ONoC-SPL: Customized Network-on-Chip (NoC) Architecture and Prototyping for Data-intensive Computation Applications

Akram Ben Ahmed, Kenichi Mori, Abderazek Ben Abdallah
The University of Aizu
Graduate School of Computers Science and Engineering
Aizu-Wakamatsu 965-8580, Japan
E-mail: d8141104@u-aizu.ac.jp

Abstract—Network-on-Chip (NoC) has emerged as a promising paradigm to largely alleviate the limitations exhibited by the shared-bus based systems in current System-On-Chip (SoC). These problems include the lack of scalability, clock skew, lack of support for concurrent communication, and increasing power consumption. Based upon a packet/filt switching scheme, NoC allows a concurrent transmission of data providing a higher bandwidth and performance.

In this paper, we present an optimized version of our earlier design OASIS-2, called ONoC-SPL, that employs a Short-Pass-Link (SPL) customization which aims to reduce the communication latency for performance enhancement in data-intensive computation applications. In order to evaluate its performance and hardware complexity accurately, we prototyped ONoC-SPL on FPGA using two traffic patterns (Dimension-reversal and Hotspot) and also a real application (JPEG encoder).

Evaluation results showed that ONoC-SPL reduces the execution time by 30.1% and enhances the throughput by 32.3% when compared with the earlier design. This performance improvement came with only an area utilization cost under the 5% and a slight 0.49% power overhead.

Keywords—NoC; Parallel; Customized; Architecture; Prototyping; Look-ahead routing;

I. Introduction

Network-On-Chip (NoC) [1], [2] offers a good way of realizing interconnections on silicon and significantly solves the problems of bus-based solutions. This architecture also provides extremely high bandwidth by distributing the propagation delay across multiple switches, effectively pipelining the packet transmission. Moreover, transmission can be conducted in parallel if the network provides more than one transmission channel between a sender and a receiver. Thus, unlike bus-based Systems-on-Chip, NoC presents a scalable architectural platform with a huge potential to handle the increasing complexity that can easily provide reconfigurability [3].

One of the important design choices for NoC is the topology. NoC's topology can be classified into two categories: a regular one [4], such as Grid-Topology (Mesh), Torus, Star, and Clos. These architectures are not concerned for dedicated applications where each processing element (PE) is connected uniformly without consideration of the network load balance or hotspots [5]. This may cause packet dropping or stalling, then significantly degrading the overall performance. The second category is custom topologies [6], whose router cannot be reused and the routing algorithm is difficult to implement [4], [6]. In this kind of topologies, the design time is longer than the regular ones. This significant increased design complexity causes an unacceptable simulation time compared with the traditional simulation methods [7], [8]. Thus, there is a design time and performance trade-off.

Some works focused on topology's optimization techniques. For instance, Murali et al [9] introduced a tool named SUNMAP. This tool automatically selects the best topology for a given application and produces the mapping of cores onto that selected topology. The evaluation is executed with Traffic Generator (TG), and xpipes compiler [10] is used as a hardware library. The evaluation is executed with Traffic Generator (TG), and xpipes compiler [10] is used as a hardware library. Fully customized topologies are shown in [11] where the authors implemented some proposed algorithms in a C++ package. These topology optimizations are mostly evaluated in software level.

Another problem that can rise when designing a NoC is how to verify the system correctness and integrity. The NoC architecture research includes a lot of trade-offs between topology, routing, flow control, buffer size, packet size, and other optimization techniques. It is difficult to analyze these trade-offs using only high-level simulations. Therefore, NoC prototyping is an essential design phase for evaluating the performance of NoC architecture under real applications [12].

Genko et al [7] presented a flexible HW/SW emulation environment implemented on FPGA. However, they used TG and did not use real applications. An implementation on FPGA and detailed explanation about how to implement NoC architecture with multi cores are proposed in [8]. But its execution time measurement was in terms of assembly language instructions execution. These two works mentioned.
how to deeply implement NoC on FPGA but no parameter trade-offs are presented. To the best of our knowledge, none of the previous works proposed architecture and real design of a NoC based on our proposed customized topology and Look-ahead routing scheme.

Previously, a Network-on-Chip called OASIS-2 [13] has been designed by our group. It is a 4x4 mesh topology system, using Wormhole-like switching and Stall-Go flow control. However, OASIS-2 suffers from high communication latency due to the increasing number of hops. In this paper, we propose an architecture and design of an optimized version of OASIS-2 called ONoC-SPL. ONoC-SPL aims to reduce the communication latency for long range and high frequency communications (source, destination) pairs. We prototyped ONoC-SPL on FPGA and evaluated its performance and hardware complexity using two traffic patterns (Dimension-reversal and Hotspot) and also a real application (JPEG encoder). The obtained results are compared with the original OASIS-2 architecture.

II. OASIS-2 System Architecture Overview

OASIS-2 is a Mesh topology system based upon our earlier OASIS-NoC design [2], [14]. It adopts Wormhole-like switching [15]. The forwarding method, which is chosen in a given instance, depends on the level of the packet fragmentation. For instance, when the buffer size is greater than or equal to the number of flits, Wormhole is used. But, when the buffer size is less than the number of flits, Virtual-Cut-Through switching is employed. In this way, packet forwarding can be executed in an efficient way while guaranteeing a small buffer size.

A. Router architecture

Figure 1 illustrates OASIS-2’s router block diagram and its three pipeline stages. The left five modules are the input modules where we can find the input buffers and the routing circuit. In the middle, the switch_allocator module including the scheduler and flow control circuits. Finally, the crossbar module handles the transfer of flits to the next neighboring node.

The input_module contains two main components: Input-buffer and Route modules. Incoming flits from different neighboring routers, or from the connected computation tile, are first stored in the Input-buffer and waiting to be processed. This step is considered as the first pipeline stage of the flit’s life-cycle (Buffer Writing). Each input-buffer can host up to four flits. Figure 2 demonstrates the OASIS-2 flit format. First, three bits are used to store destination information of each xdest and ydest. The next five bits are dedicated to indicate the Next-Port identifier, as it will be explained later. The following 64 bits are dedicated to store the payload. Finally, the last bit indicates the tail informing the end of the packet. In addition, the architecture does not provide a separate head flit and every flit therefore identifies its destination X and Y addresses and carries an additional single bit to indicate whether it is a tail flit or not.

<table>
<thead>
<tr>
<th>Tail</th>
<th>Payload</th>
<th>Next_Port</th>
<th>X-dest</th>
<th>Y-dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Bit</td>
<td>64 Bit</td>
<td>3 bit</td>
<td>3 bit</td>
<td>3 bit</td>
</tr>
</tbody>
</table>

Figure 2. OASIS-2 flit format.
After being stored, the flit is read form the FIFO buffer and advances to the next pipeline stage. The $x_{dest}$ and $y_{dest}$ fields are decoded in order to extract the information about the destination address in addition to the Next-Port information embedded in the flit. These values are then sent to the Route circuit where the routing algorithm Look-ahead-XY (LA-XY) is executed to determine the New-next-Port direction for the next downstream node. The New-next-Port direction is calculated by comparing the flits' destination address with the next router's address.

LA-XY aims to minimize the flit's latency in the router. Using the Next-Port identifier pre-calculated in the previous upstream node and embedded in each flit, LA-XY computes first the address of the next neighboring node on the flit path. After knowing the next node’s address, LA-XY calculates its output-port direction (New-next-Port). The resulted identifier is then re-embedded in the flit and later used by the next downstream node for arbitration. By doing so, LA-XY allows both Routing Calculation and Switch Arbitration stages to be performed in parallel. This reduces the flit delay in the router, eventually reducing the system overall communication latency while enhancing its throughput.

![Figure 3. Stall-Go block diagram.](image)

While the routing module calculates the New-next-Port direction for the next downstream node, the input_module uses at the same time the Next-Port identifier read from the flit to generate the request for the switchAllocator to use the selected output-port via sw_req and port_req signals as shown in Fig.1. These two signals are transmitted to the switch Allocator module to perform the arbitration between the different requests. When more than two flits from different input-ports are requesting the same output-port at the same time, the switchAllocator manages to decide which output-port should be granted to which input-port, and when this grant should be allocated. This process is done in parallel with the routing computation done in the Input-port to form the second pipeline stage (Routing Calculation / Switch Arbitration).

At the end, the switchAllocator sends a sw_ctrl signal that contains all the information needed by the crossbar circuit about the scheduling result. This latter forming the last pipeline stage (Crossbar Traversal) fetches the correspondent flit from the granted input_module and sends it to its allocated output-channel. The switchAllocator module is composed of the Matrix-allocator and the Stall-Go flow control circuits.

**Stall-Go flow control:** As a flow control, OASIS-2 employs Stall-Go instead of the previous OASIS retransmission flow control (RFC) [14]. Figure 3 illustrates Stall-Go block diagram. The used flow control technique efficiently prevents buffer from overflowing [16]. Data transfer is controlled by signals indicating the buffer condition. In the absence of Stall-Go function [14], receiver cores need to judge whether dropped packets occurred or not. If they did, the transmitter must resend the dropped packets using a receiving request signal from master cores. Stall-Go scheme might incur a communication blocking, but at the same time it can considerably reduce the latency.

**Matrix-allocator scheduler:** OASIS-2 employs Matrix-allocator as a least recently served priority scheme [3] via the packet transmission layer. Thus, it can treat each communication as a partially fixed transmission latency [17]. In order to adopt the Matrix-allocator scheduling for OASIS-2, we implemented a 4x4 scheduling-matrix for each output-port. The scheduling module accepts all the requests from the different connected input-ports and their desired output-ports. Then it assigns a priority for each request. The scheduling module verifies the scheduling-matrix, compares the priorities of the different requests, and gives the grant to the one possessing the highest priority in the matrix. The scheduling module should update the matrix with the new priorities of each request to be sure that every input-port will be served and get the grant to use the output-port in a fair way.

**B. Short-Pass-Link (SPL) Approach**

In OASIS-2 design, as in Mesh-based NoC systems, there is a possibility where some transactions may need larger latency or many hops that a flit should traverse to reach its destination. This may become a bottleneck. To solve this problem, we propose an Optimized OASIS-NoC (OnoC-SPL) system with SPL customization to decrease the latency caused by the big number of hops between some specific nodes. OASIS-2 router has 5 direction's in-out ports (Local, North, East, South and West). But, in order to realize the SPL, an additional sixth port is needed for the Extra-port. However, the additional port increases the area occupation and power consumption. Therefore, the SPL is carefully inserted only between specific communications which include the possibility to become a bottleneck among the whole communication patterns. If many SPLs are inserted without a special care, it may not only consume larger area, but a power overhead might be also observed.

The SPL insertion algorithm shown in Fig.4 selects the communications which need the SPL. First, we should decide the available SPL resource budget so we can prevent any unacceptable increasing area utilization or power consumption, which are the drawbacks of SPL addition. Next,
the communication cost for all communications patterns with their communication frequencies and distances. Depending on these calculations, the SPL is inserted to the highest communication cost. Simulation results compare the updated design with the initial one without SPL. After adding the SPL, the algorithm goes back around all remaining communication cost calculations again until the available SPL budget is exhausted.

\[ f_{ij} = \max \left( \sum_p \sum_{pq} V_{pq} \right) \]

\[ d_M(i,j) = |i_x - j_x| + |i_y - j_y| \]

\[ C_{ij} = f_{ij} \times d_M(i,j) \]

Formula 1 is used to measure the communication frequency by calculating the total system communication between all neighbor nodes transaction volume and target communication neighbor nodes usability volume. This means that \( p \) is always a neighbor of \( q \). The total communication volume is expressed by \( \sum_p \sum_{pq} V_{pq} \); where \( p \) indicates the sender node and \( q \) indicates the receiver node. Then, the target communication frequency is expressed by \( V_{ij} \); where \( i \) indicates the sender node, and \( j \) indicates the receiver node. To calculate the communication distance, Manhattan distance is employed (formula 2). The address of \( i \) and \( j \) is expressed by \( (i_x, i_y) \) and \( (j_x, j_y) \). Finally, the calculation cost is computed according to formula 3 and by using the values obtained from Formula 1 and Formula 2.

The main ONoC-SPL system modifications are: an in-out port addition to the router, the modification of the network’s connections design and the flit structure. To configure an SPL commutation, it is essential to add an Extra-port for the routers hosting the SPL. However, other routers should keep having the same number of 5 ports to prevent from any unnecessary increasing area utilization caused by the Extra-port addition.

Another necessary modification is the flit structure. OASIS-2 has 5 bits dedicated to define the next port direction used by LA-XY routing. So, it is necessary to extend the next port field from 5 bits to 6 bits for the SPL, as an additional direction. Consequently, the whole flit’s width also needs to be extended by one bit. The final modification is the network connections, between two routers in Mesh topology which are connected by SPL.

III. Evaluation

A. Evaluation methodology

ONoC-SPL system was designed in Verilog HDL, synthesized and prototyped on commercial CAD tools and FPGA board respectively [18]. We evaluate the hardware complexity of the proposed design in terms of area utilization, power consumption and speed. As we previously stated, before starting the SPL insertion we should first determine our SPL budget. This budget generally depends on the application requirements or the system constraints. In our case, we set the SPL budget to 5% of extra hardware obtained with the SPL insertion when compared with the original OASIS-2 design’s hardware complexity. This choice has been taken due to the used FPGA’s area capacity constraints.

To evaluate the performance of the proposed system, three target applications have been selected: Dimension-reversal [19], Hotspot [20] and JPEG encoder [21]. Using the three benchmarks mentioned above, we evaluated the execution time and throughput. All the results obtained with ONoC-SPL are compared with the early OASIS-2 system.

Table I presents the configuration parameters used for the synthesis of ONoC-SPL and OASIS-2 designs. When executing the SPL insertion algorithm previously explained in Fig. 4, Dimension-reversal and Hotspot budgets are exhausted at two SPLs. On the other hand, JPEG encoder can
insert three SPLs. In addition, each node which needs an SPL is located at the edge of the network. This means that unused ports can be exploited for the SPL insertion.

B. Evaluation results

1) System complexity evaluation: Table II illustrates the hardware complexity results and their variation with each additional SPL insertion (SPL1, SPL2 and SPL3). When inserting two SPLs, the area utilization of the Dimension reversal has increased by 2.93% and by 2.60% with Hotspot. It also has risen with JPEG encoder by 4.28% when inserting three SPLs. This area utilization respects our SPL insertion budget’s restriction (5% of extra hardware). As we previously mentioned, the increasing area utilization is caused by the additional Extra-port added to the router in order to host the SPL insertion.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>OASIS-2</th>
<th>ONoC-SPL1</th>
<th>ONoC-SPL2</th>
<th>ONoC-SPL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR Area (ALUTs)</td>
<td>9,008 (8%)</td>
<td>9,119 (8%)</td>
<td>9,212 (8%)</td>
<td>9,650 (10%)</td>
</tr>
<tr>
<td>Speed (MHz)</td>
<td>182.78</td>
<td>169.98</td>
<td>171.53</td>
<td>167.28</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>660.61</td>
<td>663.12</td>
<td>662.54</td>
<td>661.46</td>
</tr>
<tr>
<td>Extra hardware</td>
<td>0</td>
<td>1.22%</td>
<td>2.93%</td>
<td>6.63%</td>
</tr>
</tbody>
</table>

When compared to OASIS-2 in terms of power consumption, we observed in ONoC-SPL a slight increasing with 1.93 mW (0.29%), 1.32 mW (0.20%), and 8.02 mW (0.98%) for Dimension-reversal, Hotspot and JPEG encoder respectively. As the results show, ONoC-SPL exhibits a very slight power consumption overhead which was under the 1% for each one of the three applications. The power consumption includes dynamic power, static power, and I/O power.

2) Performance analysis evaluation: Figure 5 (a) represents the comparison results between the base architecture OASIS-2 and the proposed ONoC-SPL in terms of execution time. It also depicts the variation of the execution time with each additional SPL insertion (SPL1, SPL2 and SPL3). We may see in this figure that the execution time is reduced by 29.7%, 16.1% and 43.7% for Dimension-reversal, Hotspot and JPEG encoder respectively. In average, we obtained 30.1% execution time reduction. This reduction can simply be related to the reduction of the number of hops offered by the SPL insertion. Since a flit has less hops to traverse to go from one source to its destination, this incurs less buffer writing, less switch arbitration, and less link and crossbar traversal. All of these factors lead to decrease the flit latency and eventually decreasing the execution time.

Figure 5 (b) depicts the throughput comparison results between OASIS-2 and ONoC-SPL when we vary the number of SPLs inserted. The throughput has improved by 49.6%, 24.8% and 22.6% with ONoC-SPL when evaluated with Dimension-reversal, Hotspot and JPEG encoder respectively. In average, we obtained 32.3% throughput enhancement. This amelioration is also another consequence of the number of hops reduction offered by the SPL insertion and the eventual communication latency reduction obtained. Besides, the Extra-port added for the SPL insertion, allows the flits traveling the network to have better routing choices which eventually will decrease the congestion and improve the traffic balance along the whole network.

Finally, Fig.6 shows the floorplan generated for the designed ONoC-SPL implemented on the Altera Stratix III EP3SL150F1152C2 device.

IV. Conclusion

In this paper, we presented an architecture, design, and evaluation of a Network-on-Chip architecture, named ONoC-SPL, which utilizes a Short-Pass-Link (SPL) insertion customization to reduce the communication latency which directly affects the overall system performance. We also prototyped the system on FPGA and evaluated its performance in terms of hardware complexity, execution time and throughput.

From the performance evaluation results, we observed that the execution time has decreased with 30.1% and the throughput has enhanced by 32.3% in average when comparing the proposed system with the previously designed OASIS-2 design using the three selected benchmarks. This performance enhancement came with a small area utilization degradation (less than 5%) and a very slight 0.49% power.
consumption overhead. According to the results obtained, we may conclude that our proposed system provides a very effective architecture for balancing the area utilization, power and performance for NoC designs, especially for data-intensive computation applications.

As a future work, we plan to optimize the proposed system by employing fault tolerance techniques, especially with the LA-XY routing adopted for our design. Although, ONoC-SPL showed better performance, it is still not an ideal solution for future large scale systems due to the increasing hop count, especially with large network size. Therefore, we plan to exploit the benefits of the proposed customization with more efficient architectures such as three dimensional Network-on-Chips (3D-NoC).

REFERENCES


