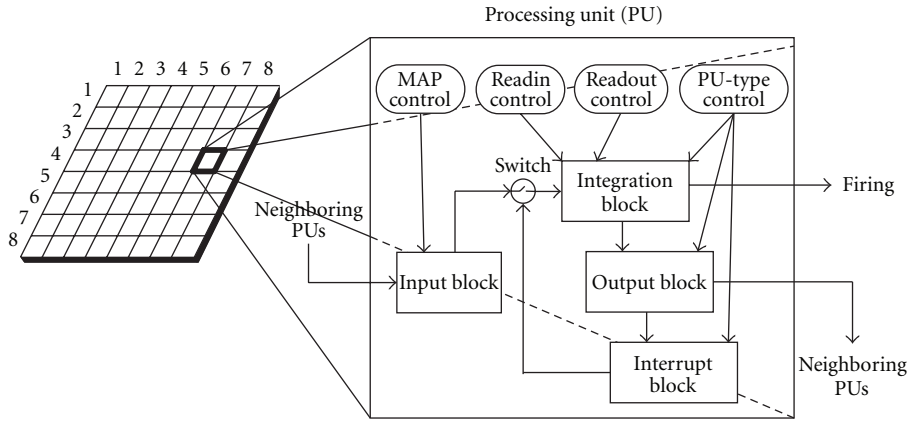


Figure 2: Conceptual architecture of a PU. The input block is illustrated in Figure 3, the interrupt block in Figure 4(a), the output block in Figure 4(b), and the integration block in Figure 6.

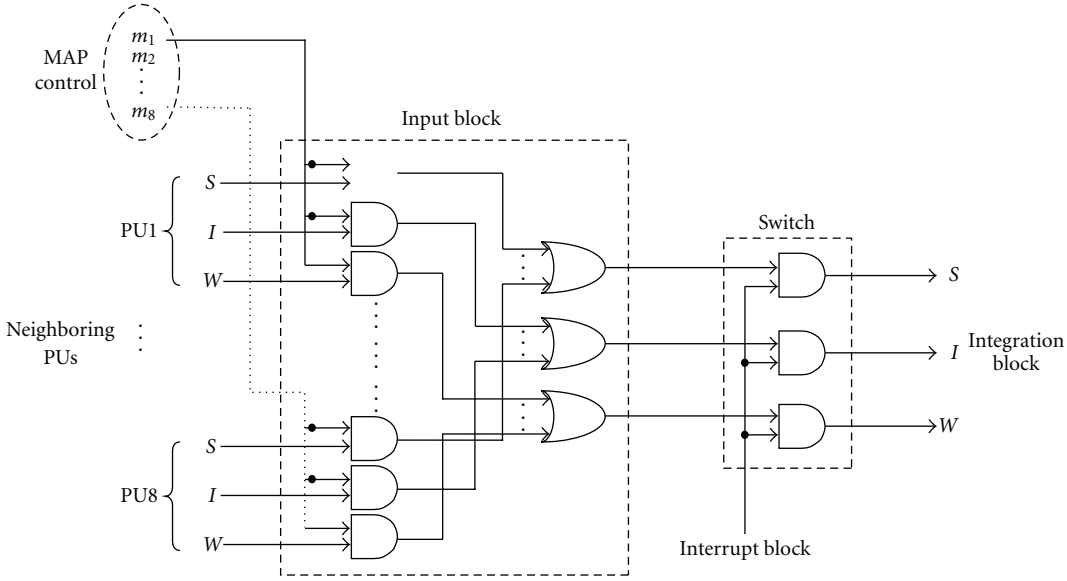


Figure 3: Conceptual architecture of the input block.

will ignore any stimuli during the refractory period. This would prevent a signal from bouncing repeatedly between two neighboring PUs. We note that the refractory time is an important parameter that may affect the performance in the present hardware design. That is, different output is possible when length of the refractory time of each type is varied. But our goal at this stage is to fulfill the function that a PU serves as a signal integrator that combines different signals in space and time (the detail will be described in the next section). That is, the refractory time is fixed and will not be involved in the evolutionary change. In the present implementation, we have not performed a systematical experiment along this line. But it would be interesting to perform the experiment in the near future.

A switch controlled by the interrupt block is used to regulate the signal flow from the input block to the integration block (Figure 4(a)). The timing of control is through the output block (Figure 4(b)). When a PU is in the state of being ready to take any signals from its neighboring PUs, its output block will turn on the switch (through the interrupt block) by sending it a high-voltage signal. This indicates that any signal from its neighboring PU is allowed to change the state of a PU. However, the switch will be turned off if a PU is either in the state of processing a neighboring signal or in the refractory state.

2.3.3. Signal Integration. As mentioned earlier, there are three types of PUs for transmitting signals. Our implementation of signal integration is that, to fire a neuron, it requires at least two different types of signals to rendezvous at a PU within a short period of time. In other words, a PU serves as a signal integrator that combines different signals in space and time. To capture this feature, two hypotheses are used. The first is that different PU types have different transmission speeds. The second hypothesis is that an activated PU can influence the state of its neighboring PU through MAP

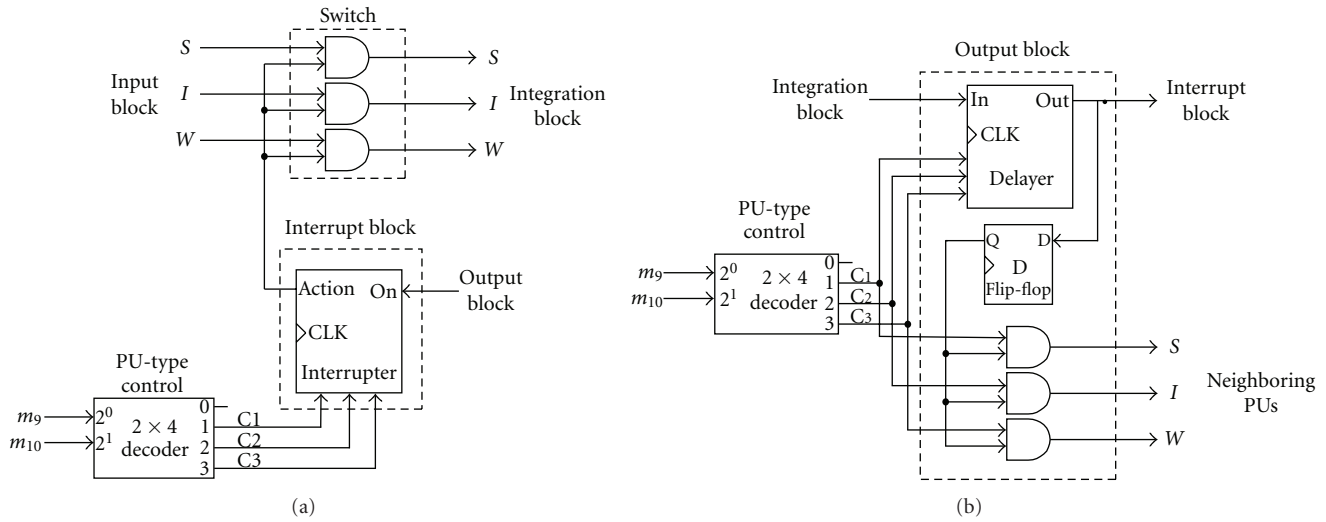


Figure 4: The interrupt block (a) and the output block (b).

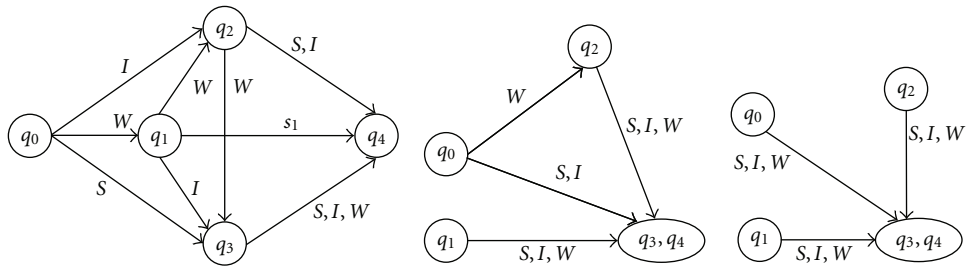


Figure 5: Transition rules of a PU. S, I, and W indicate a signal from a highly activated C1-, C2-, and C3-type PU, respectively. For example, if C1-type PU in the state q_0 receives an I signal it will enter the moderately activated state q_2. If it receives a W signal it will enter the more activated state q_3. For example, a C1-type PU in the state q_0 will enter the less active state q_1 when it receives a signal from a neighboring C3-type PU (i.e., signal W). We note that if a PU does not receive a stimulus before its next update time, it will go into state q_1 if it was in state q_2, or enter state q_0 if it was in q_1.

linking them together. That is, the latter will make a state transition when it receives a signal from the former.

We assume that a PU has six possible states: quiescent (q_0), active with increasing levels of activity (q_1, q_2, q_3, and q_4), and refractory (q_r). Certainly, the complexity of intraneuronal dynamics will be greater when a larger number of PU states are allowed. Correspondingly, it will increase the complexity of the hardware design. We note that six states are sufficient for present use. The following describes the transition rules of each PU. A PU in the highly active state (q_3 or q_4) will return to the refractory state (q_r) at its next update time, and then go into the quiescent state (q_0) at the following next update time. The next state for a less active PU (q_0, q_1, or q_2) depends on the sum of all stimuli received from its active neighboring PUs (Figure 5). If a PU does not receive a stimulus before its next update time, it will go into state q_1 if it was in state q_2, or enter state q_0 if it was in q_1.

In the present implementation, a signal traveling along C1-type PUs has the slowest speed, but also has the greatest degree of influence on the other two PU types. In contrast, a signal traveling along C3-type PUs has the fastest speed, but also has the least degree of influence on the other two PU types. The speed and the degree of influence of a C2-type signal are between those of a C1- and C3-type signal. We note that the degrees of influence between two different PU types are asymmetrical. For example, a C1-type PU in the state q_0 will enter the less active state q_1 when it receives a signal from a neighboring C3-type PU (i.e., signal W). By contrast, a C3-type PU in the state q_0 will enter a highly activated state q_3 if it receives a signal from a neighboring C1-type PU (i.e., signal S). Roughly speaking, the signal with the greatest degree of influence serves as the major signal flow in a neuron while the other two types of signals provide modulating effects.

The integration block is the major component of the ANM design that integrates signals transmitting in space and time (Figure 6). The comparator is responsible for processing either an external signal linked with its application domain or a cytoskeletal signal from its neighboring PU. For each external signal sent to a PU, we assume that the latter will directly go into a highly active state (q_3) if there is a readin enzyme sitting at the same site. This allows the initiation of a new cytoskeletal signal. As to a signal sending from its

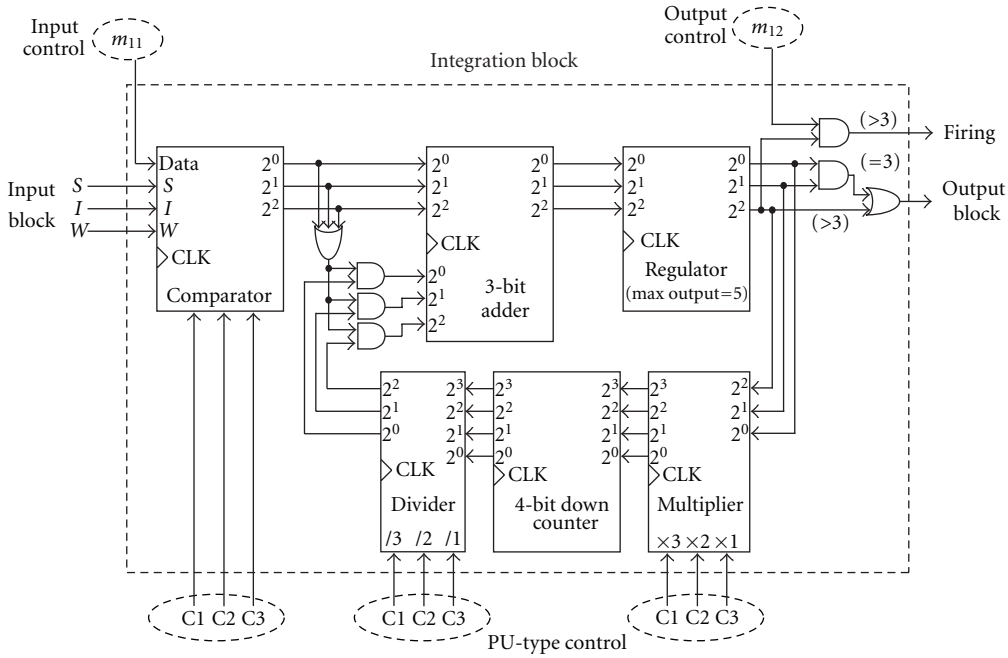


Figure 6: Conceptual architecture of the integration block.

neighboring PU, the comparator determines the state change of a PU. The adder serves as a recorder that keeps a record of a PU's present state. Different numbers represent different states. Different types of PUs have different transmission speeds. The present implementation is that the ratio of transmission speeds of C1-, C2-, and C3-type PUs is 1 : 2 : 3. To comply with the transmission speeds, the ratio of state update times of C1-, C2-, and C3-type PUs is 3 : 2 : 1. Our hardware implementation of speed control comprises a multiplier, a downcounter, and a divider. The multiplier magnifies a signal sent from the regulator and generates an output for the downcounter. The degree of magnification is determined by its update time. For each clock, the downcounter decreases by one. The divider restores the magnified signal in the downcounter back to its original range.

All of the digital circuit modules were design with Verilog using Quartus II software, a digital circuit design tool developed by the Altera Corporation (San Jose, CA). The final design of circuits were downloaded into an FPGA device which produced by the Altera Corporation.

2.4. Interneuronal Architecture-Orchestral Learning. The reference neuron scheme is basically a Hebbian model, in which the connection between two neurons is strengthened when they are active simultaneously. This model also has a hierarchical control feature. With this feature, reference neurons are capable of assembling cytoskeletal neurons into groups for performing specific tasks. Orchestration is an adaptive process mediated by varying neurons in the assembly which selects appropriate combinations of neurons to complete specific tasks. Currently, cytoskeletal neurons are divided into a number of comparable subnets. By comparable subnets, we mean that neurons in these subnets are similar in terms of their inter-neuronal connections and intraneuronal structures. Neurons in different subnets that have similar inter-neuronal connections and intraneuronal structures are grouped into a bundle.

Two levels of reference neurons are used to manipulate these bundles of neuron. The two levels form hierarchical control architecture (Figure 7). The first is referred to as the low-level reference neurons that directly control the bundles of cytoskeletal neurons. Each of these controls a specific bundle (we note that only the bundles activated by the reference neurons are allowed to perform information processing). The second level is referred to as the high-level reference neurons that play the role of grouping the low-level reference neurons. The activation of a high-level reference neuron will fire all of the low-level reference neurons that it controls, which in turn will activate some of these bundles of cytoskeletal neurons (i.e., neurons in different subnets that have similar intraneuronal structures). For example, when R_2 fires, it will fire r_1 and r_{32}, which in turn causes E_1 and E_{32} in each subnet to fire.

The connections among low-level reference neurons and cytoskeletal neurons are fixed. However, the connections between high-level reference neuron and low-level reference neuron layers are subjected to change during evolutionary learning. The above process is called orchestral learning.

2.5. Evolutionary Learning. Processing units are responsible for transmitting and integrating cytoskeletal signals. Evolution at the level of PU configurations is implemented by copying (with mutation) the PU configurations of neurons in the best-performing subnets to those of comparable neurons in the lesser-performing subnets. Variation is implemented by varying the PU configurations during the copy procedure.

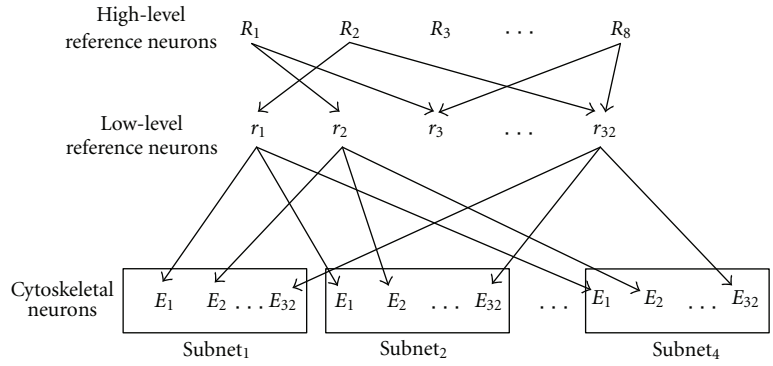


Figure 7: Hierarchical inter-neuronal control architecture.

(1) **Generate** at random the initial *MAP*, *PU-Type*, *readin enzyme*, and *readout enzyme* patterns of each neuron in the reproduction subnet. Each neuron is denoted by neuron (s, b) where s is the subnet number and b is the bundle number.
(2) **Copy** the *MAP*, *PU-Type*, *readin enzyme*, and *readout enzyme* patterns of each neuron in the reproduction subnet to those of comparable neurons in the competition subnets.

$$\text{Copy neuron } (5, b) \text{ to } \left\{ \begin{array}{c} \text{neuron } (1,b) \\ \text{neuron } (2,b) \\ \text{neuron } (3,b) \\ \text{neuron } (4,b) \end{array} \right\}, \text{ for } b = 1, 2, \ldots, 32$$

(3) **Vary** the *MAP* pattern of each neuron in the first subnet, the *PU-type* pattern in the second subnet, the *readin enzyme* pattern in the third subnet, and the *readout enzyme* pattern in the fourth subnet.

$$\text{Vary} \left\{ \begin{array}{c} \text{the } MAP \text{ pattern of neuron } (1,b) \\ \text{the } PU\text{-}Type \text{ pattern of neuron } (2,b) \\ \text{the } readin\ enzyme \text{ pattern of neuron } (3,b) \\ \text{the } readout\ enzyme \text{ pattern of neuron } (4,b) \end{array} \right\}, \text{ if } U \leq P, \text{ for } b = 1, 2, \ldots, 32$$

where P is the mutation rate and U is a random number generated between 0 and 1.
(4) **Evaluate** the performance of each competition subnet and select the best-performing subnet.
(5) **Copy** the *MAP*, *PU-Type*, *readin enzyme*, and *readout enzyme* patterns of each neuron in the best-performing subnet to those of comparable neurons in the reproduction subnets, if the former shows better performance than the latter.
(6) **Go to Step2** unless the stopping criteria are satisfied.

Algorithm 1: Evolutionary learning algorithm.

We note that different PU configurations exhibit different patterns of signal flows.

In the present implementation, the ANM system has 256 cytoskeleton neurons, which are divided into eight comparable subnets. As we mentioned earlier, comparable subnets are similar in terms of their inter-neuronal connections and in-traneuronal structures. Thus, they also can be grouped into 32 bundles. The copy process occurs among neurons in the same bundle. The initial patterns of readin enzymes, readout enzymes, MAPs, and PU-types of the reproduction subnet are randomly decided. That is, the initial value of each bit is randomly assigned as 0 or 1. The evolutionary learning algorithm is shown in Algorithm 1. Note that the mechanism controlling the evolutionary process does not have to be so rigid. Instead, there are several possible alternatives to train this system. In this study, we simply pick out one of these alternatives and precede our experiments. In the future it would be interesting to investigate the impacts of varying the number of learning cycles assigned to each level and the level opening sequence on the learning.

Evolution of reference neurons is implemented by copying (with mutation) the patterns of low-level reference neuron activities loaded by the most fit high-level reference neurons to less fit high-level reference neurons (details can be found in [14]). The copying process is implemented by activating a most fit high-level reference neuron, which in turn reactivates the pattern of low-level reference neuron firing. This pattern is then loaded by a less fit high-level reference neuron. Variation is implemented by introducing noise into the copying process. Some low-level reference neurons activated by a most fit high-level reference neuron may fail to be loaded by a less fit high-level reference neuron. Or some low-level reference neurons that are not activated may fire and be "mistakenly" loaded by a less fit high-level reference neuron. In the present implementation, evolutionary learning at the reference neuron level is turned

off, as we have not yet implemented it on digital circuits. The realization of the evolutionary learning on digital circuit at the reference neuron level is definitely a must-do step, but undoubtedly a complicated job. But in the present implementation, we are not ready to fulfill the design and prefer to focus our study on the internal dynamics, instead of the interneuronal dynamics.

3. Input/Output Interface and Application Domain

We applied the chip to the IRIS dataset, one of the best known datasets found in the pattern recognition literature. The dataset was taken from the machine learning repository at the University of California, Irvine. The dataset contains 3 classes (Iris Setosa, Iris Versicolour, Iris Virginica) of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other two; the latter are not linearly separable from each other. There are four parameters in each instance: sepal length, sepal width, petal length, and petal width.

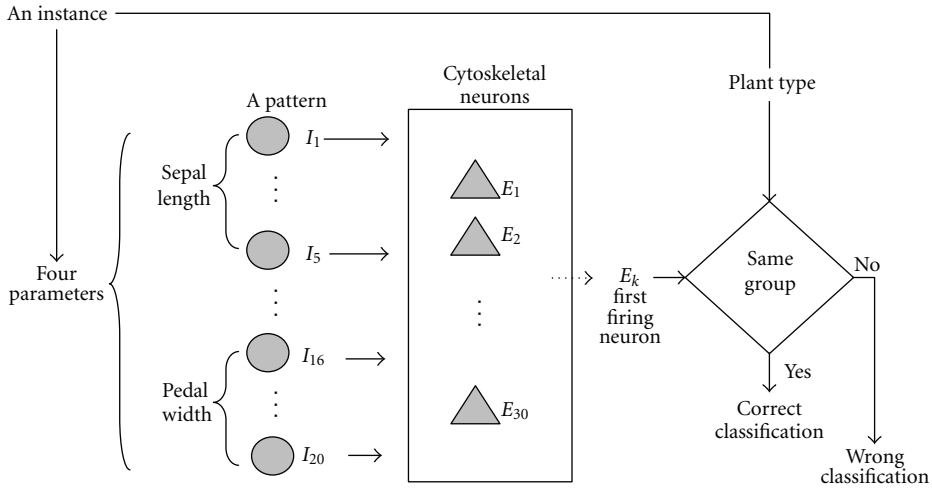
The initial connections between these 4 parameters and cytoskeletal neurons were randomly decided, but subject to change as learning proceeded. Through evolutionary learning, each cytoskeletal neuron was trained to be a specific input-output pattern transducer. That is, each of these neurons became responsible for processing only a small subset of stimuli generated from these 4 parameters. We used five bits to encode each of these 4 parameters. In total, there were 20 bits required to encode all of them. For each parameter, the minimal and maximal values of these 150 instances were determined (to be denoted by MIN and MAX, resp.), and the difference between these two values was divided by 5 (to be denoted by INCR). The transformation of each actual parameter value (to be denoted by ACTUAL) into the corresponding 5-bit pattern was shown to be

$$\begin{aligned}
&00001, \quad \text{if} \quad \text{MIN} \le \text{ACTUAL} < (\text{MIN} + \text{INCR}),\\
&00010, \quad \text{if} \quad (\text{MIN} + \text{INCR}) \le \text{ACTUAL} < (\text{MIN} + \text{INCR} \times 2),\\
&00100, \quad \text{if} \quad (\text{MIN} + \text{INCR} \times 2) \le \text{ACTUAL} < (\text{MIN} + \text{INCR} \times 3),\\
&01000, \quad \text{if} \quad (\text{MIN} + \text{INCR} \times 3) \le \text{ACTUAL} < (\text{MIN} + \text{INCR} \times 4),\\
&10000, \quad \text{if} \quad (\text{MIN} + \text{INCR} \times 4) \le \text{ACTUAL} \le \text{MAX}.
\end{aligned} \tag{1}$$

Each bit corresponded to a specific pattern of stimuli for cytoskeletal neurons. All cytoskeletal neurons that had connections with a specific bit would receive the same pattern of stimuli simultaneously. When a readin enzyme received an external stimulus, a cytoskeletal signal was initiated. It was randomly decided in the beginning and subject to change during the course of learning as to which readin enzymes of a neuron would receive the stimuli from a parameter. For each instance, all stimuli were sent to cytoskeletal neurons simultaneously. In other words, all cytoskeletal signals were initiated at the same time. The cytoskeleton integrated these signals in space and time. For each instance, the class of the first firing cytoskeletal neuron was assigned as its output. Cytoskeletal neurons were equally divided into three classes, corresponding to these three different groups of instances. For each instance, we defined that the chip made a correct response when the class of the first firing neuron was in accordance with the group shown in the dataset (Figure 8). The ANM design was tested with each of these 150 instances in sequence. The greater the number of correct responses made by the chip, the higher its fitness.

4. Experimental Results

4.1. Evolvability. The proposed hardware architecture incorporated several parameters PU, MAP, readin, and readout (denoted as P, M, I, and O, resp.) that allowed us to turn them on or off independently for evolutionary learning. We first study the manner in which problem-solving capability (or evolvability) depended on each level of parameter changes. And then we investigated the effects of increasing the number of evolutionary learning parameters (levels) opened for evolution. For each case of parameter changes (to be described later), five runs were performed. For each run, 100 out of these 150 instances in the IRIS set were selected at random as the training set whereas the remaining 50 instances were grouped as the testing set. All the results reported below were the average differentiation rates of five runs. We first trained the chip for 1200 cycles (at which point learning appeared to slow down significantly). Then, the chip after learning for 1200 cycles was tested with the testing set. The IRIS dataset includes 150 instances. In the evolvability experiment, the dataset were performed for five runs. For each run, the IRIS dataset was randomly divided into two parts: training set and testing set. The training set includes 100 instances and testing set 50 instances. We trained the chip for 1,200 cycles per run. It takes about 30 seconds to complete a learning cycle. The total running time of this experiment takes about 50 hours (30 s $*$ 1200 cycles $*$ 5 runs). When compared with BP neural network and SVM, the time needed to perform the experiment with our system is much longer than we expect. This is because the whole system has been simulated in a computer that simulation has to be performed in a step-by-step manner. When the hardware design has been totally realized with a real digital chip, it might take only microsec-onds to accomplish the assigned tasks.



($E_{1–10}$: Iris Setosa; $E_{11–20}$: Iris Versicolour; $E_{21–30}$: Iris Virginica)

Figure 8: Interface of the ANM design with the IRIS dataset.

To investigate the significance of each parameter, we first allowed only one parameter to change during the course of learning whereas evolution at the other three levels was turned off. In total, there were four experiments performed. Among these four parameters, the chip after learning when only readin enzymes were allowed to evolve alone achieved the highest recognition rate (i.e., 91.6%), when only PU types were allowed the second highest (i.e., 90.0%), when only MAPs were allowed the third highest (i.e., 84.4%), and when only readout enzymes were allowed the lowest rate (83.6%). This provided us some preliminary information about which levels of parameter changes might be friendlier to evolution than others.

The following experiment was to study the manner in which problem-solving capability (or evolvability) depended on different combinations of parameter changes. We first increased the number of parameter changes to two (i.e., two parameters were allowed to evolve simultaneously). There were six combinations of two parameter changes. As shown in Figure 9, the chip after learning when two levels were allowed to evolve simultaneously achieved higher recognition rates than when only one level was allowed to evolve alone. For example, the recognition rate was higher when PUs and MAPs were allowed to evolve simultaneously than when either PUs or MAPs were allowed to evolve alone. This implied that each level of parameter changes more or less contributed in facilitating evolutionary learning, and that synergy occurred among different levels of learning. We then performed the experiment that allowed three parameters to evolve at the same time. There were four possible combinations when three parameters were allowed to evolve simultaneously. The chip after learning when PMI (PU, MAP, and readin enzyme) were allowed to change at the same time achieved the best recognition rate (94.0%) among these four possible combinations (note that a parameter that was not opened for evolutionary changes will be held constant during the course of learning). The implication was that synergies among different levels of evolution became more important as more levels of parameter changes were allowed; implying that learning at one level opened up opportunities for another. However, it did not necessarily mean that learning with more levels of evolutionary changes could always achieve effective performance, in particular when the limitation of learning time was imposed. This was because the power of a multilevel system was not attained by simply summing up the contributions of each constituting element together, but by developing the synergy that might occur among the interactions of different levels. Learning with more levels of evolutionary changes would enhance the repertoire of the system, but did not guarantee that effective learning could be achieved with a limited amount of time. When we looked into what levels (operators) of evolution contributed to the learning progress and how the interactions occurring between different levels exerted control over the tempo of evolution, the result showed that each operator more or less contributed to learning progress, and that learning proceeded in an alternate manner. That is, synergies percolated through different combinations of evolutionary learning operators, implying that learning at one level opened up opportunities for another. We noted that synergy was more likely to occur when a comparatively small number of parameter changes was involved. However, synergy occurred only in a selective manner when more parameter changes were involved.

4.2. Comparison with Other Neural Models. For comparison we applied SVM and BP to the same training and testing sets. As above, five runs were performed. In the former model the average differentiation rates of the training and testing sets were 95.6% and 91.9%, respectively, while in the latter were 96.7% and 91.7, respectively. The above result suggested that the chip had performance comparable to either BP or SVM (Table 1).

4.3. Noise Tolerance. This experiment was to test the capability of the ANM design after substantial learning in tolerating

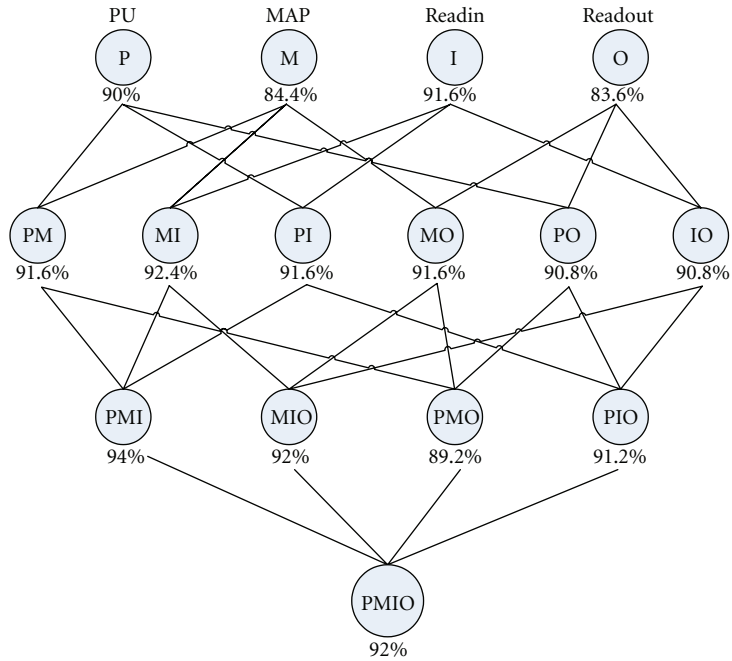


Figure 9: Learning performance of each learning mode.

Table 1: Average differentiation rates of different models with the Iris dataset.

Model	Training	Testing
BP neural network	96.7%	91.7%
Support vector machine	95.6%	91.9%
ANM	94.8%	94.0%

Table 2: Effect of increasing the degree of noise on the rate of the TCD value growth.

Level of variations imposed	5.0%	10.0%
Rate of the TCD value increased	5.7%	11.1%

noise. Through gradually increasing the degree of noise, we observed the input/output relationship of the ANM design. For each test, we kept the system's structure unchanged but varied the pattern of signals sent to cytoskeletal neurons. If the system's outputs changed gradually with the extent of the increase in pattern variations, this in part supported that the system's structure embraced some degree of noise tolerance capability. The ANM design trained for 1,200 cycles was used.

In the following, we first tested the system with spatial noise imposed on the training patterns. To generate a test set, we made a copy of the training set, but altered some bits during the copy process (changing a bit into "1" if it was "0" and into "0" if it was "1"). Five levels of variations were imposed during the copy process: 5%, 10%, 15%, 20%, and 25%. For example, at the 5% level of variations, we mean that each bit has a 5% possibility of being altered. For each level of variations, ten test sets were generated. The total clock difference (TCD) value is the measure pointer indicates the difference of firing time in the circuit. The TCD value increased slightly as we imposed a 5% level of variations on the patterns. Even when we increased the variation level to 25%, the system still demonstrated acceptable results. The TCD value was increased from 38 to 310 at the 5% level of variation (i.e., increased 272), to 569 at the 10% level of variation (i.e., increased 531), to 633 at the 15% level of variation (i.e., increased 625), to 626 at the 20% level of variation (i.e., increased 588), and to 1030 at the 25% level of variation (i.e., increased 992). If we divided the increments by the TCD value before learning (i.e., 4781), the rate of the TCD values increased gradually as we augmented the noise levels (Table 2). This implied that the system had good noise tolerance capability in dealing with spatial noise.

5. Conclusions

When a system is running in the real world, it is inevitable to be confronted with noise generated either from the environment or the system itself. When noise is made only temporarily, structural changes of a system may not be necessary (i.e., a system may ignore this noise). By contrast, a system is required to alter its structure in responding to this noise if it leads to a permanent change in the environment. In such a case, a system has to learn in a moving landscape when environmental change occurs from time to time. Two main results are obtained in the noise tolerance experiment. One is that learning is more difficult in a noisy environment than in a noiseless environment, and that the system is able to learn continuously when noise is made in a temporary manner. The other result is that the system demonstrates

a close structure/function relation. We note that noise occurring at different levels of the cytoskeletal structure has different degree of influence on its outputs. As we gradually modify the structure, its outputs change accordingly. On the other hand, we examine the system's outputs by gradually imposing noise in space and time on its input patterns. The output changes gradually (i.e., proportionally) as the degree of noise is increased. An interesting result is that its system's output does not necessarily change accordingly as we increase the degree of noise generated in time. Note that delaying a signal may alter a neuron's firing activity. However, this may not be true when several signals are delayed simultaneously as these signals may integrate at a later time (undoubtedly, this will delay its firing timing). The above results demonstrate that this system has good noise tolerance capability in dealing with spatiotemporal changes in its inputs, implying that it possesses an adaptive surface that facilitates evolutionary learning. With this feature, the ANM design can be applied to various real-world problems. We note that the ability to separate patterns is clearly a prerequisite for pattern recognition. However, it is equally important to recognize a family of patterns that are slightly varied in space and time. If a system is trained on a particular training set, any ability that it has to respond correctly to noise induced variations in this set will be a form of generalization. The manner of generalization depends on its integrative dynamics (i.e., the flow of signals in the cytoskeleton). This is directly or indirectly influenced by a neuron's PU configuration. Generally speaking, the input patterns recognized by a neuron with internal dynamics will be generalized in a more selective way than simple threshold neurons.

Acknowledgment

This paper was supported in part by the R.O.C. National Science Council (Grant NSC 98-2221-E-224-018-MY3).

References

[1] M. Conrad, "Bootstrapping on the adaptive landscape," *BioSystems*, vol. 11, no. 2-3, pp. 167–182, 1979.

[2] M. Conrad, "The geometry of evolution," *BioSystems*, vol. 24, no. 1, pp. 61–81, 1990.

[3] M. Sipper and E. M. A. Ronald, "A new species of hardware," *IEEE Spectrum*, vol. 37, no. 3, pp. 59–64, 2000.

[4] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti, "Toward robust integrated circuits: the embryonics approach," *Proceedings of the IEEE*, vol. 88, no. 4, pp. 516–540, 2000.

[5] T. Higuchi, M. Iwata, D. Keymeulen et al., "Real-world applications of analog and digital evolvable hardware," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 3, pp. 220–234, 1999.

[6] T. Higuchi and N. Kajihara, "Evolvable hardware chips for industrial applications," *Communications of the ACM*, vol. 42, no. 4, pp. 60–66, 1999.

[7] M. Murakawa, S. Yoshizawa, I. Kajitani et al., "The GRD chip: genetic reconfiguration of DSPs for neural network processing," *IEEE Transactions on Computers*, vol. 48, no. 6, pp. 628–639, 1999.

[8] H. de Garis, "An artificial brain ATR's CAM-Brain Project aims to build/evolve an artificial brain with a million neural net modules inside a trillion cell Cellular Automata Machine," *New Generation Computing*, vol. 12, no. 2, pp. 215–221, 1994.

[9] H. de Garis, "Review of proceedings of the first NASA/Dod workshop on evolvable hardware," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 304–306, 1999.

[10] J. Torresen, "A divide-and-conquer approach to evolvable hardware," in *Proceedings of the 2th International Conference on Evolvable Systems: From Biology to Hardware*, vol. 1478 of *Lecture Notes in Computer Science*, pp. 57–65, Lausanne, Switzerland, 1998.

[11] M. Conrad, R. R. Kampfner, K. G. Kirby et al., "Towards an artificial brain," *BioSystems*, vol. 23, no. 2-3, pp. 175–218, 1989.

[12] J. C. Chen and M. Conrad, "A multilevel neuromolecular architecture that uses the extradimensional bypass principle to facilitate evolutionary learning," *Physica D*, vol. 75, no. 1–3, pp. 417–437, 1994.

[13] S. Rasmussen, H. Karampurwala, R. Vaidyanath, K. S. Jensen, and S. Hameroff, "Computational connectionism within neurons: a model of cytoskeletal automata subserving neural networks," *Physica D*, vol. 42, no. 1–3, pp. 428–449, 1990.

[14] I. Baradavka and T. Kalganova, "Assembling strategies in extrinsic evolvable hardware with bidirectional incremental evolution," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2610, pp. 276–285, 2003.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 509465, 15 pages
doi:10.1155/2012/509465

Review Article

Networks on Chips: Structure and Design Methodologies

Wen-Chung Tsai,[1] Ying-Cherng Lan,[1] Yu-Hen Hu,[2] and Sao-Jie Chen[3]

[1] *Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan*
[2] *Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI 53706-1691, USA*
[3] *Department of Electrical Engineering and Graduate Institute of Electronics Enginering, National Taiwan University, Taipei 106, Taiwan*

Correspondence should be addressed to Sao-Jie Chen, csj@cc.ee.ntu.edu.tw

Received 18 September 2011; Accepted 1 October 2011

Academic Editor: Jiang Xu

The next generation of multiprocessor system on chip (MPSoC) and chip multiprocessors (CMPs) will contain hundreds or thousands of cores. Such a many-core system requires high-performance interconnections to transfer data among the cores on the chip. Traditional system components interface with the interconnection backbone via a bus interface. This interconnection backbone can be an on-chip bus or multilayer bus architecture. With the advent of many-core architectures, the bus architecture becomes the performance bottleneck of the on-chip interconnection framework. In contrast, network on chip (NoC) becomes a promising on-chip communication infrastructure, which is commonly considered as an aggressive long-term approach for on-chip communications. Accordingly, this paper first discusses several common architectures and prevalent techniques that can deal well with the design issues of communication performance, power consumption, signal integrity, and system scalability in an NoC. Finally, a novel bidirectional NoC (BiNoC) architecture with a dynamically self-reconfigurable bidirectional channel is proposed to break the conventional performance bottleneck caused by bandwidth restriction in conventional NoCs.

1. Introduction

As the density of VLSI design increases, the complexity of each component in a system raises rapidly. To accommodate the increasing transistor density, higher operating frequencies, and shorter time-to-market pressure, multiprocessor system on chip (MPSoC) and chip multiprocessor (CMP) architectures, which use bus structures for on-chip communication and integrate complex heterogeneous functional elements on a single die, are more and more required in today's semiconductor industry. However, today's SoC designers face a new challenge in the design of the on-chip interconnects beyond the evolution of an increasing number of processing elements. Traditional bus-based communication schemes, which lack for scalability and predictability, are not capable to keep up with the increasing requirements of future SoCs in terms of performance, power, timing closure, scalability, and so on. To meet the design productivity and signal integrity challenges of next-generation system designs, a structured and scalable interconnection architecture, network on chip (NoC), has been proposed recently to mitigate the complex on-chip communication problem.

An application can be represented as a set of computational units that require a set of communication blocks to pass information between the units. To distinguish the performance impact of these two major components, computation time is dominated by gate delay whereas communication time is dominated by wire delay. When the amount of computational units is low, the communication blocks can be done on an ad-hoc basis. However, with the shrinking size of transistors in recent years, gate delay is ever decreasing with respect to wire delay. Thus, we need a structured and scalable on-chip communication architecture to fit the increasingly complex applications on a single chip. This translates to the design of on-chip communications architecture as being more and more important and promotes the design concept from computation-centric design to communication-centric design.

System on chip (SoC) is an architectural concept developed in the last few decades, in which a processor or few

processors along with memory and an associated set of peripherals connected by busses are all implemented on a single chip. According to the Moore's law, the trend toward many-core processing chips is now a well established one. Power-efficient processors combined with hardware accelerators are the preferred choice for most designers to deliver the best tradeoff between performance and power consumption, since computational power increases exponentially according to the calculation of dynamic power dissipation [1]. Therefore, this trend dictates spreading the application tasks into multiple processing elements where (1) each processing element can be individually turned on or off, thereby saving power, (2) each processing element can run at its own optimized supply voltage and frequency, (3) it is easier to achieve load balance among processor cores and to distribute heat across the die, and (4) it can potentially produce lower die temperatures and improve reliability and leakage. However, while ad-hoc methods of selecting few blocks may work based on a designer's experience, this may not work as today's MPSoC and CMP designs which becomes more and more complex. Consequently, SoC design nowadays needs techniques which can provide an efficient method of enabling a chip to compute complex applications and to fit area-wise on a single chip according to today's technology trends.

A communication scheme is composed of an interconnection backbone, physical interfaces, and layered protocols which make the on-chip communication take place among components on a MP-SoC or CMP. As the design complexity scales up, intrachip communication requirements are becoming crucial. Data-intensive systems such as multimedia devices, mobile installations, and multiprocessor platforms need a flexible and scalable interconnection scheme to handle a huge amount of data transactions on chip. Customarily, dedicated point-to-point wires are adopted as sets of application-specific global on-chip links that connect the top-level modules. However, as wire density and length grow with the system complexity, the communication architecture based on point-to-point wires becomes no more feasible due to its poor scalability and reusability. Specifically, as signals are carried by the global wires across a chip, these metal wires typically do not scale in length with technology. Propagation delay, power dissipation, and reliability will be the serious issues of global wires in deep submicron VLSI technology. According to [2], as silicon technologies advance to 50 nm and beyond, global wires will take 6 to 10 cycles to propagate, which will then far outweigh gate delays and make cross-chip long wire timing difficult to meet. Keeping track of the status in all elements and managing the global communication among top-level modules by a centralized way are no longer feasible. Therefore, reusable on-chip bus interconnect templates such as ARM's AMBA [3] and IBM's CoreConnect [4] are commonly used in current MP-SoC amd CMP designs, such that the modules can share the same group of interconnection wires in a bus-based communication architecture.

However, on-chip bus allows only one communication transaction at a time according to the arbitration result; thus, the average communication bandwidth of each processing element is in inverse proportion to the total number of IP cores in a system. This character makes a bus-based architecture inherently not scalable for a complex system in today's MP-SoC and CMP designs. Implementing multiple on-chip buses in a hierarchical architecture or in a separated manner may alleviate this scalability constraint, but it requires application-specific grouping of processing elements and design of different communication protocols to meet the application requirements. Furthermore, whenever a new application needs to be designed for, or a new set of peripherals needs to be added, a chip designed with only simple buses will lack means of efficiently determining feasibility, not to mention optimality [5]. In addition, attempts to guarantee quality of service (QoS) for system performance will be a manually intensive task. Therefore, bus-based design needs to be exchanged with a method that is flexible, scalable, and reusable.

Since the latest process technology allows for more processors and more cores to be placed on a single chip, the emerging MP-SoC and CMP architectures, which demand high throughput, low latency, and reliable global communication services, cannot be met by current dedicated bus-based on-chip communication infrastructure. Trying to achieve such designs with a bus structure could be problematic for a number of reasons including timing closure, performance issues, and scalability. Specifically, as the feature size of modern silicon devices shrinks below 50 nanometers, global interconnection delays constrain attainable processing speed. Device parameter variations further complicate the timing and reliability issues. A paradigm shift focusing on communication-centric design, rather than computation-centric design, seems to be the most promising approach to address these communication crises [6–11]. Consequently, in the past few years, a new methodology called network on chip has been introduced as a means of solving these issues by introducing a structured and scalable communication architecture.

In the sequel, Section 2 will introduce the NoC architecture and its function layers. In Section 3, we will discuss the NoC design methodologies. Then, a bidirectional network-on-chip (BiNoC) architecture will be given in Section 4. Finally, conclusion will be drawn in Section 5.

2. Network-on-Chip Architecture and Function Layers

Network on chip is the term used to describe an architecture that has maintained readily designable solutions in face of communication-centric trends. In this section, we will briefly review some concepts on the design of an NoC communication system. Moreover, the NoC function can be classified into several layers, which will be introduced sequentially.

2.1. Network-on-Chip Architecture. A typical NoC architecture consists of multiple segments of wires and routers as shown in Figure 1. In a tiled, city-block style of NoC layout, the wires and routers are configured much like street grids of a city, while the clients (e.g., logic processor cores) are placed on city blocks separated by wires. A network interface (NI) module transforms data packets generated from the client

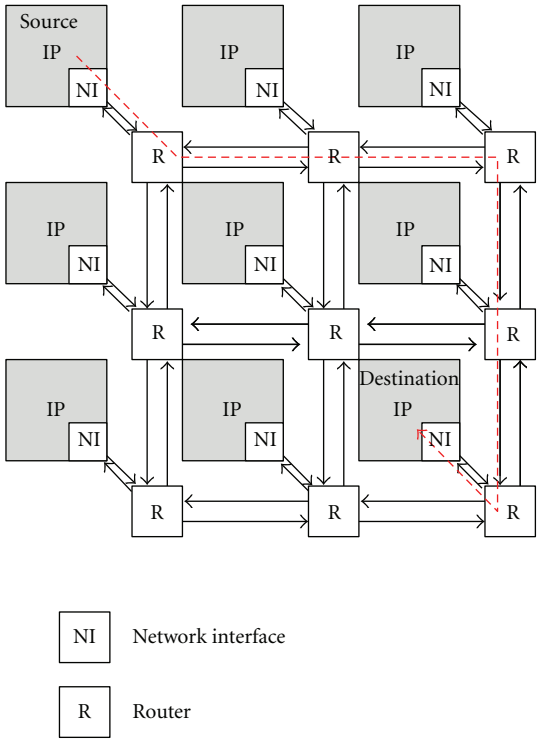


Figure 1: Typical NoC architecture in a mesh topology.

logic (processor cores) into fixed-length flow-control digits (flits). The flits associated with a data packet consist of a header (or head) flit, a tail flit, and a number of body flits in between. This array of flits will be routed toward the intended destination in a hop-by-hop manner from one router to its neighboring router.

In a city-block style NoC, each router has five input ports and five output ports corresponding to the north, east, south, and west directions as well as the local processing element (PE). Each port will connect to another port on the neighboring router via a set of physical interconnect wires (channels). The router's function is to route flits entering from each input port to an appropriate output port and then toward the final destinations. To realize this function, a router is equipped with an input buffer for each input port, a 5×5 crossbar switch to redirect traffic to the desired output port and necessary control logic to ensure correctness of routing results as shown in Figure 2.

Usually, for each data packet, the corresponding head flit specifies its intended destination. After examining the head flit, the router control logic will determine which output direction to route all the subsequent (body and tail) flits associated with this data packet according to the routing algorithm applied.

2.2. Network-on-Chip Function Layers. The NoC function can be classified into several layers: application, transport, network, data link, and physical layers. An NoC router should contain both software and hardware implementations to support functions of the layers.

2.2.1. Application Layer. At the application layer, target applications will be broken down into a set of computation and communication tasks such that the performance factors like energy and speed can be optimized. Placement of cores on an NoC has to be optimized to reduce the amount of total communication or energy but at the same time recognizing the limitations of any one particular link. The task mapping and communication scheduling problem is an instance of a constrained quadratic assignment problem which was known to be NP-hard [12]. Given a target application described as a set of concurrent tasks with an NoC architecture, the fundamental questions to answer are (1) how to topologically place the selected set of cores onto the processing elements of the network and (2) how to take into consideration the complex effects of network condition, which may change dynamically during task execution, such that the metrics of interest are optimized [13]. To get the best tradeoff between power and performance, application mapping and scheduling should be considered with several kinds of architecture parameters.

2.2.2. Transport Layer. To prevent buffer overflow and to avoid traffic congestion, some management schemes should be applied to guide the transport of packets in an NoC. The transport layer addresses the congestion and flow control issues [14]. Key performance metrics of an NoC include low packet delivery latency and high-throughput rate, and these metrics are critically impacted by network congestions caused by resource contentions. Accordingly, contention resolution is a key to avoid network congestions [14]. One of the most crucial issues for the contention resolution is, under a premise of a deadlock- and livelock-free routing algorithm, to enhance the utilization efficiency of available network resources in order to come up with a better communication performance.

2.2.3. Network Layer. Network topology or interconnect architecture is an important issue in this layer, which determines how the resources of network are connected, thus, refers to the static arrangement of channels and nodes in an interconnection network. Irregular forms of topologies can be derived by mixing different forms of communication architectures in a hierarchical, hybrid, or asymmetric way by clustering partition, which may offer more connectivity and customizability at the cost of complexity and area. In addition, optimization of a topology, which affects the connectivity of the routers and the distance of any one core to the other, is difficult. Furthermore, the tradeoff between generality and customization that, respectively, facilitate scalability and performance is important. As future designs become more complex, the non-recurring costs of architecting and manufacturing a chip will become more and more expensive. A homogenous NoC is one where the cores and routers are all the same, while a heterogeneous NoC selects individual cores from an IP library and may have its communication architecture customized to suit the needs of an application. Since NoC designs must be flexible enough to cover a certain range of applications, most of the state-of-the-art NoC designs use a mesh or torus topology because of its performance benefits

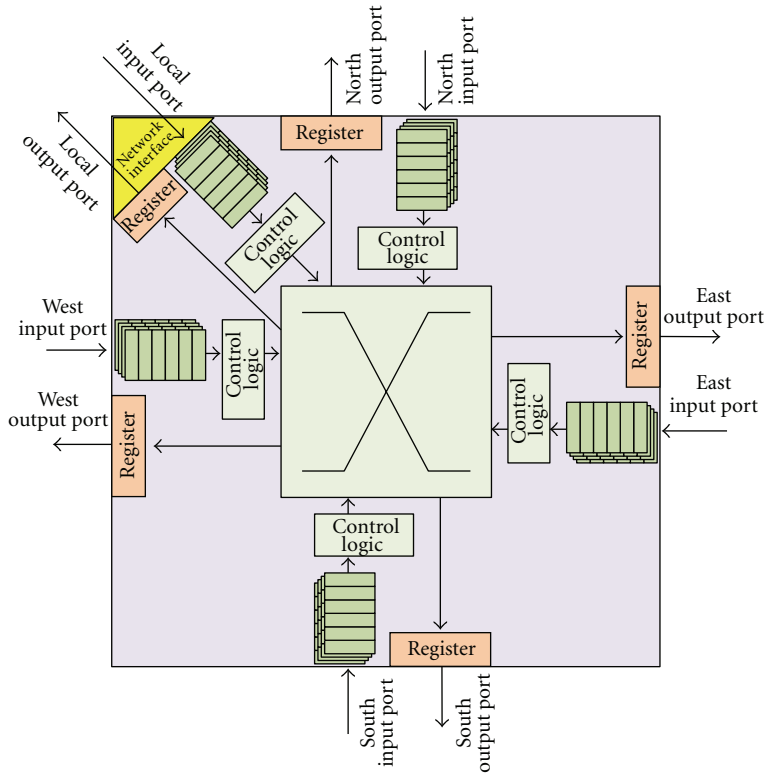


Figure 2: Typical NoC router architecture.

and high degree of scalability for two-dimensional systems, yet it may not achieve the best performance for a single application [15, 16].

In addition, the network layer also needs to deal with the routing data between processing elements. First, packetizing algorithms deal with the decomposition of a message into packets at source nodes and their assembly at destination nodes. Then, the transmission of packets can be executed by the choice of routing algorithms based on different network topologies [6]. Routing algorithm determines the path strategy of a packet from its source node to the destination node. Determining packet routes and resolving conflicts between packets when the same route is requested, with respect to improving on-chip communication performance, are two of the important responsibilities of a router.

Conventional design of a router consists of circuit-switched fabrics and an arbitration controller. In each arbitration decision, more than one path can be constructed by the crossroad switch as long as no contention exists between these paths. For most existing switch designs, virtual-channel flow-control-based router design, which provides better flexibility and channel utilization with smaller buffer size, is a well-known technique from the domain of multiprocessor networks [17–24].

2.2.4. Data Link and Physical Layers. The main purpose of data-link layer protocols is to increase the reliability of the link up to a minimum required level, under the assumption that the physical layer by itself is not sufficiently reliable [14]. The emphasis on physical layer is focused on signal drivers and receivers, as well as design technologies for resorting and pipelining signals on wiring. In addition, as technology advanced to ultradeep submicron (DSM), smaller voltage swings and shrinking feature size translate to decreased noise margin, which cause the on-chip interconnects less immune to noise and increase the chances of nondeterminism in the transmission of data over wires (transient fault) [2, 25–28]. Electrical noise due to crosstalk, electromagnetic interference (EMI), and radiation-induced charge injection will likely produce timing error and data errors and make reliable on-chip interconnect hard to achieve.

Error control schemes and utilization of the physical links to achieve reliability are the main concern of these layers. First, a credible fault model must be developed. Then, an error control scheme that is low power, low area, high bandwidth, and low latency must be designed. In NoC design, packet-based data transmission is an efficient way to deal with data errors because the effect of errors is contained by packet boundaries that can be recovered on a packet-by-packet basis.

3. Network-on-Chip Design Methodologies

This section discusses several prevalent NoC design methodologies, such as flow control, routing, arbitration, quality of service, reliability, and task scheduling.

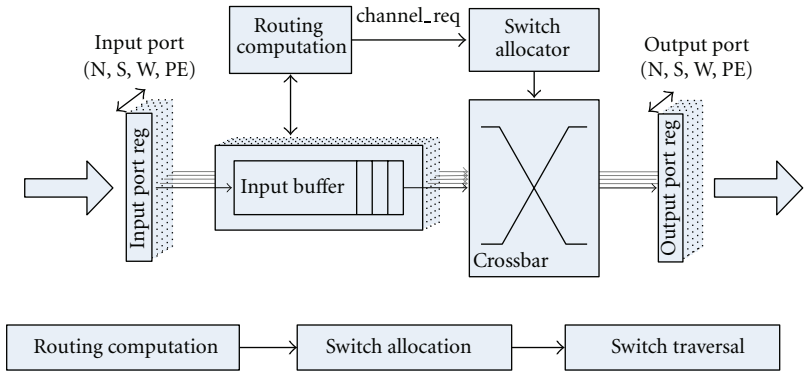


Figure 3: Typical router design based on wormhole flow control.

3.1. Flow-Control Mechanism. The performance of NoC communication architecture is dictated by its *flow-control* mechanism. Adding buffers to networks significantly improves the efficiency of a flow-control mechanism since a buffer can decouple the allocation of adjacent channels. Without a buffer, the two channels must be allocated to a packet (or flits) during consecutive cycles, or the packet must be dropped or misrouted [6]. More specifically, with *buffered flow control*, when a packet arrives at a router, it must first occupy some resources, such as channel bandwidth and buffer capacity, depending on the flow-control methodology. Each router must juggle among multiple input data streams from multiple input ports and route them to appropriate output ports with the highest efficiency.

Buffered flow-control methods can be classified into *packet-buffer flow control* and *flit-buffer flow control* based on their granularity of buffer allocation and channel bandwidth allocation [6]. Since allocating resources in unit of flit can achieve more storage utilization efficiency than that in unit of packet. Two types of *flit-buffer flow-control* architectures are commonly used in NoC: the *wormhole* flow control and the *virtual-channel* flow control.

3.1.1. Packet-Buffer Flow Control. Packet-buffer flow-control allocates network resources in a packet-by-packet basis. Examples are *store-and-forward flow control* and *virtual-cut-through flow control*. In store-and-forward method, each node must ensure that it has already received and stored an entire packet before forwarding it to the downstream node. While the virtual-cut-through scheme can forward a packet as long as there is enough buffer space to receive a packet at the downstream node. As a result, virtual cut through introduces lower communication delay than store and forward does. However, packet-buffer flow control needs larger size of buffer space in one node because of its inefficient use of buffer storage. In addition, allocating channels in units of packets will increase contention latency.

3.1.2. Wormhole Flow-Control-Based Router. Wormhole flow control improves performance through a finer granularity of message allocation at flit level instead of packet level. This technique allows more efficient use of buffer than the packet-buffer flow-control mechanism since the buffer size in each router can be reduced significantly [29, 30]. A typical three-stage pipelined NoC router architecture based on wormhole flow control is shown in Figure 3. Every input port has a FIFO-based input buffer, which can be seen as a single virtual channel used to hold blocked flits. To facilitate wormhole flow-control-based routing [6], the *routing computation* (RC) module will send a channel request signal to the *switch allocator* (SA) for data in each input buffer. If the downstream buffer at a neighboring router has vacant space, SA will allocate the channel and route the data flits through the crossbar switch toward the designated downstream router at the *switch traversal* (ST) stage.

However, wormhole flow-control-based switching technique saves buffer size at the expense of throughput since the channel is owned by a packet, but buffers are allocated on a flit-by-flit basis. As such, an idle packet may continue block a channel even when another packet is ready to use the same channel, leading to inefficient resource utilization. This is the well-known *head of line* (HoL) blocking problem. Therefore, virtual-channel flow-control-based router architecture was proposed to reduce blocking effect and to improve network latency.

3.1.3. Virtual-Channel Flow-Control-Based Router. Virtual-channel flow control assigns multiple virtual paths, each with its own associated buffer queue, to the same physical channel; thus, it increases throughput by up to 40% over wormhole flow control and helps to avoid possible deadlock problems [19, 31, 32]. A virtual channel flow-control router architecture as shown in Figure 4 can be seen as a remedy to the shortcoming of the wormhole flow-control scheme. By multiplexing multiple virtual-channels into the same input buffer, an idle packet will no longer block other packets that are ready to be routed using the shared physical channel. In a typical virtual-channel flow-control-based router, the flits are routed via a four-stage pipeline: *routing computation*, *virtual-channel allocation*, *switch allocator*, and *switch traversal*.

One incoming flit that arrives at a router is first written to an appropriate input virtual-channel queue and waits to

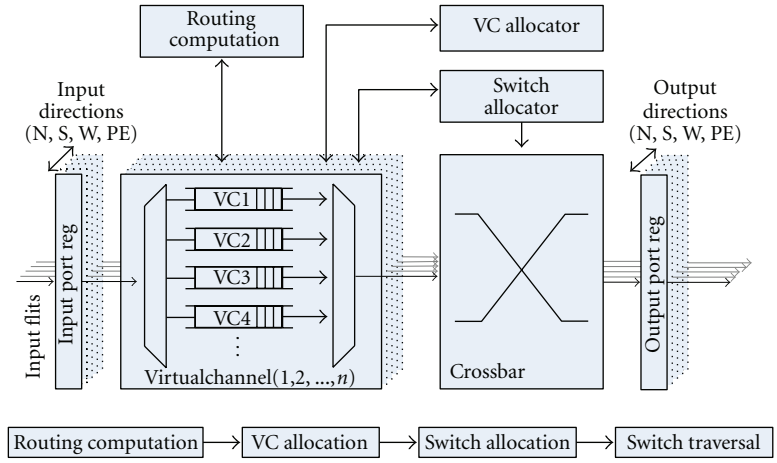


Figure 4: Typical router design based on virtual-channel flow control.

be processed. When a head flit reaches the top of its virtual-channel buffer queue and enters the RC stage, it is decoded by the RC module and generates an associated direction request. The direction request of this flit is then sent to the VA module to attain virtual channel at the downstream router. There might be some contentions among packets that request for the same virtual channel at the downstream router. The loser packets will be stalled at the VA stage, and the following flit in the previous stage will also be blocked due to this contention failure. Note that the processes of RC and VA actually take place only on the head flit. The subsequent body flits and tail flit of a packet simply accede to the routing decision acquired by the head flit and require no further processing at the RC and VA stages. Once a decision on the output virtual-channel selection is made at the VA stage, the SA module will assign physical channels to intrarouter flits. Flits granted with a physical channel will traverse through the crossbar switch to the input buffer of the downstream router during the ST stage, and the process repeats until the packet arrives at its destination.

3.2. Routing and Arbitration Techniques. A general problem pertaining to the routing and arbitration algorithms can be stated as follows: given an application graph, which can be represented by a unique traffic pattern, and a communication architecture, find a decision function at each router for selecting an output port that achieves a user-defined objective function.

3.2.1. Problem Decomposition. The above problem has three main parts: a traffic pattern, an NoC communication architecture, and an algorithm which best satisfies a set of user-defined objectives. First, the traffic patterns known ahead of time can be dealt with by a scheduling algorithm. On the other hand, dynamic or stochastic traffic patterns rely on the use of a routing algorithm with a varying degree of adaptation to route packets. Our focus will be on the patterns not known ahead of time.

Second, NoC communication architectures can have different topologies. The most common one is a regular 2D mesh, frequently used to display the behavior of adaptive routing algorithms. Other work, such as [33], deal with irregular regions in meshes. Our focus is independent of topology.

The third part deals with the algorithms themselves and the objectives to achieve. Two primary algorithms used to determine where and when a packet will move are routing and arbitration. A routing algorithm decides which direction each input packet should travel. Arbitration is the process of deciding which input packet request should be granted when there are more than one input packet requests for the same output port.

3.2.2. State of the Art. A typical router in an NoC is responsible for moving the received packets from the input buffers, with its routing and arbitration algorithms, to the output ports. The decisions which a router makes are based on the information collected from the network. Centralized decisions refer to making decisions based on the information gathered from the entire network [34]. Distributed decisions refer to making decisions based only on the information generated by the local router or nearby routers. Distributed routing, the focus of this paper, allows NoCs to grow in size without worrying about the increasing order of complexity within a centralized routing unit. An example of centralized routing is the AntNet algorithm [35], which depends on global information to make routing decisions, thus, needs extra ant buffers, routing tables, and arbitration mechanisms at each node.

There are some distributed routing algorithms which only rely on local information. They have been proposed as being efficient and still maintaining low overhead and high scalability. Routing algorithms in this category include deterministic and adaptive algorithms. Under realistic traffic patterns which pose the problem of hotspot traffic congestion areas, XY deterministic routing failed to avoid hotspots and resulted in high-average latencies [36]. Adaptive routing guides the router to react to hotspots created by different

traffic patterns, by allowing a packet at the input buffer to request more than one output port or direction [37]. While minimal routing algorithms prevent livelock from occurring, adaptive routing introduces the possibility of deadlock, which can be prevented by applying odd-even turn model restrictions to the routing decision [38].

As presented in [36], the DyAD router dynamically switches from deterministic to adaptive routing when congestion is detected, since deterministic routing achieves low packet latency under low packet injection rates. Neighboring nodes send indication to use adaptive routing when their buffers are filled above a preset threshold. Under these conditions, the router dictates that packets are routed in the direction with more available input buffer slots. This minimal adaptive algorithm, used in the presence of hotspots and increasing congestion rates, pushes back the saturation point of the traffic in the network. Another extension of adaptive routing is the neighbors-on-path (NoP) algorithm [39], which allows each router to monitor two hops away the input buffers of the routers in order to detect potential congestion earlier. By earlier detection of the buffer fill level, routes can avoid congestion better. DyXY is an algorithm which utilizes a history of buffer fill levels to make decisions [40]. The algorithms presented in [41, 42] utilize variants of buffer fill level to make decisions.

In addition to making a routing decision based on the buffer information of downstream packets, the other part of a router's decision making is the arbitration of packets. When multiple input packets are designated to be forwarded to the same next hop destination, arbitration algorithms such as round-robin or first-come first-serve (FCFS) have been proposed to resolve the output port contention. These arbitration algorithms could be designed to relieve upstream buffers with higher congestion. contention-aware input selection (CAIS) algorithm [43] is an improved arbitration algorithm that contributes to reduce the routing congestion situation by relieving hotspots of upstream traffic, determined by requests from the upstream traffic.

More works have been proposed to deal with some variance of the routing or arbitration algorithms. Sometimes, we categorize the former ones as methods of congestion avoidance; in other words, they evaluate downstream network conditions to avoid sending packets towards the congested areas so as not to aggravate the congestion conditions. We categorize the latter as methods of congestion relief; in other words, they evaluate upstream network conditions to determine which area had the most congestion to send first in order to quickly diffuse the congested situation.

3.3. Quality-of-Service Control. There is a wide range of possibilities for implementing guaranteed services on a network. Referring to the state-of-the-art QoS mechanisms for NoCs, they can be categorized into two types of schemes: connection oriented (circuit switching) and connection less (packet-switching).

3.3.1. Connection-Oriented Scheme. In connection-oriented schemes, guaranteed-service (GS) packets traverse on some particular channels or buffers that were reserved for them. Specifically, the connection path between the source and destination pair of GS packets is built at the time before they are injected onto the network [44–51]. However, this kind of static preallocation may result in high service latency and does not consider hotspots created by temporal shifts in data requirements, thus, leads to a rather unscalable NoC.

Connection-oriented QoS mechanism is reliable to achieve QoS requirement, since connections are created guaranteeing tight bounds for specific flows. Two types of the programming models for constructing the set-up phase were presented: centralized programming and distributed programming. Centralized programming sets up the reservations by a configuration manager which takes over all the resources in the network. On the contrary, distributed program models let all the resource reservations to be handled by each local router. The centralized method is simpler to achieve while it is only suitable for small-size systems. Despite the hardware overhead in routers, distributed program models have acquired popularity in a large system because of its better flexibility.

However, connection-oriented QoS mechanism comes with greater hardware overhead in control and storage for resource reservations and poor scalability because complexity grows with each node added. Furthermore, bandwidth usage is inefficient, and resource allocation has to be considered on a worst case basis. Moreover, the set-up phase of guaranteed traffic presents a timing overhead which may result in inefficiency for nondeterministic applications.

3.3.2. Connection-Less Scheme. The connection-less scheme is an alternative way to support different service levels in NoCs where the resource authorities are prioritized according to the QoS requirement of a traffic flow [48]. This is a distributed technique which allows traffic to be classified into different service levels. These service levels can often coincide with different virtual channels inside the switch. As two traffic flows with different QoS requirements are presented on the same channel simultaneously, the higher prioritized flow can interrupt the lower one and traverse this channel antecedently [48, 52]. It is more adaptive to network traffic and potential hotspots and can better utilize the network.

Different from the connection-oriented schemes, connection-less schemes do not execute any resource reservation. In contrast, multiple traffic flows share the same priority or the same resource, thus, could cause unpredictable conditions [53]. The traffic with higher service level is guaranteed in a relative fashion in a connection-less scheme by prioritizing each type of traffic flow. However, while the connection-less scheme provides a coarser QoS support as the connection-oriented schemes, they can offer a better adaptation of communication to the varying network traffic. Furthermore, better bandwidth utilization and less hardware cost can be achieved since the traffic is allocated with network resources dynamically. With the consideration of performance requirements for each service level, a network designer can select an appropriate bandwidth implemented in an NoC to both meet the QoS constraints and save the wiring cost [48, 54, 55].

Although connection-oriented communication guarantees tight bounds for several traffic parameters, an erroneous decision of resource reservation might cause an unexpected performance penalty. While in a connection-less network, a nonoptimal priority assignment has less degradation of throughput though it provides coarse QoS support. As pointed out in [20], guaranteed services require resource reservation for the worst case in a connection oriented, which causes a lot of wasted resource. In addition, some quantitative modeling and comparison of these two schemes, provided in [56], has shown that under a variable-bit-rate application, connection-less technique provides a better performance in terms of the end-to-end packet delay. These comparisons can help to design an application-specific NoC using a suitable QoS scheme.

3.4. Reliability Design. The trend towards constructing large computing systems incorporated with a many-core architecture has resulted in a two-sided relationship involving reliability and fault tolerance consideration. While yield has always been a critical issue in recent high-performance circuitry implementation, the document of the International Technology Roadmap for Semiconductor (ITRS) [57] states that "*Relaxing the requirement of 100% correctness for devices and interconnects may dramatically reduce costs of manufacturing, verification and test*." The general principle of fault tolerance for any system can be divided in two categories:

(1) employment of hardware redundancy to hide the effect of faults,

(2) self-identification of source of failure and compensating the effect by appropriate mechanism.

If we can make such a strategy work, a system will be capable of testing and reconfiguring itself, allowing it to work reliably throughout its lifetime.

3.4.1. Failure Types in NoC. Scaling chips, however, increase the probability of faults. Faults to be considered in an NoC architecture can be categorized into permanent (hard fault) and transient fault (soft fault) [13, 58]. The former one reflects irreversible physical changes, such as electro-migration of conductor, broken wires, and dielectric breakdowns. In this case, permanent damages in a circuit cannot be repaired after manufacture. Therefore, the module which is suffering a permanent fault should turn off its function and inform neighboring modules of this information. Then, rerouting packets with an alternative path will be re-calculated deterministically or dynamically according to the need. However, this may induce nonminimal path routing and increase the complexity of routing decision. Hardware redundancy such as spare wire or reconfigurable circuitry can also be used to avoid using of faulty modules [59–62]. In the latter case, several phenomena, such as neutron and alpha particles, supply voltage swing, and interconnect noise, induce the packet invalid or misrouted. Usually, a transient fault is modeled with a probability of bit error rate under an adequate fault model. In an NoC system, intrarouter or interrouter functionality errors may happen, to understand how to deal with the most common sources of failures in an NoC; Park et al. provided comprehensive fault-tolerant solutions relevant to all stages of decision making in an NoC router [63].

3.4.2. Reliability Design in NoC. A number of fault-tolerant methods were proposed in [64, 65] for large-scale communication systems. Unfortunately, these algorithms are not suitable for an NoC, because they will induce significant area and resource overhead. Dumitras et al. proposed a flood-based routing algorithm for NoC, named *stochastic communication*, which is derived from the fault-tolerance mechanism used in the computer network and distributed database fields. Such stochastic-communication algorithm separates computation from communication and provides fault tolerance to on-chip failures [57, 66]. However, to eliminate the high communication overhead of flood-based fault tolerance algorithm, Pirretti et al. promoted a redundant random-walk algorithm which can significantly reduce the overhead while maintaining a useful level of fault tolerance [67]. However, the basic idea of sending redundant information via multipath to achieve fault tolerance may cause much higher traffic load in the network, and the probabilistic broadcast characteristic may also result in additional unpredictable behavior on network loading.

Therefore, in a distributed NoC router considering practical hardware implementation, the error control scheme used to detect/correct interrouter transient fault in an NoC is required to have smaller area and shorter timing delay. An error control code that adapts to different degrees of detection and correction and has a low timing overhead will ease its integration into a router. The fault-tolerant method utilizing error detection requires an additional retransmission buffer specially designed for NoCs when the errors are detected. Error control schemes, such as the Reed-Solomon code proposed by Hoffman et al., have been used on NoCs [68]. But as their results show, the long delay would degrade the overall timing and performance of an NoC router.

3.5. Energy-Aware Task Scheduling. The availability of many cores on the same chip promises a high level of parallelism to expedite the execution of computation-intensive applications. To do so, a program must first be represented by a *task graph* where each node is a coarse-grained task (e.g., a procedure or a subroutine). Often, a task needs to forward its intermediate results to another task for further processing. This intertask data dependency is represented by a directed arc from the origin task to the destination task in the task graph. Tasks that have no intertask data dependency among themselves can be assigned for multiple processor cores to execute concurrently. As such, the total execution time can be significantly shortened.

A *real-time application* is an application in which execution time must be smaller than a *deadline*. Otherwise, the computation will be deemed a failure. To implement an application on an MC-NoC platform for parallel execution, each task in the task graph will be *assigned* to a processor core. Depending on the city-block distance between two tiles,

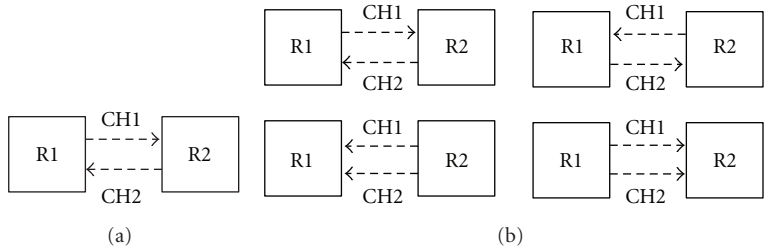


Figure 5: Channel directions in a typical NoC and proposed BiNoC.

intertask communication will take different amount of communication delay. For a particular application, proper task assignment will reduce communication delay while maximizing parallelism such that the total execution time can be minimized. For a real-time application, if the total execution time is less than the predefined deadline of the application, the *slacks* between them could be exploited to reduce energy consumption.

The execution time of a task may vary depending on the clock frequency the processor core is running. One technique to adjust the clock frequency of individual time on an MC-NoC is dynamic voltage scaling (DVS). When the clock frequency slows down, often the associated energy consumed by a running task is also reduced. Hence, in addition to assigning tasks to the processor cores located at appropriate tiles, another design objective would be to use DVS to save some energy while conforming to the deadline constraint, with perhaps smaller slacks.

Previously, it has been shown that the minimum energy multiprocessor task scheduling problem is NP-hard [69–71]. For real-time applications, it was proposed that execution of some tasks can be slowed down using DVS on corresponding tiles without violating the deadline timing constraint [72]. Several DVS-enabled uniprocessors have been implemented. Test results running real-world applications showed significant power saving up to 10 times [73]. For multiprocessor core systems implemented to execute a set of real-time dependent tasks, Schmitz et al. [74–76] presented an iterative synthesis approach for DVS-enabled processing element based on genetic algorithms (GA). They proposed a heuristic PV-DVS algorithm specifically for solving the voltage scaling. Kianzad et al. improved the previous work by combining assignment, scheduling, and power management in a single GA algorithm [77]. However, GA-based design optimization suffers slow convergence and lower desired quality. Chang et al. [78] proposed using Ant Colony Optimization (ACO) algorithm. Common to these approaches is that when PV-DVS is applied for power reduction, it is applied to one task (tile) at a time and is done after assignment and scheduling. Zhang et al. [79] and Varatkar and Marculescu [80] proposed using a list scheduling algorithm to find an initial task schedule, and the DVS problem was solved by integer linear programming. The idea behind these methods is to maximize the available slack in a schedule so as to enlarge the solution space of using DVS. However, the communication infrastructures used in these works are either a point-to-point interconnect or abus architecture. Hu and Marculescu [81] proposed an energy-aware scheduling (EAS) algorithm that considers the communication delay on an NoC architecture. However, DVS frequency adjustment was not considered.

4. Bidirectional Network-on-Chip (BiNoC) Architecture

A bidirectional channel network-on-chip (BiNoC) architecture is proposed in this section to enhance the performance of on-chip communication. In a BiNoC, each communication channel allows itself to be dynamically reconfigured to transmit flits in either direction. This added flexibility promises better bandwidth utilization, lower packet delivery latency, and higher packet consumption rate. Novel on-chip router architecture is developed to support dynamic self-reconfiguration of the bidirectional traffic flow. The flow direction at each channel is controlled by a channel-direction-control protocol. Implemented with a pair of finite state machines, this channel-direction-control protocol is shown to be of high performance, free of deadlock, and free of starvation.

4.1. Problem Description. In a conventional NoC architecture, each pair of neighboring routers uses two unidirectional channels in opposite direction to propagate data on the network as shown in Figure 5(a). In our BiNoC architecture, to enable the most bandwidth utilization, data channels between each pair of routers should be able to transmit data in any direction at each run cycle. That is, four kinds of channel-direction combinations should be allowed for data transmission as shown in Figure 5(b). However, current unidirectional NoC architectures, when facing applications that have different traffic patterns, cannot achieve the high bandwidth utilization objective.

Note that the number of bidirectional channels between each pair of neighboring router in BiNoC architecture is not limited to two. The more the channels that can be used, the better the performance results. In order to provide a fair comparison between our BiNoC and the conventional NoC that usually provided two fixed unidirectional channels, only two bidirectional channels were used in BiNoC as illustrated in Figure 5.

4.2. Motivational Example. As shown in Figure 6(a), an application task graph is typically described as a set of concurrent tasks that have already been assigned and scheduled

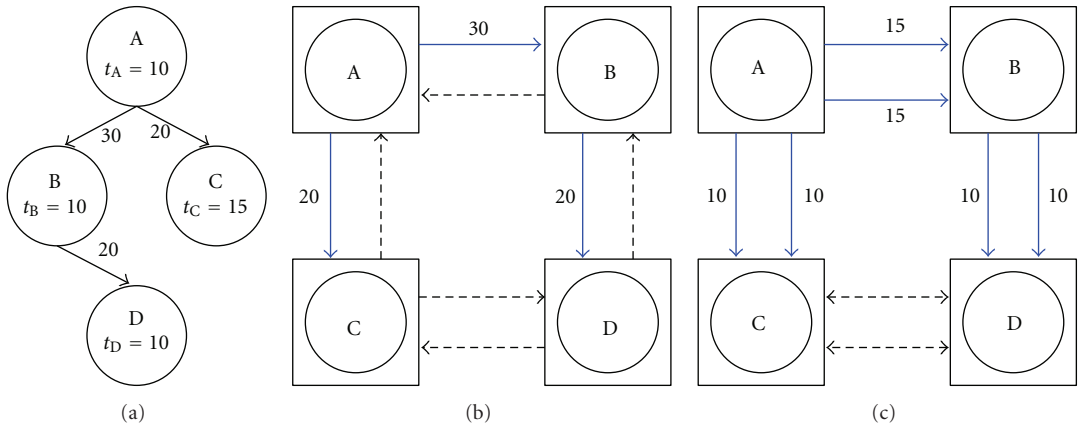


Figure 6: Example of task graph mapping on typical NoC and BiNoC.

onto a list of selected PEs. Each vertex represents a task with a value t_j of its computational execution time, and each edge represents the communication dependence with a value of communication volume which is divided by the bandwidth of a data channel.

For the most optimized mapping in a 2×2 2-dimensional mesh NoC as shown in Figure 6(b), the conventional NoC architecture in this case can only use three channels during the entire simulation and result in a total execution time of 80 cycles. However, if we can dynamically change the direction of each channel between each pair of routers like the architecture illustrated in Figure 6(c), the bandwidth utilization will be improved and the total execution time be reduced to 55 cycles. Figure 7 shows the detailed execution schedules, where the required communication time between nodes in BiNoC is extensively reduced.

4.3. Channel Bandwidth Utilization. During the execution of an application, the percentage of time when a data channel is kept busy is defined as channel bandwidth utilization U. To be more specific,

$$U = \frac{\sum_{t=1}^{T} N_{\text{Busy}}(t)}{T \times N_{\text{Total}}}, \tag{1}$$

where T is the total execution time, N_{Total} is the total number of channels available to transmit data, and $N_{\text{Busy}}(t)$ is the number of channels that are busy during clock cycle t. It is obvious that $U \leq 1$.

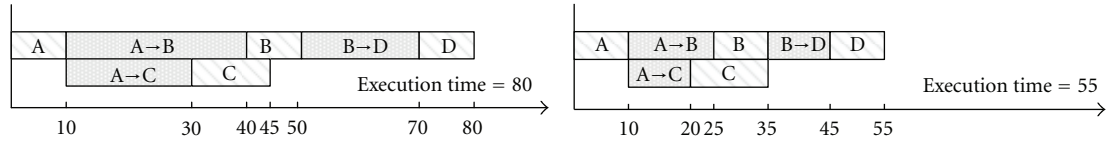
We have developed a cycle-accurate NoC simulator to evaluate the performance of a given NoC architecture. Additional implementation details of this NoC simulator will be elaborated in later sections. Using this simulator, we measured the channel bandwidth utilizations of a conventional NoC with respect to three types of synthetic traffic patterns: uniform, regional, and transpose. The channel utilization against different traffic volumes is plotted in Figure 8 under both XY and odd-even routings.

Figures 8(a) and 8(b) plot the bandwidth utilizations of a conventional NoC router with virtual-channel flow control. Four virtual-channel buffers, each with a depth of 8-flits, are allocated in each flow direction. Figures 8(c) and 8(d) give the percentage of time in which exactly one channel is busy and another channel is idle among time intervals when there is at least one channel busy. Figures 8(e) and 8(f) give the percentage of time that a bidirectional channel may help alleviating the traffic jam when exactly one channel is busy and the other is idle. Figures 8(a), 8(c), and 8(e) results are obtained using XY routing; and Figures 8(b), 8(d), and 8(f) use odd-even routing.

From Figures 8(a) and 8(b), it is clear that, even with the most favorable uniform traffic pattern, the channel bandwidth utilization peaks under XY routing and odd-even routing are only around 45% and 40%, respectively, under heavy traffic. For the transpose traffic pattern under XY routing, which is considered the worst case scenario, falls even below 20%. In other words, in a unidirectional channel setting even with two channels between a pair of routers, at most one channel is kept busy on average during normal NoC operation despite the deterministic routing algorithm such as XY or adaptive routing algorithm such as odd-even.

One possible cause of the low-bandwidth utilization as shown in Figures 8(a) and 8(b) is due to few bottleneck channels that take too long to transmit data packets in the designated direction. To validate this claim, we examine how often both channels between a pair of routers are kept busy simultaneously. In Figures 8(c) and 8(d), the percentage of time in which exactly one channel is busy and the other is idle given that one or both channels are busy is plotted, respectively, under XY and odd-even routings. As traffic load increases, it is clear that a significant amount of traffic utilizes only a single channel while the other channel is idle.

However, the situation where one channel is busy and the other is idle could be the case where there are no data that need to transmit in the opposite direction of the busy channel. It does not reveal whether there are additional data packets waiting in the same direction as the busy channel. These data packets are potential candidate to take advantage of the idle channel if the idle channel's direction can be reversed. In Figures 8(e) and 8(f), the percentage of time, in which there are data packets needed to transmit along the same direction as the busy channel while the other channel remains idle out



(a) (b)

Figure 7: Detailed execution schedules of typical NoC and BiNoC.

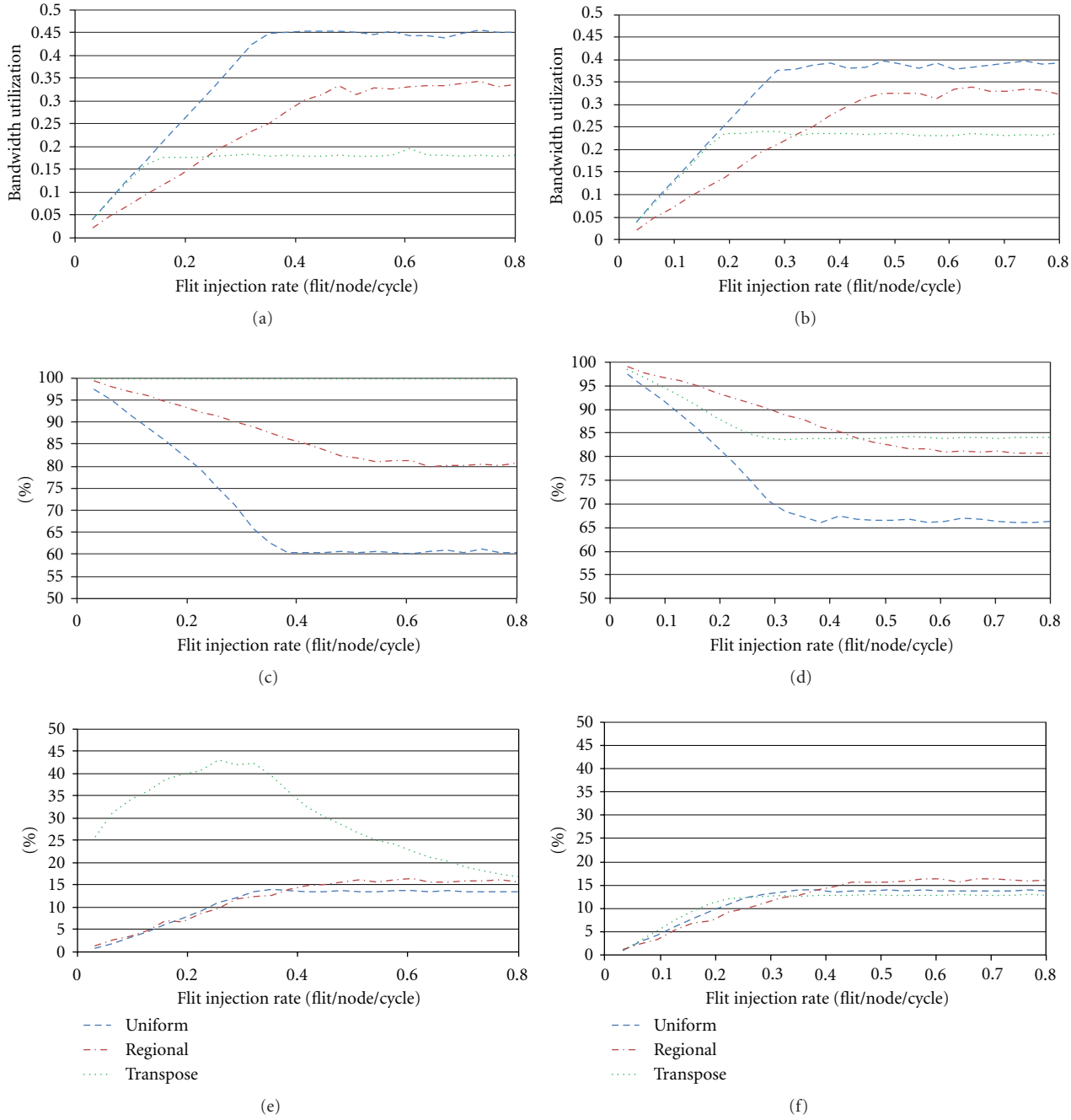


Figure 8: Bandwidth utilization analysis of a conventional NoC router.

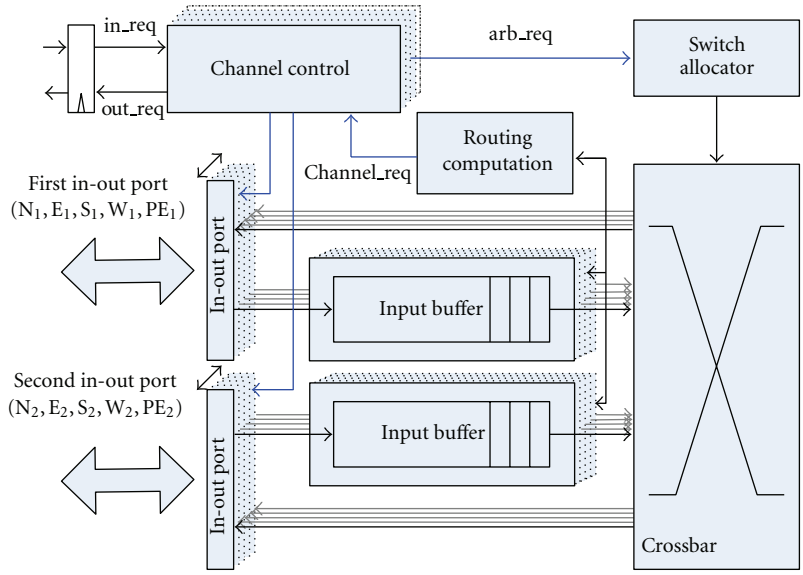


Figure 9: Proposed BiNoC router with wormhole flow control.

of all situations where exactly a single channel is busy, is plotted. An important observation is that, for large traffic volume, this situation happens for about 15% of time despite the type of traffic patterns or the routing methods.

Figure 8 gives ample evidence that the unidirectional channel structure of current NoC cannot fully utilize available resources (channel bandwidth) and may cause longer latency. This observation motivates us to explore the BiNoC architecture that offers the opportunity to reverse channel direction dynamically to relieve the high traffic volume of a busy channel in the opposite direction.

4.4. Design Requirements. Bidirectional channels have been incorporated into off-chip multiprocessor high-speed interconnecting subsystems over years. Recently, bidirectional on-chip global interconnecting subsystems have also been studied quite extensively for supporting electronic design automation of system-on-chip platforms [82–85]. Hence, physical layer design of an NoC channel to support bidirectional data transmission should be of little difficulty. The real challenge of embracing a bidirectional channel in an NoC is to devise a distributed channel-direction-control protocol that would achieve several important performance criteria as follows.

(1) *Correctness*. It should not cause permanent blockage of data transfer (deadlock, starvation) during operation.

(2) *High Performance*. Its performance should be *scalable* to the size of the *NoC* fabric and *robust* with respect to increasing traffic volume. In addition, it is desirable that the performance enhancement can be achieved across different characteristics of application traffic patterns.

(3) *Low Hardware Cost*. The hardware overhead to support the bidirectional channel should be small enough to justify the cost effectiveness of the proposed architecture.

4.5. The Proposed BiNoC Router Design. To realize a dynamically self-reconfigurable bidirectional channel NoC architecture, we initially modified the input/output port configuration and router control unit designs based on the conventional router using wormhole flow control as we proposed in [86]. In order to dynamically adjust the direction of each bidirectional channel at run time, we add a *channel control* module to arbitrate the authority of the channel direction as illustrated in Figure 9.

Each bidirectional channel which is composed of an in-out port inside is the main difference from the conventional router design where unidirectional channel employs a hard-wired input port or output port. However, the total number of data channels is not changed as its applicable bandwidth for each transmission direction is doubled.

In our design, each channel can be used as either an input or an output channel. As a result, the width of a channel request signal, *channel_req*, generated from the RC (routing computation) modules is doubled. Two bidirectional channels can be requested in each output direction. In other words, this router is able to transmit at most two packets to the same direction simultaneously which decreases the probability of contentions.

The channel control module has two major functions. One is to dynamically configure the channel direction between neighboring routers. Since the bidirectional channel is shared by a pair of neighboring routers, every transition of the output authority is achieved by a channel-direction-control protocol between these two routers. The control protocol can be implemented as FSMs. The other responsibility is that whether the channel request (*channel_req*) for the corresponding channel is blocked or not will depend on the current status of channel direction. If the channel is able to be used, the *arb_req* will be sent to the switch allocator (SA) to process the channel allocation.

The most important point of this architecture is that we can replace all the unidirectional channels in a conventional

NoC with our bidirectional channels. That will increase the channel utilization flexibility without requiring additional transmission bandwidth compared to the conventional NoC.

5. Conclusion

In the first part of this paper, we introduced the detail of an on-chip interconnection framework, namely, network on chip (NoC), used in the design of multiprocessor system-on-chip (MPSoC) and chip multiprocessor (CMP) architectures. Then, the NoC architecture and its function layers were reviewed, and some prevalent NoC design methodologies were discussed. Last, we proposed a novel bidirectional channel NoC (BiNoC) backbone architecture, which can be easily integrated into most conventional NoC designs and successfully improve the NoC performance with a reasonable cost.

Acknowledgments

This work was partially supported by the National Science Council, under Grants 99-2220-E-002-041 and 100-2220-E-002-012.

References

[1] F. N. Najm, "Survey of power estimation techniques in VLSI circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 446–455, 1994.

[2] H. O. Ron, K. W. Mai, and A. Fellow, "The future of wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490–504, 2001.

[3] ARM, *AMBA Specification Rev 2.0*, ARM Limited, 1999.

[4] IBM, *32-bit Processor Local Bus Architecture Specification Version 2.9*, IBM Corporation.

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

[6] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, Waltham, Mass, USA, 2004.

[7] 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.

[8] M. Kistler, M. Perrone, and F. Petrini, "Cell multiprocessor communication network: built for speed," *IEEE Micro*, vol. 26, no. 3, pp. 10–23, 2006.

[9] L. Seiler, D. Carmean, E. Sprangle et al., "Larrabee: a many-core x86 architecture for visual computing," *IEEE Micro*, vol. 29, no. 1, pp. 10–21, 2009.

[10] D. Wentzlaff, P. Griffin, H. Hoffmann et al., "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, 2007.

[11] J. Howard, S. Dighe, Y. Hoskote et al., "A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS," in *Proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers, (ISSCC '10)*, pp. 108–109, San Francisco, Calif, USA, February 2010.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, Calif, USA, 1979.

[13] R. Marculescu, U. Y. Ogras, L. S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives," *IEEE Transactions on Computer*, vol. 28, no. 1, pp. 3–21, 2009.

[14] G. DeMicheli and L. Benini, *Networks on Chips: Technology and Tools*, Morgan Kaufmann, Waltham, Mass, USA, 2006.

[15] S. Kumar, A. Jantsch, and J. P. Soininen, "Network-on-chip architecture and design methodology," in *Proceedings of the International Symposium on Very Large Scale Integration*, pp. 105–112, April 2000.

[16] C. Grecu, M. Jones, P. P. Pande, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, 2005.

[17] A. M. Rahmani, M. Daneshtalab, A. Afzai-Kusha, S. Safari, and M. Pedram, "Forecasting-based dynamic virtual channels allocation for power optimization of network-on-chips," in *Proceedings of the 22nd International Conference on VLSI Design—Held Jointly with 7th International Conference on Embedded Systems*, pp. 151–156, New Delhi, India, January 2009.

[18] N. Kavaldjiev, G. J. M. Smit, and P. G. Jansen, "A virtual channel router for on-chip networks," in *Proceedings of the IEEE International SOC Conference*, pp. 289–293, September 2004.

[19] W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, 1992.

[20] E. Rijpkema, K. G. W. Goossens, and A. Radulescu, "Trade-offs in the design of a router with both guaranteed and best-effort services for networks-on-chip," in *Proceedings of the Design Automation and Test in Europe Conference*, pp. 350–355, March 2003.

[21] H. S. Wang, L. S. Peh, and S. Malik, "A power model for routers: modeling alpha 21364 and InfiniBand routers," *IEEE Micro*, vol. 23, no. 1, pp. 26–35, 2003.

[22] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Proceedings of the 31st Annual International Symposium on Computer Architecture*, pp. 188–197, June 2004.

[23] K. Kim, S. J. Lee, K. Lee, and H. J. Yoo, "An arbitration look-ahead scheme for reducing end-to-end latency in networks on chip," in *Proceedings of the IEEE International Symposium on Circuits and Systems, (ISCAS '05)*, pp. 2357–2360, May 2005.

[24] 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*, pp. 250–256, March 2000.

[25] 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.

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

[27] N. Cohen, T. S. Sriram, N. Leland, D. Moyer, S. Butler, and R. Flatley, "Soft error considerations for deep-submicron CMOS circuit applications," in *Proceedings of the IEEE International Devices Meeting, (IEDM '99)*, pp. 315–318, December 1999.

[28] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proceedings of the International Conference on Dependable Systems and Networks, (DNS '02)*, pp. 389–398, June 2002.

[29] W. J. Dally and C. L. Seitz, "The torus routing chip," *Distributed Computing*, vol. 1, no. 4, pp. 187–196, 1986.

[30] P. Kermani and L. Kleinrock, "Virtual cut-through: a new computer communication switching technique," *Computer Networks*, vol. 3, no. 4, pp. 267–286, 1979.

[31] L. S. Peh, W. J. Dally, and P. Li-Shiuan, "Delay model for router microarchitectures," *IEEE Micro*, vol. 21, no. 1, pp. 26–34, 2001.

[32] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547–553, 1987.

[33] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Routing table minimization for irregular mesh NoCs," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pp. 1–6, Nice, France, April 2007.

[34] M. A. Yazdi, M. Modarressi, and H. Sarbazi-Azad, "A load-balanced routing scheme for NoC-based systems-on-chip," in *Proceedings of the 1st Workshop on Hardware and Software Implementation and Control of Distributed MEMS, (DMEMS '10)*, pp. 72–77, Besan, TBD, France, June 2010.

[35] M. Daneshtalab, A. A. Kusha, A. Sobhani, Z. Navabi, M. D. Mottaghi, and O. Fatemi, "Ant colony based routing architecture for minimizing hot spots in NOCs," in *Proceedings of the Annual Symposium on Integrated Circuits and System Design*, pp. 56–61, September 2006.

[36] J. Hu and R. Marculescu, "DyAD—smart routing for networks-on-chip," in *Proceedings of the 41st Design Automation Conference*, pp. 260–263, June 2004.

[37] 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.

[38] 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.

[39] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Neighbors-on-path: a new selection strategy for on-chip networks," in *Proceedings of the IEEE/ACM/IFIP Workshop on Embedded Systems for Real Time Multimedia, (ESTIMEDIA '06)*, pp. 79–84, Seoul, Korea, October 2006.

[40] M. Li, Q. A. Zeng, and W. B. Jone, "DyXY: a proximity congestion-aware deadlock-free dynamic routing method for network on chip," in *Proceedings of the Design Automation Conference*, pp. 849–852, July 2006.

[41] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, "Load distribution with the proximity congestion awareness in a network-on-chip," in *Proceedings of the Design Automation and Test in Europe Conference*, pp. 1126–1127, December 2003.

[42] 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, June 2005.

[43] D. Wu, B. M. Al-Hashimi, and M. T. Schmitz, "Improving routing efficiency for network-on-chip through contention-aware input selection," in *Proceedings of the Asia and South Pacific Design Automation Conference, (ASP-DAC '06)*, pp. 36–41, January 2006.

[44] 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 the Design, Automation and Test in Europe Conference and Exhibition, (DATE '04)*, pp. 890–895, February 2004.

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

[46] P. Vellanki, N. Banerjee, and K. S. Chatha, "Quality-of-service and error control techniques for mesh-based network-on-chip architectures," *ACM Very Large Scale Integration Journal*, vol. 38, no. 3, pp. 353–382, 2005.

[47] N. Kavaldjiev, G. J. M. Smit, P. G. Jansen, and P. T. Wolkotte, "A virtual channel network-on-chip for GT and BE traffic," in *Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, pp. 211–216, Karlsruhe, Germany, March 2006.

[48] 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.

[49] 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, October 2003.

[50] D. Bertozzi and L. Benini, "Xpipes: a network-on-chip architecture for gigascale systems-on-chip," *IEEE Circuits and Systems Magazine*, vol. 4, no. 2, pp. 18–31, 2004.

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

[52] M. D. Harmanci, N. P. Escudero, Y. Leblebici, and P. Ienne, "Providing QoS to connection-less packet-switched NoC by implementing diffServ functionalities," in *Proceedings of the International Symposium on System-on-Chip*, pp. 37–40, November 2004.

[53] A. Mello, L. Tedesco, N. Calazans, and F. Moraes, "Evaluation of current QoS mechanisms in networks on chip," in *Proceedings of the International Symposium on System-on-Chip, (SOC '06)*, pp. 1–4, Tampere, Finland, November 2006.

[54] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Efficient link capacity and QoS design for network-on-chip," in *Proceedings of the Design, Automation and Test in Europe, (DATE '06)*, pp. 1–6, March 2006.

[55] P. Vellanki, N. Banerjee, and K. S. Chatha, "Quality-of-service and error control techniques for network-on-chip architectures," in *Proceedings of the ACM Great lakes Symposium on VLSI, (GLSVLSI '04)*, pp. 45–50, April 2004.

[56] M. D. Harmanci, N. P. Escudero, Y. Leblebici, and P. Ienne, "Quantitative modelling and comparison of communication schemes to guarantee quality-of-service in networks-on-chip," in *Proceedings of the IEEE International Symposium on Circuits and Systems, (ISCAS '05)*, pp. 1782–1785, May 2005.

[57] P. Bogdan, T. Dumitras, and R. Marculescu, "Stochastic communication: a new paradigm for fault tolerant networks on chip," *VLSI Design*, vol. 2007, Article ID 95348, 17 pages, 2007.

[58] M. Ali, M. Welzl, S. Hessler, and S. Hellebrand, "A fault tolerant mechanism for handling permanent and transient failures in a network on chip," in *Proceedings of the 4th International Conference on Information Technology-New Generations, (ITNG '07)*, pp. 1027–1032, Las Vegas, Nev, USA, April 2007.

[59] M. Yang, T. Li, Y. Jiang, and Y. Yang, "Fault-tolerant routing schemes in RDT(2,2,1)/α-based interconnection network for networks-on-chip designs," in *Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks, (I-SPAN '05)*, pp. 1–6, December 2005.

[60] T. Lehtonen, P. Liljeberg, and J. Plosila, "Online reconfigurable self-timed links for fault tolerant NoC," *VLSI Design*, vol. 2007, Article ID 94676, 13 pages, 2007.

[61] H. Kariniemi and J. Nurmi, "Fault-tolerant XGFT network-on-chip for multi-processor system-on-chip circuits," in *Proceedings of the International Conference on Field Programmable Logic and Applications, (FPL '05)*, pp. 203–210, August 2005.

[62] T. Schonwald, J. Zimmermann, O. Bringmann, and W. Rosenstiel, "Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures," in *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, (DSD '07)*, pp. 527–534, Lübeck, Germany, August 2007.

[63] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das, "Exploring fault-tolerant network-on-chip architectures," in *Proceedings of the 2006 International Conference on Dependable Systems and Networks, (DSN '06)*, pp. 93–104, Philadelphia, Pa, USA, June 2006.

[64] Y. Hatanaka, M. Nakamura, Y. Kakuda, and T. Kikuno, "A synthesis method for fault-tolerant and flexible multipath routing protocols," in *Proceedings of the International Conference on Engineering of Complex Computer Systems*, pp. 96–105, September 1997.

[65] W. Stallings, *Data and Computer Communications*, Prentice Hall, New York, NY, USA, 2007.

[66] T. Dumitras, S. Kerner, and R. Marculescu, "Towards on-chip fault-tolerant communication," in *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 225–232, January 2003.

[67] 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 the IEEE Computer Society Annual Symposium on VLSI*, pp. 46–51, February 2004.

[68] J. Hoffman, D. A. Ilitzky, A. Chun, and A. Chapyzhenka, "Architecture of the scalable communications core," in *Proceedings of the First International Symposium on Networks-on-Chip, (NOCS '07)*, pp. 40–49, Princeton, NJ, USA, May 2007.

[69] E. S. H. Hou, N. Ansari, and H. Ren, "Genetic algorithm for multiprocessor scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 2, pp. 113–120, 1994.

[70] C. M. Krishna and K. G. Shin, *Real-Time Systems*, WCB/McGraw Hill, New York, NY, USA, 1997.

[71] H. El-Rewini, H. H. Ali, and T. Lewis, "Task scheduling in multiprocessing systems," *Computer*, vol. 28, no. 12, pp. 27–37, 1995.

[72] T. Burd and R. W. Brodersen, "Energy efficient CMOS microprocessor design," in *Proceedings of the Hawaii International Conference on System Sciences*, pp. 288–297, January 1995.

[73] G. Quan and X. Hu, "Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors," in *Proceedings of the 38th Design Automation Conference*, pp. 828–833, June 2001.

[74] M. T. Schmitz and B. M. Al-Hashimi, "Considering power variations of DVS processing elements for energy minimisation in distributed systems," in *Proceedings of the 14th International Symposium on System Synthesis (ISSS '01)*, pp. 250–255, October 2001.

[75] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, "Energy-efficient mapping and scheduling for DVS enabled distributed embedded systems," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 514–521, March 2002.

[76] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, "Iterative schedule optimization for voltage scalable distributed embedded systems," *ACM TECS*, vol. 3, no. 1, pp. 182–217, 2004.

[77] V. Kianzad, S. S. Bhattacharyya, and G. Qu, "CASPER: an integrated energy-driven approach for task graph scheduling on distributed embedded systems," in *Proceedings of the IEEE 16th International Conference on Application-Specific Systems, Architectures, and Processors, (ASAP '05)*, pp. 191–197, July 2005.

[78] P. C. Chang, I. W. Wu, J. J. Shann, and C. P. Chung, "ETAHM: an energy-aware task allocation algorithm for heterogeneous multiprocessor," in *Proceedings of the 45th Design Automation Conference, (DAC '08)*, pp. 776–779, Anaheim, Calif, USA, June 2008.

[79] Y. Zhang, X. Hu, and D. Z. Chen, "Task scheduling and voltage selection for energy minimization," in *Proceedings of the 39th Design Automation Conference*, pp. 183–188, June 2002.

[80] G. Varatkar and R. Marculescu, "Communication-aware task scheduling and voltage selection for total systems energy minimization," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design, (ICCAD '03)*, pp. 510–517, November 2003.

[81] J. Hu and R. Marculescu, "Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, (DATE '04)*, pp. 234–239, February 2004.

[82] J. Lillis and C. K. Cheng, "Timing optimization for multisource nets: characterization and optimal repeater insertion," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 2-3, pp. 322–331, 1999.

[83] S. Bobba and I. N. Haj, "High-performance bidirectional repeaters," in *Proceedings of the Great Lakes Symposium on Very Large Scale Integration*, pp. 53–58, March 2000.

[84] A. Nalamalpu, S. Srinivasan, and W. P. Burleson, "Boosters for driving long onchip interconnects—design issues, interconnect synthesis, and comparison with repeaters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 1, pp. 50–62, 2002.

[85] H. Ito, M. Kimura, K. Miyashita, T. Ishii, K. Okada, and K. Masu, "A bidirectional- and multi-drop-transmission-line interconnect for multipoint-to-multipoint on-chip communications," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 1020–1029, 2008.

[86] Y. C. Lan, S. H. Lo, Y. C. Lin, Y. H. Hu, and S. J. Chen, "BiNoC: a bidirectional NoC architecture with dynamic self-reconfigurable channel," in *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip, (NoCS '09)*, pp. 266–275, May 2009.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 697039, 19 pages
doi:10.1155/2012/697039

Research Article

Self-Calibrated Energy-Efficient and Reliable Channels for On-Chip Interconnection Networks

Po-Tsang Huang and Wei Hwang

Institute of Electronics, National Chiao-Tung University, Hsin-Chu 300, Taiwan

Correspondence should be addressed to Po-Tsang Huang, bug.ee91g@nctu.edu.tw

Received 15 July 2011; Accepted 17 August 2011

Academic Editor: Jiang Xu

Energy-efficient and reliable channels are provided for on-chip interconnection networks (OCINs) using a self-calibrated voltage scaling technique with self-corrected green (SCG) coding scheme. This self-calibrated low-power coding and voltage scaling technique increases reliability and reduces energy consumption simultaneously. The SCG coding is a joint bus and error correction coding scheme that provides a reliable mechanism for channels. In addition, it achieves a significant reduction in energy consumption via a joint triplication bus power model for crosstalk avoidance. Based on SCG coding scheme, the proposed self-calibrated voltage scaling technique adjusts voltage swing for energy reduction. Furthermore, this technique tolerates timing variations. Based on UMC 65 nm CMOS technology, the proposed channels reduces energy consumption by nearly 28.3% compared with that for uncoded channels at the lowest voltage. This approach makes the channels of OCINs tolerant of transient malfunctions and realizes energy efficiency.

1. Introduction

As design complexity of multicore system-on-chip (SoC) continues to increase, a global approach is needed to effectively transport and manage on-chip communication traffic, and optimize wire efficiency. In addition to shrinking processing technologies, the ratio of interconnection delay to gate delay will increase in advanced technologies [1], indicating that on-chip interconnection architectures will dominate performance in future SoC designs. Therefore, modern SoC designs face a number of problems caused by the communication among multiple processor elements. Additionally, in current multicore SoC designs, reducing power consumption is the primary challenge for advanced technologies. Therefore, process-independent network-on-chip (NoC) has been considered an effective solution for integrating a multicore system. NoC was investigated for dealing with the challenges of on-chip data communication caused by the increasing scale of next-generation SoC designs [2, 3]. The most important characteristics of NoC can be considered as a packet switched approach [4] and a flexible and user-defined topology [5]. Furthermore, on-chip interconnection networks (OCINs) provide the building blocks and the microarchitecture for NoCs [6, 7]. However, some physical effects in nanoscale technology unfortunately degrade the performance and reliability of OCINs. Moreover, channels in OCINs dominate the overall power consumption [8, 9].

On-chip physical interconnections will comprise a limiting factor for performance and energy consumption. For on-chip interconnections, three critical issues, delay, power, and reliability must be addressed. For the delay issue, propagation decreases by coupling capacitances. For long global lines, discharging large capacitances takes considerable time. For the power issue, power dissipation increases due to both parasitic and coupling capacitances. Finally, the reliability issue for on-chip interconnections will be degraded due to noise. In advanced technologies, circuits and interconnects degrade further due to noise with decreasing operating voltages. Furthermore, increasing coupling noise, the soft-error rate, and bouncing noise also decrease the reliability of circuits. Thus, self-calibrated circuitry has become essential for near-future interconnection architecture designs.

In this paper, we propose a novel self-calibrated energy-efficient and reliable channel design for OCINs. The proposed channels reduce the energy consumption while maintaining reliability. The channels are developed using the *self-calibrated voltage scaling technique* with the *self-corrected green (SCG) coding scheme*. The rest of this paper is organized as follows. Section 2 will analyze previous reliable and low-power coding schemes. The self-calibrated low-power coding and voltage scaling channels will be presented in Section 3. Sections 4 and 5 will describe the proposed SCG coding scheme and self-calibrated voltage scaling technique, respectively. Additionally, the simulation results will be given in Section 6. Finally, we will conclude the paper in Section 7.

2. Previous Low-Power and Reliable Interconnect Techniques

To achieve low latency and reliable and low-energy on-chip communication, energy efficiency is the primary challenge for current OCIN designs with nanoscale effects. First, coupling capacitance increases significantly in nanoscale technology. Second, decreasing operating voltage makes the interconnection susceptible to noise increasingly. Due to crosstalk noise, the coupling effect not only aggravates the power-delay metrics but also deteriorates the signal integrity. Many techniques have been developed to reduce the coupling capacitance effect using bus encoding schemes [10–18]. Bus encoding is an elegant and effective technique for eliminating the crosstalk effect, and provides a reliability bound for on-chip interconnects. Moreover, in order to provide a reliability bound for on-chip interconnects, forward error correction (FEC) and automatic repeat request (ARQ) techniques are widely used in NoC [5, 19]. Additionally, a joint error correction coding and bus coding technique is an effective solution to resolve delay, power, and reliability. Encoding schemes for low-power and reliability issues were proposed in [20–25]. The designers increased reliability for on-chip interconnections. Moreover, robust self-calibrating transmission schemes were proposed in [19, 26–28], which examined some physical properties of on-chip interconnects, with the goal of achieving fast, reliable, and low-energy communication.

Incorporating of different coding schemes was being investigated to increase system reliability and to reduce energy dissipation. The crosstalk avoidance codes incorporated with forward error correction coding is a solution to provide the low-power and reliable on-chip interconnection. Therefore, duplicate-add-parity (DAP) [20], modified dual rail (MDR) [23], boundary shift code (BSC) [22, 23], and hamming codes [20] are the forward error correction coding to increase the reliability of interconnections. A unified framework of coding with crosstalk avoidance codes (CAC), error control codes (ECC), and linear crosstalk codes (LXC) was proposed in [20, 21]. It provides practical codes to solve delay, power, and reliability problems jointly as shown in Figure 1. CAC avoids specific code patterns or code transitions to reduce delay and power consumption by decreasing crosstalk effect. ECC is able to detect and correct the error bits. However,

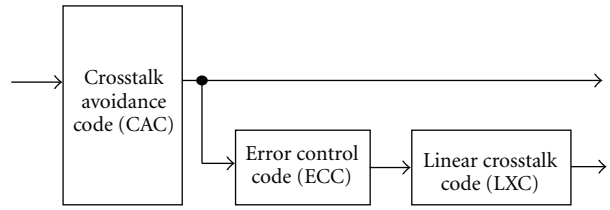


Figure 1: A unified framework for joint crosstalk avoidance code and error correction code.

the parity bits of CAC cannot be modified. In order to reduce the coupling effect of parity bits, LXC is applied without destroying the parity bits. Other approaches are based on the unified framework to improve the ability of error correction and to address signal integrity in OCINs [20–25].

CACs are designed to improve the signal integrity and to reduce the coupling effect. The purpose of CAC is to reduce the worst-case switching patterns, which are forbidden overlap condition (FOC), forbidden transition condition (FTC), and forbidden pattern condition (FPC) [20]. FOC represents a codeword transition from 010 to 101 or from 101 to 010. In addition, FTC represents a codeword transition from 01 to 10 or from 10 to 01, and FPC represents a codeword having 010 or 101 patterns. In order to reduce or avoid the worst-case switching patterns, many coding schemes are proposed to be directed against the three conditions [25]. Forbidden overlap code provides a 5-bit codeword for a 4-bit dataword to eliminate FOC. And forbidden pattern code is also a 5-bit codeword for a 4-bit dataword to avoid FPC in codeword. Additionally, forbidden transition codes provide a 4-bit codeword for a 3-bit dataword to prevent FTC. However, these three coding schemes do not satisfy the forbidden adjacent boundary pattern condition, which is defined as two adjacent bit boundaries in the codes cannot both be of 01-type and 10-type. Hence, one lambda codes is proposed not only to avoid FTC and FPC but also to satisfy the forbidden adjacent boundary pattern condition [25]. However, it needs an 8-bit codeword to transfer a 4-bit dataword.

Joint coding schemes based on the unified framework as shown in Figure 1 provide better communication performance. However, these schemes just combine different kinds of codes directly, since the intrinsic qualities of CACs and ECCs are mutually exclusive, except for duplicating codes (DAP, MDR, and BSC) [20, 23]. In DAP coding, nevertheless, the critical path of the priority bit is much longer than others. Moreover, CAC must be a code that does not modify the parity bits in any way as decoding of ECC has to occur before any other decoding in the receiver. In order to reduce the coupling effect of the parity bits, the linear crosstalk code could be applied without destroying the parity bits.

3. Self-Calibrated Low-Power and Energy-Efficient Channel Design

The self-calibrated energy-efficient and reliable channels are developed using a self-calibrated voltage scaling technique

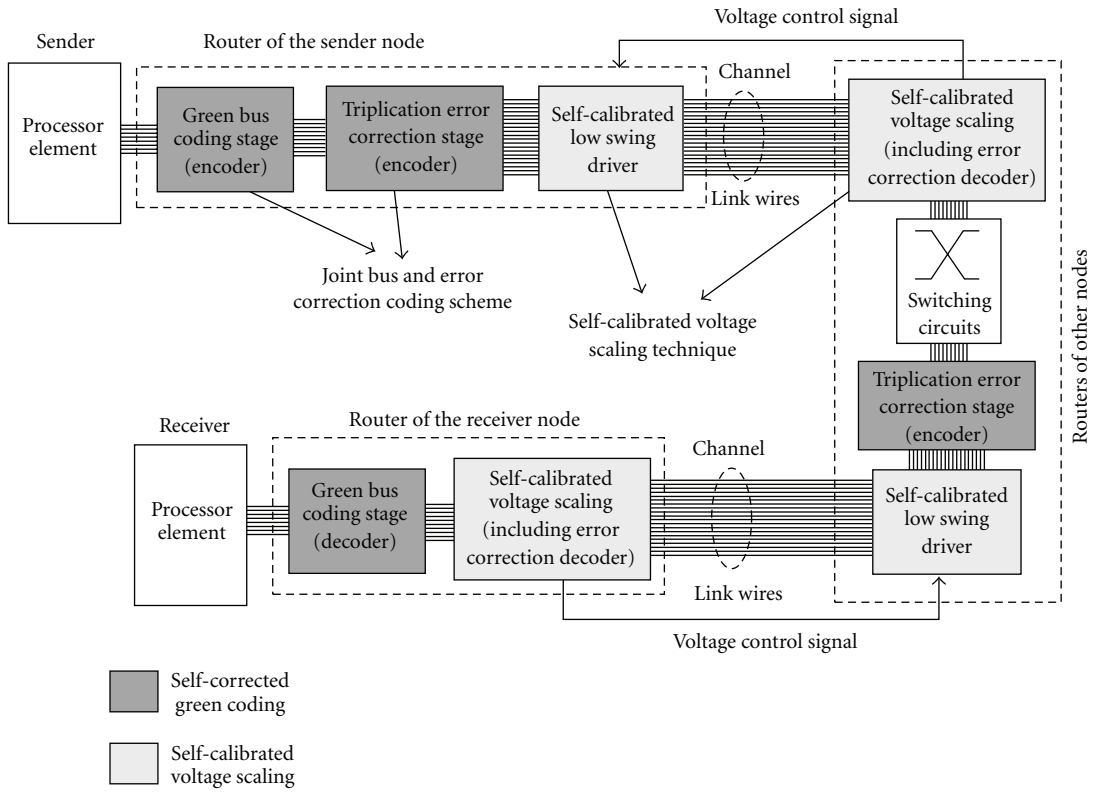


Figure 2: Self-calibrated energy-efficient and reliable channels for on-chip interconnection networks with self-corrected green (SCG) coding scheme and self-calibrated voltage scaling technique.

and a joint bus/error correction coding scheme, which is called the SCG coding scheme. Figure 2 shows the block diagrams of the proposed channels for OCINs. The SCG coding scheme reduces coupling effects and has a rapid correction ability that reduces the physical transfer unit size in routers. The self-calibrated voltage scaling technique achieves the optimal operating voltage for link wires in channels according to the SCG coding scheme. Additionally, the proposed technique overcomes increasing variation in advanced technologies and facilitates the energy-efficient on-chip data communication. Therefore, the proposed self-calibrated low-power coding and voltage scaling realize energy-efficient and reliable channels for OCINs.

The SCG coding scheme is a joint bus and error correction coding scheme that provides low-energy and high reliability channels for OCINs. The SCG coding scheme is constructed in two stages, the green bus coding stage and the triplication error correction coding stage. In routers, an undecoded code increases the area and energy dissipation of switching circuits by large physical transfer unit sizes. Therefore, the error correction code should be decoded in routers to reduce power dissipation and the area of switching circuits and buffers. The triplication error correction coding stage achieves rapid correction to reduce the physical transfer unit size in routers via a self-corrected mechanism at the bit level. To efficiently reduce the coupling effect, the green bus coding stage is developed using the joint triplication bus power model, which depends upon the characteristics of triplication error correction coding. The SCG coding can avoid the FOC and FPC, and reduce the FTC to achieve the power saving of channels. The bit width in the self-calibrated low-power coding and voltage scaling varies. The green bus coding encodes packets in accordance with a 4-to-5 codec. To increase the reliability of channels, the triplication error correction stage increases bit width from k-bit to $3k$-bit. Although the SCG coding increases link wires in channels, on-chip wires are cheap and plentiful with the increasing metal layers in advanced technologies [29, 30].

Designers can tradeoff between power consumption and reliability by reducing the operating voltage as the error correction coding increases the reliability of channels. Therefore, the operating voltage of the link wires in channels is adjusted according to the SCG coding scheme using a self-calibrated voltage-scaling technique. This technique detects error conditions of channels in the triplication error correction stage, and thus feeds the control signals back to the low swing drivers and adjusts the operating voltage of the link wires. The self-calibrated voltage scaling technique determines the optimal operating point to trade off between energy consumption and reliability. The SCG coding scheme and self-calibrated voltage scaling technique are described in Sections 4 and 5, respectively.

4. Self-Corrected Green (SCG) Coding Scheme

This section describes the SCG coding scheme, a joint bus and error correction coding scheme. This proposed scheme generates low-energy and reliable channels for advanced technologies. The SCG coding scheme is constructed via two stages, the green bus coding stage and triplication error correction coding stage. The green bus coding has the advantages of shorter delay for error correction coding, greater energy reduction, and smaller area than other approaches. The green bus coding is developed using the joint triplication bus power model to achieve additional energy reductions for triplication error correction coding.

4.1. Triplication Error Correction Stage. The triplication error correction coding scheme as shown in Figure 3 is a single error correcting code by triplicating each bit. Based on information theory, a code set with a hamming distance of h has an $h-1$ error-detect ability and a $[(h-1)/2]$ error-correction ability. For triplication error correction coding, the hamming distance of each bit is 3. Therefore, each bit can be corrected individually when no more than one error bit exists in the three triplicated bits, which are defined as a triplication set. The error bit can be corrected by a majority gate. Figure 3 also shows the function of the majority gate. Compared with other error correction mechanisms, the critical delay of the decoder is a constant delay of a majority gate and significantly smaller than that of other approaches [19–25]. Restated, the triplication error correction coding has rapid correction ability via self-correction mechanism at the bit level. Therefore, triplication error correction coding is more suitable to OCINs because data can be decoded and encoded in each router using the small delay of triplication correction coding.

Additionally, one advantage of incorporating error correction mechanisms in an OCIN data stream is that the supply voltage of channels can be reduced without compromising the system reliability. Reducing supply voltage, V_{dd}, increases bit error probability. To simplify error sources, we assume bit error probability, ε, is as in the following equation when a Gaussian distributed noise voltage, V_N, with variance σ_N^2 is added to the signal waveform:

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

where $Q(x)$ is given as

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

Each triplication set can be error-free if and only if no error transmission exists or just 1-bit error transmission exists. For each triplication set, $P_{1\text{-bit correct}}$ is given as

$$P_{1\text{-bit correct}} = (1-\varepsilon)^3 + \binom{3}{1}\varepsilon(1-\varepsilon)^2. \tag{3}$$

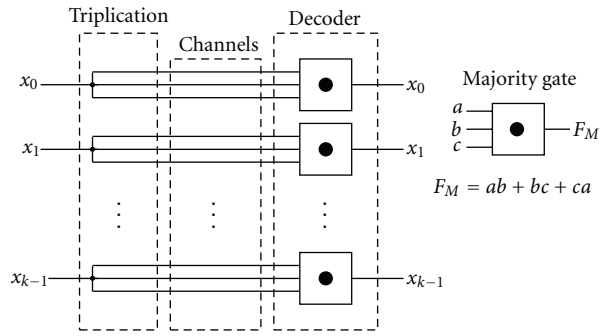


Figure 3: Triplication error correction stage of SCG coding scheme.

For k-bits data, transmission is error-free if and only if all k triplication sets are correct. Thus, $P_{k\text{ bits correct}}$ is given by

$$P_{k\text{ bits correct}} = \prod_{i=1}^{k} P_{i\text{-bit correct}} = \left(1-3\varepsilon^2+2\varepsilon^3\right)^k. \tag{4}$$

Hence, word-error probability is

$$P_{\mathrm{triplication}} = 1-\left(1-3\varepsilon^2+2\varepsilon^3\right)^k. \tag{5}$$

For a small probability of bit error, ε, (5) is simplified to

$$\mathrm{P}_{\mathrm{triplication}} \approx 3k\varepsilon^2 - 2k\varepsilon^3. \tag{6}$$

By contrast, word-error probability is much smaller than that in the Hamming code and Duplicate-add-parity (DAP) [20, 21] which are directed to $k^2\varepsilon^2$. Triplication error correction coding can avoid the FOC and FPC which increase energy dissipation via the coupling effect.

Because error-correction coding increases the reliability of on-chip interconnections, designers can tradeoff between power consumption and reliability by reducing operating voltage. In simplifying the cumulative effect of noise sources, the noise model on interconnects assumes Gaussian distributed noise with voltage V_N and variance σ_N^2 is added to the signal. In addition, we assume errors on different link lines are independent. The bit error probability, ε, is given in (1) and (2), where V_{dd} is signal voltage swing. With given the same σ_N^2, the bit error probability is increasing as the signal voltage swing decreases. However, some specific error control/correct coding schemes can decrease signal voltage swing, and guarantee the reliability of interconnections, if and only if the following equation is satisfied:

$$P_{\mathrm{uncode}}(\varepsilon) \geq P_{\mathrm{ECC}}(\hat{\varepsilon}), \tag{7}$$

where ε is bit error probability with full swing voltage of 1.0 V, and $\hat{\varepsilon}$ is bit error probability with a lower swing voltage. To obtain the lowest supply voltage for specific error

correction coding under the same level of reliability of the uncoded code, supply voltage can be revised as

$$\hat{V}_{\mathrm{dd}} = V_{\mathrm{dd}} \frac{Q^{-1}(\hat{\varepsilon})}{Q^{-1}(\varepsilon)}, \qquad P_{\mathrm{uncode}}(\varepsilon) = P_{\mathrm{ECC}}(\hat{\varepsilon}). \tag{8}$$

The inverse function of the Gaussian distributed function is also called a probit function $\Phi(x)$. The probit function has proved that the function does not have primary primitive. To solve the problems, this work first approximates the bit error probability by varying voltage swing. By integrating from $100 - V_{\mathrm{dd}}/2$, the integral range on the x-axis is divided into 0.0001 (V) segments, and each segment can produce a trapezoid. The areas of all trapezoids are then summed, which is the approximation of bit error probability. Therefore, the lowest voltage swing for a specific error correction coding that satisfies (8) can be obtained.

When an uncoded code is operated at full swing supply voltage (1.0 V), different levels of bit error probability, ε, can be obtained by altering the variance of the Gaussian distributed function. Figures 4(a) and 4(b) show the voltages of specific error correction coding versus different uncoded word error rates with $k = 8$ and $k = 32$, respectively. Factor, k, is bit width. If bit error probability of an uncode word, ε, is 10^{-20}, the specific voltage of hamming code [20], duplication-add-parity code [20, 21], joint crosstalk avoidance and triple-error-correction code (JTEC) [24] and the proposed SCG code are 0.705 V, 0.710 V, 0.579 V, and 0.696 V, respectively. The JTEC code uses a double error correction coding stage to enhance error correction and obtains lower voltages. However, delay and area overheads of the JTEC are much worse than those of other approaches. Compared to other ECC codes, the proposed SCG code has better characteristics in that the lowest supply voltage increases slowly when the uncoded word error rate increases.

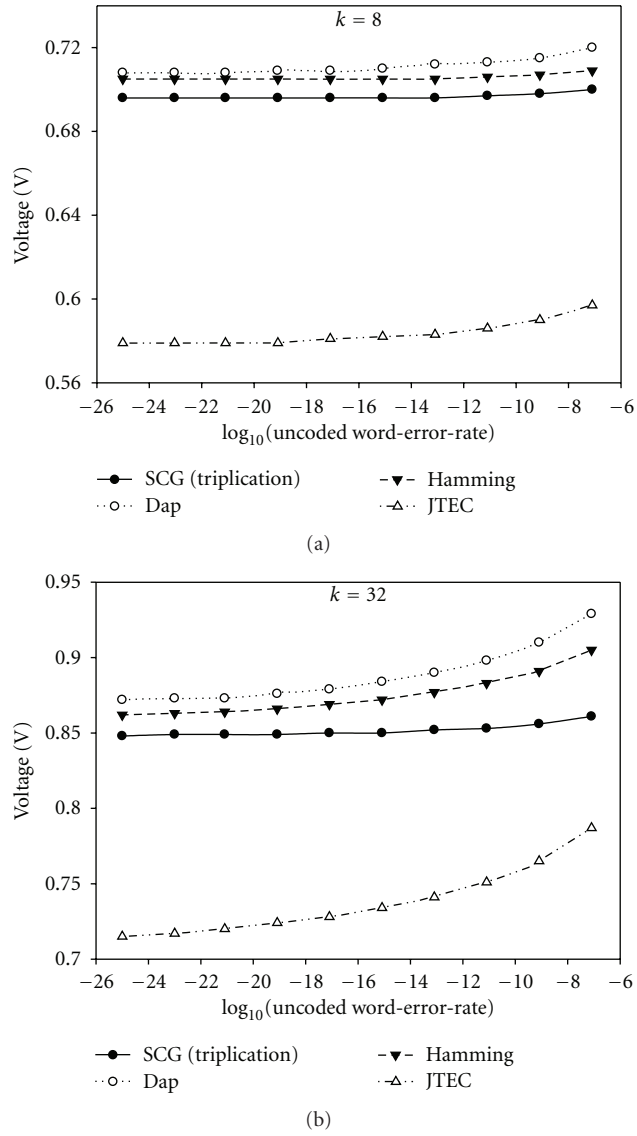


Figure 4: The corresponding voltages of specific error correction coding versus different uncoded word-error-rate with (a) $k = 8$ and (b) $k = 32$.

4.2. Joint Triplication Bus Power Model. Although triplication error correction coding can avoid many forbidden conditions, some power-hungry transition patterns cannot be eliminated entirely. These patterns are mainly generated by the FTC and self-switching activity. The FTC can be satisfied when a bit pattern does not have a transition from 01 to 10 or from 10 to 01. This work modified the RLC cyclic bus model in [31] by considering loading capacitances and coupling capacitances. Figure 5(a) shows the modified model with a four-bit bus, where C1 means the loading capacitance of line 1 and the C12 is the coupling capacitance between line 1 and line 2. Moreover, the bus lines are parallel and coplanar. Most of the electrical field is trapped between adjacent lines and the ground. Figure 5(b) shows an approximate bus power model that ignores the parasitic capacitances between nonadjacent lines.

We assume all grounded capacitors have the same value without considering the fringing effect of boundary lines, because fringing capacitors are much smaller than loading and coupling capacitors, even for the wide buses. Therefore, this work utilized a joint triplication bus model to implement the bus coding stage to further reduce energy consumption. For a 4-bit triplication bus, the capacitance matrix C^t can be expressed as

$$C^t = \begin{bmatrix} 3+\lambda & -\lambda & 0 & 0 \\ -\lambda & 3+2\lambda & -\lambda & 0 \\ 0 & -\lambda & 3+2\lambda & -\lambda \\ 0 & 0 & -\lambda & 3+\lambda \end{bmatrix} C_L, \qquad \lambda = \frac{C_X}{C_L}. \tag{9}$$

The parameter, λ, is defined as the ratio of coupling capacitance, C_x, to loading capacitance, C_L. Therefore, the λ parameter depends on the technology, the specific geometry, the metal layer, and bus shielding. λ has some important properties; for example, the parameter λ typically increases with

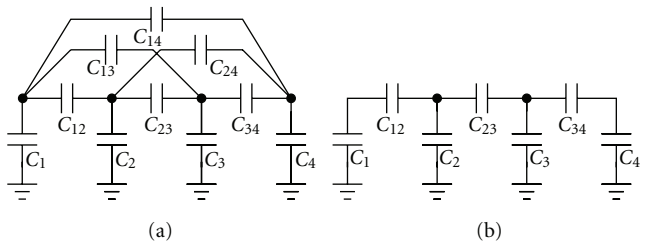


Figure 5: (a) Bus model for 4 bits. (b) The approximate bus power model.

technology scaling. For instance, the value of λ is between 6 and 10, depending on the metal layer for standard 65 nm CMOS technology and the minimum distance between wires. The parameter λ should be much larger in advanced technologies. Additionally, the coefficient of loading capacitances is 3 for the three triplicated bits.

Five transition states exist between two adjacent lines, four of which are described in [32]. These five types can be separated into two cases. The first case is static transitions, including type I (single line switching), type II (two lines switching in opposite directions), and type III (no switching or two lines switching in the same direction) as shown in Figure 6. The other case is dynamic transitions which include type IV and type V with signal aliasing for type II and type III, respectively. The static transition is defined as two adjacent lines switching at the same time without noise or different delays. The dynamic transition means that the two adjacent lines may be misaligned.

The power consumption formula is shown in (10), where E and P are energy and power density, respectively; f and V (V_{dd}) are frequency and voltage (voltage supply), respectively. B_i is the current voltage level (1 or 0) for line i, and B_i^{-1} is the previous voltage level for the line i;

$$E = \left(V^f\right)^T C^t \left(V^f - V^i\right),$$
$$P = f^* V_{\mathrm{dd}}^2 * \sum_i \sum_j C^t \left\{ \left(B_i - B_i^{-1}\right) * \left(B_j - B_j^{-1}\right) \right\}. \tag{10}$$

Power density, P, can be transferred into

$$P = f^* C_L^* V_{\mathrm{dd}}^2 * \begin{Bmatrix} 3\left(B_1 - B_1^{-1}\right)^2 + 3\left(B_2 - B_2^{-1}\right)^2 + 3\left(B_3 - B_3^{-1}\right)^2 \\ +3\left(B_4 - B_4^{-1}\right)^2 + \lambda\left[\left(B_1 - B_1^{-1}\right) - \left(B_2 - B_2^{-1}\right)\right]^2 \\ +\lambda\left[\left(B_2 - B_2^{-1}\right) - \left(B_3 - B_3^{-1}\right)\right]^2 \\ +\lambda\left[\left(B_3 - B_3^{-1}\right) - \left(B_4 - B_4^{-1}\right)\right]^2 \end{Bmatrix}. \tag{11}$$

The items in (11) are defined and identified as follows:

$$\left(B_i - B_i^{-1}\right)^2 = B_i \oplus B_i^{-1} = r_i,$$
$$\left[\left(B_i - B_i^{-1}\right) - \left(B_j - B_j^{-1}\right)\right]^2 = r_i \oplus r_j + 4 \times d_{ij}, \tag{12}$$
$$\text{where } d_{ij} = \overline{B}_i B_i^{-1} B_j \overline{B}_j^{-1} \cup B_i \overline{B}_i^{-1} \overline{B}_j B_j^{-1}.$$

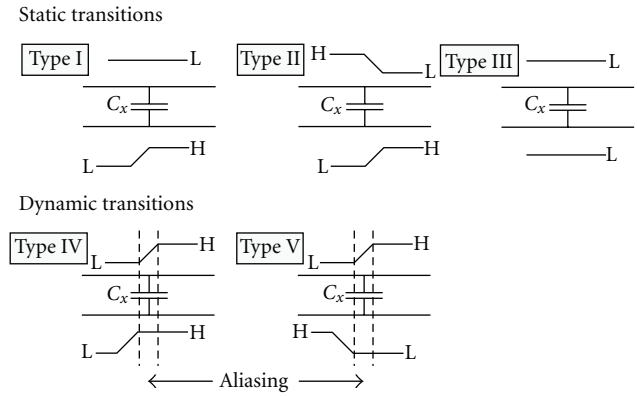


Figure 6: Five transition types for two adjacent wires.

The r_i means that a switch of line i *exists* and is not concerned with the direction of change and adjacent lines. This item, r_i, only considers loading capacitances. The meaning of $r_i \oplus r_j$ is that only one line is changing between two lines of i and j (Type I). Additionally, d_{ij} indicates that two lines change in opposite directions (Type II and Type V). Moreover, compared with the other two definitions, r_i and $r_i \oplus r_j$, the voltage difference across the coupling capacitance is double and when squared it factors 4 for d_{ij}. Using (12), the power formula can be obtained as (13) with the parameter of λ. The term α is the coefficient of coupling effects and switching activities. Except for Type IV, the five transition states are all considered in this power formula:

$$P = f \times C_L \times V_{\mathrm{dd}}^2 \times \alpha,$$
$$\alpha = 3(r_1 + r_2 + r_3 + r_4) + \lambda(r_1 \oplus r_2 + r_2 \oplus r_3 + r_3 \oplus r_4) + 4\lambda(d_{12} + d_{23} + d_{34}). \tag{13}$$

4.3. Green Bus Coding Stage for Crosstalk Avoidance. The purpose of the green bus coding stage is to minimize the value of α in (13) by encoding signals when $\lambda > 2$. Figure 7 shows design flow of green bus coding. First a triplication capacitance matrix is established using the RLC cyclic model. Then the power formula with coefficient α is derived, where α represents the switching factor by considering coupling capacitances. The green bus coding stage only affects coefficient α. Furthermore, the codeword minimizes the value of α and maps the codeword to the dataword. Depending on the mapping between the codeword and dataword, the green bus coding stage can be implemented.

According to the design flow of the green bus coding stage, the modified switching activity, α, should be minimized. Therefore, to converter the 4-bit dataword into a 5-bit codeword, a 32×31 transition state table is established by calculating α. Thus, 16 transition patterns are selected with minimal values of α as the codeword to eliminate crosstalk. The green bus coding chooses a 4 : 5 code to minimize α depending on the energy saving bound and the latency of codec. In a data bus, the bit width of a data is usually

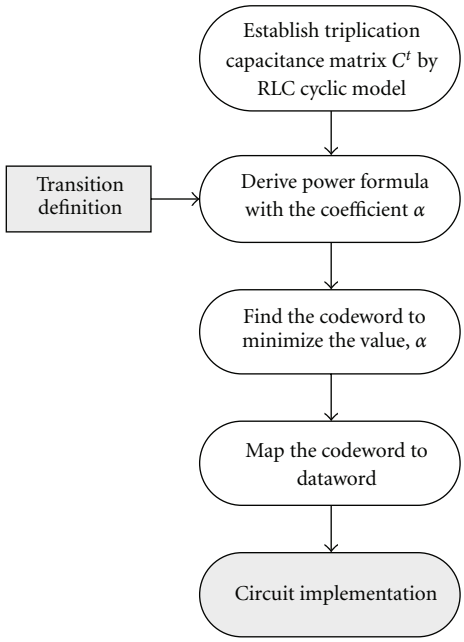
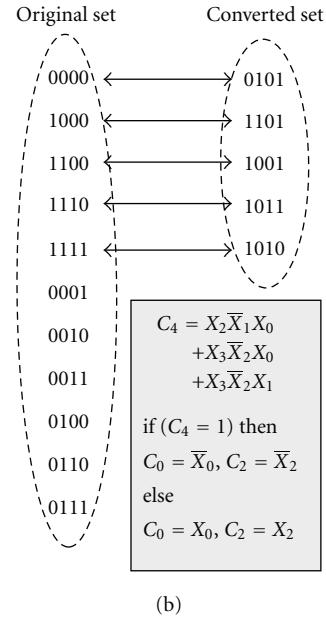


Figure 7: The design flow of the green bus coding stage.

Datawords $X_3 X_0$	Codewords $C_4 C_0$
0 0 0 0	0 0 0 0 0
0 0 0 1	0 0 0 0 1
0 0 1 0	0 0 0 1 0
0 0 1 1	0 0 0 1 1
0 1 0 0	0 0 1 0 0
0 1 0 1	1 0 0 0 0
0 1 1 0	0 0 1 1 0
0 1 1 1	0 0 1 1 1
1 0 0 0	0 1 0 0 0
1 0 0 1	1 1 1 0 0
1 0 1 0	1 1 1 1 1
1 0 1 1	1 1 1 1 0
1 1 0 0	0 1 1 0 0
1 1 0 1	1 1 0 0 0
1 1 1 0	0 1 1 1 0
1 1 1 1	0 1 1 1 1

(a)



(b)

Figure 8: (a) The mapping table between 4-bit dataword and 5-bit codeword of the green bus coding stage. (b) The two sets and Boolean expression of the green bus coding stage.

a multiple of 4. Therefore, the energy-saving bound of 4 : 5 to 4 : 8 codes are between 40% to 55% from the energy-saving bound analysis of [33]. However, the latency of the codec will increase significantly as the size of a codeword increases.

Figure 8(a) shows the relationships between the 4-bit dataword and 5-bit codeword. According to the relationships, the data-word can be grouped into two sets, the original set and the converted set as shown in Figure 8(b). When transmitted data are in the converted set, the green bus coding stage converts the data into the original set via one-on-one mapping. Meanwhile, the converted bit, $c4$, will be asserted, and $c0$ and $c2$ will be inverted and mapped to the original set. Notably, $x1$ and $x3$ will always not be modified.

Figure 9 shows the circuit implementation of green bus coding, including the *encoder* and *decoder*. The circuitry of green bus coding is more simple and effective than other approaches using the joint triplication bus model. An extra shielding line to reduce the coupling effect is not needed between two adjacent 5-bit codewords because the boundary data of the 5-bit codeword are set to roughly 0. Table 1 shows the comparisons between green bus coding and increasing wire spacing when $\lambda = 8$. Although increasing wire spacing can achieve more energy reduction than green bus coding, it has great amount of area overhead. Additionally, the energy-delay product (EDP) of green bus coding is smaller than that of double wire spacing.

The proposed green bus coding stage has the following properties.

(1) Use $c4$ as the detection bit to decode $c0$ and $c2$. It can simplify the circuitries of encoder and decoder, especially that of the decoder.

(2) The encoded bit always equals the data bit at certain bit positions, where $c1 = x1$ and $c3 = x3$.

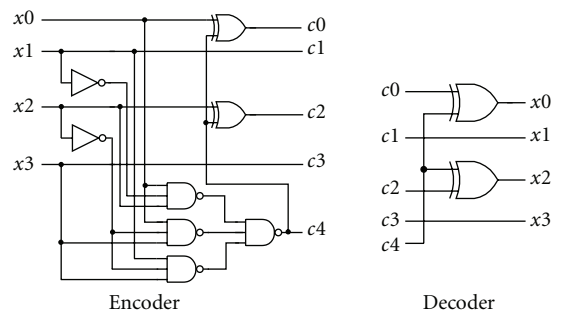


Figure 9: The encoder and decoder of green bus coding stage.

(3) By focusing on the joint bus and error correction coding scheme, the SCG coding scheme can avoid FOC and FPC and reduce FTC to further reduce power consumption.

(4) Adding extra shielding lines to reduce the coupling effect between two adjacent codeword with increasing coding bits is unnecessary.

(5) According to the delay model and energy model given by [33], the energy dissipation and critical delay are reduced from $(1 + 1.5\lambda)CV^2$ to $(1.18 + 1.17\lambda)CV^2$ and $(1 + 4\lambda)\tau_0$ to $(1 + 2\lambda)\tau_0$ via the green bus coding, respectively. τ_0 is defined as the delay of a crosstalk-free wire.

5. Self-Calibrated Voltage Scaling Technique

The proposed self-calibrated voltage scaling technique is applied to reduce the operating voltage of channels for energy

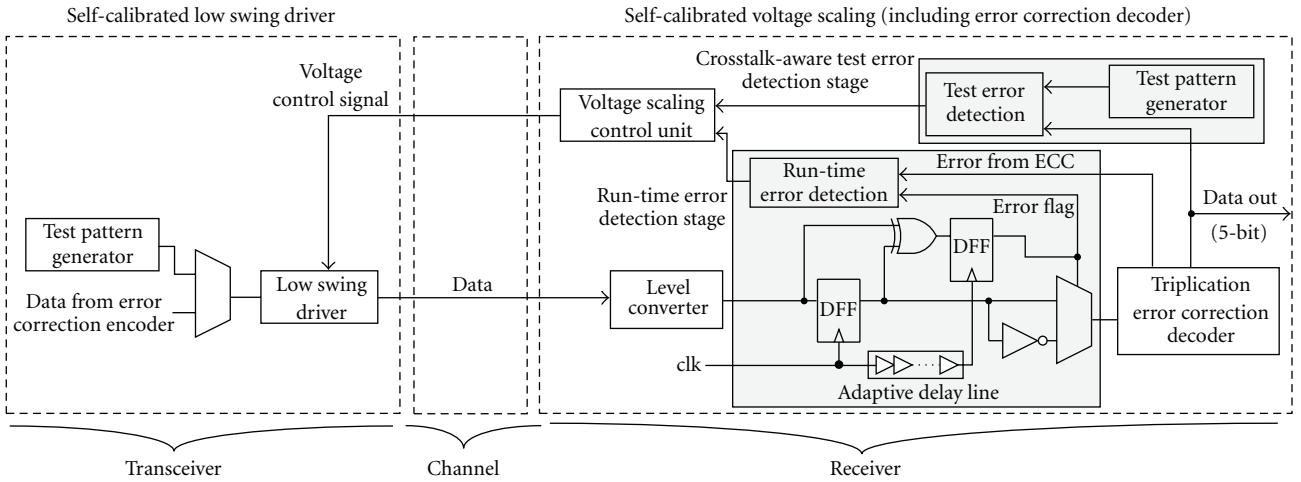


Figure 10: The block diagrams of self-calibrated voltage scaling technique with crosstalk-aware test error detection stage and run-time error detection stage.

Table 1: Comparisons between green bus coding and increasing wire spacing.

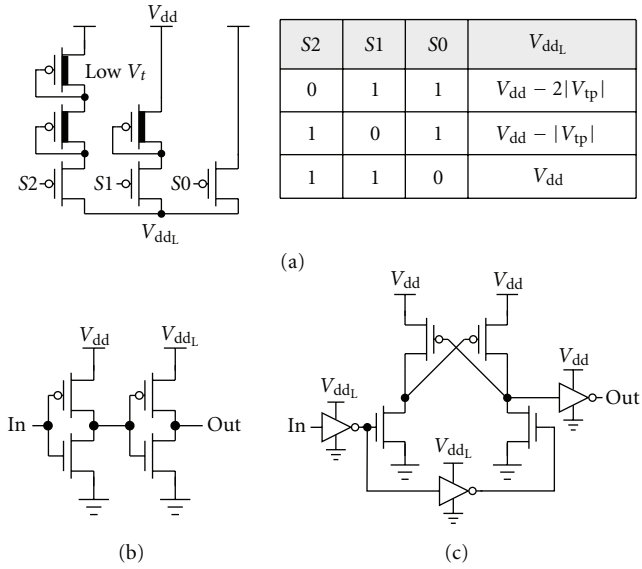
(4-bit, $\lambda = 8$)	Area overhead	Energy reduction	Delay reduction	EDP reduction
Green bus coding	19%	19.7%	49.5%	59.3%
Double spacing	23%	23.2%	30.1%	46.4%
Quadruple spacing	129%	37.3%	39.2%	61.9%

reduction and ensure the reliability based on the SCG coding scheme. The self-calibrated voltage scaling technique will identify the optimal operating voltage to trade off between energy consumption and reliability for the self-calibrated circuitry. Figure 10 presents the block diagrams of the self-calibrated voltage scaling technique. This technique is constructed by comprising low swing drivers, level converters, voltage scaling control unit, crosstalk-aware test error detection stage, and run-time error detection stage. Depending on the detections about the two error detection stages, the voltage control unit adjusts voltage swing levels of the link wires. The crosstalk-aware test error detection stage detects errors by maximal aggressor fault (MAF) test patterns in the test mode. The run-time error detection stage detects errors using the double sampling data checking technique and the adaptive delay line. Moreover, the self-calibrated voltage scaling technique is tolerant of timing variations by the adaptive timing borrowing technique. In response to detected errors, the self-calibrated voltage scaling technique can reduce voltage swing for energy reduction and guarantee the reliability is still in the confidence interval simultaneously.

Based on the SCG coding scheme, the triplication error correction coding stage can correct errors for link wires. The SCG coding scheme allows for reductions in signal voltage swing and, at the same time, achieves the same word error rate of uncoded link wires. When the bit error rate is in the range from 10^{-20} to 10^{-10}, a 0.7 V signal swing for link wires can maintain the same reliability with the uncoded code at 1.0 V as shown in Figure 4. Therefore, a low swing driver and level converter are implemented with three voltage levels as shown in Figure 11, which are high voltage (HV = V_{dd}), middle voltage (MV = $V_{dd} - V_t$), and low voltage (LV = $V_{dd} - 2V_t$). The PMOS diodes are utilized to produce low swing voltages as shown in Figure 11(a) by low-V_t PMOS. In UMC 65 nm CMOS technology, the threshold voltage of normal-V_t and low-V_t PMOS are 0.25 V and 0.15 V, respectively. Therefore, the voltage level will be two levels by normal-V_t device. In order to realize the lowest voltage, 0.7 V, low-V_t PMOS, and three voltage levels are selected. Three control signals, S0–S2, determine the voltage swing of link wires, and Figure 11(a) shows the relationships between control signals and voltages. Based on the different voltages, the low swing driver and level converter can be implemented as shown in Figures 11(b) and 11(c), respectively. Therefore, the timing overhead of switching voltage can be in one cycle.

Figure 12 shows the control policy and voltage state diagram of the self-calibrated voltage scaling technique. Therefore, the crosstalk-aware test error detection stage is triggered by T_start, and crosstalk-aware test vectors are generated. Test results are compared by the test error detector. Initially, the crosstalk-aware test vectors are transmitted at the lowest voltage level of 0.7 V. In terms of error correction coding, the error should be zero by the test error detector. If the error detector detects errors, the test vectors will be transferred again with a relatively higher voltage (0.85 V or 1 V). The initial voltage swing of link wires is determined until the test result is free of errors. When the test is finished, the run-time error-detection stage will be activated.

After the crosstalk-aware test error detection stage, the run-time error detection stage raises V_scale to trigger a scaling mechanism within every N clock cycles window. Based on the error rate, the voltage control unit can further increase or decrease the signal voltage swing during run-time. But



$S2$	$S1$	$S0$	V_{dd_L}
0	1	1	$V_{dd} - 2\|V_{tp}\|$
1	0	1	$V_{dd} - \|V_{tp}\|$
1	1	0	V_{dd}

Figure 11: (a) Low swing voltages. (b) Low swing driver. (c) Level converter.

the voltage in the run-time error detection stage cannot be lower than the voltage level determined by the crosstalk-aware test error detection stage. The error rate is defined as the ratio of the error data to the total transmission data in one window. If the error rate is less than 5%, signal voltage swing is reduced one level or kept at the lowest safe signal. However, if the bit error rate is larger than 5% but less than 15%, the signal voltage swing level is the same as that for the previous window. If the error rate is larger than 15%, signal voltage swing is increased one level or kept at the highest signal swing level. The range of bit error rate detection depends on properties of SCG coding scheme. If uncoded input data are random, the probability of the forbidden pattern condition (two adjacent lines switch in opposite directions, e.g., ↑↓ or ↓↑) of the coding scheme is roughly 15%. Additionally, the 5-bit voltage scaling control unit can determine 5% and 15 % error rate by an 8-bit adder in 256 cycles (detection window).

5.1. Crosstalk-Aware Test Error Detection Stage. The crosstalk-aware test error detection stage is composed of a test pattern generator (TPG), a test error detector (TED), and a control unit that generates the control voltages for the low swing driver. The crosstalk-aware test error detection stage is triggered by T_start, and then generates crosstalk-aware test vectors. Conventional test pattern generators, such as the liner feedback shift register (LFSR) [34, 35], generate pseudorandom pattern sequences. By changing the feedback polynomial of the LFSR, the LFSR generates different subsets of the maximum-length LFSR (maximum $2^n - 1$ patterns when the LFSR tests n-bits data with primitive polynomials). However, test patterns generated by the LFSR-based TPG are complicated and require a long test time to achieve high error coverage. Hence, a better self-test methodology is needed to achieve low hardware overhead, fast test time, and high error coverage.

Depending on test vectors, therefore, the test error detector can detect error data following error correction coding. The crosstalk-aware test vectors are generated by a test pattern generator with the maximal aggressor fault (MAF) model as shown in Figure 13 [36]. The MAF-based test patterns are a simple pattern stream that represents six different crosstalk effects: rising speedup (Sr), falling speedup (Sf), rising delay (Dr), falling delay (Df), positive glitch (Gp), and negative glitch (Gn). For test wires with n-bits, one victim line and $n - 1$ aggressor lines exist. All aggressor lines switch simultaneously to generate speedup, delay, or glitch error on the victim line. The MAF test vectors can achieve high error coverage. Additionally, the MAF-based test can be considered as an aggressive test that covers other pattern transition cases. To test n-bit on-chip interconnects, six fault models must be tested on each line. Therefore, testing n-bit needs $6n$ test pattern transitions to complete an MAF-based test.

The test pattern generator of the MAF-based self-test methodology is implemented by the finite state machine (FSM). The FSM needs a minimum of 8 cycles to complete six faults tests on one victim line, indicating that the test pattern generator requires $8n$ cycles to complete an n-bit MAF test. Test time is much shorter than that of the linear feedback shift register. The FSM, which is triggered by T_start signal, generates the values of the victim line and the aggressor line, counter reset (C_reset) and counter enable (C_enable). After each circle (states S1–S8) of the FSM, C_enable triggers the victim counter. The decoder and output 2-to-1 MUX are selected to ensure that the data bit (Di) selects the correct value (victim or aggressor value) during the test. When the value of the victim counter (C_value) is equal to $n - 1$ in the S8 state, the test is finished and returns to the S0 state.

5.2. Run-Time Error Detection Stage. The run-time error detection stage detects timing variations of link wires. Timing delay variations of on-chip interconnections are due to crosstalk noise, process variations, temperature variations, and other noises. To overcome timing error, the master-slave flip-flop (MSFF) [37] and double sampling data checking technique [38] have been proposed to detect timing errors. The MSFF contains a master flip-flop and a slave flip-flop, both of which operate at the same frequency. However, the slave flip-flop is positively triggered by a delay clock (Δt) which is proportion to master flip-flop. We assume the data captured by the slave flip-flop is correct. The data captured by the master flip-flop and the slave flip-flop are compared using an XOR gate; an error-flag is generated when the two data are not identical. When an error occurs, the control circuit stalls pipeline data flow for 1 clock and the slave flip-flop resends correct data to the master flip-flop. The principle of the double sampling data checking technique is similar to that of the MSFF.

The timing delay variation of on-chip interconnects affects the design on Δt. The different propagation delay on the on-chip interconnection caused by crosstalk is due to different pattern transients. For the increasing timing variation of

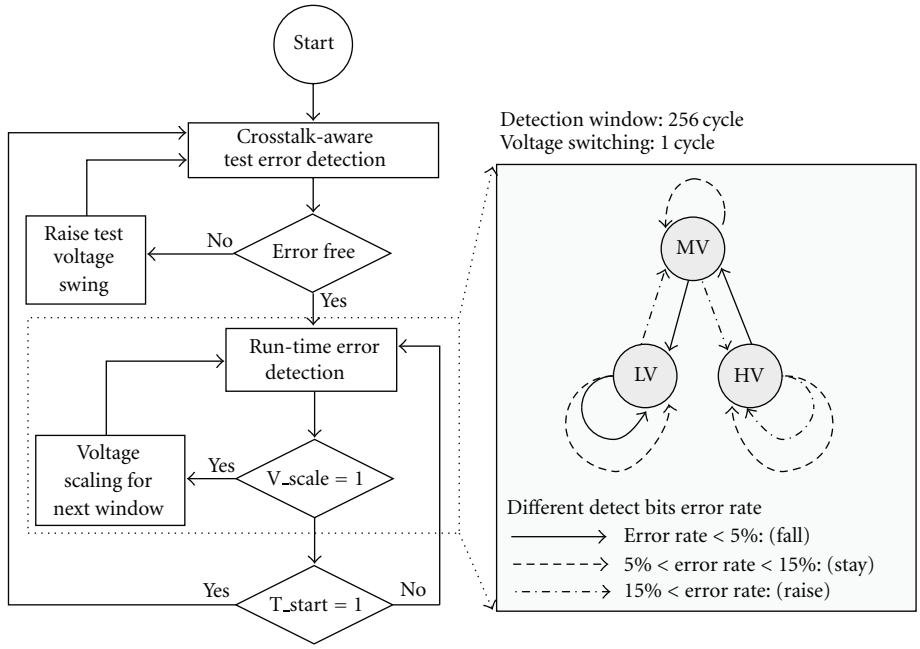


Figure 12: The control policy of self-calibrated voltage scaling technique.

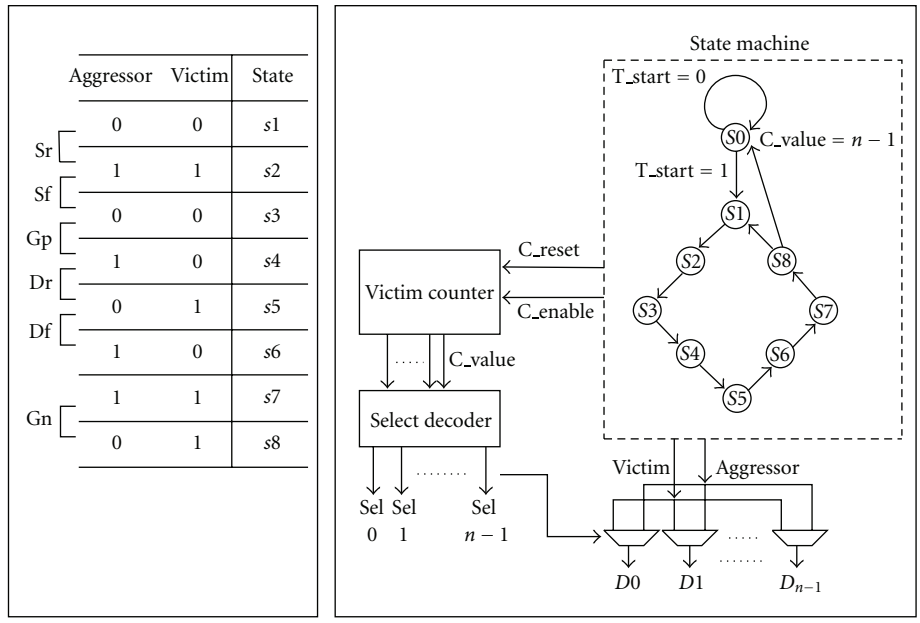


Figure 13: MAF-based test pattern generator (a) 8 states complete 6 faults test of MAF model. (b) Hardware implementation.

on-chip interconnections, detecting timing error is difficult for various voltage levels. However, the MSFF and double sampling data checking technique are limited by the clock period and fixed delay line, respectively. Therefore, the run-time error detection stage is constructed using the adaptive timing borrowing technique as shown in Figure 10. The adaptive timing borrowing technique modifies the double sampling data checking technique with the adaptive delay line. In addition, the adaptive timing borrowing technique also has correction ability via a multiplexer. The modified double sampling data checking technique with the adaptive delay line has the adaptive timing borrowing ability to borrow timing from the next clock period.

Figure 14 presents analytical results for timing constraints. To ensure that functionality of the modified double sampling data checking technique is correct, time interval Δt must be set appropriately, and each pipeline stages must be considered. If the delay between DFF1 and DFF2 exceeds l

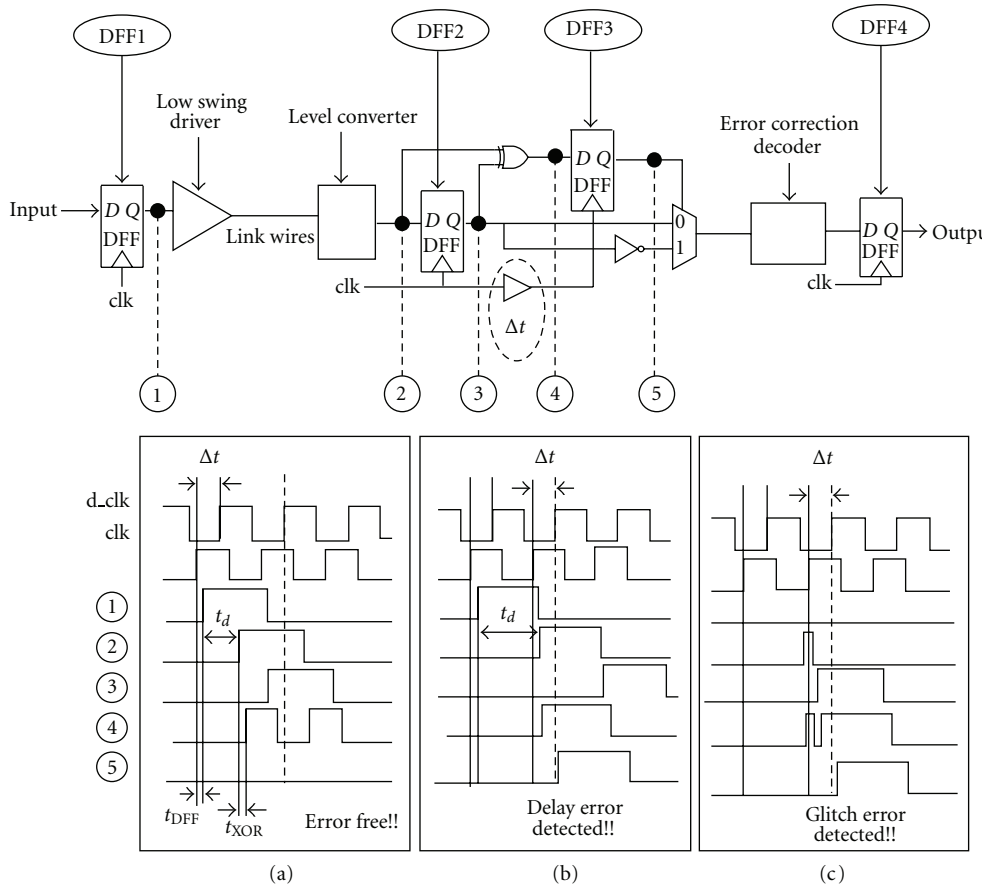


Figure 14: Modified double sampling data checking circuit and waveforms. (a) Error-free. (b) Delay error. (c) Glitch error.

clock cycle, error sampling data of DFF1 are induced. The maximum data path delay can be extended to 1 clock cycle plus time interval Δt, as in (14), where t_{DFF} is the clock to Q delay of the D flip-flop, and t_d is the data path delay (from the input of the low swing driver to the output of the level converter), t_{XOR} is the XOR propagation delay, and t_{setup} is the setup time of the D flip-flop,

$$t_{\mathrm{DFF1}} + t_d + t_{\mathrm{XOR}} + t_{\mathrm{setup3}} < \tau_{\mathrm{clk}} + \Delta t. \tag{14}$$

DFF3 samples the comparison signal, which compares sampling data before DFF2 and after DFF2. In addition, DFF3 must sample the comparison signal before next datum arrives. Therefore, Δt should be satisfied as

$$t_{\mathrm{DFF2}} + t_{\mathrm{XOR}} + t_{\mathrm{setup3}} < \Delta t < t_{\mathrm{DFF1}} + t_d + t_{\mathrm{XOR}} + t_{\mathrm{setup3}}. \tag{15}$$

Additionally, the pipeline stages after the double sampling data checking stage must satisfy basic constraints, as in the following equation, to avoid the excessive timing borrowing:

$$\Delta t + t_{\mathrm{DFF3}} + t_{\mathrm{MUX}} + t_{\mathrm{Decoder}} + t_{\mathrm{setup4}} < \tau_{\mathrm{clk}}. \tag{16}$$

Equations (14) and (15) are the timing conditions that avoid error detections, (16) is the timing condition that prevents setup timing violation of the sequential circuitry. According to (14)–(16), the upper and lower bounds of time interval Δt are derived by the following equation. When the time interval Δt is appropriate, the run-time error detection stage corrects error data and provides run-time error rate information, allowing the self-calibrated voltage scaling technique to adjust the voltage swing levels of link wires:

$$\begin{aligned}&\mathrm{Max}\Big\{\Big(t_{\mathrm{DFF1}} + t_d + t_{\mathrm{XOR}} + t_{\mathrm{setup3}} - \tau_{\mathrm{clk}}\Big),\\ &\quad\Big(t_{\mathrm{DFF2}} + t_{\mathrm{XOR}} + t_{\mathrm{setup3}}\Big)\Big\}\\ &< \Delta t < \mathrm{Min}\Big\{\Big(t_{\mathrm{DFF1}} + t_d + t_{\mathrm{XOR}} + t_{\mathrm{setup3}}\Big),\\ &\qquad\Big(\tau_{\mathrm{clk}} - t_{\mathrm{DFF3}} + t_{\mathrm{MUX}} + t_{\mathrm{Decoder}} + t_{\mathrm{setup4}}\Big)\Big\}.\end{aligned} \tag{17}$$

If (14) is not satisfied, a type I statistical error occurs. The double sampling data checking technique cannot detect true errors, and suppose that the sampling data would be correct. On the other hand, if (15) is not satisfied, the type II statistical error occurs. The double sampling data checking technique then misjudges and asserts an error flag when the transferred data is correct.

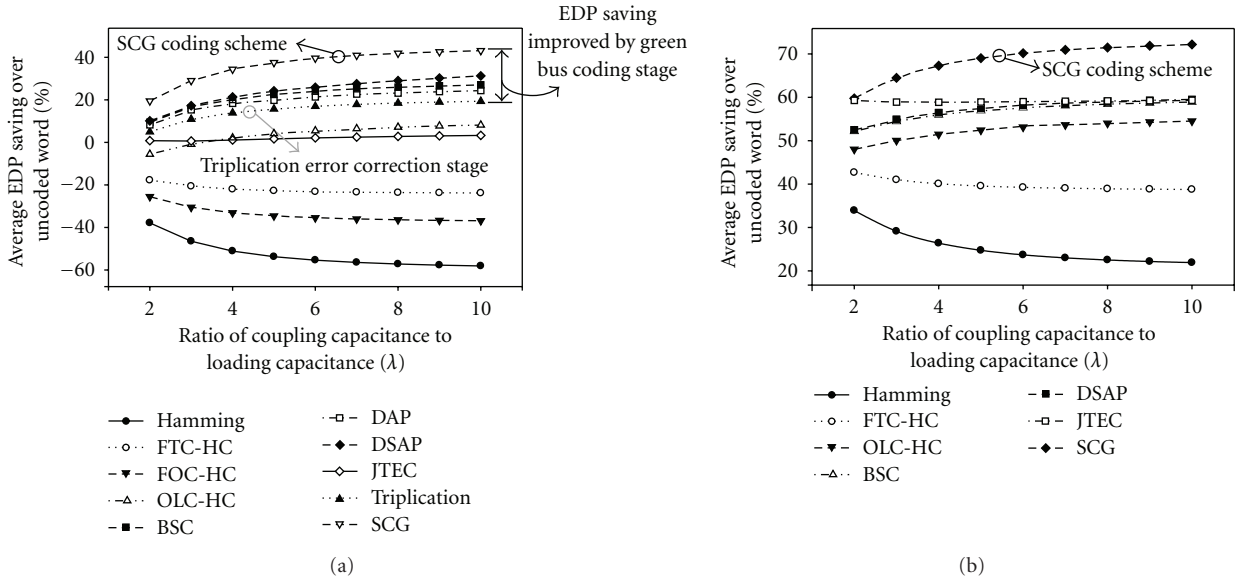


Figure 15: The energy-delay product (EDP) reduction to uncoded code under different values of λ with (a) full swing signal and (b) the lowest swing signal.

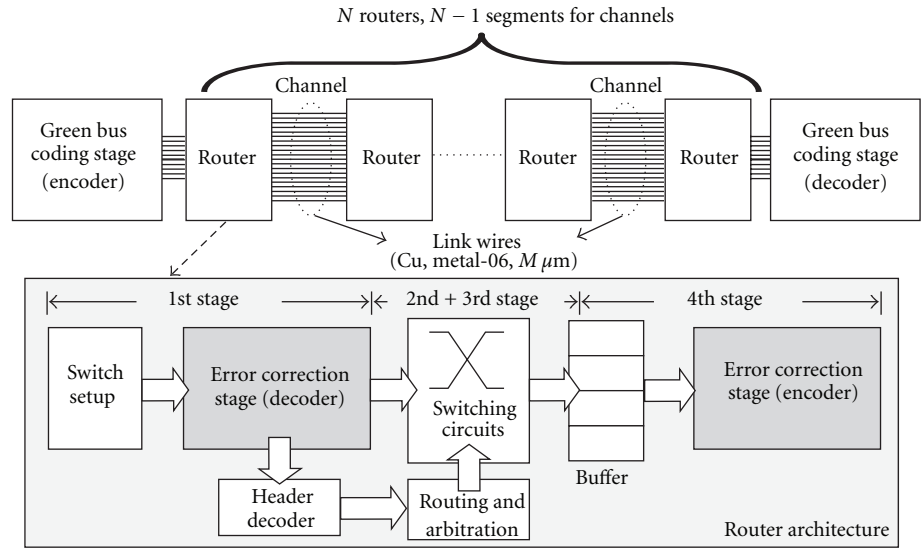


Figure 16: Simulation environment setup with different number of routers (N) and different lengths (M) of link wires.

Timing delay variation is caused by the crosstalk effect, process variation, width variation, and voltage variation. In view of increasing timing variation, the adaptive delay line is an effective solution that satisfies these conditions. Furthermore, data path delay t_d is affected significantly by operating voltages and input vectors. Therefore, the adaptive delay line can generate three time intervals Δt for different signal voltage levels to satisfy the timing condition in (17); thus, the adaptive delay line can be implemented by a digital control delay line with MUXs. Adjusting the time interval Δt guarantees the functionality of double sampling data checking technique with different voltage swing levels and process variations.

6. Simulation Results

This section presents simulation results demonstrating the improvement in energy and reliability via the SCG coding scheme and the self-calibrated voltage scaling technique. All simulation results are based on UMC 65 nm 1P9M CMOS technology. For OCINs, the metal layers can be categorized into upper-level, middle-level, and lower-level, respectively. In most cases [39–41], the upper-level metal layers are routed for power grids and global clock distribution via low resistance metals. Additionally, the lower-level metal layers are routed for local resources. Therefore, the characteristics of link wires between interprocessor elements are set as metal-6 with a minimum width and spacing of

0.10 μm in UMC 65 nm 1P9M CMOS technology. Simulation results include analysis of different error-correction coding schemes, energy-delay product (EDP) of different joint coding schemes, energy saving of SCG coding in an 8×8 mesh network, process-variation timing analysis, and analysis of the self-calibrated voltage scaling technique.

Table 2 lists different combinations of joint coding schemes, such as the hamming code (HC), FTC+HC, FOC+HC and boundary shift code (BSC) in [23], one lumbda code (OLC)+HC and DAP+shielding (DSAP) in [25], JTEC in [24], and the proposed SCG coding scheme. Additionally, Table 2 summarizes different joint coding schemes for 8-bit link wires, which consist of the physical transfer unit size in channels and routers, the maximum delay and average energy of link wires, and the corresponding lowest supply voltage. Table 2 also summarizes the codec of different approaches, including the corresponding codec area, power, and latency. The lowest supply voltages are theoretical values from Figure 5 when $\varepsilon = 10^{-20}$. The JTEC uses double error correction coding to enhance error correction. However, codec overhead and energy dissipation (unoptimized JTEC for 8-bit) are much worse than those of other approaches. Although the JTEC can reduce the supply voltage to the lowest point at the same uncoded word-error-rate, the latency is larger than others due to long chains of XOR gates. Furthermore, the lowest voltage of JTEC increases rapidly as bit error rate increases.

Except for the SCG coding, DAP and DSAP, the critical delays of other codec are larger than 0.5 ns. Consequently, these codecs are not appropriate for integration into high-speed routers. Therefore, the physical transfer unit sizes in routers of these codecs are bigger than that of proposed coding scheme; thus network area and energy consumption increase. The delay of green coding stage and triplication error correction stage are 0.28 ns and 0.09 ns, respectively. And the power consumption of triplication error correction stage is only 41.5 μW. Hence, the proposed SCG coding scheme has the smallest codec overhead. Additionally, the green bus coding stage is only integrated in the sender node and receiver node.

The delay and energy of link wires are calculated via the delay model and energy model given by [33], where τ_0 is defined as the delay of a crosstalk-free wire. The proposed SCG coding scheme achieves the most energy reduction by reducing coupling effects on link wire, and avoids the FOC and FPC by the triplication error correction coding stage. Additionally, the SCG coding scheme can reduce the FTC and self-switching activities using the green bus coding stage depending on the joint triplication power model. Although the triplication error correction stage triplicates transferred data and increases the physical transfer unit size on link wires, it also enhances data reliability and avoids the worst crosstalk patterns. Moreover, the delay can be reduced from $(1 + 4\lambda)\tau_0$ to $(1 + 2\lambda)\tau_0$.

Figure 15(a) shows the energy-delay product (EDP) reduction compared to uncoded code under different λ values. Coefficient λ is defined as the ratio between coupling capacitance of two adjacent lines and loading capacitance. The energy and the delay are measured as the average energy

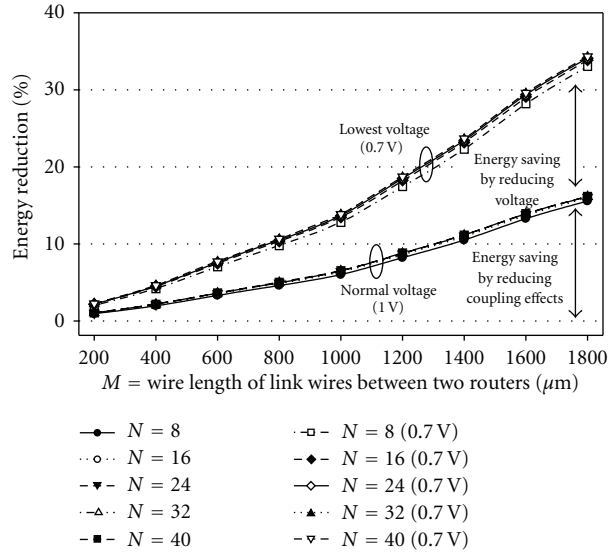


Figure 17: Energy reduction under different lengths of link wires and different number of routers.

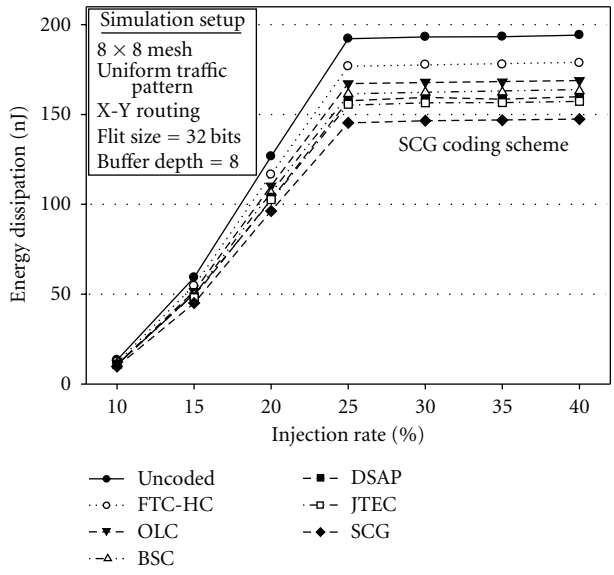


Figure 18: Energy dissipation of an 8×8 mesh-NoC with different joint CAC and ECC coding schemes.

dissipation in 1ns and the propagation delay from the transmitter to the receiver, respectively. The proposed SCG coding achieves the highest EDP reduction regardless of the value of λ. Through the tradeoff between reliability and power consumption, the signal swing levels of specific codes can be reduced further to the lowest values based on the error correction abilities. The lowest signal swing guarantees the same level of word error rate as that of the uncoded code. Figure 15(b) shows the energy reduction compared to uncoded code under different λ values and the lowest signal swing level. Simulation results indicate that the proposed SCG coding realizes more EDP saving than other joint coding

Table 2: Summaries of different joint coding schemes for 8-bit link wires.

Category	Coding scheme	Crosstalk avoidance coding	Error correction coding	Linear crosstalk coding	Phit size (wire)	Phit size (router)	Link wires (8-bit)			Codec		
							Delay (τ_0)	Avg. energy (CV_{dd}^2)	Lowest V_{dd} (V)	Area (μm^2)	Delay (ns)	Power (μW)
ECC	Hamming	—	Hamming	—	12	12	$1 + 4\lambda$	$3.00 + 5.50\lambda$	0.705	253.3	0.73	190.9
ECCx2	JTEC	Duplication	Hamming + Parity	—	25	25	$1 + 2\lambda$	$6.25 + 4.00\lambda$	0.579	512.2	0.93	311.1
CAC + ECC	FTC-HC	FTC	Hamming	Shielding	21	21	$1 + 2\lambda$	$3.38 + 4.77\lambda$	0.705	465.5	0.83	253.2
	FOC-HC	FOC	Hamming	—	16	16	$1 + 3\lambda$	$3.19 + 5.14\lambda$	0.705	421.3	0.59	250.1
	OLC-HC	OLC	Hamming	Shielding	34	34	$1 + \lambda$	$6.76 + 4.91\lambda$	0.710	961.6	0.62	321.3
	BSC	Duplication	Parity	—	17	17	$1 + 2\lambda$	$4.13 + 3.81\lambda$	0.710	488.4	0.73	207.6
	DAP	Duplication	Parity	—	17	8	$1 + 2\lambda$	$4.25 + 4.00\lambda$	0.710	146.3	0.35	68.8
	DSAP	Duplication	Parity	Shielding	25	8	$1 + \lambda$	$4.25 + 4.00\lambda$	0.710	149.2	0.35	68.9
	SCG green/ triplication	Green	Triplication	—	30	10	$1 + 2\lambda$	$7.05 + 2.77\lambda$	0.696	266.3	0.28/0.09	103.0/41.5

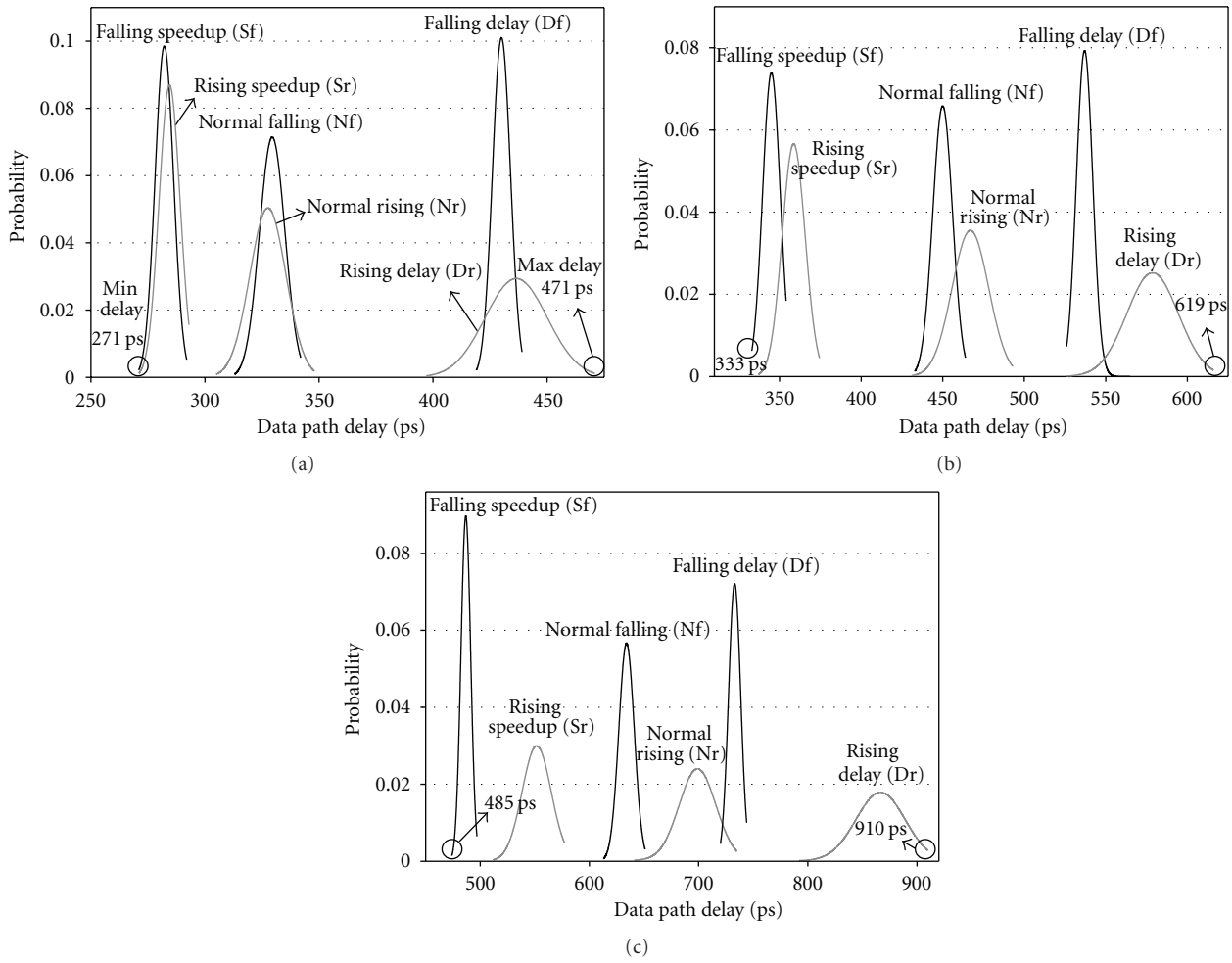


Figure 19: The data path delay (t_d) distributions of rising speedup, falling speedup, rising delay, falling delay, normal rising, and normal falling cases under (a) high voltage (1.0 V), (b) medium voltage (0.85 V) and (c) low voltage (0.7 V).

schemes. When λ equals 4 with a full swing signal (1.0 V), the SCG coding scheme can achieve a 34.34% EDP reduction compared to uncoded word and a 56.54% EDP reduction relative to that achieved by traditional hamming codes. The coding schemes can further increase EDP savings at the lowest operating voltages. In Figure 15(b), the proposed SCG coding achieves a 67.29% EDP saving relative to that achieved by the uncoded word when λ is 4 and operating voltage is 0.69 V.

The proposed SCG coding is also simulated with different lengths of link wires. Figure 16 shows the simulation environment setup with different number of routers (N) and various lengths (M) of link wires. The green bus coding stage is only integrated in the routers of the sender node and receiver node. The architecture of the routers is set as 5 input/output ports with 4-stage pipeline for mesh interconnection networks. The first stage includes switch setup, error correction decoder, and header decoder. The second stage and third stage are routing traversal and arbitration, respectively. The final stage is error correction encoder and link wires. The length of link wires is set as M μm of metal-6 with a minimum width and spacing of 0.10 μm. The clock frequency is as high as 1 GHz. Figure 17 illustrates energy reduction with different number of routers (N), different lengths (M) under the normal voltage (1.0 V), and lowest voltage (0.7 V). According to some NoC chips [39–41], the length of link wires is set from 200 μm to 1800 μm. The energy reduction increases while the length of link wires increases. Additionally, both reducing coupling effect and supply voltage can achieve significant energy saving by the SCG coding scheme.

Figure 18 shows the energy dissipation of an 8×8 mesh interconnection network with different joint CAC and ECC coding schemes under their lowest supply voltages. The simulation environment is set as an 8×8 mesh topology with uniform random patterns. The routing and arbitration

algorithms are XY routing and round robin, and The FIFO depth of each output buffer is 8 flits. The size of each flit size is 32 bits. The length of link wires is set as 800 μm of metal-6 with a minimum width and spacing of 0.10 μm. The clock frequency is as high as 1 GHz. In order to reach 1 GHz, the 32-bit uncoded data is divided into four 8-bit groups for different joint CAC and ECC coding schemes. The proposed SCG coding scheme can realize the most energy saving compared to other joint CAC and ECC coding schemes.

The self-calibrated voltage scaling technique is designed and simulated with the SCG scheme based on UMC 65 nm CMOS technology. The length of link wires is set as 800 μm of metal-6 with a minimum width and spacing of 0.10 μm. The clock frequency is as high as 1 GHz. Therefore, the timing of link wires should be analyzed with different voltage levels and process variations. The different transient patterns must also be considered. This analysis can help designers implement the adaptive delay line and guarantee correct function of the double sampling data check mechanism. The modified double sampling data checking circuit provides error information for the self-calibrated voltage scaling mechanism during run-time. However, the time interval, Δt, must satisfy the constraint discussed in Section 5. The data path delay, t_d, is clearly affected by voltages (swing levels of link wires) and input data vectors. Additionally, PVT (process, voltage, and temperature) variation affects both devices and on-chip wires. Therefore, the delays of link wires are analyzed using Monte Carlo simulations of PVT variation at different voltage levels.

Figures 19(a)–19(c) show the data path delay, t_d, of rising speedup (Sr) case, falling speedup (Sf) case, rising delay (Dr) case, falling delay (Df) case, normal rising(Nr) case and normal falling (Nf) case under high voltage (1.0 V), medium voltage (0.85 V), and low voltage (0.7 V), respectively. The supply voltages have a 15% variation in 3σ range and the means are 1.0 V, 0.85 V, and 0.7 V. The maximum value and minimum value of t_d occur in the Dr case and Sf case. The maximum and minimums value under 0.7 V, 0.85 V and 1 V are 910/485 (ps), 619/333 (ps), and 471/271 (ps), respectively. According to (12)–(15), the upper bounds of Δt under 0.7 V, 0.85 V and 1 V are about 485 ps, 333 ps, and 271 ps, respectively. Operating voltage obviously influences the timing interval. Therefore, the adaptive delay line can generate three time intervals, Δt, for different signal voltage levels: 450 ps, 300 (ps), and 200 (ps), which are 45%, 30%, and 20% of a clock period. Therefore, the adaptive delay line can be designed using a digital control delay line. Adjustments to the time interval guarantees functionality of double sampling data checking technique at different voltage swing levels and process variations. Nevertheless, analysis indicates that timing delay variation on link wires is much smaller under high operating voltage. In other words, if the error rate detected by the double sampling data checking technique increases, the control unit will increase the voltage to narrow the timing variation and enhance reliability.

Figure 20 illustrates the adaptive voltage by the self-calibrated voltage scaling technique under six phases with different noise distributions and timing variations. The noise distributions (σ_v) and timing variations (σ_d) are distributed

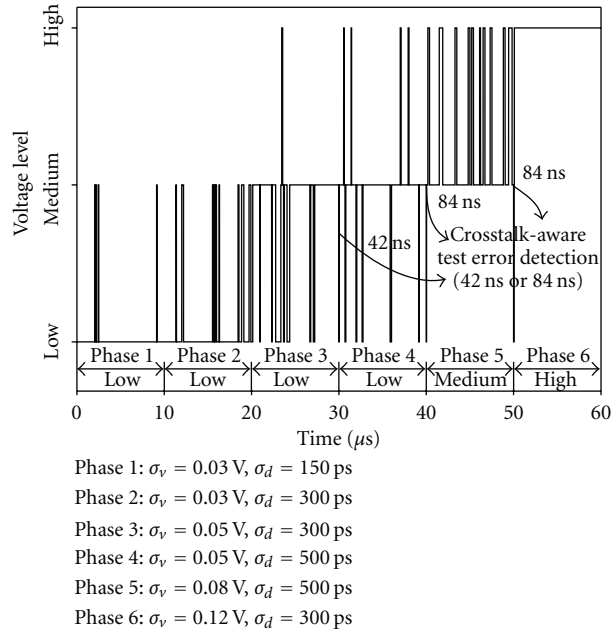


Figure 20: Voltage levels of the self-calibrated voltage scaling technique under six phases with different noise distributions and timing variations.

in $|3\sigma|$ range. The timing variations may be caused by process variation, temperature variation, large current density, and coupling effect. The control policy of the proposed self-calibrated voltage scaling technique is well described in Section 5. The test time of the crosstalk-aware test error detection stage is 42 cycles (40 cycles for testing, 2 cycles for feedback and adjusting voltage) or 84 cycles. In phases 1–4, the initial voltage level is the lowest voltage determined by the test stage. Additionally, the initial voltage levels in phase 5 and phase 6 are medium and high, respectively. The voltage in the run-time error detection stage cannot be lower than the voltage level determined by the crosstalk-aware test error detection stage. Therefore, in phase 6, the voltage level is always high in the run-time stage. Based on the error rate, the voltage control unit can further increase or decrease the signal voltage swing during run-time. The timing overhead of voltage switching is 1 cycle over (256 + 2) cycles.

In OCINs, link wires in channels dominate the overall power consumption in advanced technologies. The proposed SCG coding scheme eliminates most crosstalk effects and achieves energy reduction. From Figure 15(b), the EDP reduction of low swing link wires can reach above 60% compared with that of an uncoded bus when low swing drivers are operating at 0.7 V. The proposed self-calibrated voltage scaling technique finds the optimal operating voltage, and the tradeoff between energy consumption and reliability is determined by the self-calibrated circuitry. However, the power overhead of the self-calibrated voltage scaling technique reduces the energy efficiency of the channels. Figure 21 shows the energy analysis of the proposed self-calibrated energy-efficient and reliable channels at different voltages. The wire length is set as 1800 μm. The SCG coding

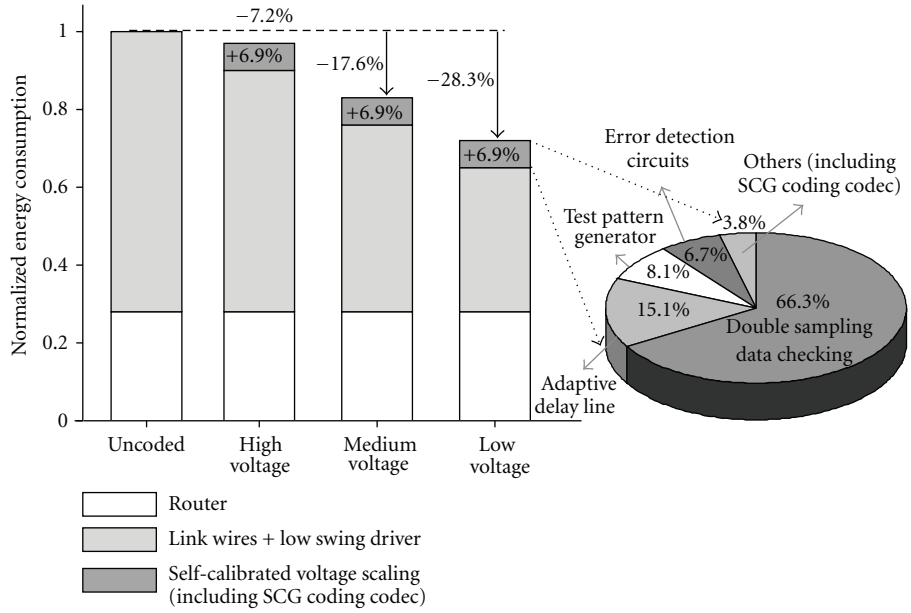


Figure 21: Energy analysis of the self-calibrated energy-efficient and reliable interconnection architecture.

stage reduces the energy consumption about 14.1% by decreasing the coupling effect and self-switching activities. From Figure 21, the total overhead of the SCG coding scheme and self-calibrated voltage scaling technique is roughly 6.9%. To elucidate the energy overhead, the right side in Figure 21 shows the energy breakdown of the SCG coding and self-calibrated voltage scaling. The double sampling data checking mechanism with the adaptive delay line accounts for almost 80% of energy overhead as a large number of flip-flops is needed. If error correction decoders are moved to before the run-time error detection stage, energy overhead can be reduced by decreasing the number of flip-flops to one-third. However, not only reliability will deteriorate, but the range of adaptive timing borrowing will degrade. Therefore, this is again a tradeoff between reliability and energy consumption.

Table 3 lists the summaries of the SCG coding scheme and self-calibrated voltage scaling technique, including area overhead in a router, energy overhead and energy reduction in channels. The wire length is also set as 1800 μm. The energy reduction of the self-calibrated voltage scaling technique is due to the low swing of link wires. The total area overhead is about 14.4% related to a router, which is using X-Y routing and round-robin arbitration. The router architecture is set as 5 input/output ports with 4-stage pipeline. And the FIFO depth of each output buffer is 8 flits. The size of each flit size is 32 bits. The area breakdown of adaptive double sampling data checking, MAF-based test generator and voltage control unit in the self-calibrated voltage scaling are 71%, 8%, and 21%, respectively.

Table 3: Summaries of SCG coding and self-calibrated voltage scaling.

(length = 1800 μm, low voltage)	Area overhead in a router	Energy overhead	Energy reduction (channels)
SCG Coding	1.21%	0.26%	14.1%
Self-calibrated voltage scaling	13.2%	6.62%	21.1%

7. Conclusion

The physical effects of crosstalk and PVT variations in nanoscale technologies degrade the performance of on-chip interconnection networks (OCINs). This work uses a combination of a *self-calibrated voltage scaling technique* and a *self-corrected green (SCG) coding scheme* to overcome increasing variations and achieve energy-efficient on-chip data communication. The SCG coding scheme is used to construct reliable and energy-efficient channels. The SCG coding scheme has two stages, the triplication error correction coding stage, and the green bus coding stage. Triplication error correction coding is a reliable mechanism that achieves rapid correction ability to reduce the physical transfer unit (phit) size in routers via self-correction at the bit level. Green bus coding reduces energy reduction significantly via a joint triplication bus power model that eliminates crosstalk effects. The self-calibrated voltage scaling technique is designed with the SCG coding scheme. The self-calibrated voltage scaling technique adjusts the voltage swing of link wires via two error detection stages, the crosstalk-aware test error detection stage and run-time error detection stage. Furthermore, the self-calibrated voltage scaling technique is tolerant to timing variations of channels. Based on UMC 65 nm CMOS technology, the proposed self-calibrated energy-efficient and reliable channels reduce energy consumption by nearly 28.3% compared with that of uncoded channels at the lowest voltage.

Acknowledgments

This paper was supported by the National Science Council, Taiwan, under project NSC 98-2220-E-009-026, NSC 98-2220-E-009-027.

References

[1] (2005–2009) International Technology Roadmap for Semiconductors. Semiconductor Industry Assoc., http://public.itrs.net/.

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

[3] R. I. Bahar, D. Hammerstrom, J. Harlow et al., "Architectures for silicon nanoelectronics and beyond," *Computer*, vol. 40, no. 1, pp. 25–33, 2007.

[4] D. Zydek, N. Shlayan, E. Regentova, and H. Selvaraj, "Review of packet switching technologies for future NoC," in *Proceedings of the 19th International Conference on Systems Engineering (ICSEng '08)*, pp. 306–311, August 2008.

[5] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, 2005.

[6] L. Benini and G. de Micheli, *Network on Chips: Technology and Tools*, Morgan Kaufmann, 2006.

[7] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2004.

[8] V. Raghunathan, M. B. Srivastava, and R. K. Gupta, "A survey of techniques for energy efficient on-chip communication," in *Proceedings of the 40th IEEE/ACM Design Automation Conference (DAC '03)*, pp. 900–905, June 2003.

[9] R. Marculescu, U. Y. Ogras, L. S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3–21, 2009.

[10] H. Lekatsas and J. Henkel, "ETAM++: extended transition activity measure for low power address bus designs," in *Proceedings of the VLSI Design Conference*, pp. 113–120, 2002.

[11] K. H. Baek, K. W. Kim, and S. M. Kang, "A low energy encoding technique for reduction of coupling effects in SoC interconnects," in *Proceedings of the 43rd Midwest Circuits and Systems Conference (MWSCAS '00)*, vol. 1, pp. 80–83, August 2000.

[12] C.-G. Lyuh and T.-W Kim, "Low-power bus encoding with crosstalk delay elimination," in *Proceedings of the International ASIC/ SoC Conference*, pp. 389–393, 2002.

[13] T. Lv, J. Henkel, H. Lekatsas, and W. Wolf, "An adaptive dictionary encoding scheme for SOC data buses," in *Proceedings of the Design, Automation, and Test in Europ Conference Exhibition (DATE '02)*, pp. 1059–1064, 2002.

[14] K. Lee, S. J. Lee, and H. J. Yoo, "Low-power network-on-chip for high-performance SoC design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 2, pp. 148–160, 2006.

[15] J. Yang and R. Gupta, "FV encoding for low-power data I/O," in *Proceedings of the International Symposium on Low Electronics and Design (ISLPED '01)*, pp. 84–87, August 2001.

[16] R. B. Lin, "Inter-wire coupling reduction analysis of bus-invert coding," *IEEE Transactions on Circuits and Systems I*, vol. 55, no. 7, pp. 1911–1920, 2008.

[17] C. S. D'Alessandro, D. Shang, A. Bystrov, A. V. Yakovlev, and O. Maevsky, "Phase-encoding for on-chip signalling," *IEEE Transactions on Circuits and Systems I*, vol. 55, no. 2, pp. 535–545, 2008.

[18] G. Chen, S. Duvall, and S. Nooshabadi, "Analysis and design of memoryless interconnect encoding scheme," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '09)*, pp. 2990–2993, May 2009.

[19] B. Fu and P. Ampadu, "On hamming product codes with type-II hybrid ARQ for on-chip interconnects," *IEEE Transactions on Circuits and Systems I*, vol. 56, no. 9, pp. 2042–2054, 2009.

[20] 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. 8, pp. 655–667, 2005.

[21] S. R. Sridhara and N. R. Shanbhag, "Coding for reliable on-chip buses: a class of fundamental bounds and practical codes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 5, pp. 977–982, 2007.

[22] K. N. Patel and I. L. Markov, "Error-correction and crosstalk avoidance in DSM busses," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 10, pp. 1076–1080, 2004.

[23] P. P. Pande, A. Ganguly, B. Feero, B. Belzer, and C. Grecu, "Design of low power & reliable networks on chip through joint crosstalk avoidance and forward error correction coding," in *Proceedings of the 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 466–476, October 2006.

[24] A. Ganguly, P. P. Pande, and B. Belzer, "Crosstalk-aware channel coding schemes for energy efficient and reliable NOC interconnects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 11, Article ID 4801555, pp. 1626–1639, 2009.

[25] S. R. Sridhara and N. R. Shanbhag, "Coding for reliable on-chip buses: fundamental limits and practical codes," in *Proceedings of the 18th International Conference on VLSI Design: Power Aware Design of VLSI Systems*, pp. 417–422, January 2005.

[26] F. Worm, P. Ienne, P. Thiran, and G. de Micheli, "A robust self-calibrating transmission scheme for on-chip networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 1, pp. 126–139, 2005.

[27] F. Worm, P. Thiran, G. D. Micheli, and P. Ienne, "Self-calibrating networks-on-chip," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, vol. 3, pp. 2361–2364, May 2005.

[28] M. Simone, M. Lajolo, and D. Bertozzi, "Variation tolerant NoC design by means of self-calibrating links," in *Proceedings of the Design, Automation and Test in Europe Conference Exhibition (DATE '08)*, pp. 1402–1407, March 2008.

[29] R. Ho, *On-chip wires: scaling and efficiency*, Ph.D. dissertation, Stanford University, 2003.

[30] R. Ho, K. Mai, and M. Horowitz, "Efficient on-chip global Interconnects," in *Proceedings of the IEEE Symposium on VLSI Circuits*, pp. 271–274, June 2003.

[31] P. P. Sotiriadis and A. P. Chandrakasan, "A bus energy model for deep submicron technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, no. 3, pp. 341–350, 2002.

[32] K. W. Kim, K. H. Baek, N. Shanbhag, C. L. Liu, and S. M. Kang, "Coupling-driven signal encoding scheme for low-power interface design," in *Proceedings of the IEEE/ACM International*

Conference on Computer Aided Design (ICCAD '00), pp. 318–321, 2000.

[33] P. P. Sotiriadis, *Interconnect modeling and optimization in deep submicron technologies*, Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, Mass, USA, 2002.

[34] R. Pendurkar, A. Chatterjee, and Y. Zorian, "Switching activity generation with automated BIST synthesis for performance testing of interconnects," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 9, pp. 1143–1158, 2001.

[35] K. Sekar and S. Dey, "LI-BIST: a low-cost self-test scheme for SoC logic cores and interconnects," in *Proceedings of the IEEE VLSI Test Symposium*, pp. 417–422, 2002.

[36] X. Bai, S. Dey, and J. Rajski, "Self-test methodology for at-speed test of crosstalk in chip interconnects," in *Proceedings of the 37th IEEE/ACM Design Automation Conference (DAC '00)*, pp. 619–624, June 2000.

[37] R. Tamhankar, S. Murali, S. Stergiou et al., "Timing-error-tolerant network-on-chip design methodology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 7, Article ID 4237244, pp. 1297–1310, 2007.

[38] Y. Zhao, S. Dey, and L. Chen, "Double sampling data checking technique: an online testing solution for multisource noise-induced errors on on-chip interconnects and buses," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 7, pp. 746–755, 2004.

[39] D. N. Truong, W. H. Cheng, T. Mohsenin et al., "A 167-processor computational platform in 65 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, Article ID 4804961, pp. 1–15, 2009.

[40] S. R. Vangal, J. Howard, G. Ruhl et al., "An 80-tile sub-100-W teraFLOPS processor in 65-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, 2008.

[41] M. A. Anders, H. Kaul, S. K. Hsu et al., "A 4.1Tb/s bisection-bandwidth 560Gb/s/W streaming circuit-switched 8x8 mesh network-on-chip in 45nm CMOS," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC '10)*, pp. 110–112, February 2010.