Research Article

A Minimal Buffer Router with Level Encoded Dual Rail-Based Design of Network-on-Chip Architecture

Trupti Patil,1 Anuradha Sandi,1 D. M. Deepak Raj,2 S. Chandragandhi3, and Dawit Mamiru Teressa4

1Department of Electronics and Communication Engineering, Gurunank Dev Engineering College, 585403, Bidar, India
2Department of Computing, Vel Tech, Rangarajan Dr. Saganthala, R&D Institute of Science and Technology, India
3AP/CSE, Karpagam Institute of Technology, Coimbatore, India
4Department of Chemical Engineering, College of Biological and Chemical Engineering, Addis Ababa Science and Technology University, Addis Ababa, Ethiopia

Correspondence should be addressed to S. Chandragandhi; chandragandhi09@gmail.com and Dawit Mamiru Teressa; dawit.mamiru@aastustudent.edu.et

Received 20 April 2022; Revised 19 June 2022; Accepted 23 June 2022; Published 13 July 2022

Academic Editor: Kalidoss Rajakani

Copyright © 2022 Trupti Patil et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Asynchronous NOCs are most prominent in present SOC designs, due to their low dynamic power consumption, modularity, heterogeneous nature, and robustness to the process variations. Though asynchronous designs are proved efficient over synchronous counterparts, they have some severe drawbacks when area and speed are considered, due to complex handshake control circuits which increase the static power loss. Quasidelay insensitive (QDI) class of asynchronous NOCs based on 2-phase encoding is proved beneficial for speed and throughput enhancement but with complex design. The work has introduced lightweight minimal buffer router based on LEDR encoding to design a low power, high speed with compact NOC architecture. Then, minimal buffer router with FSM-based arbiter and priority assigner block is designed to enhance the speed, power, and area. This proposed work achieves zero dynamic power consumption with a total power consumption of less than 0.082 W with a router latency of 0.8 ns.

1. Introduction

In recent times, shrinking design geometry makes the synchronous design more prone to process variations which corrupts the basic timing requirements of synchronous design. Many attempts have been made to solve clock-related issues of synchronous NOC, most of them are based on clock domain crossing circuit, for example, GALS, in this chip as a whole is made asynchronous (to avoid global clock distribution) and all the cores operates on their clock (synchronous). Most GALS architectures are based on asynchronous wrapper circuits are projected as a solution to improve throughputs and reduce area and power consumption but with extra hardware expenses for wrapper circuits at each router node. In recent times, asynchronous NOCs are projected as a possible solution to the issue of traditional synchronous NOCs. Unlike synchronous NOCs in event-driven asynchronous signal switches only when the request is processed and unused circuits are left obstructed. These characteristics of asynchronous design helps in achieving low dynamic power consumption and near-zero standby power. Unlike in GALS design, the isolated timing domains do not need complex wrapper circuits as they work on simple handshake interfaces, which aids in improving modularity and reusability of design. According to the study, the clock is considered a major source of noise and EMI resulting in delay variation. The absence of such a global clock in asynchronous NOCs reduces possibilities of EMI. The quasidelay insensitive (QDI) class of asynchronous circuits...
provides timing robustness due to the liberal constraints at gate and wire level, making it an attractive option for large-scale multicore systems [1].

(1) Though asynchronous designs proved efficient over synchronous counterparts in terms of scalability, power consumption, and better timing constraints. But they have some severe drawbacks when area and speed are considered due to complex handshake control circuits which increase the static power loss [2]. In this study, we offer a proposal for a high-speed, low-power small NOC architecture based on LEDR encoding and a lightweight router

(2) The study is organized as follows: in Section 2, an efficient NOC architecture based on two-phase LEDR encoding is proposed, with an asynchronous router. In Section 3, the findings are briefly discussed. Finally, in Section 4, there are some closing observations.

2. Proposed Design

In our design methodology we are using a concept of micronetworks based on ISO/OSI model. We have divided the task to different layers of the model. Such layered design enables the design of reconfigurable NOCs. Physical layer, data link layer, network layer, and transport layer are all part of a typical NOC architecture. We are using the software-hardware codesign paradigm in our research. We are focusing three important layers in software design asynchronous NOC architecture: data link layer, network layer, and transport layer.

2.1. Data Link Layer. Error detection, framing of the data stream, and physical addressing are the major functions of the data link layer. In our design these actions are performed by two blocks: (i) PLD (platform level design) and (ii) DSM (data sorting module), as shown in Figure 1. The PLD block identifies the packet and removes the header. We are aiming to make our design compatible to all types of packet. The packet classifier in the PLD block classifies the incoming packet into different categories like LEDR, TCP, IP, Ethernet, and Jumbo packets. Our design is compatible with the Ethernet network. Once the packet is classified, the PLD block segregates the data and sends it to the DSM block, where all the headers are removed and only data packet with the source and destination address is sent to the network layer. Another main function of the data link layer is data encoding and decoding is performed in this layer. Most asynchronous circuits either uses bundled data scheme or dual rail protocols; in the first method, one single bit is encoded with single wire gives high throughput but problem is Knowlagent to ensure the validity of data is sent after worst case delay. In the second method, one bit of data is encoded using two wires and spacer bits are transmitted between two consecutive data packet and with availability of completion detector data validity is indicated as soon as the computation is done, but throughput will be low due large number of communication steps [3].

2.2. Network Layer. This layer is mainly responsible to accomplish the routing of data from source to destination via routers and links. In our design, we propose a 2-dimensional 4×4 Mesh NOC architecture as shown in Figure 2. Logical mesh topology is designed in this layer. The nodes are arranged in a column-wise direction. There are 16 router nodes, namely, N1 to N16, which facilitates communication between 16 processor cores [4]. The router in our design is a five-port router with four directional and one local port. Four directional ports provide connection in all 4 directions; one is for providing connection to the local processor. In order to reduce the area overhead of buffers in the router level, we are using a concept of minimalistic buffering method; here, we are using one 48-bit register at each port which can store one packet at each port unlike buffer less routing algorithms where packets are reflected in the network reduces the network latency which increases the latency of our network without compromising with the area of network [5]. In our router, I/O ports support five simultaneous connections. Each router consists a centralized block that handles both routing and arbitration. The internal architecture of the router is shown in Figure 2. It consists of a programmable priority encoder, priority-based scheduler (arbiter), and routing controller. We have used a single 48-bit register at each port to hold the data packet instead of IO buffers. There are two main contributions of our design; the first one is arbiter design. Generally, three types of
techniques are used for assigning the access to a request made: (i) based on priority and is served accordingly in this sometimes the modules on lower priority hardly get a chance to serve; (ii) round-robin based where each module gets a chance equally. However, it induces a lot of delays which affects the throughput. (iii) Arbiter based on random allocation, which sometimes ends up in loss of data or some modules not been serviced due to nature of arbiter design. Our method is developed in such a way that it includes the advantage of all the mentioned techniques. We have not given any fixed priority to any module or node though there is a concept of priority in our design. We would say this is a priority based on the service provided to the module/node and is used only in the condition when there are two or more requests at the same time that need to be served [6].

Our design is developed with an arbiter that has rotating priority; priority allocation is discussed in detail later in this section. Our second novelty is the design of LEDR encoder/decoder block, asynchronous NOCs based on 2-phase LEDR provides better performance compared to 4-phase encoding in terms of power and throughput, but 2-phase signal handling needs complex design [7]. In this paper, we propose a simple yet efficient encoder and decoder design for level encoded dual rail encoding-decoding the hardware requirements timing constraints are discussed in Result Analysis.

Once the packet is processed at the link layer, a 48-bit packet with source and destination address enters into the network through corresponding router. In the priority encoder block assigns priorities to all incoming packets. The highest priority is assigned to the packet from the port connected to the local processor. For all the directional ports, priorities are assigned in cyclic order in a clockwise direction. Priority encoder’s select signal decides which packet to be transferred to the output port. When arbiter receives a request for port allocation, it assigns the corresponding output ports to the requested input port. During normal conditions, there will not be any contention issue because the number of input ports is equal to the number of output ports. Hence, the router can support five transactions simultaneously. But the issue arises when more than

![Figure 2: 4 × 4 mesh architecture with asynchronous NOC router architecture.](image)

![Figure 3: Arbiter FSM design.](image)

<table>
<thead>
<tr>
<th>Present state/present request</th>
<th>Comp_0</th>
<th>Comp_1</th>
<th>Comp_2</th>
<th>Next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req_0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Gnt0</td>
</tr>
<tr>
<td>Gnt0</td>
<td>Req_1</td>
<td>X</td>
<td>1</td>
<td>Gnt1</td>
</tr>
<tr>
<td>Req_2</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>Gnt2</td>
</tr>
<tr>
<td>Req_1</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>Gnt1</td>
</tr>
<tr>
<td>Gnt1</td>
<td>Req_2</td>
<td>X</td>
<td>X</td>
<td>Gnt2</td>
</tr>
<tr>
<td>Req_0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Gnt0</td>
</tr>
<tr>
<td>Gnt0</td>
<td>Req_2</td>
<td>X</td>
<td>X</td>
<td>Gnt2</td>
</tr>
<tr>
<td>Req_1</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>Gnt1</td>
</tr>
</tbody>
</table>

Table 1: Priority update table.

![Platform Level Design (PLD)](image)
one port requests for the same output port. When the arbiter receives more than one request for the same output port, it processes the request of the packet with the highest priority. Arbiter in our design maintains internal registers to keep track of the served port request. To eliminate the starvation issues. The arbiter masks the request from the last served port during the next round of arbitration. The FSM architecture is shown in Figure 3; it has three request signals (req 0, req 1, and req 2) and three grants (Gnt0, Gnt1, and Gnt2). There are four states in FSM, each of which serves as the current and next state. For states, the FSM employs a binary encoding technique. States are set to IDLE = 00, Gnt0 = 01, Gnt1 = 10, and Gnt2 = 11 in our design. The internal registers comp 0, comp 1, and comp 2 are utilized for req 0, req 1, and req 2, respectively. These registries keep track of the requests that have been fulfilled. The arbiter FSM architecture first finds the next state, then the current state, and finally the output state, as explained in the following phases.

2.3. Transport Layer. This layer is responsible for intact and in-order delivery of the packet, ensuring both error and flow control source to destination level packet routing controller is designed at this layer. Once the packet is assigned with the requested port, the routing controller decides the path through which packet travels in the network in Table 3. In our design, we are using an XY routing algorithm. It is a distributed deterministic algorithm based on XY dimension-order routing. According to this algorithm, the packet first travels in the X direction and then in the Y direction till the destination address is matched. The routing controller block is implemented in a simple yet efficient way it consists of two comparators (X address comparator block and Y address comparator block). When the packet is injected into the network through any of the processor cores, first the destination address fields are checked by the comparators. First,
the comparator block checks \( X \) address, and the address is compared with the current address if it matches, i.e., if the packet is destined for the same row, then the \( Y \) address will be compared if not then the packet is directed to the next row either to the previous row are the next row without wasting processor time in pushing the packet in the same row. On the other hand, if the packet is destined for the same row, then the second comparator checks \( Y \) address with current address if it matches then the packet is destined for that node if not then the packet will transmit to next column till the packet reaches to destination node. Once the packet is reaches the destination node, it is decoded by the LEDR decoder block and the output appears at data out.

### 3. Result Analysis

The proposed asynchronous NOC router architecture is modelled in Verilog HDL on the xc7a100t device using Xilinx ISE 14.7 and the Virtex 7 family, utilizing the 3csg324 package. The ISE simulator (iSIM) was used to simulate the system, and the synthesis was done with the Xilinx synthesis tool (XST). Figure 4 depicts the NOC’s internal architecture.

The main feature of our router is LEDR encoding; it is implemented at the data link layer. The LEDR encoder and decoder designs play an important role. Figure 4(a) shows the top view of encoder Figure 4(b). When source PE initiates the data transfer, it sends the 32-bit data to the nearest router node. The encoder block encodes the data into 64 bit each bit value encoded into two-bit, one bit holds the data value, and another bit holds phase information (\( \text{even} = 0 \) and \( \text{odd} = 1 \)). When the data packet reached the destination node, it is sent to the LEDR decoder block where 64-bit data is decoded to extract the 32-bit data value.

Table 2 shows the device utilization summary of the LEDR encoder and decoder block. The design is efficient in terms of resource utilization; we can see that the design is using only 16 of 63400 LUTs blocks which compose nearly 0% of the total available resource whereas decoder block is using 63 of 63400 available blocks leading to 0% utilization.

Figure 5 shows internal architecture of the proposed NOC architecture; it consists of PLD (platform level design) which is responsible for packet classification’s (data sorting module) of here removes all headers and keeps only source and destination address and data. This packet is sent to NOC network. Table 3 of the proposed designs demonstrates that the entire NOC architecture is resource efficient, requiring only 3% of registers and 4% of total LUTs available.

Table 4 compares the power consumption of various NOC architectures. The power analysis of proposed the design; the power consumption at clock and logic level is zero; there is no power consumption at clock which efficiently solves the high-power consumption issue of synchronous NOCs. Our design achieved a target zero dynamic power consumption with leakage power of 0.082 W making total power consumption equals 0.082 W. The power consumption of our design is compared with the mesh NOC architectures. Comparison of power with the results is presented in paper [8] with adaptive and deterministic 2D mesh NOC with power consumption.

Table 5 shows a detailed comparison of asynchronous router designs, and the proposed router provides the least latency compared to other standard asynchronous router designs.

### 4. Conclusion

In this paper, we present an asynchronous NOC architecture with a simple buffer router based on LEDR encoding in this work. Among the available asynchronous designs, ours has the shortest latency (0.8 ns). In terms of resource use, the design is efficient, consuming only 5% of total resources. Experimental results show that our design attained zero
dynamic power consumption and total power consumption of 0.082 W (leakage) which is less compared to other mesh architectures. It gives 46–72% improvement when compared to NR (nearest neighbor) adaptive and deterministic mesh architectures. We have also compared the implemented design with standard asynchronous NOC designs for latency.

Data Availability
The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Additional Points
Future Enhancement. This work will be improved and the future design will be analyzed for area development and throughput.

Conflicts of Interest
The authors declare that they have no conflicts of interest.

References