Evaluating On-Chip Interconnection Architectures for Parallel Processing*

Henrique C. Freitas† and Philippe O. A. Navaux
Parallel and Distributed Processing Group
Graduate Program in Computer Science, Informatics Institute
Universidade Federal do Rio Grande do Sul, Brazil
{hcfreitas, navaux}@inf.ufrgs.br

Abstract

For the next processor generation, many cores and parallel programming will provide high-throughput and high-performance processing. As a consequence, research works have studied on-chip interconnection architectures to identify alternatives capable of decreasing the communication latencies. The objective of this paper is to present the evaluation of three well-known architectures (bus, crossbar switch and a conventional network-on-chip) in order to propose a multi-cluster network-on-chip architecture for parallel processing. The results show that a NoC composed of programmable routers and crossbar switches to interconnect clusters of cores has a better performance than conventional NoCs.

1. Introduction

Parallel processing [1] is the best approach to design new architectures capable of improving the computational performance. In this case, three examples can be described: i) Instruction Level Parallelism (ILP), ii) Thread Level Parallelism (TLP), and iii) computer clusters.

Instruction level parallelism [2] is a technique supported by superscalar processors, which execute more than one instruction simultaneously. Due to the multiple execution pipelines, sequential or parallel applications with high instruction parallelism achieve better performance results running on superscalar pipelines.

In contrast, there are many workloads with low instruction parallelism. In this case, instruction flows (threads) executed in parallel achieve better performance running on multiple processing units.

Multithreading [3] is an alternative to explore the throughput-based parallelism. Superscalar processors cannot perform a good work, since the application has a large region that runs sequentially. Thread level parallelism is common in single-core (thread contexts in a superscalar pipeline) and multi-core processors, but due to the limited number of functional units in superscalar pipelines, the multi-core processors support throughput-based parallelism better than single-core processors.

Basically, in ILP and TLP techniques the processing units access a shared memory. On the other hand, distributed memories and message-passing programming model are common in a computer cluster [1]. A cluster-based environment can explore the parallel processing of each node (a computer) in terms of TLP and ILP supported by processors, and in terms of scalability supported by the network. On the network, the communication among nodes is based on message-passing programming.

Due to the large-scale integration, the number of processing cores has increased a lot, and the great challenge is the many-core processor [4]. Actually, the perspective is to provide high-performance processing. For this reason, the designers have focused on an important issue: on-chip interconnection architectures.

As a consequence, for general-purpose and parallel processing contexts, there is a question: How to increase the data throughput of on-chip interconnections?

Therefore, the objective of this paper is the evaluation of on-chip interconnection architectures focusing on two of the most popular interconnections (bus and crossbar switch) [5][6] and on the NoC (Network-on-Chip) [7]. Last one is considered the trend for future generations of multi-core processors. In addition, this paper presents and evaluates a multi-cluster NoC proposal as an alternative to support parallel processing and to increase data throughput.

* This work was partially supported by CNPq, Microsoft and Intel.
† On leave from Pontifícia Universidade Católica de Minas Gerais (PUC Minas), Belo Horizonte, Brazil, cota@pucminas.br.
The evaluation of on-chip interconnections on specific parallel communication patterns [1][5] is the main contribution of this paper.

2. Related Works

According to the several topology architectures capable of improving the performance, and other issues, some research works [7] have evaluated on-chip interconnections in order to identify the trends for many-core processors.

In [8], on-chip static interconnections were evaluated through analytical models in order to verify the impact of area, power, speed, and cost. Three interconnections were studied: mesh, binary tree and cube connected cycles. The main result pointed out to better efficiency of mesh, since the other topologies have long interconnections and so, a penalty delay.

In the same way, the work [6] evaluated the area, power, performance, and design issues. The results pointed out that increasing interconnect bandwidth reduces the number of cores and caches, and does not increase the multi-core performance.

In [9], the objective is the energy and latency evaluation focusing on NoC topologies. The approach was based on a simulation engine and a Tabu-based heuristic optimization algorithm. The results show that fat-tree topology fulfills the latency constraints and the mesh topology the energy consumption. The best efficiency related to energy and latency is achieved with fat-tree topology.

In this paper, interconnection architectures (bus, crossbar switch and conventional NoC) are analyzed and evaluated in order to propose a multi-cluster NoC architecture with low latency and high data throughput.

3. On-Chip Interconnection Architectures

Design decisions related to on-chip interconnection architectures consider the same taxonomy used for computer interconnection structures [5][10]. This taxonomy can be divided into the following levels: i) Transfer Strategy, ii) Transfer Control Method, iii) Transfer Path Structure, and iv) System Architecture or Topology.

In this paper, three of the most popular interconnections (bus, crossbar switch and network-on-chip) are analyzed and evaluated in order to propose a multi-cluster NoC architecture with low latency and high data throughput.

4. Evaluation Methodology

This paper presents three models [11] in order to evaluate on-chip interconnection architectures as follows: i) analytic, ii) simulation, and iii) experimental design.

The simulation was the first model used in this paper. Basically, communication patterns modeled by Petri Nets [12] were simulated through the Visual Object Net tool [13]. This simulation tool verified the models and generated performance results to check the latency impact on packet forwarding.
Queuing models [11][14] were used to identify the main metrics during packet transmissions. In this case, components of a conventional NoC and a Multi-Cluster NoC proposal were analyzed in order to check the total transmission time to send packets with different sizes.

At last, two prototypes were implemented on an FPGA (Field Programmable Gate Array) platform [15] to check the area occupation and network bandwidth and throughput.

According to the network utilization and the limits of speedup demonstrated in other research works [1][7][11][16], this evaluation methodology shows the importance of a cluster-based NoC architecture for parallel processing and high data throughput.

5. Modeling with Petri Nets

Petri Nets [12] are state machines capable of modeling communication, concurrency and conflicts. A Petri Net has a graphical representation that consists of Places (O), Transitions (l), Arcs (→) and Tokens (●). Places represent conditions, activities or resources. Transitions represent events or actions. Arcs represent token directions from places to transitions or from transitions to places. Tokens represent the state of a Petri Net.

In this paper, Petri Nets are used to model three communication patterns common in network topologies and collective communications. The objective is to evaluate the latency impact to perform simultaneous communications (adjacent nodes), one-to-all communication (broadcast), and all-to-one communication (slaves to master). In order to show the latency, each model consists of timed transitions that represent cycles to begin a packet transmission.

For buses or crossbar switches, a simple arbiter can delay in one cycle to alter a communication. In contrast, a conventional NoC router delays three cycles to decide the packet path. If the parallel application is a shared-memory program it is necessary a new (one) cycle to convert (network adapter) the data word to a network packet, and other cycle to convert the network packet to data word.

Considering the current trend of shared-memory programming for multi-core processors, Figure 2 shows Petri Net models of simultaneous and adjacent communications according to a task logical mapping onto eight cores. The Bus bandwidth limitation is represented by the place Bus Arbiter. Any interconnection occurs in alternating slices of time, and only one communication per slice is possible. In contrast, both crossbar switch and NoC can perform simultaneous communications, but the latencies are different. Although the low bus latency and the NoC parallel communication, the simulation results, presented in Section 7, shows the high bus and NoC delays to this pattern.

Figure 2. Simultaneous communication (8 cores): (a) bus, (b) crossbar switch, (c) NoC (torus)

Figure 3 shows the broadcast communication (1-to-7) for eight cores. Due to the native features of bus and crossbar switch, all cores receive simultaneously the same packet. On the other hand, a 2x4 torus-based NoC with X-Y routing technique needs four intermediate routers or 14 cycles to forward a packet to the farthest core. In addition, according to X-Y routing, the source core previously send the same packet for all other cores in alternating slices of time. Figure 3.c shows the model for the last destination core, but the simulation results consider all previous delays to achieve this core.

Figure 3. Broadcast communication (8 cores): (a) bus, (b) crossbar switch, (c) NoC (torus)

Figure 4 shows the all-to-one communication pattern and the respective latencies for each approach. Figure 4.a can be modeled as Figure 2.a, but this new
modeling represents the total latency cost (7 cycles) considering seven source cores. Similarly, the total cost also was modeled for the crossbar switch (7 cycles). However, the NoC cost model considers only the last core (14 cycles) and according to Figure 3.c all previous communications must be considered as a transmission time overhead.

Figure 4. All-to-one communication (8 cores): (a) bus, (b) crossbar switch, (c) NoC (torus)

6. Multi-Cluster NoC Architecture

The NoC architecture proposed in this section is based on features described in the previous sections, according to the state-of-the-art and current multi-core processors. For these reasons, the main motivations are the following: i) problems and limitations of traditional interconnections, for instance, the wire size and resistance [7], ii) NoC is the alternative for many-core processors [7], iii) current multi-core processors (e.g. eight cores) have high performance through crossbar switch [17], iv) applications have a large instruction region that runs sequentially [1], v) the isoefficiency limits [16], and vi) NoC routers may increase the packet contention [7][11].

Figure 5 shows a conventional NoC router. Simple architectures with low latencies, as described in this paper, consist of dedicated routers based on handshaking flow control, wormhole forwarding strategy, and X-Y routing. This type of router manages one processing core and may cause a high packet contention for parallel and intensive communications.

Figure 5. Conventional NoC router (Queuing Model)

Figure 6 shows the MCNoC (Multi-Cluster NoC) architecture. According to the scalability problems related to the crossbar switch and the application parallelism, programmable routers capable of managing clusters of cores compose the MCNoC. Each cluster consists of one programmable router, eight cores and a crossbar switch. Central routers are responsible for interconnecting the other clusters. Therefore, regarding taxonomy and state-of-the-art, MCNoC is an indirect interconnection network.

Figure 6. Multi-Cluster NoC Architecture

Figure 7 shows the programmable router architecture composed of buffered inputs, outputs, a crossbar switch, and a Network Processor responsible for adding flexibility and performance to process a large number of general-purpose parallel communication patterns.
7. Results and Evaluation

The Petri Net models described in Section 5 were simulated through the Visual Object Net [13]. Figure 8 shows the number of packets (tokens) transmitted during period of 500 and 1000 cycles. In order to verify only the influence of latencies, a token has the same size of a data word. So, for each token only the latency influences to the transmission time.

Figure 8 shows the forwarded packets for simultaneous (adjacent), broadcast, and all-to-one communication patterns. For simultaneous communication, the crossbar switch presented the best results achieving a speed up to 8.06x and 4.06x related to bus and NoC (mesh or torus) respectively. The broadcast pattern shows the problem of the NoC X-Y routing that has a bad performance. The crossbar switch and bus achieved a speed up to 4.09x and 33.27x related to torus-based and mesh-based NoC, respectively. The all-to-one results show the smaller difference, but the crossbar switch and the bus achieved a speed up to 2.03x related to torus-based NoC. However, these results consider only the NoC farthest core total latency, and with this best condition, the NoC loses for the other interconnections.

![Figure 8. Simulation based on Petri Net (for eight cores)](image)

The Petri Net simulation shows the latency influence, but considering the bus and crossbar switch limits for many-core processors, and the NoC architecture trend, the next results show evaluations of a conventional NoC and a proposed multi-cluster NoC.

In order to evaluate the transmission time, and the impact of packet sizes, the queuing model based on references [11][14] is depicted through the equations (1), (2), (3) and (4), and Table 1.

The sender and receiver overhead are the same for all interconnections, as consequence, these values were not used. All latency values for each interconnection were presented in Section 5 (Petri Net Models), the packet path has 32 bits, the frequency is 100MHz, and the total message has 1000 packets.

\[
Nt = L + MS / B \quad (1)
\]

\[
Rt = St + Qt \quad (2)
\]

\[
Ot = So + Ro \quad (3)
\]

\[
Tt = Nt + Rt + Ot \quad (4)
\]

<table>
<thead>
<tr>
<th>Table 1. Evaluation Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Metrics</strong></td>
</tr>
<tr>
<td>Nt</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>Ms</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>Rt</td>
</tr>
<tr>
<td>Qt</td>
</tr>
<tr>
<td>Ot</td>
</tr>
<tr>
<td>So</td>
</tr>
<tr>
<td>Ro</td>
</tr>
<tr>
<td>Tt</td>
</tr>
</tbody>
</table>

According to the advantages of a small crossbar switch, the MCNoC router for eight cores is compared with two conventional NoC topologies based on 2x4 mesh and 2x4 torus. The evaluation results consist of simultaneous (adjacent) communication, tree-based concurrent communication (3 nodes to 1 node), broadcast communication, and all-to-one communication.

Figure 9 shows the equation (4) results to perform one thousand transmissions for different packet sizes (128 bytes to 4096 bytes). MCNoC decreases transmission time up to 20% for simultaneous communication related to both mesh and torus NoCs; up to 70% and 57% for tree-based concurrent communication related to mesh and torus NoCs, respectively; up to 86% to perform a broadcast related to both NoC versions; and up to 7% and 5% for all-to-one communication related to mesh and torus NoC versions, respectively.

![Figure 9. Transmission Time](image)

The results presented in Figure 9 were modeled for a NoC based on wormhole forwarding technique. However, if the NoC works with store-and-forward technique, the queue time is very high. In order to show this impact, Figure 10 presents the transmission time to send one packet with different sizes (128 bytes to 4096 bytes) for the simultaneous communication that has a low impact for all NoC versions. As a result, the store-and-forward technique shows the increase of time for mesh and torus NoCs, and for this reason, the large difference related to MCNoC. In this case, the
MCNoC decreases transmission time up to 15% for wormhole, but 67% considering store-and-forward.

![Figure 10. Transmission Time (simultaneous)](image)

**Table 2. Prototype Results (FPGA 2vp30ff1152-6)**

<table>
<thead>
<tr>
<th>NoC Versions</th>
<th>Occupation</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FFs</td>
<td>FFs + LUTs</td>
</tr>
<tr>
<td>SoCIN (2x4 torus)</td>
<td>3948</td>
<td>6486</td>
</tr>
<tr>
<td>SoCIN (2x4 mesh)</td>
<td>3425</td>
<td>5464</td>
</tr>
<tr>
<td>MCNoC 8 (1 cluster)</td>
<td>3965</td>
<td>4731</td>
</tr>
</tbody>
</table>

In order to check the previous results, two SoCIN (System-on-Chip Interconnection Network) [18] versions with the same conventional NoC features described in this paper, and a MCNoC version, all of them for 8 cores, were implemented on a Xilinx’s FPGA. Table 2 shows the same bandwidth for all NoC versions, but a high-throughput for MCNoC (7x), since the MCNoC router has parallel links to perform a broadcast. According to crossbar switch features and results, a small cluster managed by one router has better performance. Considering the network processor architecture based on open source code called Plasma [19], the MCNoC area (Flip-Flops + Look-up Tables) has a better occupation (↓2.33%, ↓16.79%) than mesh and torus SoCIN versions, respectively.

8. Conclusions

According to the wire limitations, NoCs have shown the best results for future many-core processors. However, due to the application scalability, the large instruction region that runs sequentially, and the network utilization, a NoC with clusters of cores can increase the performance significantly.

The results show the impact of the NoC latencies, according to the number of routers and the adapter (shared-memory to message-passing). For this reason, MCNoC router consists of a network processor on chip adding flexibility, but also a crossbar switch interconnecting clusters of cores. As a result, this router architecture can add the benefits of a crossbar switch increasing the throughput and so, decreasing the transmission time (up to 86%). For this reason, a multi-cluster NoC can support parallel processing better than a conventional NoC.

Future results focus on evaluation of NoC router architectures based on reconfigurable topologies [20].

9. References