Universitat Politècnica de València
Departament de Informàtica de Sistemes i Computadors

Floorplan-Aware High Performance NoC Design

A dissertation submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy
(Computer Engineering)

Author

Antoni Roca Pérez

Advisor

José Flich Cardo
Federico Silla

València, 2012

Contents

List of Figures

List of Tables

Resum

Arquitectures de múltiples nuclis com a multiprocessadors (CMP) i solucions multiprocessador per a sistemes dins del xip (MPSoCs) actuals es basen en l'eficàcia de les xarxes dins del xip (NoC) per a la comunicació entre els diversos nuclis. En aquest sentit, els CMPs i MPSoCs que s'han fabricat en els darrers anys s'han implementat basant-se en NoC amb una topologia simple, normalment malles o anells. No obstant, a mesura que els nombre de elements a connectar creixia, xarxes simples no eren capaç de satisfer els requeriments del sistema, que són principalment, una baixa latència i alt rendiment. Així, la comunitat investigadora ha proposat i analitzat NoCs ms complexes. No obstant, aquestes solucions són més difícils de implementar – especialment els enllaços llargs – fent que aquest tipus de topologies complexes siguen massa costoses o inclòs inviables.

En aquesta tesi, presentem un metodologia de disseny que minimitza la pèrdua de prestacions de la xarxa degut a la seua implementació real, principalment per la complexitat dels commutadors i dels enllaços llargs. Com a primer pas, re-dissenyem el commutador fent-lo modular. Cada mòdul del commutador és capaç de arbitrar, commutar, i emmagatzemar els missatges que li arriben. Com a segon pas, flexibilitzem la col.locació d'aquestos mòduls arran del xip, permetent que mòduls d'un mateix commutador estiguen distributs per el xip.

Aquesta metodologia de disseny l'hem aplicat en diferents escenaris. Primerament, hem transformat NoC ja conegudes com la malla 2D introduint el nostre commutador modular. Els resultats mostren com la modularitat i la distribució del commutador redueixen la latència i el consum de potència de la xarxa. A més a més, aquests beneficis són majors en aquelles topologies on la

complexitat del commutador és major, com per exemple, la *flattened butterfly* o la *concentrated mesh*.

En segon lloc, hem utilitzat la nostra metodologia de disseny per a implementar un *crossbar* distribuït. El nostre *crossbar* distribuït té unes prestacions clarament millors que la resta de configuracions, tant en latència com en rendiment. No obstant, per a xarxes amb un gran nombre de nodes, el nostre *crossbar* distribuït requereix massa recursos. En aquest cas, hem analitzat un *crossbar* distribuït jeràrquic viable que, a més a més manté pràcticament les prestacions del *crossbar* distribuït.

Abstract

Network-on-Chip (NoCs) have been widely accepted as the interconnection choice for Chip MultiProcessor (CMP) and MultiProcessor System-on-Chip (MPSoCs). In this sense, current CMPs or MPSoCs rely on simple NoC designs based on rings or meshes. As the number of cores to interconnect grows, simple NoCs are not able to fulfil system requirements, that is, low latency and high throughput. In this sense, more complex NoC solutions have been proposed and analyzed. However, complex designs means complex implementations – specially critical is link length – which makes these solutions costly or even unaffordable.

In this dissertation, we present a floorplan-aware NoC design methodology that minimizes the implementation drawbacks of complex NoC designs, that is, high-radix switches and long links. As a first step, we redesign the switch by making it modular and locally self contained. That is, each small block has its own buffering, arbitration, and crossing capabilities. As a second step, we flexibilize the placement of the basic modules, allowing a distribution of the switch basic blocs all over the chip.

We have applied our floorplan-aware NoC design into different scenarios. First, we show that it is possible to build a conventional NoC as a 2D mesh by using a floorplan-aware switch. Results show that modularizing and distributing the switch along the link reduces the network critical path and power consumption over conventional implementations. Additionally, these benefits increase as the switch radix increases, and hence, our floorplan-aware switch fits perfectly in complex NoC implementations, as the flattened butterfly or concentrated meshes.

Second, we have leveraged our floorplan-aware NoC design methodology

to implement a distributed crossbar. Our distributed crossbar clearly outperforms in terms of latency and throughput the rest of configurations. For very large networks, our crossbar resources requirements could be unaffordable. In this case, a hierarchical approach has been implemented which almost keeps all the crossbar benefits but reducing resources used.

Resumen

Las actuales arquitecturas de múltiples núcleos como los chip multiprocesadores (CMP) y soluciones multiprocesador para sistemas dentro del chip (MPSoCs) han adoptado a las redes dentro del chip (NoC) como elemento óptimo para la inter-conexión de los diversos elementos de dichos sistemas. En este sentido, fabricantes de CMPs y MPSoCs han adoptado NoCs sencillas, generalmente con una topología en malla o anillo, ya que son suficientes para satisfacer las necesidades de los sistemas actuales. Sin embargo a medida que los requerimientos del sistema – baja latencia y alto rendimiento – se hacen más exigentes, estas redes tan simples dejan de ser una solución real. Así, la comunidad investigadora ha propuesto y analizado NoCs más complejas. No obstante, estas soluciones son más difíciles de implementar – especialmente los enlaces largos – haciendo que este tipo de topologías complejas sean demasiado costosas o incluso inviables.

En esta tesis, presentamos una metodología de diseño que minimiza la pérdida de prestaciones de la red debido a su implementación real. Los principales problemas que se encuentran al implementar una NoC son los conmutadores y los enlaces largos. En esta tesis, el conmutador se ha hecho modular, es decir, formado como unión de módulos más pequeños. En nuestro caso, los módulos son idénticos, donde cada módulo es capaz de arbitrar, conmutar, y almacenar los mensajes que le llegan. Posteriormente, flexibilizamos la colocación de estos módulos en el chip, permitiendo que módulos de un mismo conmutador estén distribuidos por el chip.

Esta metodología de diseño la hemos aplicado a diferentes escenarios. Primeramente, hemos introducido nuestro conmutador modular en NoCs con topologias conocidas como la malla 2D. Los resultados muestran como la mod-

ularidad y la distribución del conmutador reducen la latencia y el consumo de potencia de la red.

En segundo lugar, hemos utilizado nuestra metodología de diseño para implementar un *crossbar* distribuido. Nuestro *crossbar* distribuido tiene unas prestaciones claramente mejores que el resto de configuraciones, tanto en latencia como en rendimiento. No obstante, para redes con un gran número de nodos, nuestro *crossbar* distribuido necesita demasiados recursos. Para solucionar este problema de escalabilidad, se ha estudiado un *crossbar* distribuido jerárquico viable que, además mantiene prácticamente las prestaciones del *crossbar* distribuido.

Chapter 1

Introduction

In this chapter, we first introduce the reasons that have motivated this dissertation (Section 1.1). Then, we briefly define the specific objectives aimed by the dissertation (Section 1.2). Finally, we outline the structure of the remaining chapters in this document (Section 1.3).

1.1 Motivation

As technology advances, more and more transistors can be included in the same die. In this scenario, conventional out-of-order superscalar uniprocessors present an excessive design complexity that do not exploit all the technology opportunities at reasonable power budgets. In contrast, Chip Multiprocessor (CMP) and Multiprocessor System-on-Chip (MPSoC) systems have been chosen as the power efficient solution for current necessities and challenges. Rather than a complex monolithic uniprocessor, a CMP system is compounded of multiple replicated simple tiles on a single die where each tile includes a processor, some cache structure, and a switch.

The complexity of conventional uniprocessor solutions is managed via replication and design reuse. Nowadays, commercial solutions are able to reach several tens of cores (see Figure 1.1). Examples of such solutions are the Polaris prototype chip [2] by Intel with 80 simple cores, the Single-chip Cloud Computing prototype [3] by Intel with 24 x86 dual-core tiles, and the Tilera 100-core chip [4]. On the other hand, an MPSoC system is built by multiple

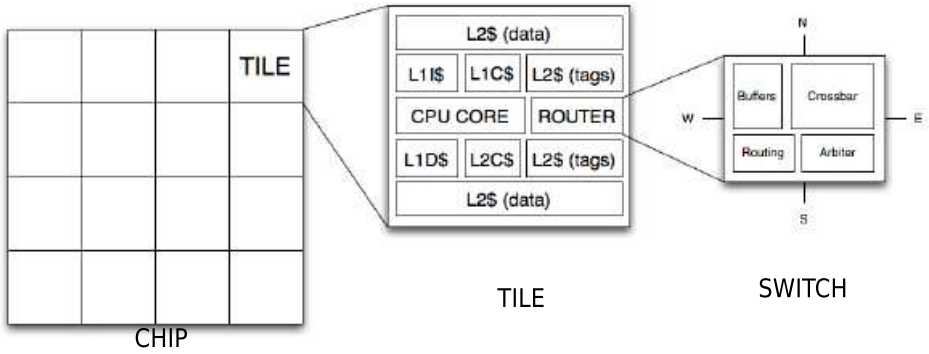


Figure 1.1: Tiled CMP.

heterogeneous processing elements with different sizes and functionalities, such as processors, memories and I/O components. MPSoCs are typically used in the embedded system market domain. In both cases, the trend is that the number of replicated cores will be increased up to thousands of cores.

1.1.1 Network-On-Chip

Aggregating several cores in a single die leads to the necessity of interconnecting them. Initially, when the number of cores to interconnect was low, ad-hoc wiring solutions or a shared bus [5, 6] were sufficient solutions to fulfil performance requirements. However, as the number of cores to interconnect increases, those initial solutions became inefficient. As an alternative, Network-on-Chip (NoC) have been widely accepted as the proper interconnection choice for CMPs and MPSoCs. NoCs are able to meet the severe constraints imposed by the new on-chip environment [7, 8]. The idea is quite simple: a point-to-point network inside the chip connects all the devices by means of switches and links. The popularity of NoCs grew also due to its physical scalability, specially when a simple 2D topology with a low-radix switch is implemented. That is, the planar structure of a 2D NoC makes them suitable for the planar silicon technology, as for example the simple 2D mesh as can be seen in Figure 1.2. For that reason, most existing NoC designs are based on simple topologies as rings [9] or the most popular 2D mesh [10, 11]. In fact, current commercial solutions rely on a 2D mesh such as the Polaris pro-

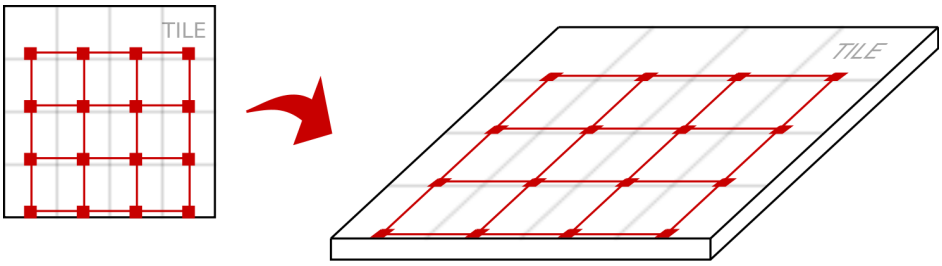


Figure 1.2: 2D mesh planar scalability schematic.

totype chip [2] and the Single-chip Cloud Computing prototype [3] by Intel. The Tilera 100-core chip [4] includes several 2D meshes. Simpler commercial solutions can be found, as for example the Intel Nehalem [12], which relies on a bidirectional ring.

A NoC is built from basic components as switches, links, and network interfaces. Switches and end nodes are connected by links, thus forming the topology and final network structure. Although the network interface must be carefully designed in order not to introduce bottlenecks, the complexity is usually shifted to the switch design. A basic switch should store its incoming flits, route them, and deliver them onto the proper output port. Despite that a switch can be designed in different manners, a crossbar-based switch design has been widely accepted [13,14,14], that is, a switch in which inputs and outputs are interconnected by a matrix of connections. Figure 1.3 shows a basic 5×5 crossbar-based canonical switch. This switch is made of input buffers, which store the incoming flits. Then, a routing module, usually replicated at each input port, is used to compute the proper output port for the stored incoming flits. Next, the switch allocator arbitrates which input port is connected to an output port. Finally, the crossbar is the interconnection structure that physically connects – based on the switch allocator decision – the input ports with the output ports. Typically, the most critical components in a switch are the crossbar and the switch allocator. That is, these elements fix the switch throughput and switch operating frequency. In contrast, the input buffering is the main responsible for the area and power consumption of the switch. Notice that, as the switch radix increases more input/output ports must be handled by the switch components, and hence, the switch complexity increases,

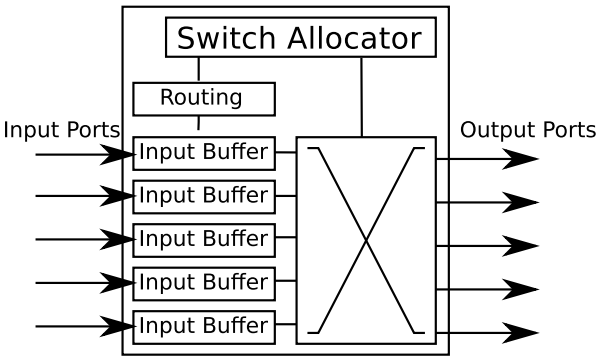


Figure 1.3: 5×5 crossbar-based canonical switch schematic.

leading to an unavoidable increment in area, and most probably a decrease in its operating frequency.

On the other hand, in the NoC scenario, link design becomes more critical than switch design. NoC severe timing and power consumption constraints have made link design a priority. Contrary to the off-chip domain, wiring delay becomes not only appreciable but significant in the message's critical path. Wire delay can be reduced by inserting repeaters along the link at the expense of increasing its area and power consumption. This fact gets worse as the wire length increases to connect more distant components (nodes or switches). Long interconnects are costly or even unfeasible [15] in terms of power consumption. In fact, the popularity of modular and scalable NoC topologies as the 2D mesh is in part due to these wiring concerns. However, despite that the 2D mesh minimizes wire length with respect to more complex architectures and topologies, link power consumption can still be more than 70% of the total network energy consumption [16].

1.1.2 NoC Topologies

Simple 2D network topologies as rings or meshes rely on low-radix switches which leads to a fast network with good performance when the number of elements to interconnect remains low. However, meanwhile a simple 2D topology offers a proper connectivity for systems with dozens of tiles (processors), those network implementations are unable to fulfil larger system requirements. As the amount of devices to interconnect increases, the performance – measured

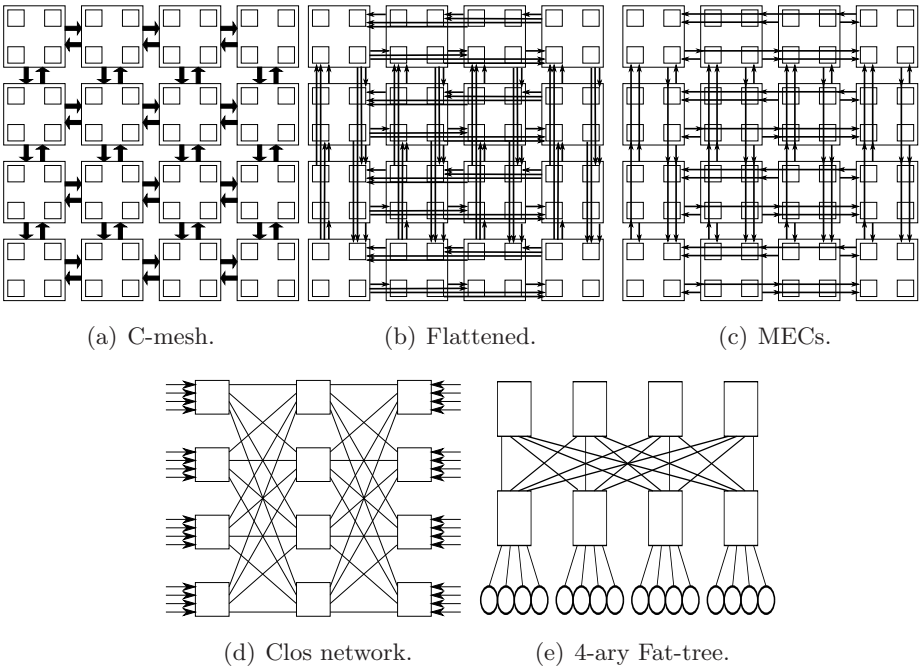
as latency and throughput [17] – of these simple topologies is highly impacted due to its lack of scalability. First, message latency increases due to the larger network diameter, increasing also the energy to transmit messages even with a very low traffic injection rate when no contention exists. But, the main limitation is the network throughput these topologies may achieve. Indeed, the throughput of these topologies is highly impacted as the bisection bandwidth[1] increases linearly with the square of the number of nodes to interconnect. As the network traffic increases, it becomes congested, and hence, message latency exponentially increases due to the queueing latency. Furthermore, the 2D mesh structure may bring an unbalanced network utilization. Switches placed at the borders are underused with respect to switches placed at the center of the network. Unbalanced network utilization reinforces the congestion problem by increasing queueing message latency and making message latency unpredictable.

The performance degradation and energy increment that those topologies suffer when the number of cores to interconnect increases, force the designers to explore new solutions based on planar processes. However, the current planar silicon substrate technology restricts the space of implementable networks [18][2]. In this sense, researchers have developed more complex topologies that overcome the inefficiencies of basic ones but still being implemented as a planar structure. A flattened butterfly topology [21, 22] (see Figure 1.4(b)), multi-stage networks such as the Clos network [23] (see Figure 1.4(d)), topologies based on fat-tree networks [24,25] (see Figure 1.4(e)), concentrated meshes (C-meshes) [26] (see Figure 1.4(a)), hierarchical topologies [27], or multidrop express channels (MEC) [28] (see Figure 1.4(c)) are some remarkable solutions recently suggested by researchers.

In general, these complex topologies reduce network latency by reducing the hop count of messages. Reducing hop count is achieved by *concentrating* nodes to a switch (C-mesh), by increasing the connectivity between switches (flattened butterfly or MEC), or by completely modifying the network con-

[1] Bisection is the minimum set of network components that split the network in two equal halves. Bisection bandwidth is the bandwidth at the bisection.

[2] Recently, a 3D silicon technology has appeared. However, this technology is currently in experimentation phase [19, 20] so will not be considered in this thesis.



(a) C-mesh. (b) Flattened. (c) MECs.

(d) Clos network. (e) 4-ary Fat-tree.

Figure 1.4: Complex network topologies.

cept (Clos or tree-based networks). From the theoretical point of view, there is no wining topology that clearly overcomes the rest in all the critical aspects of the NoC environment, mainly performance (throughput and latency), power consumption, and implementation costs (area). In Table 1.1 we can see different metrics for different 64-node topologies. As it can be seen, a 4-ary 3-tree topology (similar construction as Figure 1.4(e)) presents the maximum bisection bandwidth but the number of switches and the network diameter is higher than the C-mesh topology (see Figure 1.4(a)) and the 4-ary 3-flat topology (see Figure 1.4(b)). The 4-ary 3-flat topology has the lowest network diameter, but the switch radix (number of ports) – and hence its complexity – is the highest too. Moreover, final applications to run on top of these systems influence the topology selection process [7]. Thus, low latency, resources available, or throughput requirements fixed by applications can determine the topology pick for the network design.

In NoCs, contrary to the off-chip domain, complex topologies exacerbate two important drawbacks. First, those topologies rely on high-radix switches.

	Switches	Bisection BW	Diameter	Switch radix
2D mesh	64	16	15	5
4-ary 3-tree	48	64	5	8
C-mesh	16	16	5	8
4-ary 3-flat	16	32	3	10

Table 1.1: Analytical properties for different 64-node topologies.

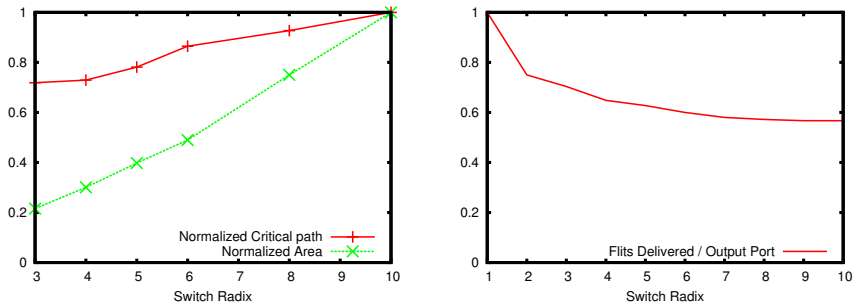
Second, more resources are needed when designing such switches which leads to an increment in power consumption. This increment in resources gets even worse since the number and length of links increases. As the switch radix increases, switch complexity –and hence switch latency – increases too, as can be seen in Figure 1.5(a). This figure shows the normalized critical path and area[3]. As the radix increases, both the switch delay and area increase – the increment in area is specially critical. Beyond 10 ports, the increment curve is even higher. Thus, topologies made by high-radix switches, although reduce hop count, they suffer from increased switch latency. Furthermore, high-radix switches suffer from traffic concentration, that is, they receive traffic from a higher number of input ports which can lead to a switch congestion, and hence, reducing overall network throughput. Figure 1.5(b) shows the throughput per output port of a switch when each input port has an injection traffic rate of one flit/cycle[4]. As it can be seen, the throughput of a switch gets worse as the number of input/output ports increases. This traffic concentration is noticeable in high-radix topologies[5]. For example, the concentrated mesh topology, the flattened topology or the hierarchical proposals presented in [27] achieve a network throughput that is even lower than the one achieved by the simple 2D mesh topology.

If we add the fact that long links (found in high-radix networks) also lead to possibly prohibitive increases in power dissipation, all the previous comments lead to questioning whether high-radix networks are feasible and preferable.

[3]Results obtained after synthesising the canonical switch presented in Appendix A with the 45nm Opensource Nangate Library.

[4]Flit (flow control bit) is the minimum amount of information that is flow controlled in the network.

[5]We refer to high-radix topologies those implemented by using high-radix switches.



(a) Normalized critical path and area. (b) Switch throughput per output port.

Figure 1.5: Switch properties as a function of switch radix.

1.1.3 Crossbars

When derived to the ideal case, a fully connected network where each node is directly connected to any node with a bidirectional link is the most aggressive high-radix network. Alternatively, a less aggressive solution is a single crossbar which can also be used to interconnect all the nodes. Figure 1.6 shows a $N \times N$ crossbar where a simple matrix connects N inputs with N outputs. The crossbar has been traditionally discarded by the common belief of its infeasibility due to its high wiring complexity, and its arbiter complexity (N inputs to N outputs). Notice that replacing an entire conventional NoC, like a 2D mesh, by a crossbar provides many advantages from a conceptual point of view. First, the bisection bandwidth of a crossbar is higher than that of the 2D-mesh network. While the bisection bandwidth of the 2D mesh is $\sqrt{N}$, the bisection bandwidth of the crossbar can be computed as $N^2/4$, where N is the number of nodes in the network. Moreover, the average latency in a crossbar is noticeably reduced (theoretically in terms of cycles) as only one hop (one cycle) is required by a node to reach any other node in the die.

In addition to the huge performance improvement, crossbars present several interesting features. As each node is directly connected to the rest of nodes a crossbar does not depends on the traffic pattern, that is, traffic to a node does not affect traffic to the rest of nodes avoiding collisions between packets addressed to different destinations (see Figure 1.7(a)). In this sense, a crossbar is insensitive to bursty traffic and network contention. A burst of

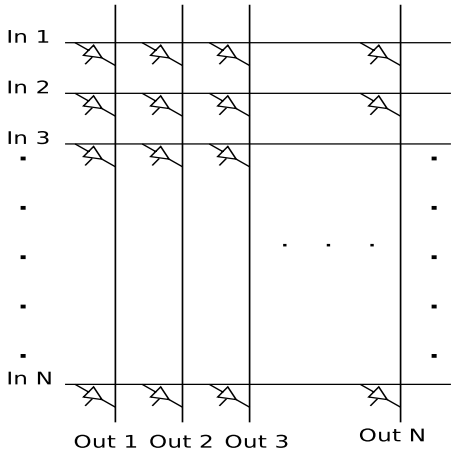
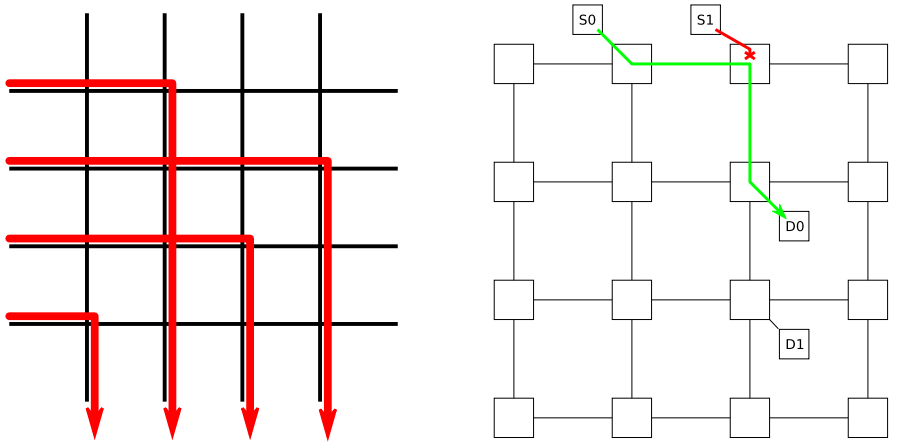


Figure 1.6: Crosspoint-based crossbar.

messages from a given input to a given output will not interfere with other communicating flows. On the contrary, in a 2D mesh (Figure 1.7(b)), the red flow is temporarily blocked by the green flow of messages, even in the case both flows are for different destinations. The absence of network conflicts reduces traditional problems as deadlock or head-of-line blocking. A crossbar ensures deadlock-freedom. Whereas Head-of-line blocking (where packets requesting free resources are blocked by packets requesting to other outputs) is removed inside the crossbar, it may occur on the NIC limiting the throughput to a 58.6% when uniform traffic and a single FIFO per input is used [29][6]. More complex NIC implementation or buffered crossbars can increase this limit up to 100% [30].

In addition, direct connection between nodes simplifies implementation. First, there is no need for routing packets that traverse the network and, hence, no routing bits must be prepended to packet headers, thus increasing the effective network throughput. Most important is that support for multicast and broadcast in crossbars is straight-forward, as only the crossbar has to be configured to multiple outputs. Figure 1.8 shows two forms to manage a broadcast message. In Figure 1.8(a) any broadcast message compete for all the outputs at a time. When a broadcast message wins it uses all the outputs.

[6]Despite that this limit is far away than the maximum – 100 % – is still higher than in other architectures, specially as the network size increases.



(a) Collision avoidance in a crossbar.

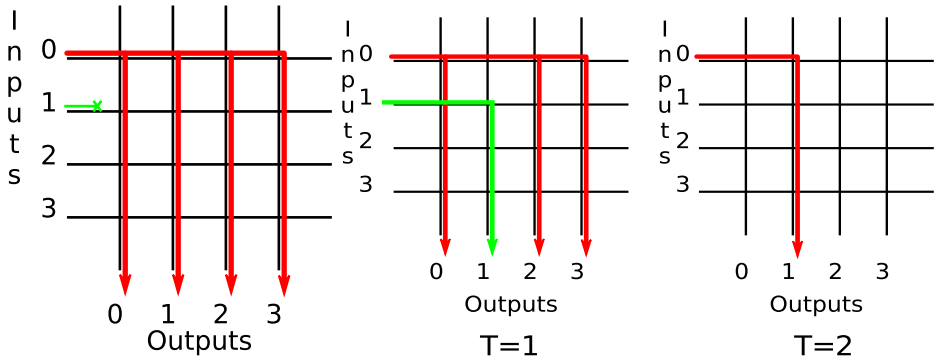
(b) Packet blocked by a collision in a 2D switch

Figure 1.7: Collision avoidance in a crossbar network.

On the other hand, in Figure 1.8(b) a broadcast message is sent at the same time a unicast message progresses. As it can be seen, at cycle one, input two requires the use of output two, and hence, the broadcast message is able to arrive to any destination but to output two. During the next cycle, the broadcast message is sent to output two. Broadcast transmission ends when the broadcast message is transmitted to all the requested outputs. Notice that to support multicast messaging, the network interface will need minor changes. The properties mentioned above – high throughput, absence of collisions – joined to the constant latency obtained from any source node to any destination node promote the crossbar as highly predictable, which allows for simpler QoS solutions being implemented in crossbars.

Although theoretically crossbar are not scalable, in [1, 31] a 128×128 centralized crossbar has been designed and synthesized. In [32] buffers are introduced inside the crossbar design to solve scalability and implementation cost issues. In these works, it is shown that a proper and meticulous design process can lead to a feasible large centralized crossbar as in Figure 1.9. Similarly, in [21, 24, 28], it is shown that high-radix topologies are feasible without an excessive cost in terms of resources.

In addition to the resources needed for these high-radix solutions, the de-



(a) Broadcast message treated as a indivisible unit.

(b) Broadcast message.

Figure 1.8: Broadcast support for a fully connected network.

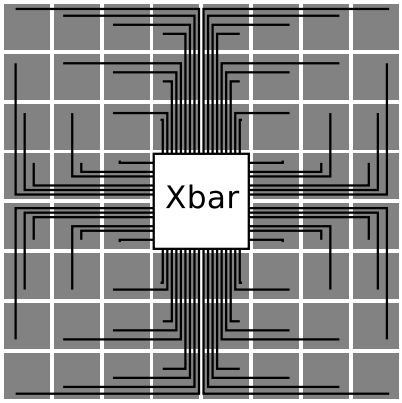


Figure 1.9: Crossbar-based fully connected network floorplan [1].

sign of each component is also critical. Indeed, a naive implementation of the switch will minimize or even cancel the theoretical benefits of these topologies. Indeed, the real problem is that those designs are not floorplan-aware, in the sense that the design tools can not take profit from the design and perform an optimal floorplan.

1.1.4 Contribution

In this dissertation, we aim to minimize the negative impact of a real implementation of a NoC by presenting a novel floorplan-aware design methodology.

We consider a complex NoC design as a concatenation of small and simple modules. The simplicity of those modules makes them easily optimized at design time or by the synthesis tools. In addition, the modularity of a large and complex design allows the designer to properly place each module along the chip. By simplifying, modularizing, and spreading along the chip a complex NoC design, higher operating frequencies can be achieved because problems related with long links are minimized and switch design is optimized. With this approach, the large gap between the theoretical performance of high-radix networks and their real performance when implemented is closed with respect to other methodologies.

Figure 1.10 shows the flow diagram of this dissertation. The starting point is a canonical switch in which its major components are buffering, routing, arbitration, and forwarding incoming flits. The first step in this thesis is to convert the canonical switch into a modular switch design made of simple and identical modules. To perform this first step, it is necessary to reorganize the canonical switch pipeline. With this approach, we mimic the canonical switch behaviour. We call this novel switch the *modular switch* (see Figure 1.11(b)). In the modular switch, each output port is independent of the rest of ports. That is, each output port is able to store, arbitrate, and forward a flit. In addition to the independence of the output ports, the *modular switch* has another interesting property. Output port tasks – arbitration, buffering, and forwarding – are splitted in smaller units thus leading to simpler and faster decisions.

Modularizing the switch allows the designer to properly place and route the switch along the chip, because each module can be handled independently. The NoC design methodology presented in this dissertation fixes the basic module (termed AC) as the basic design unit. In this sense, as it can be seen in Figure 1.12, it is possible to spread a single switch along the chip by spreading the AC modules that compound that switch. Thus, as a second contribution in this thesis, we provide a distributed switch approach where the switch components are spread over the chip. At the physical level, the network becomes a chain of AC modules uniformly distributed over the chip surface (see Figure 1.12(b)). By spreading basic modules link length is reduced by half, and hence, the link power consumption is reduced and overall network

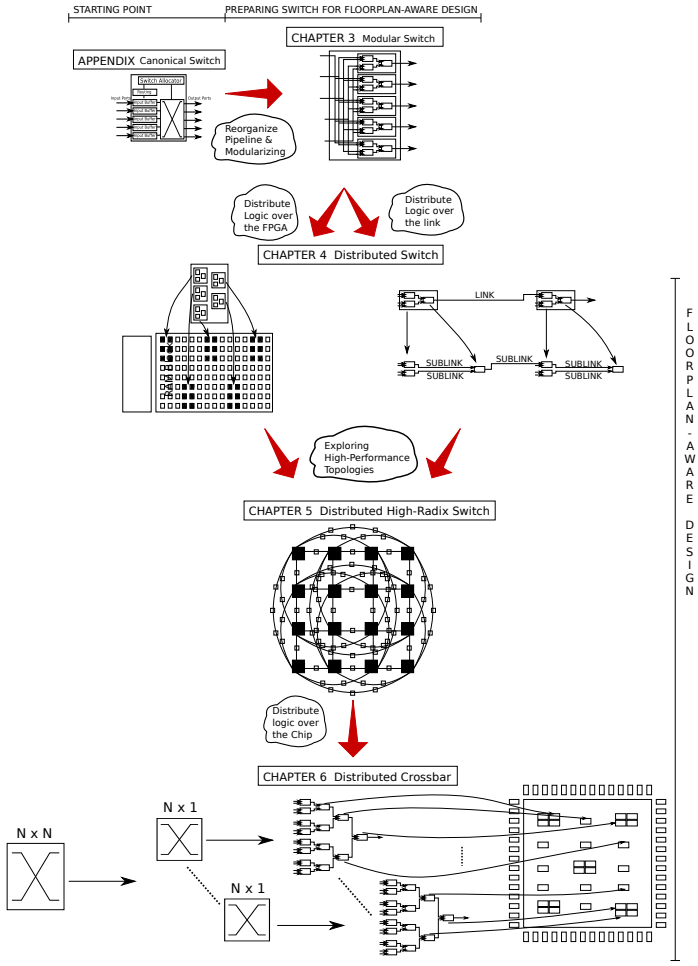
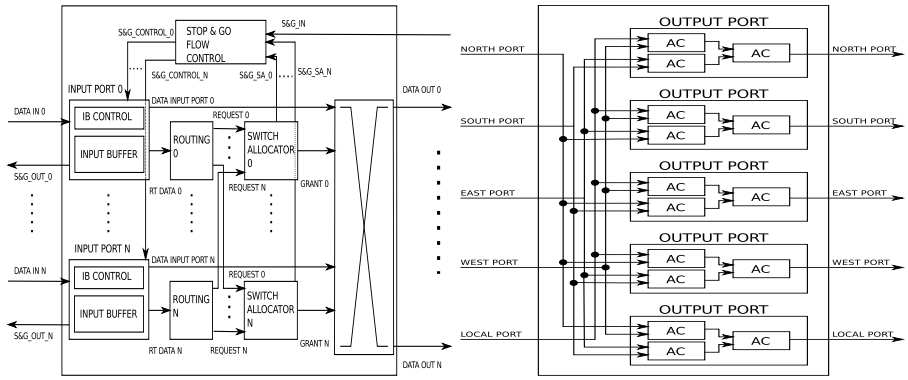


Figure 1.10: Thesis flow diagram.

frequency is increased.

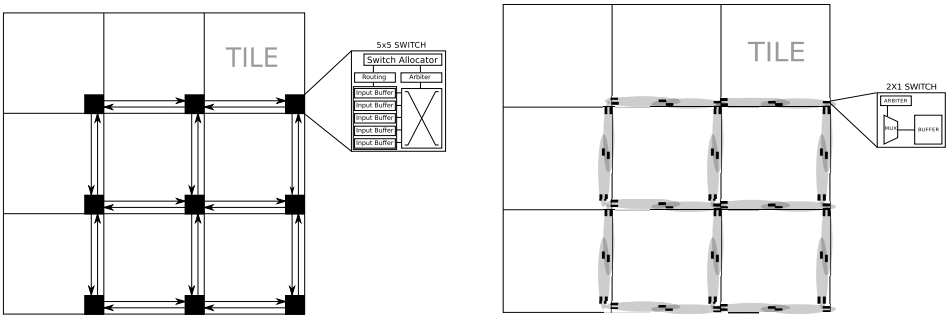
By reducing link length, and modularizing the switch, high-radix topologies can be reconsidered with this approach. Indeed, in a third contribution in this thesis, we promote the use of high-radix networks by using the distributed modular switch approach. By doing this, the gap between theoretical network performance and real network performance (when placed and routed) gets reduced.

Indeed, it is possible to reuse the *modular switch* design philosophy to implement a large crossbar in an efficient, modularized, and distributed manner



(a) Canonic switch schematic. (b) Modular switch schematic.

Figure 1.11: Canonic and modular switches schematic.



(a) Floorplan with standard switches. (b) Floorplan with distributed switches.

Figure 1.12: Floorplan of a 2D mesh with standard and distributed switches.

for small and even medium – up to 64 node – size networks. Basically, a large crossbar is decomposed in a cascade of AC modules. In fact, our floorplan-aware design we propose in this thesis for a distributed crossbar is able to use less resources than a conventional 2D mesh for small networks. In order to cover ultra-large many-core systems, a hierarchical design is proposed to overcome the scalable problems of very large crossbars. A hierarchical distributed crossbar allows a crossbar-based design for systems larger than 64 nodes.

1.2 Objectives

This section presents the objectives of this dissertation. The key objective we seek is to describe a simple and novel philosophy to design a NoC that reduces the negative impact of long links and improve overall network performance. Basically, NoC overall functionality – buffering, arbitration and forwarding– is mixed and splitted into simpler and spread –distributed – decisions. In order to achieve this overall goal, we pursue the following specific goals:

- Provide an insight of the most recent related work in switch and high-performance on-chip network design.
- Describe a new switch architecture (*modular switch*) where arbitration, buffering, and forwarding capabilities are pipelined into a cascade of small and simple modules.
- Propose a new floorplan-aware 2D mesh NoC architecture which reduces link interconnect negative impact. For that purpose we design a proper place-and-route of our *modular switch*, called the *distributed switch*. The *distributed switch* is the foundation of our high-performance NoC solutions.
- Leverage our *distributed switch* to implement and improve current high-radix 2D topologies.
- Demonstrate the feasibility of fully connected networks implemented as large crossbars for small and medium networks, when the *distributed switch* philosophy design is reused. We refer to our crossbar as the *distributed crossbar*. We show the inefficiencies of current 2D high-radix topologies with respect to crossbars for small and medium networks.
- Propose a feasible and efficient high-performance floorplan-aware hierarchical distributed crossbar design – *hierarchical distributed crossbar* – for many-core systems.

In conclusion, we propose a new floorplan-aware NoC design philosophy, that is able to improve current designs. Our philosophy can be used to implement a simple 2D mesh – *distributed switch* – or leverage this solution to

implement a feasible high-radix topology. Finally, our *distributed switch* can be leveraged to recover traditionally discarded solutions as large crossbars. Our *distributed crossbar* is suitable for small and medium networks, meanwhile our *hierarchical distributed crossbar* is suitable for high-performance large networks.

1.3 Dissertation Outline

This dissertation begins with this introductory chapter (Chapter 1). Then, we continue with Chapter 2 describing the basics of on-chip interconnection networks and an analysis of the current state of art that contributes to the matter of this dissertation. Chapter 3 presents the *modular switch* which is the baseline switch that will be used in later Chapters. Chapter 4 presents the *distributed switch* which is the solution to minimize the impact of long links. Chapter 3 and Chapter 4 focus on a simple topology as the 2D mesh, however, these chapters present the basic concepts and design methodology that will be used in Chapter 5. Indeed, in Chapter 5 we present the application of the distributed switch design to high-radix topologies. Then, Chapter 6 shows that a fully connected network designed as a distributed crossbar is feasible for medium networks. To overcome scalability concerns, a hierarchical distributed crossbar is presented in the same chapter. Finally, the dissertation ends with Chapter 7, which summarizes the conclusions and summarizes the contributions related to the research field.

Chapter 2

Technical Background and Related Work

In this chapter, the goal is to describe the basics and terminology of on-chip interconnection networks. For the sake of understanding we will cover the main aspects, but it is not the intention of this chapter to provide an in-depth view on the subject, since the on-chip network field is as complex as the general interconnection network field, and there exist several aspects that are beyond the scope of this dissertation. We refer the reader to the established textbooks on this topic and related ones for further background and introductory material [7, 8, 17, 33].

First, in Section 2.1 a brief description of the design parameters that involve networks-on-chip are presented. Then, in Section 2.2, we dive into a more extensive description of the aspects that surround this kind of networks. Finally, this chapter, in Section 2.3, shows the related work and existent contributions that serve as a reference for this dissertation.

2.1 On-chip Interconnection Networks

In the field of interconnection networks, there is a growing interest and amount of research in the on-chip domain. Since the appearance of the NoC concept, many research groups and institutions have turned their attention into it and they have contributed to a plethora of proposals in related conferences and

journals. The integrated circuit technology has evolved to accommodate a multiprocessing device capable of high-performance computation. As a result of the high integration scale in the deep sub-micron domain and the increasing number of connecting elements, on-chip interconnection has become a need and influences the performance of the final system. So, any gain in the efficiency of the on-chip interconnection layer will be highly beneficial for the entire system. Next, we describe the main design factors that should drive any research devoted to NoCs.

2.1.1 Design Factors

As aforementioned, NoCs play a major role in the design of the modern high-performance computers, nevertheless, they are not simple; there are many factors that affect the choice of an appropriate interconnection layer at design time. The main factors are:

- *Performance.* As commented, performance is a key point in interconnection networks, not only from the point of view of raw throughput, but also from the point of view of latency. Latency is a critical design issue in several systems such as real-time systems. Moreover, in on-chip networks, messages must reach destinations in terms of nanoseconds.

- *Scalability.* Scalability is the first design rule that an interconnect designer should keep in mind. Scalability in interconnection networks implies that the bandwidth of the network increases proportionally to the number of elements of the system. Latency should also be kept to reasonable limits when increasing the system size. Otherwise, the interconnection network would become a bottleneck, limiting the efficiency of the whole system. Scalability also implies that network cost and resources are proportional to the network size.

- *Reliability.* An interconnection network should be able to deliver information in a reliable manner. Interconnection networks should be designed for continuous operation in the presence of a limited number of faults. More important, as technology scales, manufacturing defects will increase, thus demanding an efficient treatment.

- *Simplicity.* Not only for the sake of cost, but making simpler designs leads to the implementation of architectures that work with higher working frequencies, thus, increasing the system performance, and occupying less area. In fact, the silicon area usage is a critical aspect in on-chip networks. Indeed, reducing the area, translates into the opportunity for making room for more devices inside the chip.

- *Power consumption.* One of the most important aspects in networks-on-chip, not critical in other network environments, is the reduction or minimization of power consumption. Indeed, effective power-aware techniques are needed to bring better management of the total power consumed by the processing cores.

All these previous factors must be specifically considered when designing an on-chip network. In this thesis all the contributions take most of these factors as a reference. In the next section we present the basics for interconnection networks.

2.2 Interconnection Network Basics

The network architecture design is the result of several design choices like network *topology*, *switching and flow control* techniques and *routing strategies*. The network topology defines the physical interconnection between nodes and other elements. The switching and flow control techniques define how and when the information is transmitted through the network resources. Finally, the routing strategies manage the different path choices of communication between the nodes.

There are some common elements that conform a network architecture. The first elements are the nodes. Nodes are the elements that communicate through the network and perform basically two main tasks: computation and/or storage. Nodes connect to other nodes through a network interface associated to a switch, depending on the topology of the network. A switch is the basic component that connects different devices. Links are used to connect the devices (network interfaces and switches) among them. Figure 2.1 shows

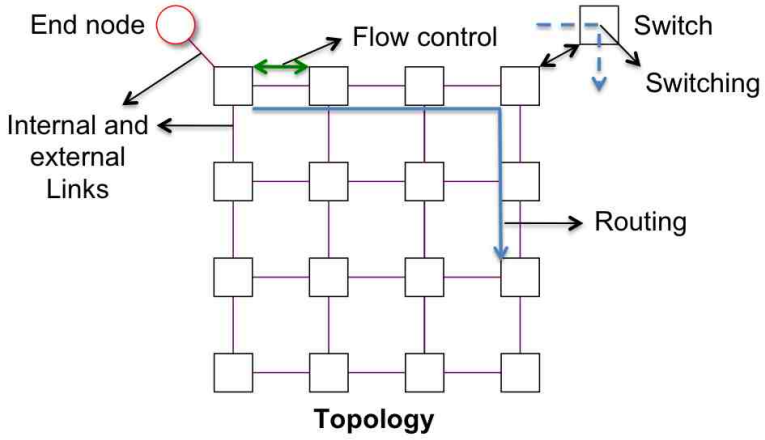


Figure 2.1: Routing, switching and flow control in a network.

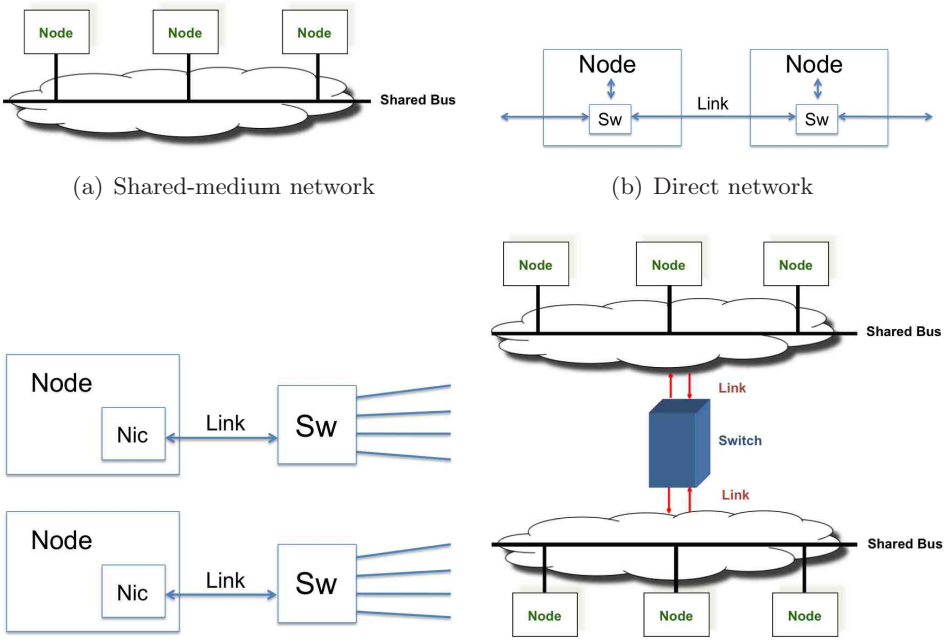
a network with a 2D-Mesh topology highlighting where switching, flow control and routing is performed. Next, we describe each network component.

2.2.1 Network Topology

Different network categories can be devised based on how all the elements of a system are connected to the network (see examples in Figure 2.2):

- *Shared-medium networks:* In this type of network there is a transmission medium that is shared by all the nodes, and only one node is able to begin communication at a time, the rest of nodes read (and monitor) the shared medium. Every device has the circuitry to handle addressing of other nodes and the data management. In these networks, the routing device is the shared medium, called also bus. Buses have limited bandwidth, so they suffer from scalability problems, as the number of connected nodes increases.

- *Direct networks:* Each node has a routing device attached, called switch, which is the component that establishes the connection to other nodes through point-to-point links. Each node is directly connected usually to a small subset of nodes in the network. The concept of network interface is weak in this type of networks as the end node and the switch (also



(a) Shared-medium network (b) Direct network

(c) Indirect network (d) Hybrid network

Figure 2.2: Network architectures.

called router) are tightly connected. Nodes are connected according to a certain interconnection pattern (topology).

- *Indirect networks:* Instead of directly connecting the nodes through point-to-point links, the communication between a pair of nodes can be performed by intermediate stand-alone switches. Every node has a network interface that connects to a switch (through a point-to-point link) and switches are connected between them (also through point-to-point links).

- *Hybrid networks:* This type of networks is a mixture of the previous approaches. In general, they combine mechanisms from shared-medium-networks and direct or indirect networks.

Although there are very subtle differences between direct and indirect networks, the functionality is similar in many aspects. An indirect network in which every switch is connected to a single node is equivalent to a direct net-

work. Also, terms router and switch, although having different meanings, are used with no distinction by the community, so both terms for the routing devices are interchangeable. In the rest of the dissertation, unless noted, the term switch is assumed.

There are also some common aspects to all these types of networks. Although links are usually formed by two communication channels, one in each direction, one of the basic aspects of a network is how communication channels are arranged. Network performance significantly differs if links are bidirectional or unidirectional. This choice impacts directly on the routing techniques and algorithms and associated issues, like deadlock avoidance. We assume the use of bidirectional channels on every link, though.

Each type of network can also be categorized with different properties:

- *Switch degree:* This property refers to the number of channels that connect a switch to its neighbours.
- *Diameter:* Is the maximum distance between a pair of end nodes in the network.
- *Regularity:* A network is defined as regular when all the switches have the same degree.
- *Bisection Bandwidth:* Bisection of the network encompasses the minimum set of links that split the network in two equal halves. Bisection bandwidth is the resulting bandwidth at the bisection.
- *Homogeneity:* A network is homogeneous if every node is equal in all aspects to the rest of nodes.

There are three common basic topologies used in interconnection networks. The first one is the *crossbar*. A crossbar (see Figure 2.3) allows the connection from any node to any other node simultaneously at the same time other connections are established (as long as the requested input and output are free). Crossbar networks, typically, are used for high-performance computing multiprocessor solutions and in the design for switches in direct networks. The drawback with crossbar topologies is that they do not scale well as system grows due to the quadratic requirement of connections.

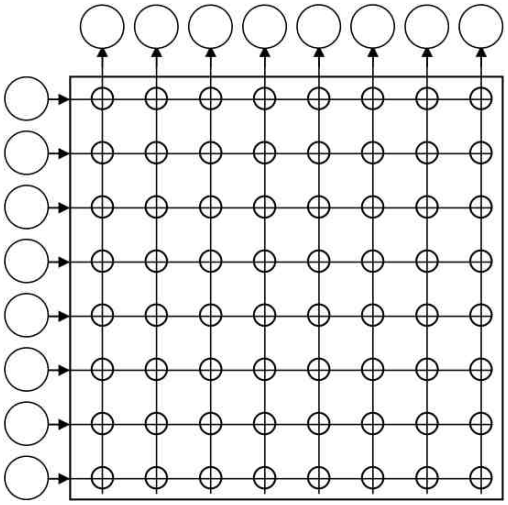
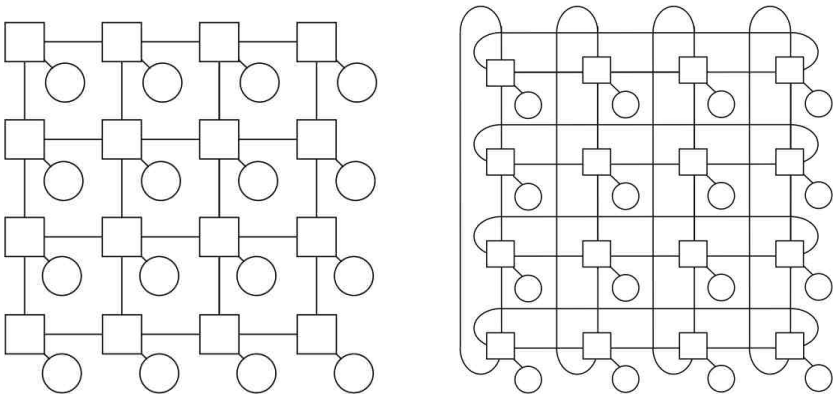


Figure 2.3: A crossbar network.

Strictly orthogonal topologies are the second common type. In this kind of networks we can find the *n-dimensional meshes and tori* (see Figure 2.4). A n-dimensional mesh or torus has k nodes placed along each dimension. A mesh differs from a torus because it does not have the wraparound channels that connect the nodes at the borders of the topology. Note that the torus topology duplicates the bisection bandwidth of the mesh topology and reduces the diameter. These topologies are the most common example of direct networks.

Multistage interconnection networks (MINs) are topologies driven by the concept of indirect networks as seen in Figure 2.5. Between input and output devices there are several switch stages. Examples of these networks are Clos networks, tree based networks or the butterfly network [7]. The arrangement of stages and the connection patterns determine the routing in these networks. MINs have been heavily used to interconnect parallel computers with large number of processors in commercial and high-performance solutions. However, for on-chip networks mapping of such topology pattern in the 2-dimensional surface of the chip is a big challenge [23].



(a) Mesh (b) Torus

Figure 2.4: A 4×4 2-dimensional mesh and torus.

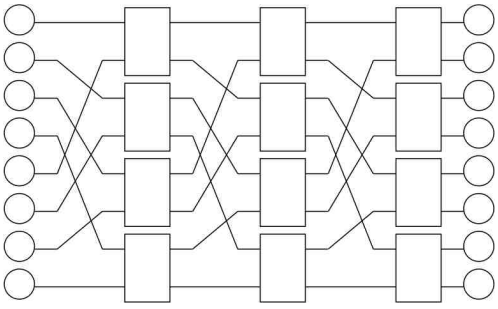


Figure 2.5: A multistage network topology.

Networks-on-chip Topologies

Earlier on-chip communication architectures fall on the share-medium network paradigm, that included buses as the communication subsystem. Examples of this kind of architecture is the Cell processor [34], that is well suited for a small number of processing elements (or nodes). But the trend nowadays in the industry of high-performance computing is to include a reasonably large number of processing cores inside the chip, and shared-medium network designs have poor scalability and bandwidth impacting heavily on the network performance.

Networks-on-chip emerged, thus, as a response to effective on-chip communication. On-chip networks are based on a paradigm that is a mixture of

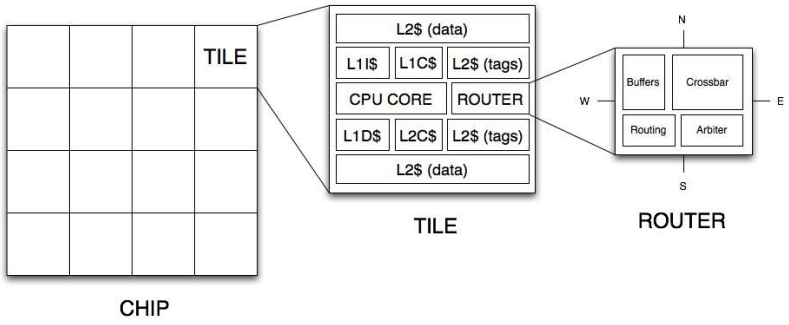


Figure 2.6: The processing element in a tile-based CMP.

the concept of direct and indirect networks. Current multicore architectures are composed as a wall made of elemental brick nodes that work together to achieve the high-performance computing goal, i.e, the chip is formed by several processing devices. These devices are called usually *tiles*. A tile, fundamentally, apart from the processing elements, has also a switch attached that handles the communication between tiles. See a simplified schematic of a tile in Figure 2.6.

As the chip can be seen as a collection of tiles, there is a major taxonomy where chips can be differentiated between homogeneous (inducing regular topologies) and heterogeneous designs (more suited with irregular topologies). Every tile is connected to a subset of other tiles through an on-chip network. An example of homogeneous configurations are the tiled chip multiprocessors (CMPs) where all the tiles are equal, i.e, tiles are replicated along the chip (see Figure 2.6). Instead, high-end multiprocessor systems-on-chip (MPSoCs) are an example of heterogeneous designs where components are different in many aspects: size, functionality, performance, throughput, etc. In this thesis, we focus on CMP systems with regular structures.

A popular choice in NoC design is the use of orthogonal topologies as most of the direct network architectures are implemented with this property in mind. Orthogonal topologies, which are associated with regular patterns, allocate the nodes in a n-dimensional space, with k nodes along each dimension. Every switch has at least one link crossing each dimension, and is labelled with an identifier depending on the coordinates. All links that communicate to other switches are bidirectional (formed by two channels, one in each di-

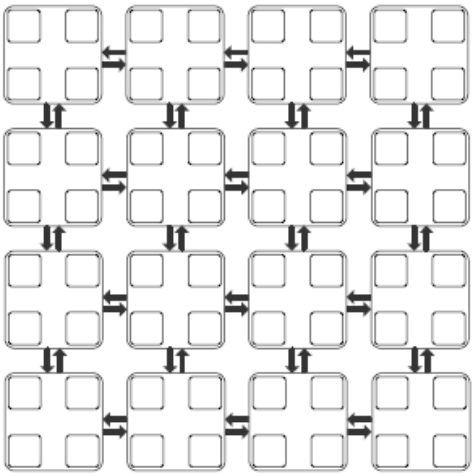
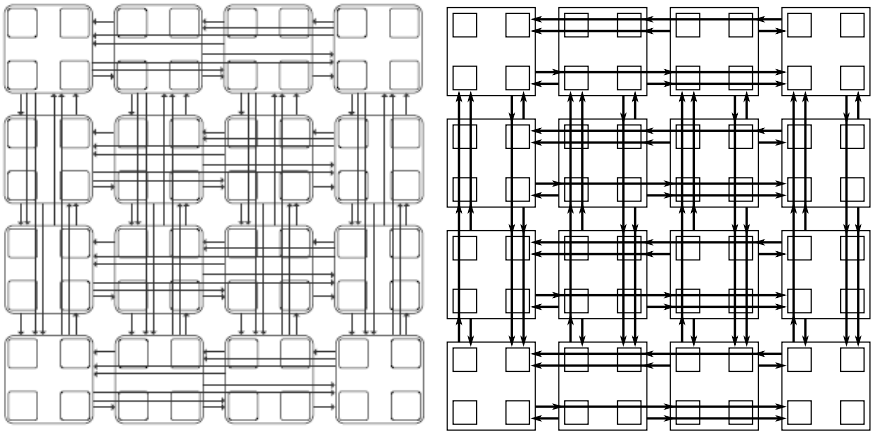


Figure 2.7: A 8×8 concentrated Mesh.

rection). As the distance between a pair of switches is the sum of the offsets in all dimensions, the routing strategy is usually implemented as a function of selecting the links that decrement the absolute value of the coordinate offsets between the current node and a destination node, a very simple mechanism. The most popular design in NoCs is the n-dimensional mesh, used in most of the commercial and non-commercial (prototypes) NoC designs. The most suitable topology is the 2-dimensional mesh (Figure 2.4(a)). This kind of topology is vastly used (or at least assumed) because it fits the chip layout.

However, a 2D mesh network does not scale in performance as its bisection bandwidth increases linearly with the square of the number of nodes. To overcome this issue, more complex topologies has been proposed to implement a NoC. In this dissertation three solutions are mentioned: the concentrated mesh or C-mesh (see Figure 2.7), the flattened butterfly (see Figure 2.8), and the fat-tree (see Figure 2.9). The C-mesh network [26] is an evolution of a 2D mesh. Basically, the C-mesh *concentrates* a fixed number of cores to a switch (see Figure 2.7). By concentrating cores to a switch, hop count is reduced. In contrast, switch radix, and hence, traffic concentration is increased. In this sense, lower network latencies are obtained at the expense of reduced network throughput.

To solve C-mesh drawbacks, other high-radix topologies that use *concen-*



(a) Flattened butterfly. (b) Multidrop express channel.

Figure 2.8: A 8×8 flattened butterfly and multidrop express channel.

tration have been proposed [21]. The flattened butterfly is an evolution of a butterfly network. Basically, the flattened butterfly consist in *flattening* a common butterfly network, that is, converting a multistage network into an orthogonal network (see Figure 2.8(a)). A flattened butterfly has the same bisection bandwidth than the butterfly network but reduces the hop count [7]. As the C-mesh, the switch radix increases. Additionally, the flattened butterfly presents another concern as the number and length of links increase, and hence, power consumption. To solve wiring issues, Multidrop Express Channels are proposed in [28] (see Figure 2.8(b)). Basically, all the links along a row or column are condensed into a single unidirectional bus, one in each direction. By condensing links in a bus, link rate utilization is optimized and power consumption reduced.

Finally, tree-based networks are feasible if some simplifications are made [24, 25], creating a fat-tree topology. A fat-tree topology (see Figure 2.9) is an indirect topology where the bandwidth is doubled at each upper layer, achieving full bisection bandwidth. However, as before, the switch radix is increased, and hence, the physical implementation becomes complex, or alternatively, the number of switches increases dramatically.

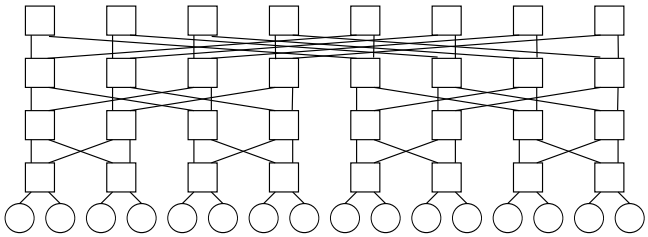


Figure 2.9: A 4×4 fat-tree topology.

2.2.2 The Switch

As aforementioned, each tile is composed of several elements. The switch is in charge of the communication between the input ports of the switch and its output ports. Typically, a switch architecture is structured by the following general parts (Figure 2.10):

- *Buffers:* Buffers are a key component and its design and position inside the switch affect other aspects of the switch design. The task of a buffer is to store temporarily units of information (typically called flits). Buffers are typically associated to the channels of the switch. Channels, also called ports, are divided into input ports, that receive messages, and output ports, that send the flits to other switches or nodes. Note that, to save area and power, buffers at the output ports are usually not implemented.

- *Crossbar:* The crossbar is the switching element and is non-blocking. Crossbars allow the connection between all inputs of the switch to all the outputs. Crossbars are classified by their radix, i.e. the maximum numbers of connections they can make.

- *Routing unit:* This unit is the responsible for decoding the unit of information provided by the incoming message, and based on the routing function and destination of the message, computes the most suitable output ports for transmitting the message.

- *Arbiter unit:* This unit takes its input from the routing unit and configures the crossbar accordingly to the requests between input and output

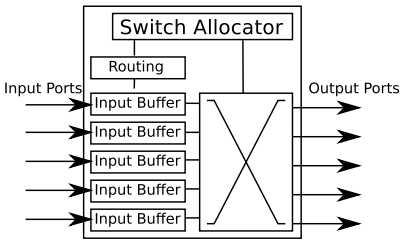


Figure 2.10: Canonical switch architecture.

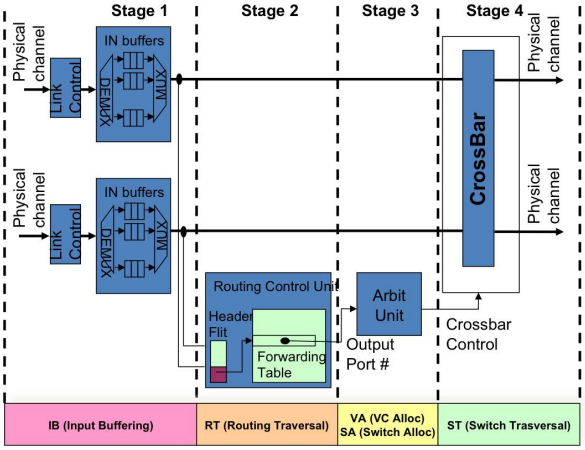


Figure 2.11: Switch stages.

ports, taking into account *switching* and *flow control* issues (both will be explained later).

Pipelining is a typical design method for high-performance switches. The different stages work in parallel with different data streams, thus providing parallelism and, thus, high throughput. A typical pipeline design of a switch can be seen in Figure 2.11 where four stages are shown (IB, RT, VA/SA, and ST). In this thesis we assume this pipeline design for the canonical switch. The baseline switch is described in the appendix of the dissertation.

2.2.3 Data Units

In an interconnection network, the typical unit of information between nodes is the *message* (see Figure 2.12). A message is a collection of bits that the

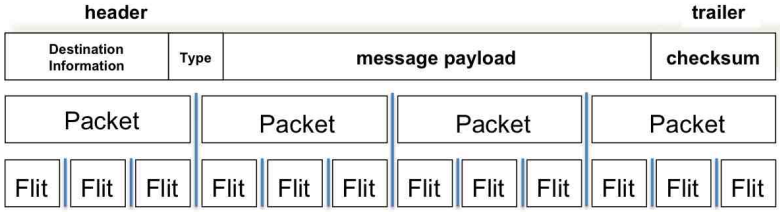


Figure 2.12: Data units.

sender wishes to transmit to a destination (or a set of destination nodes), i.e. it contains the data that must be transmitted. This information unit, however, due to resource restrictions affected by design choices, may need to be divided into smaller units, called *packets*, through a packetization process (usually performed at the network interface). A packetization of a message implies some reassembly and reordering at the destination node. A packet (or the message) is comprised of a header, which contains the information for routing and control, to be used by the routing devices, a body which contains the data, and optionally a tail, for flow control or arbitration purposes. Often, packet and message terms are interchangeable by the community, when both are equal in size. The term packet is usually employed even when the message has not been packetized. In this thesis we use the term message when wormhole switching is assumed (see next section) and the term packet when virtual cut-through switching is used (see also next section).

A packet is divided further into *flits* (flow control digits), which are the smallest unit of information that is flow controlled. As the width of the link can be lower than the size of a flit, the flit is further divided at the physical level, into *phits* (physical digits). It is left to the designer and the parameters involved, the size of every unit. However, in on-chip networks, due to the vast amount of bandwidth available, the phit size usually equals the flit size.

2.2.4 Switching

Switching techniques are responsible for the allocation of network resources to messages/packets inside the switches. Their basic function is to perform the setting of connections between buffers at the input and the output ports. The choice imposes several design constraints in the switch that impact per-

formance, fabrication costs and power consumption of the elements in the network. Next, we describe the main switching techniques used in on-chip networks.

Circuit Switching

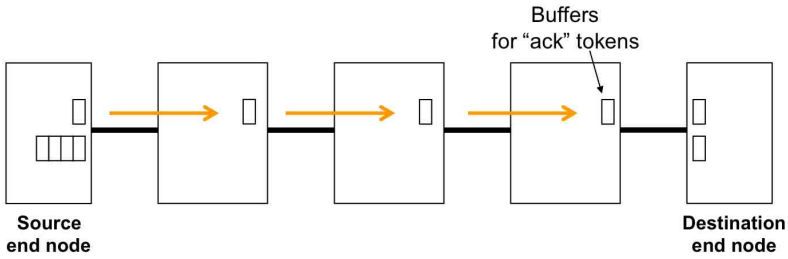
In circuit switching (Figure 2.13), the network establishes a reserved path between source and destination nodes prior to the transmission of the message. This is performed by injecting in the network a flit header, which contains the destination of the transmission. This header acts as some kind of routing probe that progresses towards the destination node reserving the channels that it gets. When the probe reaches its destination, a complete path between destination and source node has been set up due to the acknowledgement that is sent back to the source node. As the path has been reserved for this flow, messages cross the network avoiding buffer needs and collisions with other flows. The circuit is torn down when the transmission finishes. An example of a circuit switching-based on-chip network is described in [35].

Circuit switching can be very advantageous when messages are very frequent and long. Nevertheless, this switching technique has several important drawbacks. If circuit set up time is long compared to transmission time of the data, it will strongly penalize the performance of the network since links will be poorly used. Additionally, as channels are reserved for a given flow, no other flows can use them even if the connection is idle, thus channels may become even more underutilized.

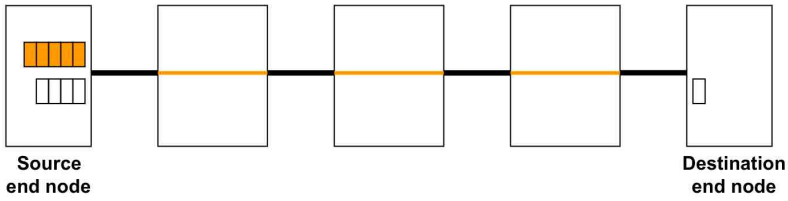
Store and Forward

Instead of reserving all the path for a certain flow, there are some techniques that operate at message/packet granularity. These techniques are referred as packet switching. The most basic technique related to packet switching is *store and forward* (SAF). When a packet arrives to a switch, the switch waits to store the whole packet in its input port buffer before the packet is forwarded. So, input port buffers must be large enough to store a packet (see Figure 2.14).

As can be deduced SAF has longer buffer requirements than circuit switching. In addition, latency of packets is multiplicative with hop count along the



(a) Request for circuit establishment



(b) Acknowledgement and circuit establishment

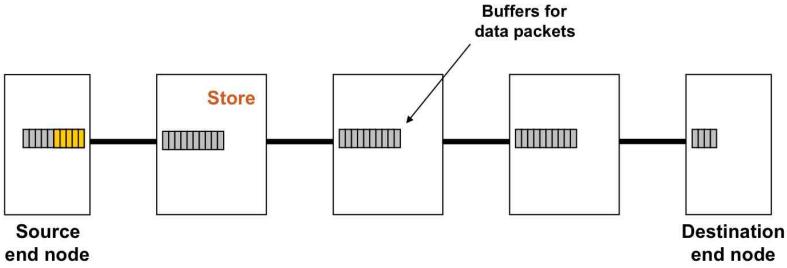
Figure 2.13: Circuit switching.

path (as the forward operation waits for the completion of the store operation).

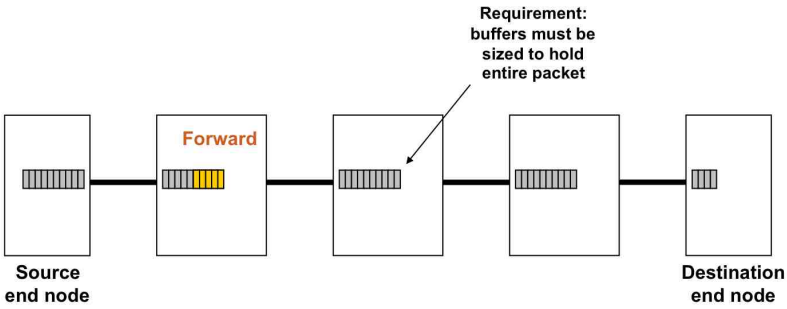
Virtual Cut-Through Switching

SAF switching is based on completely receiving a packet before any routing decision is made. But, this is not a very practical decision, since the packet header contains all the required information to perform the routing, and it is physically located at the beginning of the packet (typically in the first flit). So, the routing process can be started as soon as the packet header arrives to the input buffer, without waiting for the rest of the packet. Thus, the packet can be forwarded provided the selected output port chosen by the routing strategy is free. This is what is done in *virtual cut-through* (VCT) switching (Figure 2.15).

In this case, as packets can advance through the switches of the network once the packet header has arrived to each buffer (and has been decoded), the base latency for this switching technique is mostly additive to the distance between the nodes (hop count). Despite this, buffer requirements are the same for VCT and SAF. VCT requires there is enough free buffer space to store the



(a) Store phase



(b) Forward phase

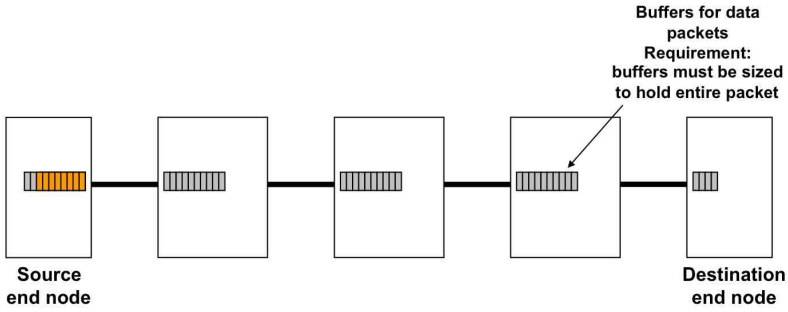
Figure 2.14: Store and forward switching.

entire packet. In fact, VCT behaves like SAF when the output link is busy. The switch needs to completely allocate the entire packet. This is the switching technique commonly used in off-chip high-performance interconnects [17, 33] as buffer size is not as critical as in NoCs.

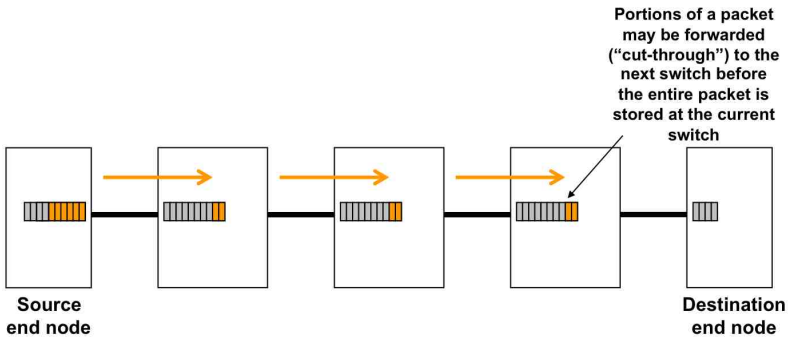
Wormhole Switching

VCT switching is an improvement over SAF, but in some network architectures, the choice of a buffer size to hold an entire packet could be critical. The requirement to completely store a packet in the buffer of a switch may prevent to design a small, compact, and fast switch [33]. In wormhole switching (WH) buffers at the ports of a switch only have to provide enough space to store only few flits, depending on the round-trip time delay (RTT) [1], instead of the

[1]Round-trip time delay can be defined as the elapsed time between a unit of information is sent and the acknowledgement of that transmission is received.



(a) Packet stored in the source node



(b) Portions of packet being forwarded

Figure 2.15: Virtual cut-through switching.

whole message. In WH switching (Figure 2.16), the message is forwarded immediately before the rest of the message is entirely received, but as opposed to VCT, there is no need to have enough space for the rest of the message in case the message blocks. In that case, the entire message remains stored through the buffers of several switches. The major advantage of WH switching is the low storage requirements at switches. However, the most important drawback is that WH switching could lead to high contention levels at the network, because a message may block several resources when is traversing the network, causing low utilization of links and buffers.

Virtual Channels

To overcome the problem of contention induced by wormhole switching, *virtual channels* [36] were proposed. Buffers basically are operated as FIFO (First-in,

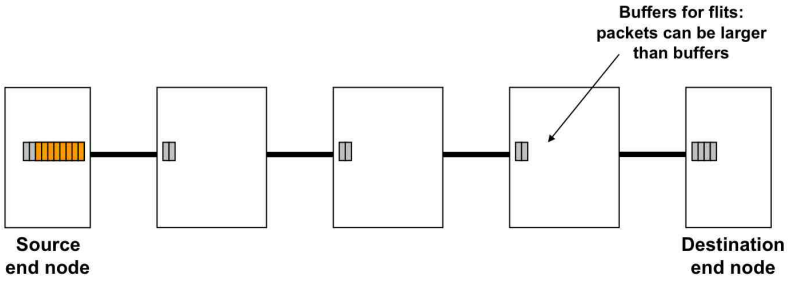


Figure 2.16: Wormhole switching.

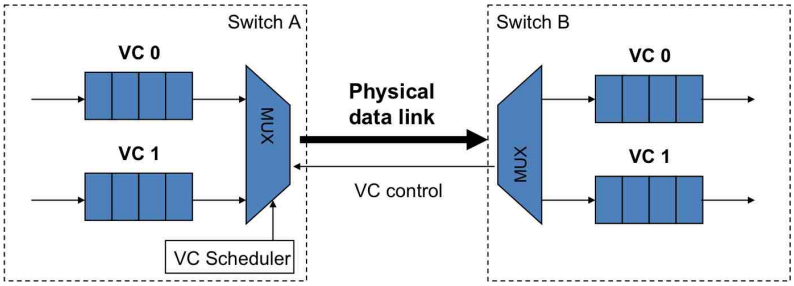


Figure 2.17: Virtual channels.

First-out) queues. Therefore, if a message reserves the channel but due to the saturation of the network it remains blocked at the current switch, no other message behind this message can use the physical channel even if its requested output port is available. This problem is known as *head-of-line blocking*.

When using virtual channels, the buffer at the input port is divided into different virtual buffers and the channel is shared by all the virtual buffers (see Figure 2.17). Of course this virtual multiplexing requires some local arbitration and must be taken into account by flow control and switching techniques. Virtual channels can be used to improve message latency and network throughput. Their major drawback is that the available link bandwidth is distributed over all the virtual channels sharing a physical link, resulting in lower speeds. Again, in the on-chip network domain, the designer must evaluate the trade-off and the impact overhead on the network. Virtual channels are not restricted to wormhole switching, the concept can be extrapolated to other design choices, depending on the need of their functionality (examples are deadlock-free routing algorithms and quality-of-service protocols).

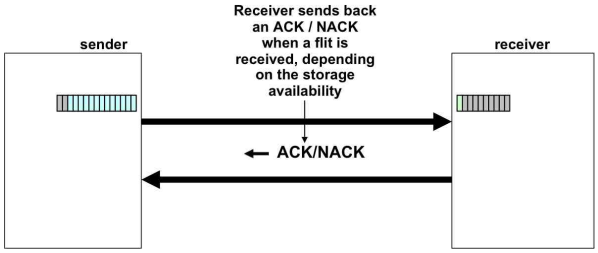


Figure 2.18: Ack/nack flow control.

2.2.5 Flow Control

Transmission of a flit between the input and output ports in a switch is a task performed by the switching technique. *Flow control*, however, is in charge of administering the advance of information between switches. Buffers are a temporary resource where to store flits, but they are finite. Flow control techniques are in charge of determining when the flits can be forwarded evaluating the capacity of the buffers and the link bandwidth.

There are three flow control mechanisms that are commonly used: *ack/nack*, *stop & go* and *credit-based*. The ack/nack flow control mechanism is based on data acknowledgements. When a flit arrives to a buffer, if the buffer has space available, then the flit is accepted and an acknowledgement signal (ack) is sent back. Instead, if there is no space available, the flit is dropped and a negative acknowledgement is sent. The flit must be retained at its origin until it receives a positive acknowledgement.

Stop & go emerged as an alternative to reduce the signalling (control traffic) between the sender and the receiver. Stop & go flow control is based on every buffer having two thresholds corresponding to certain sizes computed from the round-trip time. When the space occupied in the buffer reaches the *stop* threshold, a signal is sent back to the sender precisely to stop the transmission, taking into the account that still remains enough buffer space for the flits that are still being transmitted by the sender. When the buffer occupancy diminishes under or equal to the second threshold, *go*, then another signal is sent to reactivate the flow of flits.

With *credit-based* flow control, each sender, at its end of the link, maintains a count of credits, which is equal to the number of flits that can still be

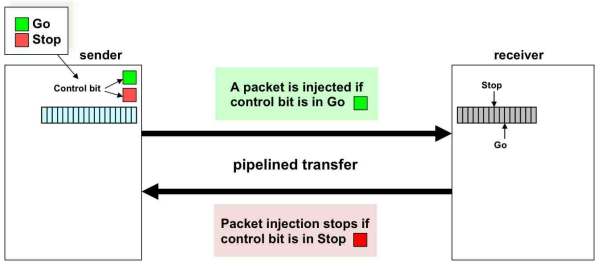


Figure 2.19: Stop & go flow control.

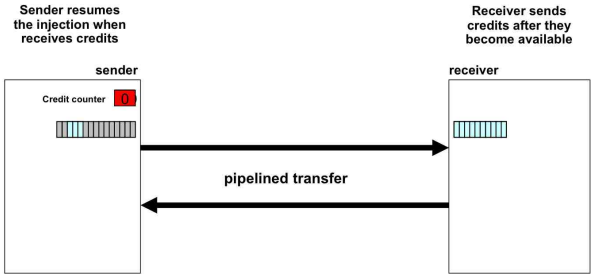


Figure 2.20: Credit-based flow control.

stored at the buffer on the receiver side. Whenever a flit is forwarded to the receiver buffer, as it occupies a slot, then the counter is decremented. If the counter reaches zero, it means that there is no available buffer space at the other end, and no flit can be forwarded. On the other hand, whenever a flit is forwarded and frees the associated buffer space, a credit is sent back to increment the counter. The drawback of this flow control mechanism is the significant amount of credit signalling sent backwards, which could impact on network performance.

2.2.6 Arbitration

A switch is composed of multiple input and output ports with their associated buffers and channels. Multiple inputs, according to routing decisions, may request the same output port. In this scenario, an arbitration operation is required to decide which one of the requests is allowed to connect to the output port. The arbitration mechanism must ensure the output to only one of the inputs that have requested it, and the others must wait until they are granted.

As the arbitration operation introduces a latency to determine the assignment of the different output ports, it is critical for a network-on-chip environment that these operations are performed fast enough to keep low latencies.

The main goal of an arbitration mechanism is to provide fairness between all the ports while achieving maximal matchings between requests and resources. Although there are many proposals for arbitration algorithms and implementations, we can distinguish two general arbitration techniques that differentiate on how to assign priorities between the requestors.

The first one is *fixed priority*. An arbiter with fixed priorities assigns the requests in an established order to the different input ports. This order is determined by the static priority assigned to each input port. In this mechanism, the arbitration is simple, but introduces unfairness and potentially, starvation. If one of the input buffers with higher priority keeps requesting the associated output, the inputs with lower priority get blocked, even, inducing the chance that the inputs with low priority never get the request satisfied.

The second one is called *round-robin*. An arbiter that implements round-robin arbitration cycles priorities between all the input ports by assigning the lowest priority to the input port which request was last served. This arbitration technique introduces fairness between the requestors, but is more complex to implement. Usually, represents a trade-off between implementation cost and performance.

2.2.7 Routing

As we have described before, topology defines the physical organization of the network composed by the nodes. In fact, a given topology defines the available paths between all the nodes. The routing algorithm is the responsible of deciding which path has the message to follow to be effectively routed from its source to its destination. The choice of the routing algorithm becomes of outmost importance for performance reasons. Indeed, in the on-chip network domain not all solutions from the off-chip network domain are suitable due to environmental restrictions. The designer must find a trade-off between efficiency, flexibility and implementation cost of routing.

As this thesis does not delve into routing algorithms, we only hint the most used routing algorithm in NoCs. The DOR routing algorithm routes messages

in an established dimension order. For instance, in a 2D mesh topology, the routing algorithm sends messages through the X dimension until the offset in that dimension is cancelled, then the message is routed through the Y dimension. This algorithm is also known as XY routing algorithm. In this thesis we assume XY routing as the routing algorithm for 2D meshes.

2.3 Related Work

In this section, we describe those contributions that have motivated this dissertation. We focus on the state-of-the-art in three main topics. First, switch design proposals linked with the modular switch presented in Chapter 3. Second, research targeting high-radix topologies for NoCs. Finally, those works that challenge the NoC community by claiming large crossbar designs are feasible for NoC environments.

2.3.1 Switch Design

We can find in the literature, and in real implementations and prototypes, two different switch architectures. The first one is a single stage switch. In this architecture, a flit crosses the entire switch and reaches the input port of the next switch in one cycle, typically. This is the usual case for MPSoC systems [37]. On the other hand, we may find pipelined switches for high-end MPSoC and CMP systems. This is the case for the Intel Polaris chip [2].

During the last years many efforts have been done in order to reduce the latency of NoC switches for CMP systems. Initially, the knowledge and established techniques from the off-chip network domain (from high-performance interconnects) were applied to the emerging NoC field [38, 39]. First, NoC switches were designed using well-known routing algorithms (e.g. DOR routing) and switching techniques (e.g. wormhole switching). Also, tied with wormhole, virtual channels were advocated in order to time multiplex the physical channel and, therefore, reduce the blocking induced by wormhole switching. All these techniques forced the switch to become complex and slow. Different efforts have been made to reduce the complexity and latency. For example, in [13] different techniques are applied and a one-cycle switch

is achieved, or in [14] the output port is predicted rather than computed in order to minimize latency.

In order to reduce the complexity and latency of the switch, it is possible to exploit the network topology to optimize the switch design. In [40] a low latency switch that supports adaptivity is presented. Its main characteristic is that it exploits the properties of a 2D-mesh in order to perform adaptivity. Another similar proposal, presented in [41], simplifies the switch by exploiting the properties of the ring topology. The same author evolved the proposal into a low latency switch for a mesh network [42]. In [43] special features are added to simplify the switch design in a virtual cut-through environment. Finally, the Rotary Router is presented in [44]. This switch improves performance by decomposing a switch structure into two independent rings structures. Again, by decomposing a complex design into a sum of smaller and simpler ones, performance can be improved.

Additionally, it is possible to reduce the complexity and latency of the switch, by exploiting the area-performance trade-off provided by the logic synthesis principle [45] that performs an area-performance trade-off. When delay constraints are loose, area-efficient netlists can be achieved, but when more tight delay constraints are needed, high performance can be obtained at the cost of area. Then, by minimizing the complexity of an arbiter we can relax delay constraints thus achieving higher performance. Then, it is possible to split a task into several smaller tasks that are independent, and hence, they can perform in parallel. Assuming that principle, in [38], the authors remark the need of reducing the complexity of the crossbar from previous works [46], arguing that smaller switch modules achieve faster switches. Similarly, Gilabert et al. [47] propose a decoupled crossbar for each virtual channel rather than a shared crossbar for all the virtual channels. Decoupling resources in a switch relaxes delay constraints thus improving area and power consumption. Kim et al. [48] proposed an efficient microarchitecture for the implementation of a high-radix switch. This proposal is based on the implementation of a decoupled fully buffered crossbar that enables scalable performance with the number of ports. This work, however, is not directly suitable for an on-chip network scenario, where technology constraints impose new limitations, specially on the maximum operating frequency of the design [25].

It is possible to apply decoupling techniques to any part of the switch. In fact, it is possible to replace virtual channels by physical parallel ports. This has been inherited from the off-chip domain [49]. Carara et al. [50] reuse this concept for NoCs. In particular, they take advantage of the abundance of wires in current and expected deep sub-micron technologies. Carara et al. based their work in *spatial division multiplexing* (SDM) introduced by Leroy et al. [51] and Lane Division Multiplexing (LDM) technique introduced by Wolkotte et al. [52]. However, despite the critical path is reduced resources are increased which is the main problem of its decoupling technique.

Jongman Kim et al. reuse the decoupling concept to design a low-latency switch [53, 54]. In [53], the Row-Column (RoCo) switch is presented. This switch achieves low latency decomposing a 2D switch into two halves, one in each dimension. That is, incoming messages from one dimension are handled independently from incoming messages in the other dimension. Additional logic is used to handle messages from one dimension to the other. Decomposing the switch into dimension slices is extended to the 3D scenario [54], where the Dimensionally-Decomposed (DimDe) switch is presented. This switch design optimization was previously considered in [55], where the adaptive Chaos Router is presented. Chaos switch decoupled X direction from Y direction. In [56] a hierarchical optimal implementation (Partitioned Crossbar Input Queued) of a crossbar is presented. In this paper it is shown that a partitioned crossbar is able to obtain higher switch efficiency than a conventional crossbar design.

Buffering management determines the behaviour of a switch. Initial switch designs rely on input buffer queues that store incoming messages before the switch manages them. However, Input Queued switches [57] present congestion problems that tend to reduce overall network performance. Furthermore, these kind of switches are exposed to the Head-of-Line (HOL) blocking problem. A simple solution to remove that risk to HOL blocking is to introduce the Virtual Output Queueing technique [58, 59] which maps flits addressed to different output ports into different queues, that is, using one dedicated queue per output port. The problem of this solution is that buffering resources should be hugely increased, making this solution unfeasible for high-performance networks. Other similar solutions as Combined Input Output Queued switches,

place buffers at both sides of the switch, that is at the input side and the output side. This kind of solution to be efficient, requires a complex circuitry to speed-up the switch in order to take the most from buffer at both sides. A more revolutionary solution is to integrate buffering and crossbar in the same structure, creating a buffered crossbar. This novel idea places memories or registers in the crosspoints of the crossbar. However, this technique leads to a complex switch being unaffordable in an NoC environment [60]. In contrast, buffered crossbars have been redesigned in order to become an optimum solution for high-radix switches or large crossbar-based NoC. Kim et al. proposed a buffered crossbar for high-radix switches [61]. In this paper, it is confirmed that buffering a crossbar improves overall network performance as it was previously stated in [30]. However, the switch design environment is different than the network design presented in this dissertation. Meanwhile, in this dissertation we focus on NoCs, in [61] authors design each high-radix switch on a chip, where multiple chips are connected to develop a multicore system.

In this thesis we start decomposing a switch in basic and equal modules and then distributing them over the chip area. As important as the proper placement and interconnection of these modules, is how they communicate. In this sense, our proposal introduces a communication protocol between logic modules that is similar to that presented in [62]. In this work, the Ambric register is presented. In addition of a conventional clock signal that charges/discharges the register, two control signals – valid and accept signals – are added setting communication between registers, modelling an asynchronous communication protocol reinforced by the clock signal. Similarly, Jacobson et al. [63] use the same concept to design a complete synchronous pipeline – controlled by a clock signal – where data transfer is fixed by an asynchronous signal. These works recall the elastic concept in digital design [64]. Elasticity refers to the flexibility of a system to adapt to the variability of delays. In this sense, any synchronous design managed by asynchronous signals that adapt data transfer rate to the pipelined design can be considered a elastic circuit. In this sense, it is possible to adapt any design to elasticity [65, 66].

2.3.2 High-Performance NoC Topologies

The design space of complex network topologies has been widely explored and built in the domain of off-chip interconnection network. In fact, some networks used were adopted from the well-know multistage networks used in telephony, as for example Clos network [67]. However, the severe constraints imposed by the new on-chip environment – low latency and power consumption requirements – keep NoC designers from using those complex topologies. Recently however, some papers demonstrate that it is feasible to introduce complex network topologies to NoC environment.

In [26], Balfour and Dally proposed the use of concentrated mesh topologies in combination with express channels for large networks. *Concentration* means connecting several nodes to the same switch. By using a concentrated mesh, network interface complexity is slightly increased since is able to receive messages from different nodes. On the other hand, the number of switches, and hence, hop count is reduced, reducing network latency. In contrast, link length is increased as the switches are physically moved further away. thus, in [26] researchers show that improvements in area and energy efficiency can be obtained when using a *concentrated mesh* over conventional 2D meshes. To reduce power consumption of complex networks, an on-off technique is used in [68]. In this paper, a *Parallel Concentrated mesh* (PC-mesh) is presented. The PC-mesh is just several concentrated meshes working in parallel. To optimize power consumption, a proper algorithm turns off those parallel concentrated meshes that are not required. Then, by properly using different concentrated meshes working in parallel is possible to improve latency without increasing power consumption.

Another solution to overcome performance scalability of 2D meshes is the use of high-radix switches, that is, network topologies where each switch is connected to its neighbouring switches –as the 2D mesh – and extra switches placed in the same row or column. This kind of solutions requires the use of long links – express channels – to connect switches that are placed at two hops of distance. In [69], extra long links connecting 2-hop away switches are used in a 2D mesh baseline topology. Increasing the switch connectivity and reducing the hop count, network latency is reduced, and throughput is increased at the expense of an increment in power consumption. In the same way, Kim

et al. [21, 22] proposed the use of a flattened butterfly to solve the scalability limitations of low-radix topologies. The flattened butterfly network connects a single switch with all the switches placed in the same row and column.

Some papers, redefined the NoC scenario and introduce some complex network topologies that are far away from the conventional 2D mesh. Networks as the fat-tree [70] or the multistage Clos network [67] are considered. In [25], Ludovici et al. analyzed the feasibility of the on-chip fat-tree implementation. Fat-trees are indirect topologies based on complete trees. In [25], the authors show how technology constraints impose severe limitations to the practical utilization of tree-based high-radix topologies. In [24], the authors go beyond the previous simplification, and reduce the fat-tree complexity by forcing a deterministic routing algorithm, developing the Reduced Unified Fat-Tree (RUFT). Results show that such simplification allows RUFT to achieve lower latency than an original fat-tree for low and medium traffic loads.

However, as stated in [27, 28], all these high-dimensional on-chip networks designs suffer from traffic concentration, that causes at the end a reduction in the maximum network throughput. In [23, 71] a high-radix Clos network-on-chip is implemented. In this paper, authors stated that a proper placement of Clos network switches should be done to obtain good performance. Thus, a proper floorplan algorithm is required to minimize power consumption of long interconnects. Results show that, an implemented Clos network obtains better results than conventional network topologies, reducing network latency and increasing network throughput. Grot et al. target the flattened butterfly implementation drawbacks by introducing a Multidrop Express Channel network [28] where a high degree of inter-node connectivity provides a lower latency and bandwidth is handled efficiently. In this paper, each switch is connected to a group of switches using a unidirectional bus – multidrop express channel. The use of Multidrop Express Channels reduces power consumption of conventional express channels. Alternatively, in [27] a different approach is presented. In this work, a hierarchical network-on-chip that suits the particular requirements of CMP traffic demands is proposed. The main advantage of this hierarchical on-chip network is its low latency specially when considering high traffic locality. As a drawback, the network throughput provided by this hierarchical network, when no local traffic distribution is considered, is lower

than the one provided by a regular 2D mesh topology. Finally, a complete NoC topology comparison and analysis can be found in [7].

2.3.3 Long Link Issues

The implementation of complex topologies in a 2D silicon substrate requires the use of long interconnection links. Long links suffer from an excessive power consumption and large latency, that makes them costly or even unfeasible. To minimize the impact of long links, in [72–74] long links are replaced by elastic buffers, that is, buffered links where a trade-off between latency and power consumption is made, because a flit requires several hops to traverse an elastic buffer. In [75], authors focus on 3D technology as an ideal scenario to implant high-radix topologies because long 2D links evolve to short and power efficient 3D vias. Similarly, a 3D packaging is considered in [22] in order to implement the flattened butterfly.

Despite the link length, the trade-off between link delay and link power consumption is critical. In this sense, it is necessary to insert repeaters along the wire when needed [15]. This method reduces interconnect delay and signal transition times. In order to minimize the power consumption of a wire, it is necessary to insert the proper number of minimum sized repeaters [76] that satisfies the delay constraint imposed by the operating frequency of the switch. This technique is called power-optimal repeater insertion [76]. There exist other link power optimization techniques, that are not considered in this dissertation since they focus on the technological point of view which is not in the scope of this thesis.

As important as link delay and power consumption, is the area occupied by links. As the number of links increase due to the network complexity increment, the area of those links increases as well. It is possible to reduce the area occupancy of wires by using high metallization layers to route long interconnects. By using high metallization layers, it is possible to place other functional blocks as SRAM below the link interconnects, because the other functional blocks as SRAM require the use of the lowest metallization layers [77]. Using this technique, wires do not require dedicated area except for the repeaters inserted along the wires.

Crossbar-based NoC have been rejected as a valid solution for NoC design

due to the common believe that their unaffordable cost in terms of resources used, wires and crosspoints. Indeed, some papers reject crossbar-based NoCs due to its inefficient behaviour to unbalanced traffic [78]. But, in general, designers consider crossbar-based designs as a ideal network implementation unreachable due to its implementation infeasibility. For example, in [79] a one-cycle crossbar is considered as the maximum theoretic performance achievable. That is, a one-cycle crossbar based design is considered as the network baseline which the rest of solutions can be compared to.

In contrast, despite to initial reluctance, large crossbar-based designs are slowly introduced in the NoCs community. Crossbar cost can be reduced by using buffered crosspoints. That is, crossbar crosspoint are buffered, and hence, pipelined. Registering the crosspoint forces messages to require more than one cycle to traverse the network, on the other hand, it makes large crossbars feasible. Buffered crossbars appeared in the switch design optimization. Different buffered crossbars are presented in [32, 80–82]. In cite [80] buffering the crossbar allows the designer to achieve 100% throughput under uniform traffic and a simple round-robin arbitration scheme. By extrapolating this technique to the whole network a buffered crossbar-based chip is implemented in [83]. In this paper, low latency and high throughput is obtained. Using pipelined crossbar philosophy is the Mesh-of-Trees (MoT) presented in [84,85]. In these papers, conventional NoC is replaced by a mesh of buffered trees to obtain a low latency and high throughput network. Finally, in [1,31] an accurate and detailed physical implementation of a centralized crossbar is made. However, a centralized crossbar is a slow element that should handle several input requests. However, the main drawback of a centralized crossbar is that strongly depends on design constraints. That is, the results of proposal presented in [1, 31] are obtained for a fixed number of cores, where their size, shape, etc. are set by the designer. If some design constraint are modified, the whole design should be redesigned. In the next chapters we will go over the proposed designs that build the current thesis.

Notice that all these works (high-radix topologies, reducing link impact, and switch architectures) are well known. However, most of them do not take the floorplan as a central issue. And the works that consider the physical implementation, differ of our approach in the sense that we redesign the switch,

network, or crossbar (converting it into a modular design) in order to be easily distributed over the chip.

Chapter 3

Modular Switch

In this chapter we perform the first step towards a floorplan-aware NoC design. This first step focuses on redesigning the switch into modular basic blocks, each one independent of the others.

Basically, buffering, arbitration, and switch traversal tasks are mixed and partitioned into a tree of small, simple and identical modules called Arbitration-Crossbar (AC) module. An AC module is basically an M-to-1 crossbar with an output buffer, where M is the degree of the AC module. Additionally, arbitration and flow control mechanisms handle the crossbar.

This chapter is organized as follows. First, we provide a brief description of a canonical switch. Then, we describe the proposal of this chapter, the modular switch design. Finally, we provide an evaluation of the modular switch compared to the canonical switch organization.

Despite that the modular switch has a performance improvement over a canonical switch, the aim of the chapter is not to present an optimized switch but a switch that will be ready to be distributed along the links, enabling a floorplan-aware NoC design, which will be presented in the next chapter.

3.1 Canonical Switch

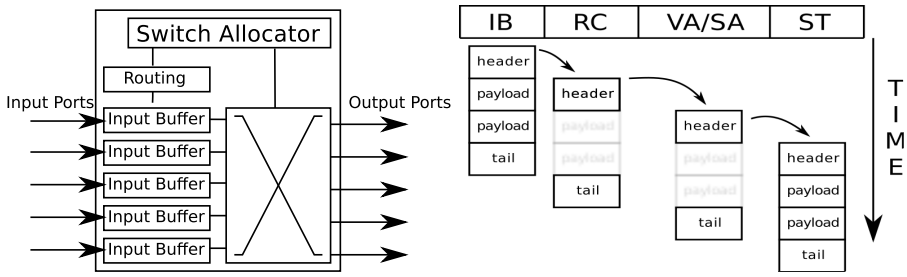
As commented in the introductory chapter, a NoC is built from basic components as switches, links, and network interfaces. Switches and end nodes are connected by links, thus forming the topology and final network structure.

Although the network interface must be carefully designed in order not to introduce bottlenecks, the complexity is usually shifted to the switch design. Indeed, many previous works have focused in different switch architectures. In CMPs the current trend is to design pipelined switch architectures with wormhole switching in order to increase clock frequency and reduce buffer requirements.

The basic pipelined wormhole switch design (see the Appendix A for a detailed description of the baseline canonical switch design) is shown in Figure 3.1(a). It performs four basic tasks to an incoming flit: buffering, routing, arbitration, and forwarding. In a conventional switch those tasks are pipelined in different stages: input buffer (IB), route computation (RC), switch allocator (SA), and switch traversal (ST). The IB stage is used to allocate an incoming flit from an input port into a queue. The RC stage is used to compute the output port the message has to take. This is usually achieved, in a 2D mesh topology, by using a small logic block implementing the DOR routing algorithm. Once the output port is computed, at the next cycle, in the SA stage the flit contends for the requested output port (with all the flits requesting the same output port). Finally, on success, the flit crosses the internal crossbar of the switch, thus reaching the output port. This is done in the ST stage. If the switch implements virtual channels, then an additional stage, named virtual channel allocation (VA) is required to arbitrate for the output virtual channel amongst the virtual channels of an input port. In parallel, the flow control mechanism communicates the local switch state to the neighbouring switches to prevent buffer overflows and optimize their utilization.

Figure 3.1(b) shows the pipeline stages of the switch. We can see how a header flit is handled in all the stages, whereas payload flits only visit the IB and ST stages. The tail flit is used to release the switch resources that have been occupied by the header flit.

In order to increase performance (higher throughput and/or lower latencies) several techniques have been proposed that can be applied to the canonical switch design. Some techniques focus on specific parts of the switch as routing [86], arbitration [17], or crossbar [47]. Some of these modify switch pipelining shown in Figure 3.1(b) by performing some stages in parallel. In contrast, other proposals modify the whole switch architecture [44, 50, 53]. In



(a) Canonical switch schematic. (b) Canonical switch pipeline.

Figure 3.1: Canonical switch schematic and pipeline.

general, switch optimization efforts lie on improving switch performance measured as throughput or latency at the expense of increasing the complexity of the switch. Our contributions redesign the switch in a different direction, taking as a design parameter modularity and simplicity. In this thesis we use as the baseline switch the one with four stages (IB, RT, VA/SA, X). Notice that the advanced switch solutions tend to add complexity to the switch, thus not being a good choice as a baseline for a modularized switch design.

3.2 Modular Switch

The modular switch is a pipelined buffered wormhole switch which is mainly made of a sum of independent modules called AC modules. Each AC module is able to store, arbitrate and forward its incoming flits. Figure 3.2 shows a modular switch consisting of five input/output ports. Each input port can reach any output port except the output port that goes in the opposite direction of the input port (U turns are not allowed). Also, the local port connecting the core to the switch can be reached from any input port. Thus, each output port works as a 4-to-1 switch (N is set to four). That is, each output port is able to buffer, route, and forward those packets that request that output port. This switch is suitable for a low-radix topology as the 2D mesh.

3.2.1 Output Port Controller

Figure 3.3 shows the block diagram of an output port controller. As it can be seen, the output port controller circuitry – and hence all the switch – is

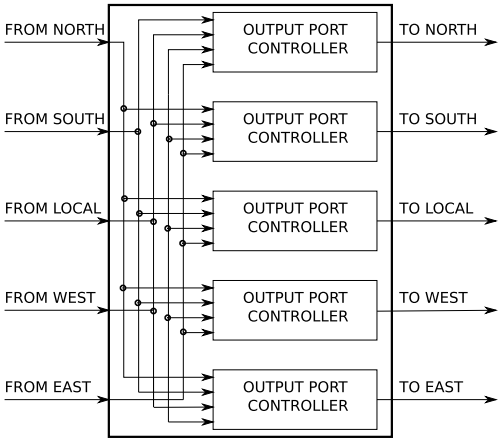


Figure 3.2: Modular switch schematic.

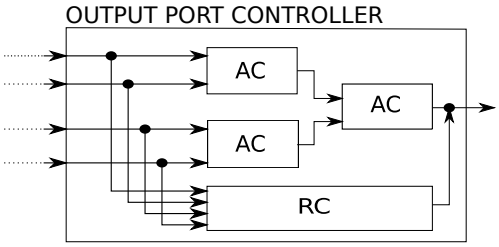


Figure 3.3: Output port controller schematic.

composed of two different modules: the Routing Computation (RC) module, and the Arbitration-Crossbar (AC) module. Thus, the modular switch is a pipelined buffered wormhole switch with $\log_M N$ stages, where M is the AC module degree and N is the number of input ports that can reach a given output port in the switch. In the example provided in Figure 3.3, M is set to two and N is set to four. An important characteristic of the switch is that each output port is managed independently. That is, each output port has its own circuitry (output port controller) which is not connected to the circuitry of the rest of output ports.

3.2.2 AC Module

The AC module, depicted in Figure 3.4, is the most important component in the switch. As each AC module is independent of the rest of the switch

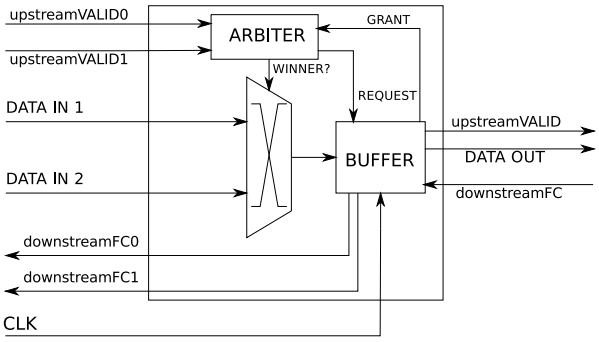
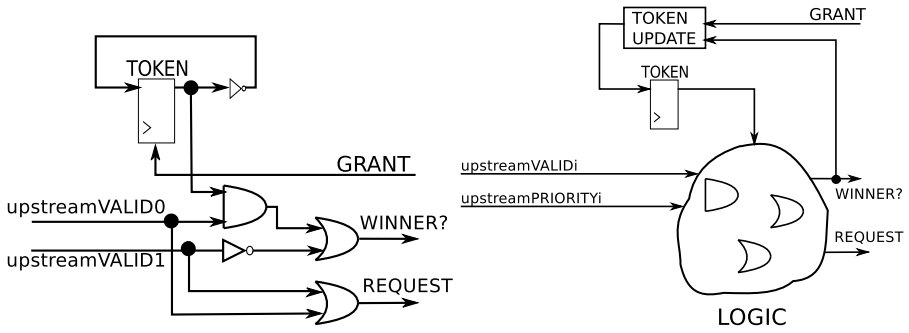


Figure 3.4: AC module of degree 2 schematic.

circuitry, the critical path of a single AC module sets the clock cycle time of the whole switch. It performs arbitration, switch traversal (crossbar), and buffering tasks. Transmission between AC modules is implemented through a customized flow control mechanism which administers buffering, which is controlled by the upstream AC module, and arbitration, which is coupled by the flow control signals from the downstream AC module (later described).

As it can be seen in Figure 3.4, the AC module has three main components: a multiplexer, an arbiter, and a buffer. The AC module is a simple M-to-one multiplexer with its output registered. The arbiter decides at every cycle which input is connected to the output of the AC module. The signal that fixes which input has granted the access to the buffer resources is *winner?*. The decision of which input crosses the AC module depends on local status – a token – and input signals from the upstream AC modules. Figure 3.5(a) shows a simple 2×1 round-robin arbiter. As it can be seen, the arbitration decision is determined by a token register amongst the inputs that request that AC module. A flow control signal – *upstreamVALIDi* – is associated to any link attached to an AC module. This signal is activated when the link has a valid flit. Only inputs with valid flits compete for the buffering resources. The arbiter has been designed to keep flits belonging to the same packet crossing the module consecutively – interleaving flits of different packets is not allowed. Additionally, one token registers the last input that accessed to local resources (buffer). The token register updates when a tail flit leaves the AC module, which is pointed when the *grant* signal gets active. The token fixes the priority



(a) Round-robin arbiter.

(b) Round-robin arbiter with traffic priority.

Figure 3.5: Round-robin arbiters.

of the inputs, when the token value changes, the input with the highest priority becomes the input with the lowest priority. Notice that, for higher AC radices the token register becomes a counter that fixes the highest priority input. The *grant* signal will increase the token counter by one. Finally, a *request* signal is created at the arbiter module. This signal is an OR of the incoming inputs. When the signal is activated, the buffer will store an incoming flit, if the upstream flow control allows this operation.

Thus, the arbiter shown in Figure 3.5(a) is a round-robin arbiter that provides a grant only to those inputs that request that AC module. Nonetheless, more complex arbitration policies can be applied, as for example, to provide some priority depending on traffic classes. To implement more complex arbitration policies, minor changes must be applied to the arbiter shown in Figure 3.5(a), just introducing extra upstream flow control signals and increase the complexity of the token counter update process. Figure 3.5(b) shows a generic arbiter schematic where some extra upstream flow control signals have been added – *upstreamPRIORITYi*. If active, the associated flit will have higher priority. However, in this thesis we do not address the priority support in AC modules.

The buffer is implemented by using a mux-in-first-out buffer [8]. Figure 3.6 shows an schematic of the buffer implemented. The output link attached to the AC module is connected to a fixed slot. That slot will be considered the master slot of the buffer, as if no contention exists, all flits will use only this

slot. Only when contention exists, flits will occupy the rest of slots that will be considered as slave slots. When contention disappears, flits stored on the slave slots leave the AC module one by one, by crossing the buffer slot by slot until they reach the master slot before leaving the AC module. Meanwhile the oldest stored flit is leaving the buffer, flits in the buffer move at the same time, thus, no bubbles are introduced neither when the communication flow is stalled nor when it is restarted. By using this structure for the buffer, the zero-load latency of the modular switch is one cycle per stage (AC module crossed), that is, when no contention exists, a flit needs $\log_M N$ cycles to cross the modular switch. As any AC module performs local decisions, at least one incoming flit will cross the AC module when no contention exists. Notice that, the minimum AC buffer size that fulfils round-trip-time constraints between AC modules is two (see Section 3.2.5 for a more detailed explanation). Then, by assuming buffer size of two flits (two slots), each input-output path of the modular switch is able to store $2\log_M N$ flits (two per stage).

Two signals control the buffer. The *request* signal created at the arbiter, and the downstream flow control – *downstreamFC* – signal that comes from the downstream AC module. The *request* signal means that an incoming flit wants to be stored. Downstream flow control signal tells the module if the downstream AC module is able to receive new flits. An AC module can receive flits if there is free space on its buffer. Additionally, *request* signal is used to enable the registers to store an incoming flit. This operation is allowed if there is an empty slot which is equivalent to the *write* pointer being active (signals to the enable input at the registers at Figure 3.6). The *write* pointer of the register measures the buffer occupancy. Finally, when buffer occupancy is reaching the maximum, it generates the flow control signals that would be sent to the upstream AC modules. Notice that, when assuming a two slot buffer, the *write* pointer logic gets reduced to a single register that determines if the master slot is available or not.

3.2.3 RC Module

The RC module performs the routing computation. Our switch is designed to perform a simple XY routing algorithm for the 2D mesh topology. However, other routing algorithms can be leveraged. Nevertheless, as deterministic

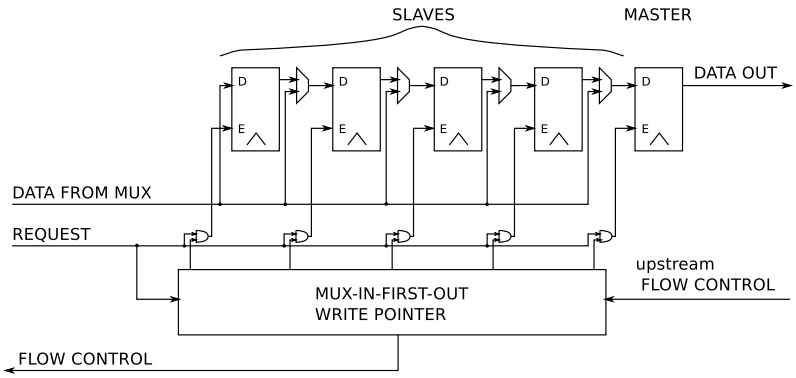


Figure 3.6: AC module buffer schematic.

routing is used in our design, each RC module is able to perform look-ahead routing [87], which increases the overall performance of the switch. Figure 3.3 also represents the pipeline of the switch. At each stage a flit visits an AC module inside the switch. Also, notice that the RC module has no timing constraints and it can be computed during the two cycles needed by a flit to cross the switch. In this dissertation, the RC module is computed in parallel with the last AC stage of the modular switch. By performing the RC tasks during the second stage of the switch, only two routing computations are needed, each one for a different flit coming from a different input port at the last AC module of an output port of the modular switch.

3.2.4 Advantages and Disadvantages of the Design

The modular switch design requires additional resources when compared to the canonical switch design (will be seen later). In contrast, a higher operating frequency will be obtained. Another penalty that this design presents is the increment in metallizations (wiring) at the input side of the switch, where each input port of the switch connects to all the output ports of the switch. Note in Figure 3.2 that each input port is connected to all output ports. That increment in wiring increases the power consumption of the switch at the benefit of increasing the operating frequency. In the next chapter we will propose a distributed floorplan strategy for the modular switch – unfeasible for the canonical switch – which will decrease the power consumption of links, and

hence, the overall network power consumption with respect to a conventional floorplaning.

As it will be shown later, the modular switch presents two main advantages. First, as each output port has its own buffer, modular switch buffering resembles output queuing technique. Furthermore, an output port buffer is distributed among its stages. Populating each stage with intermediate buffering allows the switch to increase its throughput. Additionally, the simplicity of the switch when placed and routed, will enable a faster implementation when compared to a simple canonical switch, as it will be shown later.

3.2.5 Flow Control

To fully decouple AC modules, a flow control protocol is designed between an AC module and the upstream AC modules it is connected to. In this sense, only two signals between two connected AC modules are enough (see Figure 3.7). An *upstreamVALID* signal comes from each upstream AC module and means that the flit on its output is ready for the downstream AC module. On the contrary, the *downstreamFC* signal travels in the opposite direction. This signal, when set, indicates that the flit has been accepted by the downstream AC module. If the corresponding incoming *downstreamFC* signal is inactive, it means that data flow control must be stopped, and hence, the AC module stall its own flow control communication, and incoming flits must be stored in the slave slots on the buffer.

Figure 3.8 shows an schematic of the flow control signalling that involves an AC module with its upstream modules. The *upstreamVALIDi* signals are taken by the arbiter module in order to decide which input accesses buffering resources. The arbiter decision in addition to the local buffer status are used to generate *downstreamFCi* signals. If the buffer is able to accept an incoming flit, then the winner will have its *downstreamFCi* signal active, meanwhile the rest of incoming inputs will have their *downstreamFCi* signals inactive. If the buffer is full, all the *downstreamFCi* signals are set to zero. Figure 3.8 also shows the complete path – marked line – that should be crossed by a flow control signal. The path starts at the upstream AC module, travels down to the proper AC module, crosses the arbiter, and returns up to the starting point. Flow control signalling must cross the link twice in a cycle in order to

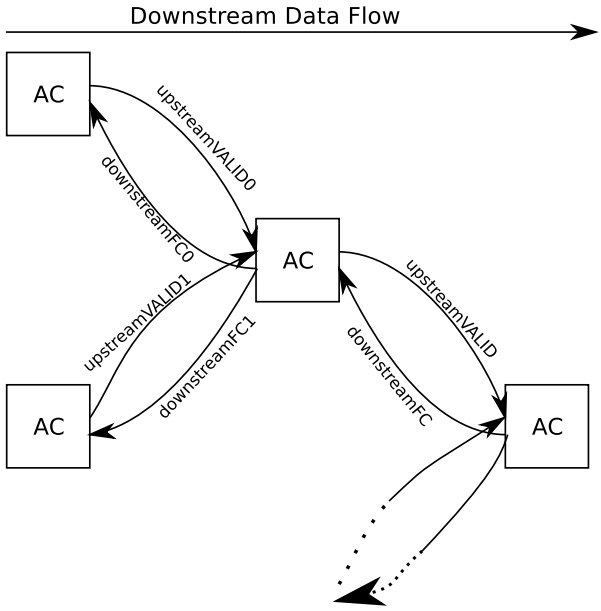


Figure 3.7: Flow control signalling.

fulfil timing constraints, in contrast of the data path that only crosses the link once. If no optimization is performed, the flow control signalling that crosses two times the link sets the critical path of the whole switch.

Finally, the buffer status occupancy must be computed to generate the proper *downstreamFCi* signals. Figure 3.9 shows how this info is retrieved. When one slot is available – penultimate slot is full – and the incoming *downstreamFC* signal is inactive, means that the next cycle is the last one that the buffer will be able to receive a flit until the data flow resumes. Notice that, the *downstreamFC* signal just crosses an AND gate before being registered. That means, that any downstream flow control signal involved crosses at most one link.

Figure 3.10 shows how a message crosses an AC module when there is no contention (see Figure 3.10(a)), and when there is contention (see Figure 3.10(b)). *data_in_1* signal refers to the first input port of the AC module (see Figure 3.4). If *data_in_1* signal has a valid flit, the *upstreamVALID1* signal (see Figure 3.7) is set. In this example, only this input is receiving incoming flits. Note that buffer size has been set to two and the occupancy of each slot is

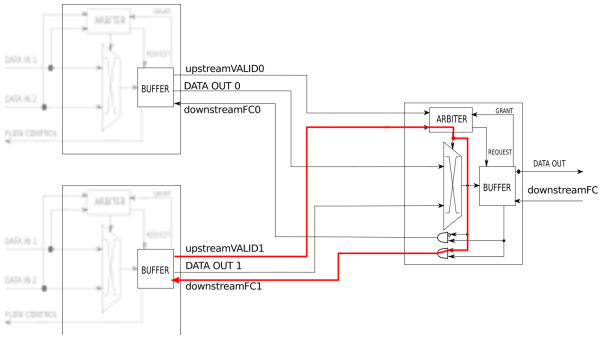


Figure 3.8: Flow control signalling generation.

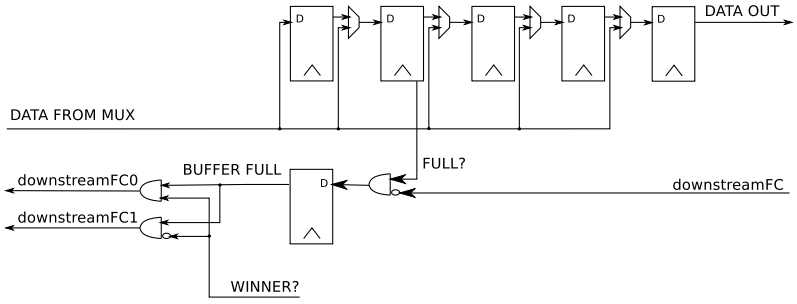
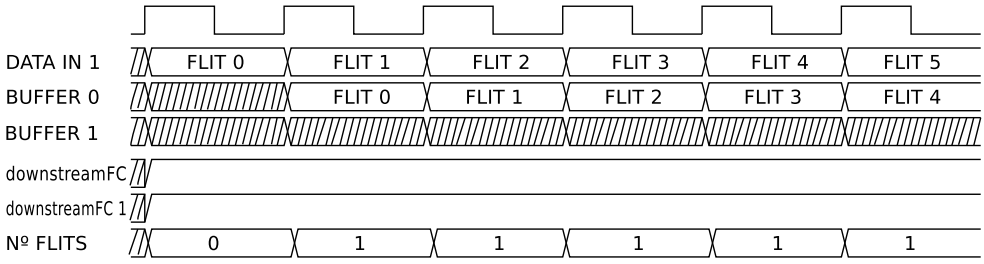
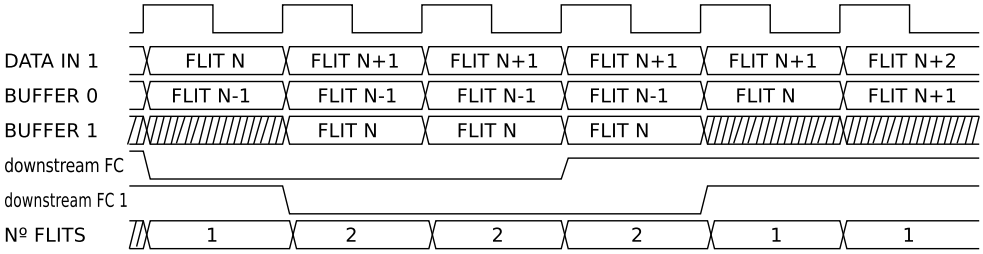


Figure 3.9: Buffer state signal generation.

recorded by the signals *buffer_0* and *buffer_1*. FC signals refer to flow control signals. The *downstreamFC* is the input flow control signal that comes from the downstream AC module. When set to one it means the downstream AC module is able to receive more flits. The *downstreamFCi* signals are the flow control signal sent to the upstream AC modules connected through *data_in_i* ports. As it can be seen in Figure 3.10(a), when no contention exists only a single slot in the buffer is used. In this case, flow control signals are active which means that communication flow is active. However, when the *downstreamFC* signal gets inactive (see Figure 3.10(b)), communication flow stalls. To avoid discarding flits flying on the link, the second slot on the AC module is used in this case. Additionally, *downstreamFCi* signals are recomputed, and if necessary, propagate the Stall state to the upstream AC modules, thus stopping the whole communication flow. When the *downstreamFC* signal is deasserted, the AC module drains its flits one by one, and communication flow



(a) Flit Transmission without contention.



(b) Flit transmission with contention.

Figure 3.10: Example of flow control operation between two AC modules.

resumes without introducing bubbles (see Figure 3.10(b)).

The *downstreamFC* and *downstreamFCi* signals are independent from each other. The *downstreamFCi* signals are computed by using the local buffer occupancy information and a registered version of the *downstreamFC* signal (see Figure 3.9). In a two slot buffer, *downstreamFCi* signals are a one-cycle delayed version of the *downstreamFC* signal. Decoupling both signals allows any flow control signal to involve at most two adjacent AC modules, and hence, any flow control signal may traverse at most one link in one clock cycle. Relaxing the delay constraint imposed to flow control signalling reduces the impact of the flow control into the whole network critical path.

The use of local flow control mechanisms reduces the communication overhead. That is, an AC module only requires to communicate with its connected AC modules. Another important benefit is the reduction of the critical path of the whole network. The flow control mechanism resembles the one previously used in [62–64, 88]. In that work, flow control signals allow messages to cross a synchronous network resembling asynchronous networks, by implementing a handshake protocol between adjacent stages. Then, flits cross the

network by using an asynchronous communication protocol but clock signal is not removed keeping the network synchronous which reinforces communication between modules.

In this flow control scheme, the storing capacity of the AC module can be set to any value at design time. However, if the buffer size (slot) of the AC module is lower than two, the round-trip-time constraint between modules cannot be met and, hence, there is no possibility to transmit consecutive flits without introducing bubbles, and thus impacting performance. Setting buffer size equal to two ensures minimum buffer size with no data loss [89]. By increasing the buffering resources the network throughput will increase at the cost of increasing the modular switch area and power consumption.

3.2.6 AC Module Radix Comparison

The AC module design can be parametrized by its radix (the number of input ports). Both area and delay depend on the AC radix. Also, the pipeline depth of the modular switch will vary accordingly. In this section, we analyze how area and delay are affected by the AC radix. Later we analyze the impact on performance (taking into account the pipeline depth). For this analysis, AC modules with different radices have been synthesized using the 45nm technology open source Nangate [90] with Synopsys DC [91]. The posterior Place&Route has been made with the Cadence Encounter tool [92]. From the complete set of metallization layers – M1-M10 – available in this library, only M1-M4 layers have been used to implement the logic of the AC modules. The reason is that high metallization layers will be used in next chapters to implement general purpose signals – supply signal and ground signal – or long links.

Table 3.1 shows the area and the delay for an AC module with different radices. The increment in area and delay of the different AC-x with respect the AC-2 are also included. As expected, as the AC-radix increases, area and delay increases as well. However, this increment is not linear. This is because the buffering remains constant (we need only two buffer slots regardless of the AC radix). Notice that, the radix increment does not modify the flow control and buffer behaviour. That is, the same conclusion obtained in the previous section is applied to AC modules with radix higher than two. However, as

AC Radix	Area (um^2)	Increment (%)	Delay (ns)	Increment (%)
2	2025	0.00	0.53	0.00
4	2704	33.53	0.65	22.64
8	3364	66.12	0.87	64.15
16	6084	200.44	1.08	103.77

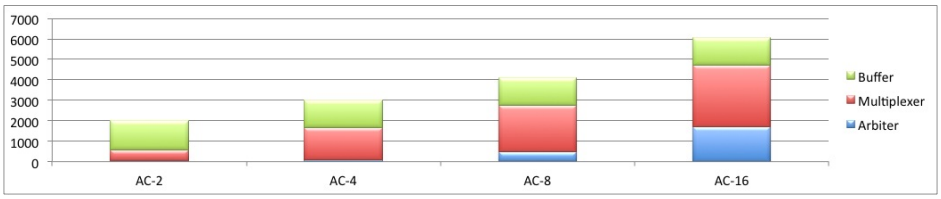
Table 3.1: Area requirements and delay for different AC-radix modules. Increment with respect to AC-2 included.

the number of inputs increases the number of flow control signals increases affecting the critical path of the AC module. Notice that, there exists an *upstreamVALIDi* and a *downstreamFCi* signal that relates the AC module with its connected upstream AC modules. Additionally, the multiplexer and the arbiter are modified. Thus, an AC-16 module is only three times bigger than an AC-2 module, despite it is able to handle eight times more inputs. Similarly, the delay does not increase linearly. An AC-16 module is just two times slower than an AC-2 module. Figure 3.11 shows the area and delay breakdown of each component for different AC-radix modules.

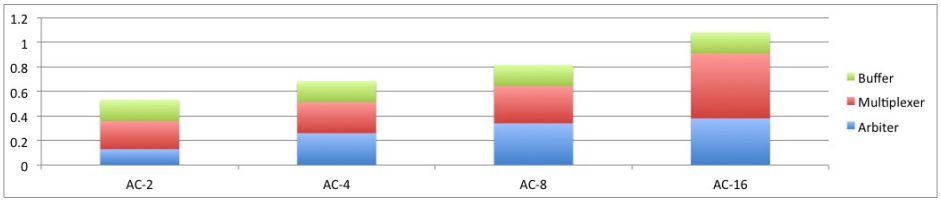
As it can be seen in Figure 3.11(a) the main contributor to AC area is the multiplexer. However, as the AC radix increases the complexity of the arbiter increases, and hence, it requires more resources. On the other hand, the area of the buffer remains almost constant as the buffering size only depends on the flit size which is invariant with the AC radix. Similarly, the delay related with buffering tasks remains almost unalterable as it deals with charging/discharging a FIFO. On the other hand, multiplexing and arbitrating delay depends on the AC radix, because as the AC radix increases more inputs must be handled.

3.3 Modular Switch Analysis

As each output port controller is independent and identical to the other output port controllers, the critical path of an output port controller is equivalent to the critical path of the entire switch. If we analyze the delay of the paths of a switch, we observe that the critical path are those that interconnect



(a) Area.



(b) Delay.

Figure 3.11: Area and delay for different AC module radices.

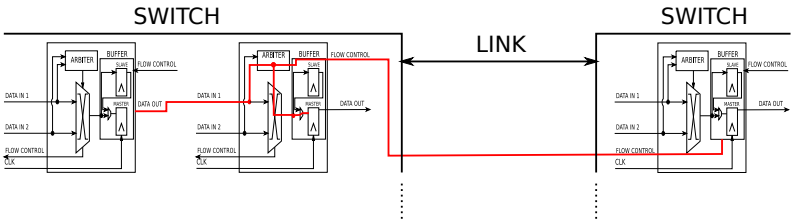


Figure 3.12: Modular switch delay path.

two adjacent switches (see Figure 3.12). That is, the slowest path is the one between two switches and the flow control logic that connects the two AC modules each at a different switch. Identically to the canonical switch, the delay of the modular switch is contributed by a combinational logic that creates the signals and the wire delay to transmit those signals from one switch to another.

In order to know such delays, a 5-port switch with AC-2 modules has been implemented using the 45nm technology open source Nangate [90] with Synopsys DC. We have used M1-M4 metallization layers to perform the Place&Route with Cadence Encounter. No link customization has been made, that is, the Place&Route tool is responsible of implementing and optimizing the links, and inserting the proper number and placement of the repeaters as necessary. We

observe, after synthesizing the modular switch that the combinational delay of the data path and the flow control path are almost identical. The minimum combinational delay obtained by the synthesis tool for those paths is 0.65ns. Then, the critical path (T) of a modular switch is

$$T = 0.65ns + \text{link delay} \tag{3.1}$$

where the link delay depends on the topology chosen and the chip layout. Figure 3.13 shows the critical path of the modular switch with AC-2 and AC-4, and the canonical switch as a function of the link length. First, notice that the modular switch delay with AC-2 is higher than the delay of a single AC-2 module shown in Table 3.1. The difference is due to the delay introduced by the flow control signalling when two AC modules are connected. Similarly, the delay of the modular switch with AC-4 is higher than the delay of the single AC-4 module. Second, the modular switch with AC-2 has a lower critical path than the rest of switches. The difference is maximum when link is short, and hence link delay barely influences the switch critical path. Thus, when no interswitch distance is considered, the modular switch with AC-2 has a critical path 20% smaller than the canonical switch. As the link length increases, the difference is reduced because link delay becomes the main factor in the whole switch critical path. However, the modular switch with AC-2 has a critical path 6.5% shorter than the canonical switch for interswitch distance of 2.4mm. In contrast, the modular switch with AC-4 has a critical path that is much close to the canonical switch critical path. It is only 3% shorter for link length equal to zero, and no difference exists for larger link lengths.

Table 3.2 shows the area and delay when link length is zero[1] of the modular switch with AC-2 and AC-4 modules as well as for the canonical switch. As expected the modular switch consumes more resources. This is due to the redundant resources that the modular switch requires to make each output port independent. The increment in area rises up to 21% when the modular switch is built with AC-2. In contrast, the modular switch with AC-4 reduces the area overhead down to 6%. Note that, the increment in area when using AC-2 is at the expense of reducing the critical path. When using AC-2 modules, the critical path is reduced by 20% meanwhile, it is reduced just by 3%

[1]This case allows to analyze the switch logic without considering the link influence.

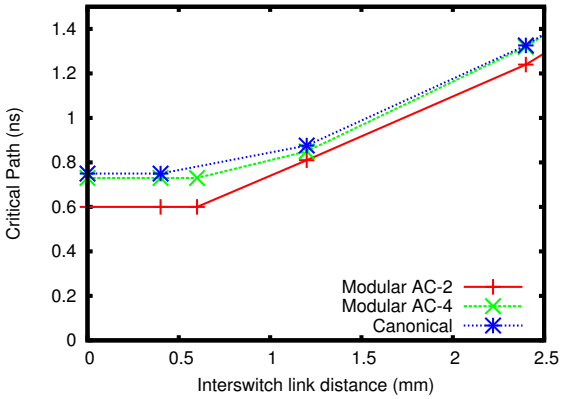


Figure 3.13: Modular switch delay.

	Area (um^2)	Increment (%)	Delay (ns)	Increment (%)
Canonical	17651	—	0.75	—
Modular AC-2	22500	21.0	0.65	-13.3
Modular AC-4	16520	-6.0	0.73	-3.0

Table 3.2: Area requirements and latency by the modular and canonical switch when link length is zero. Increment values with respect to the canonical switch shown.

when using AC-4.

3.4 Network Performance with Synthetic traffic

In order to analyze the performance of a network, we have developed a cycle-accurate network simulator, which is able to mimic the behaviour of the network when implemented with canonical switches or with modular switches with different AC radices. The simulator, called gMemNoCsim, is an event cycle-accurate and flit-accurate simulator developed in C/C++. This simulator has been used extensively during the last years within the group in different conference and journal publications. The simulator allows three different kind of evaluations: synthetic traffic patterns, coherence protocols with traces, and coherence protocols with connection to a system-level application running on top of the simulator (via GRAPHITE [93] simulation tool).

The tile, for synthetic traffic, has been modeled as an ideal NIC that injects messages into the network at a constant injection rate. Each NIC is made of a single queue that stores the flits waiting to be transmitted. Three types of traffic patterns have been used: uniform, bit-reversal, and bit-complement. With uniform traffic, each end node has equal chances to become the target of a newly generated message. With bit-reversal traffic, messages generated by node coded in binary as $b_{N-1}b_{N-2}\dots b_0$ target node $b_0b_1\dots B_{N-1}$. With bit-complement messages generated by a node target the node with the ID resulting from complementing all the bits of the source ID node (e.g. node N-1 targets node 0). Unless stated the contrary, flit size has been set to 64 bits. Two message generation scenarios will be considered. First, an ideal scenario where all messages are 1-flit long. In a second scenario, message generation pattern has been created to model a real application traffic on CMPs. In this scenario, there exist two messages sizes. Short 1-flit long messages (64 bits to allocate mainly a 32-bit address, few bits for the coherence protocol command, and the message header), and long 9-flit long messages (72 bytes to allocate mainly a 32-bit address, a 64-byte line cache, some bits for the coherence protocol command, and the message header). Short messages represent 70% of messages injected.

Network performance is measured as the accepted traffic rate (flits per cycle per node) and the average end-to-end flit latency. End-to-end latency is measured as the time elapsed since the flit generation at the source node until the reception at the destination node. In absence of contention, end-to-end latency is equal to the network latency measured in switch-to-switch hops plus two cycles (one cycle for flit injection and one cycle for flit extraction from the network). For the canonical switch we model the 4-stage design (IB, RT, VA, ST) with a buffer size of four and for the modular switch we model the AC-based pipelined design. Networks have been warmed up by injecting 20000 messages, and results are collected after 200.000 messages are received. These numbers for transient and permanent scenarios are typical and have been used in previous research works.

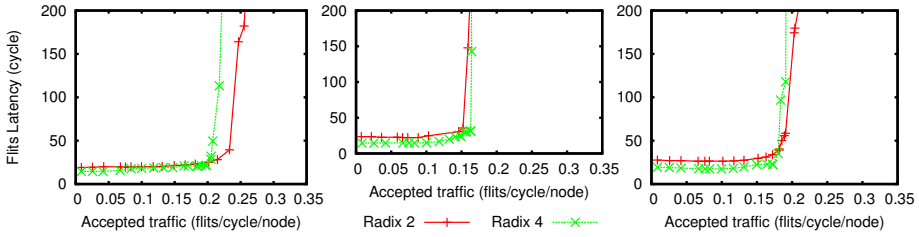
3.4.1 AC Radix Influence

The AC radix sets the network performance. First, the AC radix determines the switch pipeline depth (see Section 3.2), and hence, network latency per switch hop. Additionally, as each AC stage has its own buffer, modifying the AC radix means changing in-network buffering resources and hence network throughput will be affected. In this section, we compare network performance of a modular switch with AC modules implemented with two ports (radix 2; AC-2), and a modular switch with AC modules implemented with four ports (radix 4; AC-4). In a 2D mesh with a switch radix of 5 (four switch-to-switch ports and a local port), a modular switch with AC-2 has an output port depth equal to two, meanwhile the modular switch with AC-4 has a output port depth equal to one (the switch radix equals the AC radix, excluding U turns).

Figure 3.14 shows the average network latency measured in cycles and the accepted traffic measured in flits/cycles/node for the modular switch with two AC radices and for the three different traffic patterns. Network size is set to 64 nodes (8×8 mesh) and messages are one flit long. Dimension order routing is used.

As it can be seen, as the AC radix increases the network zero-load latency is smaller because the switch pipeline depth gets reduced. Notice that we are obtaining cycle level results. Theoretically, the zero-load latency reduces to half when increasing the AC radix from two to four (going from a depth of 2 AC-2 modules down to a single AC-4 module per switch output port). In practice, this reduction is smaller due to added and constant injection/extraction delay. Another interesting observation is that network throughput is highly affected for uniform traffic when increasing the AC radix. There are two reasons that explain this behaviour. First, as the AC radix increases the buffering resources of the modular switch decrease. Second, increasing the AC radix means to increase the number of inputs that compete for a buffer, thus, contention increases degrading the network throughput. Notice that, this worsening is not noticeable for bit-reversal or bit-complement traffic patterns.

Now, we compare the canonical switch with the modular switch with AC-2 modules. Figures 3.15 and 3.16 show the network latency measured in cycles and the accepted traffic measured in flits/cycles/node when message size is set to one flit for the three synthetic traffic patterns. As done in the previous

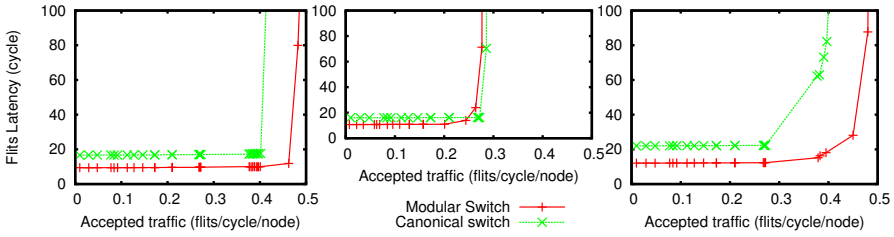


(a) Uniform. (b) Bit-reversal. (c) Bit-complement.

Figure 3.14: Cycle-level load-latency graph for a 64-node network built with a modular switch with different AC radices. Messages are one flit long.

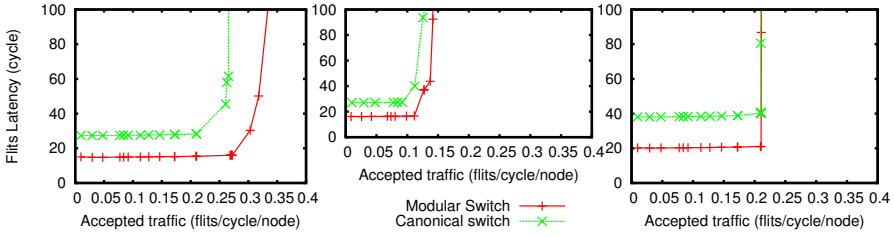
analysis, the switch operating frequency has not been taken into account to analyze the strength of the modular switch. Network size is set to 16 (4×4 mesh) and 64 (8×8 mesh) nodes. Notice that a flit requires two cycles to cross the modular switch but five to cross the canonical switch – four to cross the switch and one to cross the link. Thus, the modular switch has a lower zero-load latency. Although a more advanced switch can be engineered with lower number of pipeline stages, notice that, the aim of this chapter is not to present a switch that overcomes the canonical switch but a switch which has the same functionality than the canonical switch, and it is designed to be optimally placed along the chip, as it will be shown in next chapters.

Significant differences are obtained when focusing on the network throughput. Figure 3.15 shows that the modular switch has a higher network throughput than the canonical switch. This is due to the distributed buffering along the switch stages that reduces message contention. That is, the canonical switch has a single buffer per input port which compete for the output resources with the rest of buffers. If one message of an input port losses an output port grant, no messages from this buffer will leave the switch. On the contrary, the modular switch spreads the buffering resources along the stages. Thus, messages get more chances to advance through the network, thus, improving the network efficiency. This improvement in network throughput does not depend, however, on the traffic pattern used. For a 16-node network, the increment in throughput is, at most, 20% – for uniform traffic. As network size increases, differences increase a bit. For a 64-node network, the increment in throughput grows up to 26%.



(a) Uniform. (b) Bit-reversal. (c) Bit-complement.

Figure 3.15: Load-latency graph for a 16-node network built with the modular and the canonical switch. Messages are one flit long.

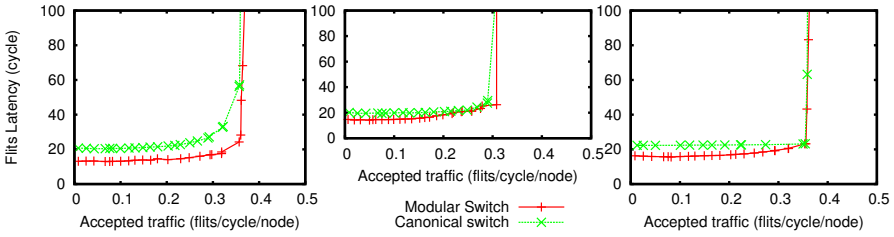


(a) Uniform. (b) Bit-reversal. (c) Bit-complement.

Figure 3.16: Load-latency graph for a 64-node network built with the modular and the canonical switch. Messages are one flit long.

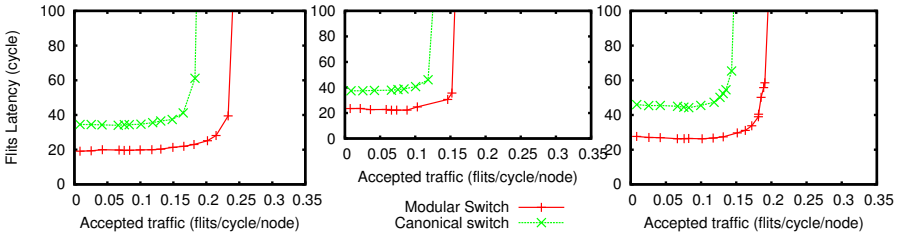
When traffic configuration is stressed differences are even higher. Figures 3.17 and 3.18 show the achieved performance for a 16- and a 64-node network, respectively, when short and long messages are considered (bimodal traffic). For a 64-node network the throughput increment of the modular switch is almost 38% when uniform traffic is considered. On the contrary, for non-uniform traffic patterns, the modular switch achieves lower throughput benefits.

The previous results do not consider implementation issues, and thus, operating frequencies. Figures 3.19 and 3.20 show the same results but now latency is measured in nanoseconds and accepted traffic in flits/nanoseconds/node for 16- and 64-node networks, respectively. As it can be seen, the modular switch presents now much lower network latency and much higher throughput. To the intrinsic benefits of using the modular switch – higher throughput due to its distributed buffering – its higher operating frequency implies that the



(a) Uniform. (b) Bit-reversal. (c) Bit-complement.

Figure 3.17: Cycle-level load-latency graph for a 16-node network built with the modular and the canonical switch. Bimodal traffic: 70% 1-flit messages and 30% 9-flit messages.



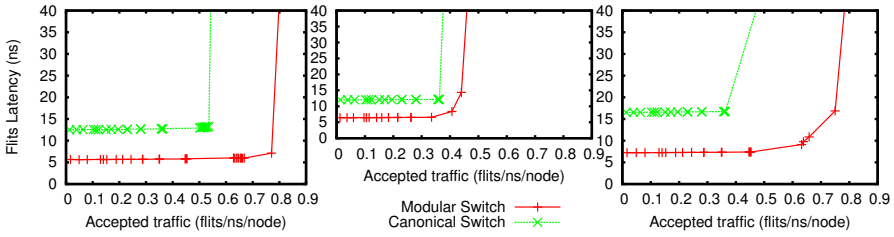
(a) Uniform. (b) Bit-reversal. (c) Bit-complement.

Figure 3.18: Cycle-level load-latency graph for a 64-node network built with the modular and the canonical switch. Bimodal traffic: 70% 1-flit messages and 30% 9-flit messages.

modular switch increases its goodness with respect to the slower canonical switch. For the uniform traffic, throughput is increased up to 47% and 60% for 16- and 64-node networks, respectively, meanwhile latencies are reduced up to 51% and 65%, respectively.

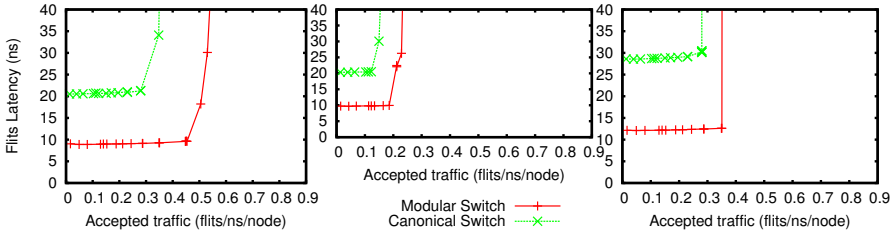
3.5 Application Performance and Power Consumption

Now, we analyze the network power consumption when running parallel applications on top of the CMP system. Different SPLASH-2 applications have been run in a 16-core CMP system using the GRAPHITE Simulator [93]. gMemNoCsim simulator has been coupled with GRAPHITE to model a 2D mesh



(a) Uniform. (b) Bit-reversal. (c) Bit-complement.

Figure 3.19: Time-level load-latency graph for a 16-node network built with the modular and the canonical switch. Messages are one flit long.



(a) Uniform. (b) Bit-reversal. (c) Bit-complement.

Figure 3.20: Time-level load-latency graph for a 64-node network built with the modular and the canonical switch. Messages are one flit long.

when using, both, the AC-2 modular approach and the canonical approach. Additionally, a coherence layer models a regular MOESI invalidation-based coherence protocol. To avoid protocol-level deadlocks we assume parallel (virtual) networks, as done in the Tilera chip. Notice that using parallel networks instead of virtual channels keeps the critical path of the network unalterable. The configuration of the CMP system is summarized in Table 3.3.

Figure 3.21 shows execution time of applications, normalized to the canonical when running different SPLASH-2 application with both switch designs. As expected, execution time of applications when using the modular switch is significantly lower. As the traffic generated by SPLASH-2 applications is low, the main contribution to the reduction in execution time comes by the fact that the modular switch reduces the number of cycles a flit needs to traverse the network (two hops per switch instead of four hops per switch with the canonical approach). Notice that these differences could be much lower if

Parameter	Values
CMP	16 cores, tiled organization, Distributed Shared Memory
Cores	x86 Architecture, Area $2.4mm^2$, in-order, single thread
L1 inst cache	private, 64KB, 4-way
L1 data cache	private, 64KB, 4-way
L2 cache	shared, 512KB per tile, 16-way
Messages	Short messages: 1-flit long. Long Messages: 9-flit long

Table 3.3: CMP configuration.

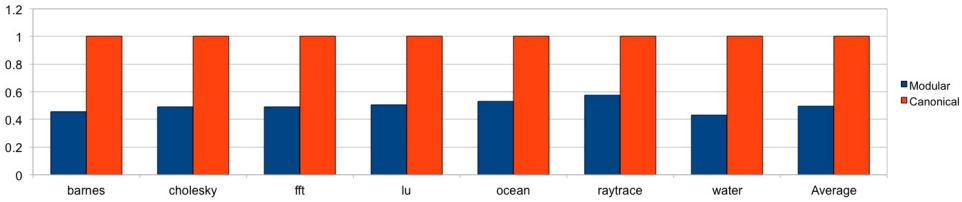


Figure 3.21: Execution time when running different SPLASH-2 applications with a modular and canonical switches.

compared to an optimized switch with less stages.

However, by running real traffic an interesting conclusion is revealed. A network implemented with modular switches significantly reduces the energy consumption despite its larger buffering requirements. Figure 3.22 shows the energy required due to links and switches when implementing a 16-node network with a modular switch and a canonical switch. Energy of each network component has been measured over the synthesised, place&route switches by using Power Compiler by Synopsys [94]. The energy is decomposed in two terms, the static energy due to the leakage power, and the dynamic energy which is caused by the flits crossing the network. As it can be seen, the static energy is a small fraction of the whole network energy, and hence, despite that the modular switch increases the network resources, that increment does not highly affect the total energy consumption. In contrast, as the modular switch reduces the contention of the network due to its distributed buffering, the energy needed by a flit to cross the network is highly reduced.

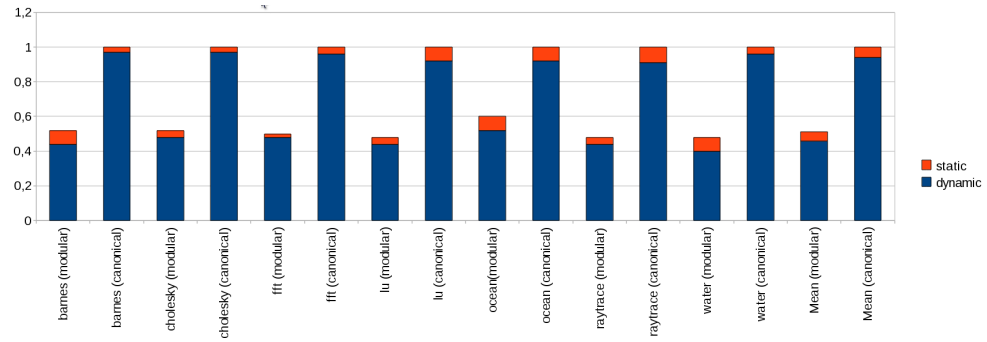


Figure 3.22: Energy consumed by network implemented with modular and canonical switches when running different SPLASH-2 applications.

3.6 Reliability Analysis

To conclude this chapter, focused on the AC-module-based design, we want to focus the attention now on the reliability offerings of such design approach. Now, in this section we analyze the robustness of the modular switch architecture against the new challenges imposed by the nanoscale manufacturing scenario. Concretely, we analyze the fault-tolerance robustness of the proposed architecture. Notice that this fact is not a central contribution of the thesis. However, the extra redundancy of the modular switch design allows for a certain level of fault tolerance that is not naturally available in the canonical switch design.

We compare fault-tolerant benefits of the modular design against the canonical switch architecture. For that purpose, we assume both switches to be composed as a sum of transistors. Additionally, each transistor has a failure probability (TFP). Based on this assumption, any sub-block of a switch has a probability of a permanent fault (R_i) given by:

$$R_i = 1 - e^{-TFP*N_i} \tag{3.2}$$

where N_i is the number of transistors in that sub-block. Transistor count is computed by dividing the area of a sub-block by the area of the 2-input NAND gate in Nangate Open Source Library [90] to obtain an approximate gate count. The transistor count of the sub-block was calculated from the gate count. This

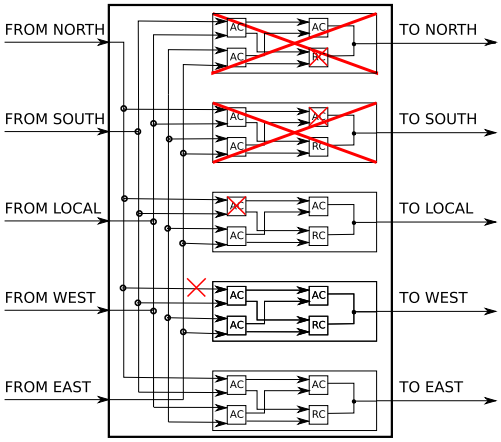
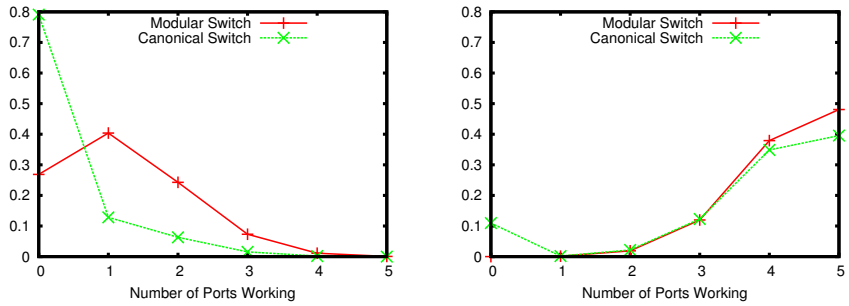


Figure 3.23: Fault tolerance of the modular switch.

methodology is based on the assumption that the density of transistors per unit area is roughly homogeneous and the same across the chip. By assuming this scenario, each transistor has a failure probability that does not depend on its location. From this analysis we have that the switch architecture with more transistors is more prone to manufacturing defects. Therefore, the probability of having a failure in the modular architecture is higher as the modular switch presents a larger transistor count than the canonical switch.

However, as the modular switch includes redundant resources (see Figure 3.23) its architecture presents inherent fault-tolerance. In this sense, some errors can be tolerated by the modular architecture despite the presence of failures at the port of the switch. Figure 3.23 shows a modular switch schematic where some failures reduce switch functionality but some ports still work. As it can be seen, *north* and *south* ports suffer some failures in the AC and RC modules of the second stage, and hence, the whole output ports get inactive. In contrast, output ports to *local* and *west* present some failures at the first stage, and the link to access to the *west* output port, respectively. However, both ports can still be operated, although not providing full service.

Figure 3.24 shows the probability of having a given number of ports working for two different failure rates, (TFP $=10^{-4}$ and FTP$=10^{-5}$). The modular switch architecture presents a more robust behaviour than the canonical switch. Note that, the canonical switch presents, for high values of TFP, a



(a) TFP=1e-4. (b) TFP=1e-5.

Figure 3.24: Probability of having different number of ports working for different switch designs.

high probability failure for the whole switch (no ports working). This probability is close to 0.8 meanwhile the probability that any port of the modular switch works remains lower at 0.3. The canonical switch relies on the use of single centralized modules as the crossbar and the arbitration modules. Those modules are shared amongst all the input/output ports, and hence, whenever the crossbar or the switch arbitration module fails, the whole switch fails. In contrast, any output port of the modular switch is independent of the rest of ports, and hence, failures affecting a given output port, do not affect the remaining ports of the switch. This redundancy makes the modular switch more robust against transistor errors than a canonical switch. For a low transistor failure rate (see Figure 3.24(b)) the probability of a correct operation is 0.48 for the modular switch and 0.4 for the canonical switch. In addition, the probability of a total failure is negligible for the modular switch, meanwhile the canonical switch has a probability larger than 0.1 to fail the whole switch. Despite the fact that the modular switch has this unexplored redundancies and fault-tolerant properties, in this thesis we pursue performance and implementation efficiency, thus leaving for future work the fault-tolerance properties of the modular switch design.

3.7 Conclusions

In this chapter we have introduced the modular switch design, which will be used and exploited as the baseline design for the rest of the chapters. The main property of the modular switch is that is mainly formed of simple AC modules. Any AC module is independent and it is able to perform all the switch tasks needed, that is, buffering, arbitration, and switch traversal.

The modular switch design allows for higher operating frequencies when implemented, thus exhibiting higher throughput and lower latency values. In addition, redundancy within the design allows for higher fault-tolerant degrees.

In the next chapter we deal with the floorplan of the AC modules in order to make a distributed and optimum placement, thus maximizing the benefits of the design. Notice that this distribution is not possible in a centralized switch design (as the one imposed by a canonical switch design).

Chapter 4

Distributed Switch

In this chapter, we present a new switch architecture that optimizes the physical implementation of the modular switch in different scenarios as FPGAs or ASICs. We refer to this novel switch architecture as the *distributed switch*. In this chapter, we will focus on 2D meshes for the sake of clarity.

On ASICs, the *distributed switch* moves the circuitry of the switch along the link in contrast to a conventional floorplan where the entire switch is placed in a single bounded area. In our distributed switch, in contrast, the switch and the link are glued. **Packets are buffered, routed, and forwarded at the same time they are crossing the link**. By properly placing the AC modules, maximum link length (between logic blocks) is reduced, reducing the whole network critical path and power consumption.

On FPGAs, the *distributed switch* is properly placed by guiding the mapping tool which is the responsible of allocating the network into the FPGA resources. In this scenario, the *distributed switch* divides each AC module into single bits, and then, each AC module bit is placed on a pre-build logic block of the FPGA. Using the logic blocks as buffering resources removes the necessity of using the slower SRAM blocks of the FPGA. Thus, the *distributed switch* allows the FPGA to reach higher operating frequencies meanwhile optimizing resources.

The *distributed switch* can be seen as an elastic switch. When using the distributed switch, each pipelined stage of the communication between two nodes is independent of the rest of the network. Additionally, each stage

communicates with its previous connected stages and its next stage, allowing a local communication that allows flits to cross the network step-by-step by using local information at each stage.

4.1 Distributed Switch

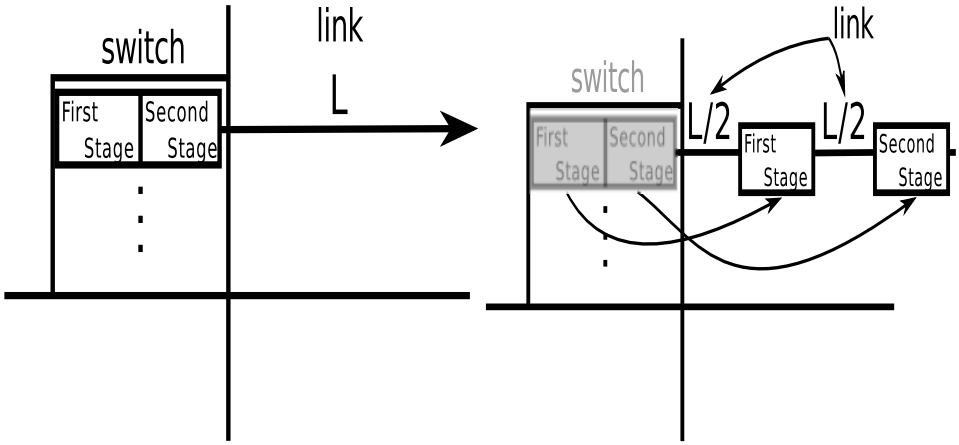
The modular switch design presented in the previous chapter has an interesting property. Each output port has its own circuitry that is independent from the circuitry of the rest of output ports (see Figure 3.2). This property allows the modular switch to be distributed over the links while keeping the connectivity of the switch with its neighbours, that is, it allows to spread the circuitry of each output port controller of the switch along the link that connects that output port circuitry with the adjacent switch. Figure 4.1 shows the spreading of the output port circuitry of the switch along the link. Note that each of the stages of the output port controller is placed at half the length of the link connecting to the adjacent switch. For the sake of clarity, the RC module has been omitted. However, in order to reduce wire redundancy, the RC module is computed in parallel with the second stage.

There are several benefits of distributing the switch over the link. First, the maximum operating frequency can be increased. As it has been seen in the previous chapter, the critical path of a switch is highly related with the link delay, and this delay depends on the link length. Distributing the switch over the link, allows to convert an interswitch link in several smaller sublinks, and hence reducing the critical path, increasing the maximum operating frequency.

Second, power consumption is reduced without increasing the delay of the communication between switches. That is, distributing the switch over the link forces the link to be pipelined (*distributed link*). However, the pipeline of the distributed link is not only introduced to minimize power consumption but to perform the switching tasks. Pipelining the link minimizes the delay constraints of the link. Then, power consumption of the link is reduced. Thus, interconnects can be designed reducing the power consumption meanwhile the pipeline of a message is not increased.

The third benefit of distributing the switch over the link is that any stage of the pipelined switch is connected to a fragment of the link (sublink). Thus, the

effects of process variation over the pipelined switch can be easily minimized, as shown in [95]. In [95] a simple technique to reduce the process variation effects of the switch was presented. Basically, any performance variation in the switch is compensated by making the link faster, which is a simple and well-known technique. However, this technique could not be used inside a pipelined switch [96]. In a distributed switch, any stage of the switch is connected to a sublink and, therefore, any process variation of any pipeline stage of the switch can be compensated by the sublink that is connected to.

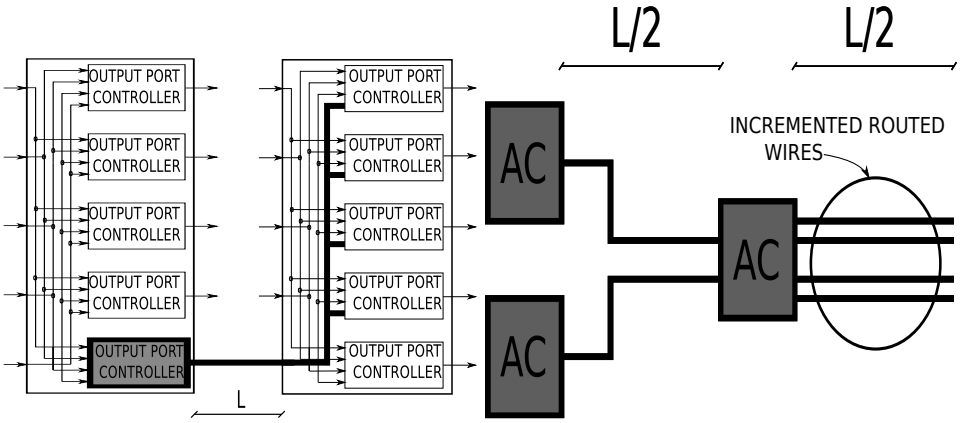


(a) Modular output port. (b) Distributed output port.

Figure 4.1: Output port controller for a modular switch and a distributed switch.

Distributing the switch over the link may have a negative consequence, the number of wires between some AC modules increases. Remember that in the modular switch, any input port is connected to all the output port controllers inside the switch. In the distributed switch, this interconnection becomes longer because AC stages are separated from each other. This effect can be seen in Figure 4.2. The figure shows the connectivity between two modular switches and its equivalent distributed link. While a typical link has a length of L plus the length of the wire inside the switch (negligible), the distributed switch presents six smaller links of length $L/2$, accounting for a total of 3L routed wires per distributed link. Then, a distributed link, presents three times more routed wires than the equivalent centralized link. Despite the

increment in routed wires, the length reduction of each sublink in a distributed link will reduce the delay constraint of these sublinks and then, reducing the total power consumption of the distributed link.



(a) Link between adjacent modular switches.

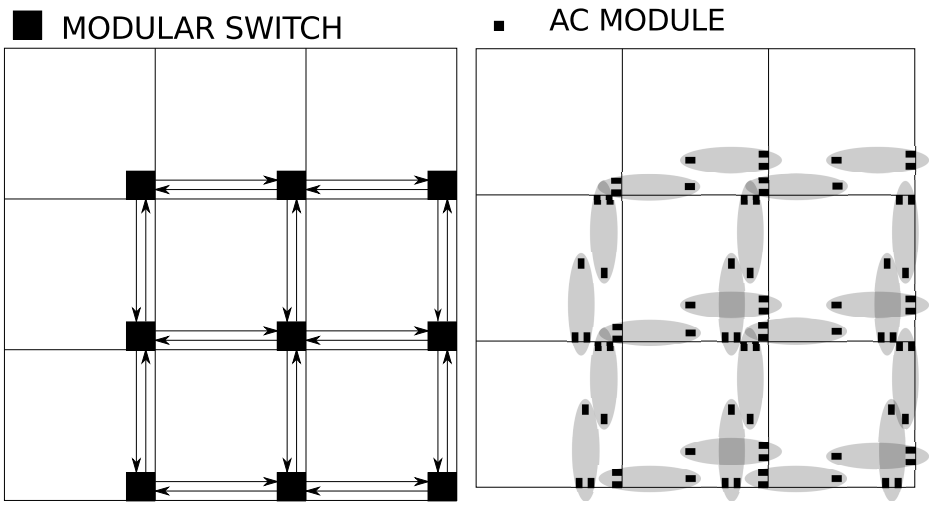
(b) Equivalent distributed link.

Figure 4.2: Link scheme for both scenarios a modular switch and a distributed switch designs.

The negative effect of these longer wires is minimized by using higher metallization layers, achieving low power consumption (see Section 4.1.3). This is the case of the distributed switch. In contrast, in the modular switch, the distances between internal AC modules are shorter as they are inside the switch area and, hence, they are routed by using lower metallization layers which have worse properties, and hence, higher power consumption numbers.

Figure 4.3(a) shows the floorplan of a 3x3 2D-mesh with standard (modular) switches. Note that, each modular switch is constrained to a bounded area, equidistant to other switches. Figure 4.3(b) shows the floorplan of a 3x3 2D-mesh when distributed switches are implemented. Interconnection wires have been omitted for clarity. Note that, the AC modules of the different distributed switches are spread on the die. AC modules connected between them are placed at distance L/2. AC modules grouped by a shaded area build a distributed link, as shown in Figure 4.2(b).

The critical path of this switch can be computed in the same way as the critical path of the modular switch. The combinational logic delay of the



(a) Floorplan with standard switches. (b) Floorplan with distributed switches.

Figure 4.3: Floorplan of a 2D mesh with standard and distributed switches.

distributed switch is identical to the combinational logic of the modular switch and therefore the delay of the combinational part of the critical path remains identical. However, remember that for the distributed switch the link length is half the length of the modular switch link. The length reduction of the distributed switch sublinks allows the designer to reduce the link delay fixing the same link delay for both approaches, the distributed sublink consumes less power than the modular link. Then, identically to the modular switch, the critical path of the distributed switch can be computed as

$$T = 0.60ns + \text{link delay} \tag{4.1}$$

4.1.1 Reducing Power Consumption

The main disadvantage of the distributed switch is the higher number of wires, that increases the leakage power consumption of the switch. To minimize leakage we can switch off or remove some sublinks based on the applied routing algorithm. Indeed, the XY algorithm avoids any YX transition. Therefore, only two ports may compete for an X output port: the local port and the west input port for the east output port, and the local port and the east port for the west output port. By leveraging the XY routing algorithm, we can

remove half of the links in the X direction. Figure 4.4 shows a distributed link where half of the wires have been removed in order to reduce the leakage power consumption.

This technique can also be applied to the modular switch. However, it only affects to logic modules inside the switch area rather than along the link. Thus, the benefit of applying this technique in the modular switch is much lower.

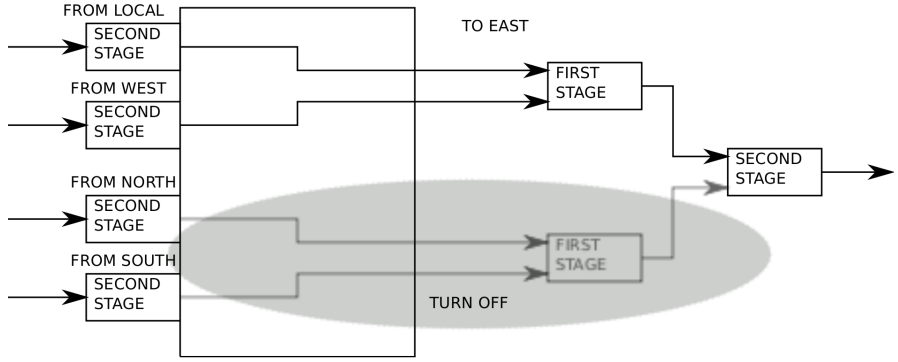


Figure 4.4: Removing intra switch sublinks when using XY to minimize power consumption.

4.1.2 Distributed Switch With Spread Buffers

As mentioned in Chapter 3 the minimum buffer requirements for a proper communication flow is set to two slots per AC module. However, we can take further advantage of these buffers. Figure 4.5 shows a distributed switch with spread buffers (DSB switch). In a DSB switch, the logic of the AC modules remains unchanged. In contrast, the buffer slots of the different AC modules are evenly spaced. In order to prevent bubbles from being introduced, the flow control remains untouched and relates to AC modules. From a behavioural point of view, each AC module still has two buffer slots (one inside the AC module and one in between two AC modules). That distance can be computed as L/N, where L is the total length of the wire and N the total number of buffer slots of a distributed switch path. Figure 4.5 shows a distributed link with spread buffers (DSB link) with four buffer slots (two per AC module). In Figure 4.5, each AC module has one buffer slot, while the second buffer slot

is shifted at a distance of $L/4$. Therefore, all the buffer slots are from each other at a distance of $L/4$.

In a DSB link, the buffer properties described in the previous section still hold, that is, if there exists no contention, the delay of the link is only two cycles (one cycle per stage). If the link blocks due to contention, then the extra slots (one per stage) are used, increasing the delay of the link up to four. To keep the buffer behaviour, it is necessary to duplicate the number of wires over the link as it can be seen in Figure 4.5, because internal wires between slots became wires. There is no extra logic inserted and hence the operating frequency of the switch remains identical.

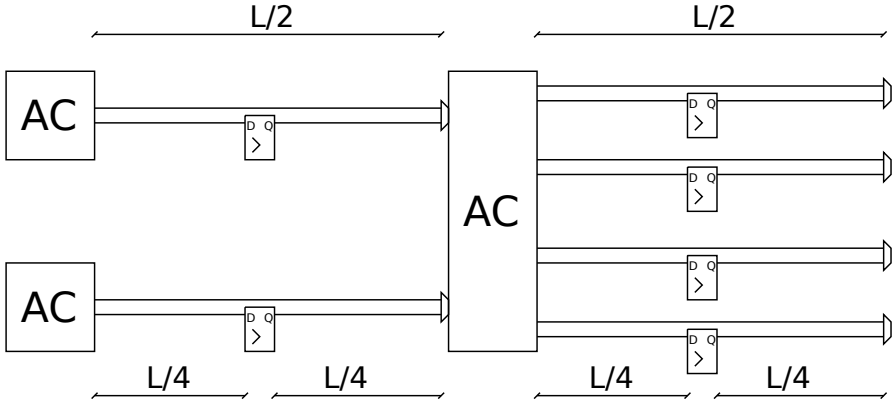


Figure 4.5: Distributed with spread buffers switch.

4.1.3 Wire Design

The proper placement and route of the components of a NoC is critical to obtain good performance. In particular, designing the links of a NoC is determinant. In fact, Placement&Route tools as Encounter by Cadence, are able to optimize the wires that interconnect different logic blocks. However, it is possible to design links manually, in order to minimize some parameter constraints as power or delay. In this section we describe the design of the wire between switches to minimize power consumption for both scenarios: when modular switches are used, and when distributed switches are implemented.

Wire delay has been set to the maximum value that allows to match the operating frequency of the switch to 1 GHz (see Equation 4.1) for both the

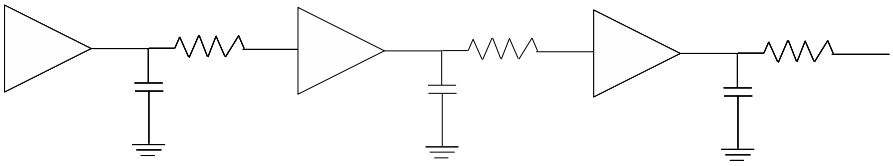


Figure 4.6: Diagram of a repeated wire composed of three sections

modular (or standard) switch and the distributed switch. That is, for the modular switch wire delay has been set to 0.35ns, meanwhile for the distributed switch has been set to 0.40ns. In contrast, wire length differs from one switch design to the other. Wire length for the modular switch is set to 2.4mm identically to the experimental scenario analyzed in the previous chapter. For the distributed switch, although the tile is identical, wire configuration is different and, therefore, in practice wire length is divided by two, being 1.2mm (see Section 4.1). We do not take into account switch area in any of both designs.

We insert repeaters along the wire when needed [15]. This method reduces interconnect delay and signal transition times. Figure 4.6 shows a wire with three inserted repeaters. Each segment of the wire is represented by a resistor and a capacitor that model the resistance and the capacitance of this segment of the wire, respectively. In our case, we have modelled the wire by using the 5-pi wire model [97]. As in the design of the switch, repeaters have been designed by using the Nangate 45nm technology library, as stated in [90]. This technology sets the supply voltage to 1.1V, and the electrical and physical parameters of the different metallizations available which are used to route the wires. MOS transistors used for the repeaters follow the properties of the PTM model for 45nm [98].

We use the higher metallization layers offered by the technology (M7-M8) to route the interconnection wires. This decision has been made because of two reasons. First, because the resistance of higher metallization layers is lower and, hence, the power consumption is lower. Second, the use of high metallization layers for switch interconnects allows the designer to use lower metallization layers for other purposes, as SRAM. By using the higher metallization layers for the interconnecting wires, it is possible to route wires

between switches over functional modules as SRAM [1,77], that is, route wires without using dedicated area for them. The repeaters inserted along a wire use the same type of resources as other functional blocks as for example the logic of the switches or the SRAM. Then, the repeaters inserted along a wire occupy some dedicated area that must be taken into account. Table 4.1 shows a summary of the parameters used to design the wires between switches.

Technode (nm)	45
Wire latency (ns)	0.35/0.40
Vdd (V)	1.1
Wire Width (um)	0.8
Wire Space (um)	0.8
M7-M8 Resistance (Ω / mm)	300
M7-M8 Capacitance (fF / mm)	190

Table 4.1: Physical and electrical data for 45nm NoC wires

In order to minimize the power consumption of a wire, it is necessary to insert the proper number of minimum sized repeaters [76] that satisfies the delay constraint imposed by the operating frequency of the switch. This technique is called power-optimal repeater insertion [76]. Table 4.2 shows the number and size[1] of the repeaters of a wire for different wire lengths, when using the optimal repeater insertion technique together with the wire parameters showed in Table 4.1. As can be seen, shorter wires require much smaller repeaters, thus consuming less power. This is the reason why the redundancy in wires for the distributed switch is compensated by the reduction in power consumption of each fragment of the wire.

Wire Length	2.4	1.2	0.6
Number of repeaters	7	6	3
Size of the repeaters	15	6	6

Table 4.2: Number and size of the repeaters of a wire.

[1]The size or gain of a repeater is the relation between the width and length of the transistors channel.

4.2 Distributed Switch Analysis

Now, we focus on the analysis of the distributed switch, compared to the modular switch. First, we perform an analysis of the link delay, then the area requirements, power consumption, and finally, variability and fault-tolerance.

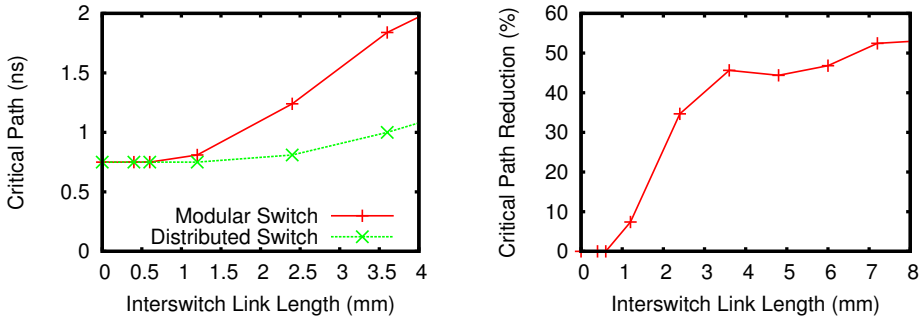
4.2.1 Link Delay vs Link Length

As it has been said in previous section, the critical path of the modular and the distributed switch are identical. That is, the critical path has two main components. The logic component fixed by the AC module, which is identical in both switches, and the delay of the link, which is different (see Equation 4.1). In this section, we analyze the maximum operating frequency – minimum critical path – of the modular switch and the distributed switch. Figure 4.7(a) shows the critical path of both switches as computed in Equation 4.1 with respect to the intercore distance computed in millimetres (mm). The wire length of the modular switch is equal to the intercore distance. In contrast, the wire length of the distributed switch is half the distance of the intercore distance. Figure 4.7(a) shows that for short wire lengths (lower than 0.5 mm), the critical path remains almost constant and equal for both switches. However, for larger wire lengths, the critical path increases as the wire length increases in both cases. However, the increment in the critical path is smaller for the distributed switch. Figure 4.7(b) shows the reduction in percentage in the critical path of the distributed switch with respect of the modular switch. As it can be seen, there is no reduction for short wire lengths. In contrast, as the wire length increases, the critical path of the distributed switch gets smaller than the critical path of the modular switch. The reduction of the critical path increases up to 53% for long intercore distances.

4.2.2 Area Results

Logic Area Results

In this section, we analyze the area of the modular and distributed switch. Note that area results provided in this section do not include the link connecting different switches. Remember that both switches are identical from a



(a) Switch delay.

(b) Distributed switch delay reduction.

Figure 4.7: Modular and distributed switch delay.

	Area (um^2)
Modular Switch	28356.72
Distributed Switch	24437.28

Table 4.3: Area occupied by a modular switch and a distributed switch

functional point of view, meaning that both switches provide the same output response for an identical input stimuli. Then, there is no difference in performance (at cycle level) when introducing both switches in a network.

Table 4.3 shows the area occupied by the logic for both switches. The distributed switch occupies a 13.82% less area than the modular switch. The main reason for the small difference is that the distributed switch presents a simpler design that helps the synthesis tool to optimize the design. Furthermore, the are of the wires inside the modular switch connecting AC modules is considered in the area numbers provided in the table. These wires, when distributed are accounted in the area of the links.

Link Area Results

In this section, we discuss the different link area resources required by both the modular switch and the distributed switch. In the area occupied by a link two elements take part: the routed wires, and the repeaters inserted along the link. As mentioned in previous sections, a modular link differs from a distributed link in the number and size of repeaters and in the length of the

routed wires. For a modular switch the total area required by wires is given by

$$\text{WireArea}_{\text{modular}} = N * (W + S) * L \tag{4.2}$$

where N is the width of the link (set to 66), W is the width of the wire (equal to 0.0008mm), S is the interwire spacing (in our case is 0.0008mm), and L is the length of the wire (equal to 2.4mm). Then, the area occupied by the wires of a modular link is 0.2534mm^2. On the other hand, the total routed wire area of a distributed link can be calculated as

$$\text{WireArea}_{\text{distributed}} = N * (W + S) * \frac{L}{2} * 6 \tag{4.3}$$

The total area of the routed wires of the distributed link is equal to 0.7603mm^2. The area occupied by the wires of a distributed link is three times the area occupied by the wires of a modular link (remember the 3L overhead mentioned before). However, this increment in the routed wire area is not a major concern because – as it is said in Section 4.1.3 – the wires of a link can be routed over other functional blocks as SRAMs. Thus, routed link wires may not require dedicated area.

Table 4.4 shows the area occupied by the repeaters inserted along the link for a modular, distributed, distributed with half the links removed, and a DSB link. As the number of sublinks increases in the distributed and DSB links, the number of repeaters increases. As a consequence, area resources required by distributed switches increase with respect to the modular switch. In fact, the distributed link area is two times the modular link area. Furthermore, the DSB link has three times more area than the distributed link. In contrast, the distributed link when removing part of the links (X-improved) requires only an 8.3% more area than the modular link.

Whole Switch Area Results

Table 4.5 shows whole switch area results for the different architectures presented. The area of the whole switch is calculated considering the area of switch modules and the area of four links – link for the local port is considered inside the logic area of the switch. As shown, the distributed switch architecture presents a 22% more area than the modular switch. The DSB

	Area (um^2)
Modular link	2697.3
Distributed link	5845.32
X improved distributed link	2922.66
DSB link	12824.83

Table 4.4: Area occupied by the repeaters of a link

	Area (um^2)
Modular	39145
Distributed	47817
X improved distributed	41973
DSB	75733

Table 4.5: Area occupied by the whole switch

switch and DSB links area requirements are greater as the area of this switch is practically twice (98%) the area of the modular switch. On the contrary, the X-improved distributed switch occupies only 7.2% more area than the modular switch.

Therefore, the distributed switch approach has an impact on area that should not be overestimated. The benefits of this area overhead will come true with the power consumption values shown in the next section.

4.2.3 Power Consumption

Switch Modules Power Consumption

Table 4.6 shows the total power consumption of a modular switch and its equivalent distributed switch. The power consumption of the switch is broken down in three components: internal power, switching power and leakage power. The internal power consumption is due to the charging and discharging of the parasitic capacitance of the cells of the switch. The switching power is due to the charging and discharging of the load capacitance of the cells of the switch. Finally, the leakage power is the power consumed by a logic cell when no activity is registered. Table 4.6 shows also the clock tree power consumption.

Power (mW)	Internal	Switching	Leakage	Clock tree	Total
Modular	45.80	25.05	0.36	14.78	71.21
Distributed	42.20	20.25	0.30	12.05	62.7

Table 4.6: Power consumption of a modular and a distributed switch

The power consumption has been obtained using the Power Compiler tool by Synopsis. Results show that the distributed switch has a lower power consumption, obtaining a reduction of 11.95%. This saving in power is due to the synthesis tool improvement due to the simplicity of the distributed switch design. In the same way, the distributed switch achieves a remarkable power reduction in the clock tree. Concretely, clock tree power consumption of the modular switch is 14.78 mW, whereas the distributed switch clock tree power is only 12.05 mW, representing a 18.47% of power reduction.

Note that despite part of the power consumption improvement achieved by the distributed switch is a direct consequence of shifting some switch parts to the link, the power of the distributed link does not increase as a consequence of this, as described in next section.

Wire Power Consumption

In this section we analyze the power consumption of a single wire for different wire lengths. The electrical and physical parameters of the wire are presented in Section 4.1.3, and the link configuration – the number and size of repeaters – is the same as explained in Section 4.1.3. Table 4.7 shows the power consumption during a clock cycle for a single wire for different wire lengths. Power consumption has been obtained using *Cadence Virtuoso*. In the same way as electrical and physical parameters, power consumption of a wire depends on the input signal. In the absence of signal transition, the wire only consumes leakage power. In this case, the difference in power consumption, when the input is set to 1.1V or it is set to 0V is small. In contrast, when a logic input transition occurs, power consumption increases enormously due to the contribution of the dynamic power. In fact, the leakage power consumption is negligible with respect to the power consumption when an input transition takes place. Table 4.7 also shows that as the length of the wire decreases, the

	Power Consumption (uW)	
Wire length (mm)	Input transition	No input transition
2.4	358	0.800
1.2	152	0.350
0.6	69.3	0.162

Table 4.7: Power consumption in a clock cycle for different link lengths

power consumption of the wire decreases. In average, there exists a reduction of 55% in power consumption when reducing the length of the wire by two.

Link Power Consumption

Wire power consumption values in Table 4.7 are used to compute the power consumption of both the modular and the distributed link (see Section 4.1.3). Define N as the total number of wires in a link. N is set to 66 (see Section 4.1.3). Define $P_s(l)$ the total power consumption during a clock cycle of a single wire of length l when there is no input transition (third column in Table 4.7). Define $P_y(l)$ the total power consumption during a clock cycle of a wire of length l when there is an input transition (second column in Table 4.7).

The probability of an input signal transition depends on both the link utilization (L_u) and the switching activity. Note, that the switching activity (α) is defined as the average probability of bit transition when flits traverse the link. Typical values of switching activity are in the range of $0.3 - 0.5$ [99]. Therefore, we define T as the probability that an input signal modifies its value. This parameter takes into consideration both the probability of transmitting a flit and the switching activity. Concretely, the value of T is given by $T = L_u * \alpha$. Finally, the power consumption of a modular link (P_m) as defined in Section 4.1.3 is:

$$P_m(T) = N * T * P_y(2.4mm) + N * (1 - T) * P_s(2.4mm) \qquad (4.4)$$

The power consumption of the modular link has two terms. The first term defines link dynamic power consumption, while the second term defines the link static power consumption, that is, when no information is crossing the link.

To measure the power consumption of a distributed link (P_d), it is necessary to remark that a distributed link is made of six sublinks of length 1.2mm. Thus, when no information is crossing the link, the static power of the distributed link is the static power consumption of six sublinks of length 1.2mm. In contrast, as mentioned before, the dynamic power consumption depends on the number of flits that crosses the link (defined by the probability T). As shown in Figure 4.2, the distributed link is two stages long. Thus, any flit that crosses a distributed link, will traverse two small sublinks. Therefore, by assuming the same traffic rate as in the modular link (defined by probability T), the distributed link is traversed by a number of flits defined by $2*T$. Then, the power consumption of a distributed link is:

$$\begin{aligned} P_d(T) = N*2*T*P_y(1.2mm)+ \\ N*2*(1-T)*P_s(1.2mm) + N*4*P_s(1.2mm) \end{aligned} \tag{4.5}$$

Figure 4.8 shows the average power consumption per clock cycle of a modular and a distributed link for different values of T. For high traffic rates (high values of T), the distributed link consumes less power than the modular link. Concretely, for $T = 1$, the distributed link is able to save 3.47mW per clock cycle (up to 14.68%). On the contrary, for low traffic rates (low values of T), the higher number of routed wires in the distributed link causes the distributed link to consume more power. However, when low traffic rate is present in the network, the main contributor of the power consumption is the leakage power of the link, which is negligible in comparison with the dynamic power consumption of those links (see Section 4.2.3). For $T = 0$ the distributed link only consumes 138.6uW per clock cycle, while the modular link only consumes 52.8uW. The difference then is only 85uW per clock cycle. Anyway, for low link utilization rates, the different links over the network are inactive. In these long inactivity periods, links can be turned off reducing its power consumption to zero. In such network scenario, the link power consumption is mainly defined by the dynamic power consumption of the link. In that case the distributed link provides a power consumption decrement of 14.68%, which is the power consumption saving when only the dynamic power consumption is taken into account ($T = 1$).

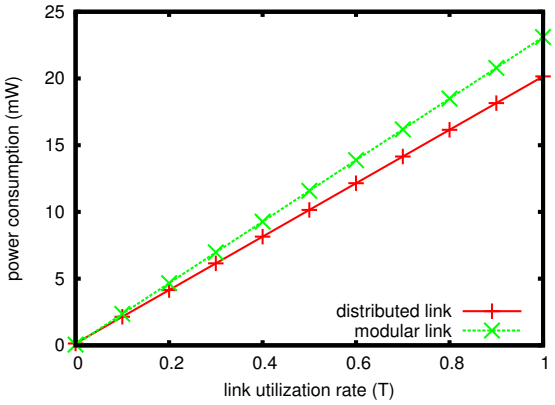


Figure 4.8: Power consumption of a modular link and a distributed link.

In order to minimize the static power consumption of the distributed link – when low link utilization rates exist – it is possible to turn off half the distributed link in all the X links (see Section 4.1.1). In this case, where half the wires are off, the power consumption is measured as:

$$\begin{aligned} P_x(T) = N * 2 * T * P_y(1.2mm) \\ +N * 2 * (1-T) * P_s(1.2mm) + N * P_s(1.2mm) \end{aligned} \tag{4.6}$$

The power consumption of the distributed link with half the wires off is identical to the power consumption of the distributed link for high link utilization rates (T). The difference in power consumption between both configurations is for low traffic rates. For low values of T, there is a reduction in the power consumption. This decrement in power consumption is 50% of the static power consumption, that is a decrement equal to 69.3uW for $T = 0$ which is only 16.5uW higher than the static power consumption of the modular switch.

Distributed Switch with Spread Buffers Link Power

In Section 4.1.2 a DSB switch is presented. When no contention exists on the link, flits cross the link using sublinks of length $L/2$ (*non-congestion sublink*). However, when there exists contention on the link, flits traverse the link by crossing sublinks of length $L/4$ (*congestion sublink*). Define C as the probability that a link is congested. Thus, the power consumption of a DSB link

(P_b) is computed as

$$P_b^c(T) = N * C * (4 * P_y(0.6mm) + 8 * P_s(0.6mm) + N * 6 * P_s(1.2mm)) + N * (1 - C) * (12 * P_s(0.6mm) + P_d(T)) \quad (4.7)$$

Note that in the absence of congestion ($C = 0$), the power consumption of the DSB link is higher than the power consumption of the distributed switch. This happens because the leakage power of the twelve sublinks of length $L/4$ take part. This difference can be reduced by turning off the *congestion sublink* in the absence of congestion. Similarly, if there exists congestion, it is possible to turn off the *non-congestion sublink*. Thus, the power consumption of the DSB switch is

$$P_b^c(T) = N * C * N * (4 * P_y(0.6mm) + N * 8 * P_s(0.6mm)) + N * (1 - C) * (P_d(T)) \quad (4.8)$$

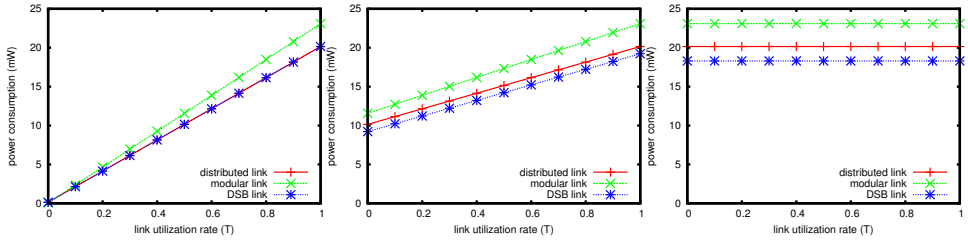
Similarly, the power consumption of the modular link and the distributed link can be computed as a function of the congestion. In this case, the power consumption of the modular switch is

$$P_m^c = N * C * P_m(T = 1) + N * (1 - C) * P_m(T) \quad (4.9)$$

and the power consumption of the distributed switch is

$$P_d^c = N * C * P_d(T = 1) + N * (1 - C) * P_d(T) \quad (4.10)$$

Figure 4.9 shows the average power consumption per clock cycle of a modular, distributed, and DSB link as a function of T. The figure shows link power consumption for three values of the congestion C parameter: when no congestion exists ($C = 0$), when half the time the link is congested ($C = 0.5$), and when the link is congested all the time ($C = 1$). Note that as congestion increases, the DSB switch has lower power consumption than the modular switch and the distributed switch. For $C = 1$, the DSB link consumes 19.31mW per clock cycle. That is a reduction of 0.75mW per clock cycle (3.78%) with respect to the distributed link, and a reduction in power consumption of 4.32mW (18.27%) with respect to the modular link. When no congestion is present ($C = 0$), the power consumption of the DSB link is identical to the distributed link.



(a) non congested link (C=0).(b) some congestion (C=0.5). (c) congested link (C=1).

Figure 4.9: Power consumption of a modular, distributed, and a DSB switch.

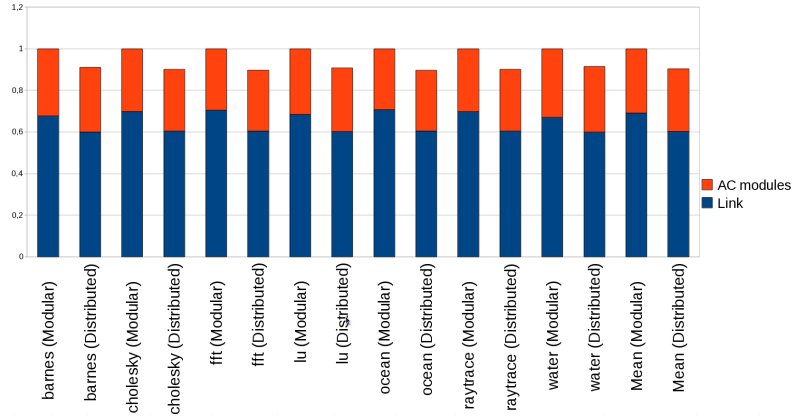
Parameter	Values
CMP	16 cores, tiled organization, Distributed Shared Memory
Cores	x86 Architecture, Area $2.4mm^2$, in-order, single thread
L1 inst cache	private, 64KB, 4-way
L1 data cache	private, 64KB, 4-way
L2 cache	shared, 512KB, 16-way

Table 4.8: CMP configuration.

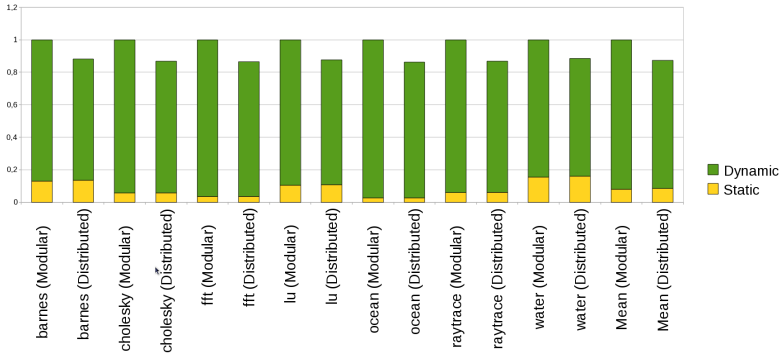
Application Power Consumption

In this section we analyze the energy results of two networks, a 4×4 2D-mesh implemented with modular switches, and a 4×4 2D-mesh implemented with distributed switches when running parallel applications on the CMP system. Different SPLASH-2 applications have been run in a 16-core modelled CMP system using GRAPHITE Simulator [93]. On top of the simulator, a detailed network simulator has been coupled to model the different network architectures. Additionally, a coherence layer models a typical MOESI invalidation-based coherence protocol. The configuration of the CMP system is summarized in Table 4.8. Remark that as the operating frequency of the switches is not considered, no differences in the execution time are obtained.

Figure 4.10 shows the network energy results for the different benchmarks. Figure 4.10(a) shows the network energy detailed as the sum of the link energy and the logic energy. In contrast, Figure 4.10(b) shows the network energy detailed as the sum of the static energy – due to the leakage power consumption – and the dynamic energy, due to the flits crossing the network. In any



(a) Logic and link network energy.



(b) Dynamic and static network energy.

Figure 4.10: Network Energy for different applications.

benchmark, results are normalized to the energy of the modular switch network. The first conclusion that can be extracted from Figure 4.10 is that, the distributed switch is most energy-efficient than the modular switch. This efficiency grows, on average, up to 13%. This is due as the distributed switch network reduces the link dynamic energy consumption which is the main component of the network energy, compensating the increment of the link static energy due to the increment in wires as described in the previous sections. These conclusions can be seen in Figure 4.10. As it shows the links are the components that consume more energy in the network. Its energy consumption is up to 69%. Additionally, the dynamic energy consumption is the main component of the network which represents up to 96%.

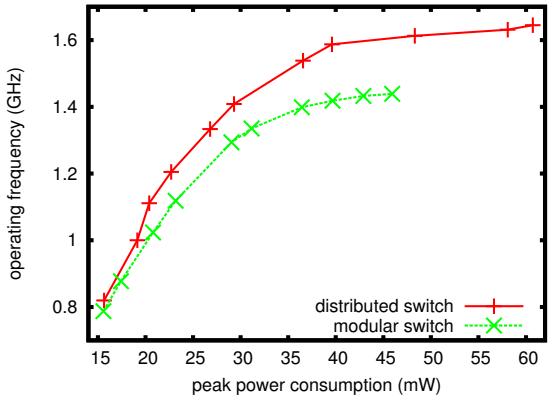
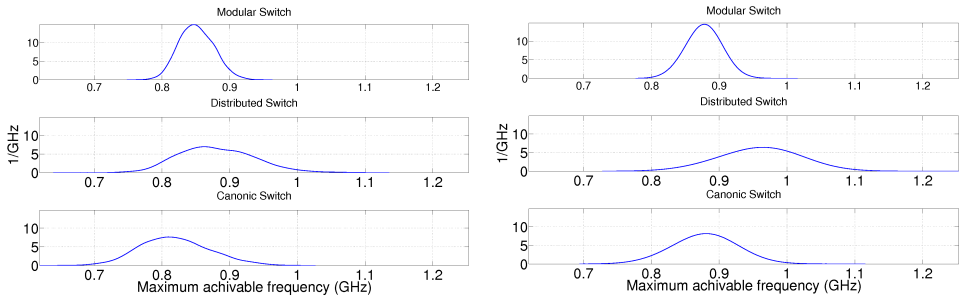


Figure 4.11: Operating frequency of a modular switch and a distributed switch.

4.2.4 Operating Frequency vs. Peak Power Consumption

In previous sections, we have analysed how by reducing link length, the power consumption of the distributed switch is reduced compared to the modular switch. The same happens for the power consumption of the distributed link over the power consumption of the modular link. Those results have been obtained when both scenarios work at the same operating frequency in order to allow a fair comparison. However, reducing link length has another important benefit. When shortening the link, the minimum link delay – that sets the total latency of the switch – can be reduced. That is, given a power budget, the distributed link is able to work at higher frequencies.

Figure 4.11 shows the operating frequency of the distributed switch and the modular switch. Note that, for the same operating frequency, the distributed switch consumes less power than the modular one. Furthermore, the distributed switch is able to reach an operating frequency higher than the modular switch. Concretely, the maximum operating frequency of the distributed switch is 1.65GHz, whereas the maximum operating frequency of the modular switch is 1.44GHz or, in other words, the maximum frequency of the distributed switch is 14.3% higher than that of the modular switch.



(a) Baseline (b) After Compensation

Figure 4.12: Probability density function of the maximum achievable frequency of switch architectures.

4.2.5 Variability Robustness

In this section we analyze the robustness of the proposed architecture against process variations. To do so, we injected systematic front-end variations, random front-end variations, and back-end variations in the canonical, the modular and the distributed switch architectures. For that purpose we used the variability model presented in [96] to generate 500 instances of each design. This model is able to accurately introduce variations into synthesized designs enabling to characterize the impact of process variations in circuit timing. The variability sources considered are: transistors channel length ($\sigma_{L_{eff}} = 12\%$), threshold voltage ($3\sigma_{V_{th}} = 33\%$) and the metal thickness ($3\sigma_t = 10\%$). These values reflect expected variations for a 45nm technology node [100].

Figure 4.12 shows the probability density function of the maximum achievable frequency for the canonical, the modular, and the distributed switch architectures. On one hand, it is possible to see how variations affect similarly the different designs (Figure 4.12(a)). This is explained by the fact that in all cases the link is involved in the critical path of the design. As a consequence of this, systematic variations caused similar effects to the components spread across the link due to spatial correlation. Concretely, the measured variation (σ/μ) – where μ is the free-variability operating frequency – for the canonical, the modular, and the distributed switch, are 14.5%, 13.8%,and 16.4%, respectively. These measurements show how the impact of parameter variations is slightly greater in the distributed architecture. However, there exist several

well known techniques that can be used to mitigate process variations [101]. Among the possible architectural solution to mitigate the impact of variations we chose the one given in [95]. This technique, an adaptive supply voltage link design, compensates the effect of variations in NoC performance by adjusting the voltage of the link. In this way, this technique is suitable for compensating variations in the architectures we are analysing as in all cases the design critical path is set by link delay. In order to analyze the ability of the different architectures to compensate variations using such technique we increased the link voltage of designs to 1.25V in those cases where the real operating frequency is lower than the design operating frequency. Figure 4.12(b) shows the maximum achievable frequency for the different architectures considered after adjusting link voltage. As shown in this figure, when the supply voltage of the link is increased more instances of the circuit are able to operate over the target frequency.

4.2.6 Fault Tolerance

Figure 4.13 shows the behaviour of the modular and the distributed link against a crosstalk noise, which is the main cause of a transient fault in a link. The Figure 4.13 shows the probability of having a bit error in a modular and a distributed link due to a crosstalk noise. The link error has been measured assuming the model developed in [102]. In this model, the noise error is assumed to have a voltage V_n defined by a normal distribution with variance σ_N. Then, the error probability of each single wire can be written as

$$\epsilon = Q(\frac{V_{dd}}{2 * \sigma_N}) \tag{4.11}$$

where $Q(x)$ is the *Gaussian pulse*. As it can be seen in Figure 4.13, the distributed link presents lower link error probability despite the increased number of wires. This is because a single wire in a distributed link is shorter, and hence, it suffers less crosstalk noise [103]. Then, the distributed link is more robust against transient errors than other longer links as those implemented in a modular or canonical switch.

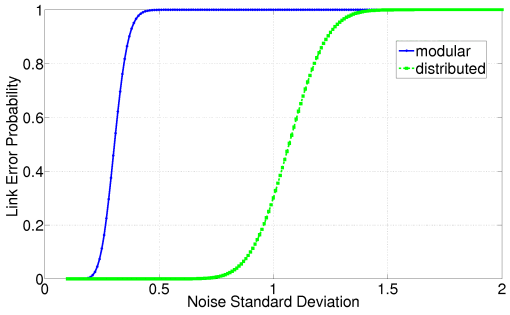
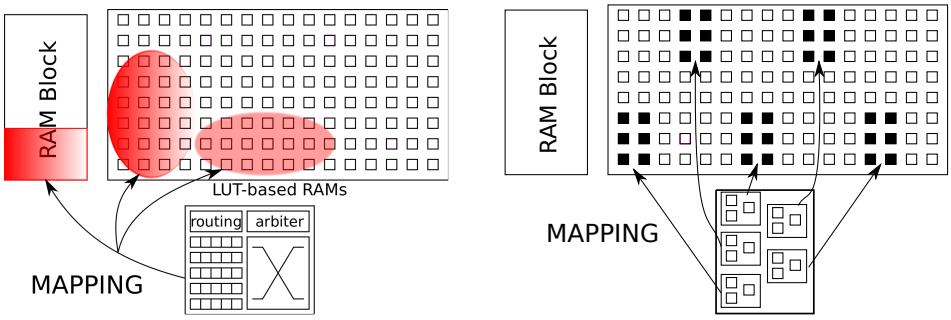


Figure 4.13: Crosstalk-induced link error probability.

4.3 Leverage Distributed Switch to FPGA Environment

In this section now we turn our attention to FPGAs. FPGA platforms are growing in terms of resources and capabilities. In recent years, researchers have focused on the possibility of synthesising an entire multi-core system on a FPGA, developing the concept of Network-on-FPGA. The first Network-on-FPGA mimicked generic NoC designs without considering the own characteristics of FPGAs [104]. In particular, the abundant number of wires and the structure of the FPGA memory resources. First, current FPGAs present abundant wiring resources, and even more important, long wires fulfilling delay constraints are feasible. With respect to storage resources, FPGAs present in general two kinds of possibilities. Using the FPGA logic to perform storage functions or using some dedicated slower SRAM blocks. A non-optimized network implementation can be seen in Figure 4.14(a). As it can be seen, a switch is placed by the placement tool by collocating the buffering resources onto the SRAM block which lead to slow network implementations.

In this scenario, the distributed switch concept can be extrapolated to an FPGA environment to optimize the network implementation. Thus, the FPGA distributed switch is a modular switch where each AC module is properly mapped bit-to-bit on the FPGA LUTs (see Figure 4.14(b)). The immediate effect is that buffering can be optimally performed by LUTs removing the necessity of using the SRAM blocks, and hence, the critical path length



(a) Non-optimized mapping of a canonical switch.

(b) Optimized mapping of a distributed switch.

Figure 4.14: Canonical and distributed FPGA switches.

decreases. Thus, packets are buffered, routed, and forwarded at the same time they are crossing the links of the network. The AC module suits perfectly to the logic blocks of the FPGA, thus optimizing the use of FPGA resources.

To validate the benefits of the FPGA distributed switch we synthesized and mapped the distributed and a canonical switch to a Virtex 5 FPGA chip. For the synthesis, mapping, and placement & routing of the designs we used the ISE 12.2 toolset of Xilinx. Table 4.9 shows the area measured in slices and delay measured in nanoseconds of our distributed switch with a canonical switch. The distributed switch is made of AC-2 modules. The canonical switch described in the Appendix A has been modified to have a pipeline depth equal to two. Two buffer sizes have been analyzed for the canonical case. First, two slots per buffer which is the minimum buffering size to fulfil the round trip time. Additionally, four buffer slots to equal the buffer resources to the distributed switch. Independently of the buffer size, the input buffers of the canonical switch are always mapped to distributed RAMs for better area efficiency after appropriately constrained by the synthesis/mapping tool. Both switches have five input/output ports. Finally, two datapaths widths have been analyzed, flit size of 16 and 32 bits.

As it can be seen in Table 4.9, for small datapaths of 16 bits, the proposed design requires the least amount of area, while in the case of 32 bits it is in the middle between the area of the canonical design with 2 and 4 buffers per input, respectively. Due to the small amount of logic in each distributed switch

Flit= 16bits	Canonical		Distributed	
	Area	Delay	Area	Delay
2buf/input	321	6.7	272	2.7
4buf/input	343	6.9		
Flit= 32bits	Canonical		Distributed	
	Area	Delay	Area	Delay
2buf/input	406	8.5	427	2.9
4buf/input	475	8.8		

Table 4.9: The area in slices and the delay in nanoseconds of the canonical and distributed switch.

pipeline stage, the critical path of the AC-based switch is 50% lower than the critical path of the canonical switch. Notice that the optimized placement of the distributed switch allows that despite the increment in the datapath the distributed switch delay remains almost constant meanwhile the critical path of the canonical switch strongly gets affected by the datapath widths.

4.4 Conclusions

In this chapter, a distributed switch for a 2D mesh architecture is presented. The distributed switch architecture is suitable for different environments as FPGAs or ASICs. In ASICs, the *distributed switch* moves the circuitry of a typical switch along the link, reducing maximum link length, and hence, reducing network critical path and power consumption. In an FPGA environment, the *distributed switch* suits the circuitry of a conventional switch on the fast logic blocks of the FPGA, removing the necessity of using the slowest SRAM blocks. When the switch placement is optimized, FPGA operating frequency is increased. In both cases, the performance improvement is at cost of increasing the resources used. In ASIC environment, the number of links inserted is increased. On FPGAs, the number of logic blocks used increases too.

In the next chapter we reuse the *distributed switch* strategy in order to implement high-radix architectures. By introducing the *distributed switch* on

these high-radix architectures, the known overheads of high-radix switch/network implementations are minimized or even removed. This is an additional dimension the distributed switch approach offers, which is fully exploited in the next two chapters.

Chapter 5

High-Radix Topologies

In this chapter, we leverage the distributed switch design to implement high-radix topologies as the *concentrated mesh* (C-mesh) or the *flattened butterfly* (FBF). We define a high-radix topology as the one using switches with more than five ports.

Distributing the switch over the links minimizes the high-radix switch design drawbacks. First, as switches are modularized and spread over the link, high-radix switch critical path is kept almost constant and independent of the link length and switch radix. Additionally, traffic concentration is minimized due to the distributed buffering of the distributed switch. Finally, infeasible long links involving distant switches become naturally pipelined.

As the switch radix increases, more modules are needed. Increasing the switch pipeline allows the designer to relax link restrictions increasing network performance. Thus, the *distributed switch* benefits shown in the previous chapter increase as the switch radix increases.

The aim of this chapter is to shown the behaviour of the high-radix topologies, and how the distributed switch approach, when applied, outperforms conventional switch designs. However, high-radix topologies – despite they overcome the commercially extended 2D mesh – do not provide the definitive solution to NoCs. In the next chapter we will be more aggressive in the design and will propose a single crossbar design.

5.1 Topology Comparison

In this section we compare different topologies and show that there is no a best configuration that overcomes the rest. Tables 5.1 and 5.2 show the main properties for each 64-node network topology analyzed. We compare the conventional 2D mesh network, a 4-ary 3-tree fat-tree network, the concentrated mesh (C-mesh) network, and the 4-ary 3-flat flattened butterfly network[1]. These topologies have been chosen as the most suitable ones for a 64-node network, as analyzed in [7]. For the sake of comparison, we assume all the topologies use switches implemented with AC modules in both cases, distributed over the links or placed at a bounded place. The effect of long links in performance is omitted.[2]

Table 5.1 shows the resources required for each topology. The first four columns show the number of switches, the switch degree, the number of ACs per output port, and the total number of ACs in the network. The switch degree determines the number of ACs per output port, and hence, the area of each switch. The fifth column shows the normalized area with respect to the 2D mesh topology. The area of each architecture is related to the number of ACs of each network. Note that, high radix topologies increase the area of a switch but reduce the number of switches. This balance allows the 4-ary 3-tree topology to have just an 11% more area and the C-mesh to have even a 10% less (compared to the 2D mesh baseline). Note that, the flattened butterfly topology presents an increment of 67% which should be taken into account.

The last two columns show the number of unidirectional links and the maximum link length between switches as a function of the intercore distance L. As it can be seen the 4-ary 3-flat and the C-mesh topologies reduce the number of links due to their concentration property, that is, the number of nodes connected to a switch (see Table 5.2). In contrast the fat-tree topology increases the number of links. As important as the number of links is the maximum link length. As the link length increases the delay of that link increases, and hence, the overall network performance suffers. In this sense, the flattened butterfly has the worst response, as its maximum link length is

[1] In this dissertation the flattened butterfly will be named as FBF.

[2] Note that any high-radix topology introduces longer links than the ones used in a conventional 2D mesh.

	Sw	Switch degree	ACs per output port	ACs total	area	Number of links	Link length
2D mesh	64	5	3	864	1	224	L
4-ary 3-tree	24	8	3/7	960	1.11	256	$2L$
4-ary 3-flat	16	10	11/15	1440	1.67	96	$3L$
C-mesh	16	8	7	784	0.90	48	L

Table 5.1: Area referred parameters for 64-end nodes for different topologies implemented using the modular switch design.

	Nodes /Switch	BB (flits/cycle)	BB Ejection Rate	Zero-load latency (cycles)	Switch cycles
2D mesh	1	1	0.5	12	2
4-ary 3-tree	4	3	0.33	11.375	2/3
4-ary 3-flat	4	32	1	10.000	4
C-mesh	4	8	1	10.500	3

Table 5.2: High-level parameters for 64-end nodes for different topologies implemented using the modular switch design.

three times the intercore distance. Some techniques have been proposed to minimize the impact of long links [72].

First columns of Table 5.2 show the maximum number of nodes per switch, the bisection bandwidth (BB), and the BB ejection rate for each architecture, that is the ratio of messages that crosses the link. Those metrics are related with the throughput of the network. As expected, the 4-ary 3-tree, and the 4-ary 3-flat network configurations exhibit the largest bisection bandwidth. Theoretically, the bisection bandwidth obtained is enough for sustaining the maximum traffic injection rate under uniform traffic distribution. However, this is not necessarily always true, as shown in [27]. The reason behind this is the traffic concentration that high-performance high-radix topologies suffer (see nodes/switch column in Table 5.2). In addition, flow control, switching and routing implementation in those topologies impacts the maximum achievable throughput. This will be seen later.

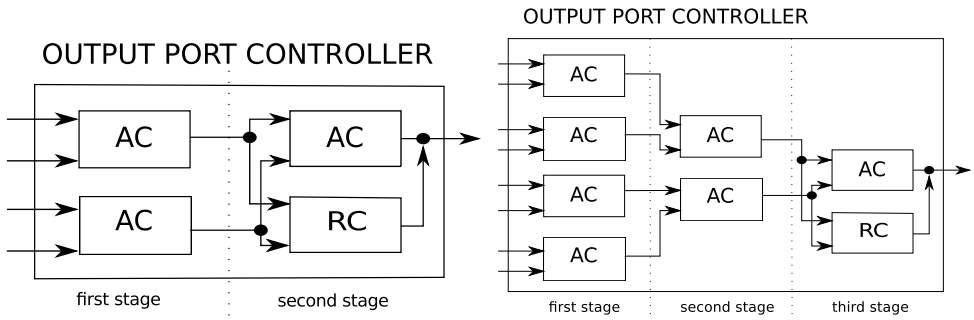
Finally, Table 5.2 shows the average zero-load latency measured in cycles. For the sake of comparison, switches have been modelled using AC modules, and consequently switches with different port degrees present different pipeline

depths. Switch pipeline depth is measured as $\log_2$(Switch Degree). As shown in the table, the best zero-load latency result is achieved by the 4-ary 3-flat. The reason behind this latency reduction is due to the optimum trade-off between switch hop latency and the number of switches/crossbars crossed.

In summary, the results presented in Table 5.1 and 5.2 show that there is not a suitable topology that overcomes the rest by looking its theoretical parameters. The flattened butterfly (FBF) has a lower zero-load latency and high throughput but suffers large penalty area. Additionally, this topology increases the number and maximum length of links which will affect the real performance –as it will be shown later. Similar problems present the 4-ary 3-tree topology. Its increment in area is lower than the flattened butterfly but, on the contrary, it is not able to reach a zero-load latency as the flattened butterfly achieves. Finally, the C-mesh topology is able to reduce the area and latency of the 2D mesh network but it reduces the BB which in practice means the throughput will be highly impacted. From now on, we will focus on two high-radix topologies, Cmesh and the flattened butterfly topologies.

5.2 Distributed High-Radix Topologies

In this section, we extend the design philosophy described in previous chapters and focus on implementing high-radix topologies as the C-mesh and the flattened butterfly. To implement a distributed high-radix topology it is necessary to leverage the previously described distributed switch. As told before, the distributed switch is an output port sliced switch. Thus, implementing a high-radix topology means to increase the number of independent output ports. The increment of input/output ports can be due to the increment in connectivity with other switches, as the flattened butterfly topology (see chapter 2), or the increment in the number of end-nodes connected to a single switch as the case of the C-mesh architecture (see chapter 2). By increasing the number of input/output ports, the number of stages – and hence the number of AC modules – of a single output port controller will be increased. Then, a high-radix distributed switch with N input/output ports has $N-1$ output port controllers attached to a single output port. Each output port controller is a mesh of trees where each stage is buffered. Each stage is made of a sum



(a) 4-to-1 output port controller. (b) 8-to-1 output port controller.

Figure 5.1: Output Port Controller.

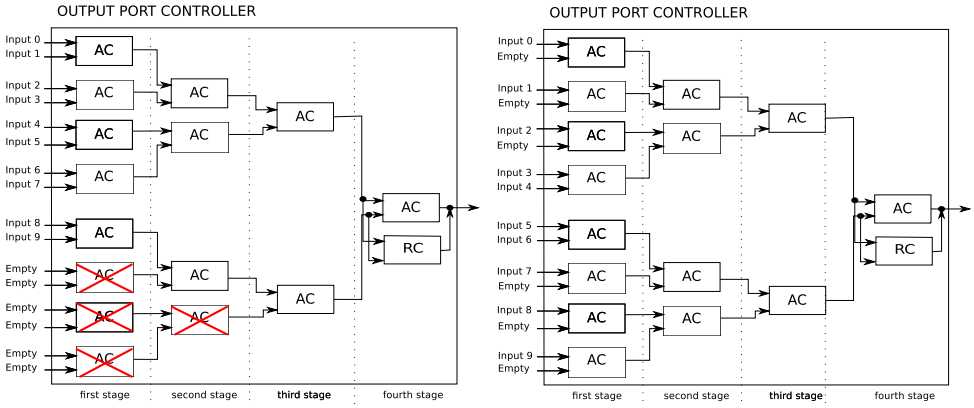
of AC modules working in parallel (see Figure 5.1). Each AC module has M input ports that define the AC module radix[3].

Figure 5.1 shows the main modifications when leveraging the 4-to-1 distributed switch suitable for a 2D mesh to a 8-to-1 high-radix switch suitable for a high-radix topology as for example the C-mesh. The AC module radix is set to 2, an hence, in this case the number of stages is fixed by the equation $S = \log_2 N$. As it can be seen, the number of AC modules is increased, increasing the area of the whole switch. In contrast, the critical path remains almost unalterable with respect to N because stages are pipelined. Then, independently of N the critical path depends on a single AC module[4]. Note that, the RC module remains at the last stage in order to minimize its impact.

A big concern arises when N is not divisible by two. Figure 5.2 shows a 16-to-1 output port controller to just handle 10 input ports. Figure 5.2(a) shows an asymmetric 16-to-1 output port controller, where 10 inputs are concentrated in the first 10 entries of the output port controller. By concentrating the input port to the first input of the output port controller, some AC modules can be removed – crossed AC modules – to minimize area overhead. In addition, some internal path unbalances occur. Note that, Inputs 8 and 9 cross the output port without conflicting their traffic with other inputs. That

[3]As analyzed in Chapters 3 and 4 it is possible to implement the distributed switch by using AC modules of different degree. However, in this chapter we focus on the AC module with degree 2.

[4]Some differences exist due to the synthesis tools inefficiencies when managing designs of different size.



(a) Asymmetric 10-to-1 output port controller.

(b) Symmetric 16-to-1 output port controller.

Figure 5.2: 16-to-1 output port controller to handle 10 input ports.

is, the fourth stage serves 50% of time to those inputs, meanwhile the rest of time is offered amongst the rest of inputs. To minimize unfairness, input signal reorganization can be done as shown in Figure 5.2(b). Notice that, despite that only 10 inputs signals are considered, all the 16-to-1 output port controller structure is required. In contrast, inputs are better balanced across the output port controller. Some unfairness exist as some input – input 3 and 4, and input 5 and 6 – contend at the first stage with other inputs, meanwhile the rest do not compete for resources until the second stage. However, traffic from different inputs is better balanced at the expense of not minimizing any buffering overhead.

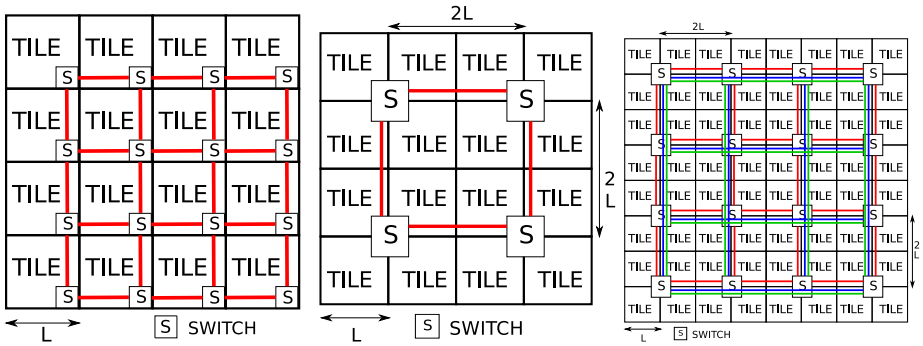
Table 5.3 shows the maximum number of ports required in a network, its corresponding number of stages in an output port controller, and the corresponding number of AC modules per output port when comparing a C-mesh, FBF and a conventional 2D mesh – no U-turns allowed. Notice that high-radix architectures increase the number of ports with respect to the 2D-mesh. This effect is due to the increment in the connectivity amongst switches – FBF – or due to the increment in end nodes connected to a switch – C-mesh and FBF. In this chapter, we assume that both topologies have 4 end nodes connected to any switch. Implementing high-radix topologies with a modular switch as described implies increasing the area of the switch (measured as a function of

Topology	Max. Ports	Pipeline Depth	AC modules per port
2D mesh	5	2	3
C-mesh	8	3	7
FBF	10	4	11/15

Table 5.3: Different topology parameters when implementing using a distributed switch.

the number of AC modules), and increasing the number of stages required to cross a single switch. However, this increment in the switch latency will be compensated by the reduction in the number of switches to cross, as it will be shown later. Note that, the number of AC modules per output port in a FBF architecture can take two values. This is due to the unbalance shown in Figure 5.2.

Similarly to the results presented in the previous chapters, the network critical path is fixed by the maximum link length between AC modules. As in the previous chapters, the maximum link length between AC modules is fixed by the maximum link length between switches divided by the number of stages of the switch. Figure 5.3 shows the maximum link length between switches, for the 2D mesh, the C-mesh, and a 64-node FBF topologies in a NoC where each tile occupies a square area of size $L \times L$. Remember that, the 2D mesh and C-mesh link length between switches remains constant with the number of nodes. As shown in Figure 5.3, the link length of the C-mesh doubles the link length of the 2D mesh because the switch of the C-mesh handles four tiles (see Figure 5.3(b)). More harmful is the increment in the link length between switches that suffers the FBF topology. As it can be seen in Figure 5.3(c), the FBF has a maximum link length between switches that is six times larger than the 2D mesh case. In general, in a squared network – same number of nodes in the X direction as in the Y direction – the relationship between the maximum link length between switches in a FBF and in a 2D mesh is equal to $\sqrt(N) - 2$, where N is the number of nodes in the network.



(a) 2D mesh floorplan. (b) C-mesh floorplan. (c) FBF floorplan.

Figure 5.3: Different topology floorplans.

Topology	Link Length between switches	Switch Depth	Link length
2D mesh	L	2	1/2 * L
C-mesh	$2 * L$	3	2/3 * L
FBF	$(N-2) * L$	4	(N-2)/4 * L

Table 5.4: Link length configuration for a 2D mesh, C-mesh, and FBF architectures.

5.3 Implementation Analysis

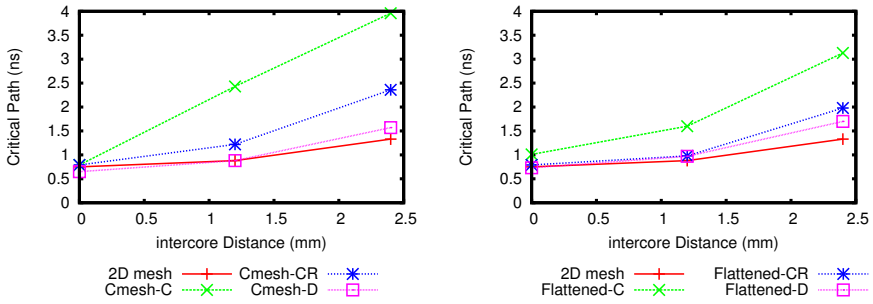
In this section, we analyze the area and critical path for the concentrated mesh and the flattened butterfly topologies implemented with the canonical switch and the distributed switch design methodology presented in this chapter. Those topologies implemented with the canonical switch are referred to as Cmesh-C and Flattened-C. Similarly, those topologies implemented by the distributed switch approach are referred to as Cmesh-D and Flattened-D. Additionally, both topologies have been implemented with the canonical switch but registering the long links, that is, any link longer than the inter-core distance is splitted into registered segments with length is at the inter-core distance. Buffering the link minimizes the maximum link length, and hence, improves the network critical path but increases network pipeline. A trade-off should be achieved to optimize zero-load latency as it will be shown in next sections. Those implementations with links registered are referred to as Cmesh-CR and Flattened-CR, respectively. Finally, these topologies are compared to the 2D

mesh case implemented with the canonical switch.

Figure 5.4(a) shows the critical path of the Cmesh topology implemented with different switch architectures when compared to the conventional 2D mesh. When no inter-core distance is considered, switch complexity fixes the critical path. As concluded in previous chapters, the modular switch has lower complexity than the canonical switch, and hence, lower critical path. In this sense, the Cmesh-D has the lowest critical path, despite it has a higher radix-degree than the 2D mesh canonical switch. In contrast, the differences in the critical path for the Cmesh-C and 2D mesh are small despite the higher complexity of the Cmesh switch. As the inter-core distance increases, link delay becomes critical. As it can be seen, the Cmesh-C is the most affected architecture. that is, a conventional high-radix switch gets highly affected by the link length. In fact, for a 2.4mm inter-core distance, the critical path of the Cmesh-C network is three times the critical path of the 2D mesh. Link length impact on the critical path can be minimized by registering the link. Then, Cmesh-CR reduces the critical path by 50% and 41% with respect to the non-registered Cmesh-C. As mentioned above, the Cmesh-D presents the lowest critical path amongst all the Cmesh architectures. The optimum placement of the AC modules allows the designer to implement a Cmesh network with no critical degradation when the inter-core distance is 1.2mm. For 2.4 mm inter-core distance, the Cmesh-D has a 18% increment in the critical path due to the inefficiencies of the Place&Route tool.[5]

Similar conclusions can be obtained when implementing the FBF. Figure 5.4(b) shows the critical path of the FBF topology implemented with different switch architectures and compared to the 2D mesh. The results for the 2D mesh architecture reflect the benefits achieved by the simpler implementation of the topology. It is the topology with the lowest critical path. As the inter-core distance increases, the difference with the rest of architectures increases. On the other hand, the critical path of the Flattened-C is the highest. This is due to its higher switch complexity, and the long links required. Registering the long links minimizes their impact. Thus, the flattened-CR highly reduces the critical path of its baseline flattened-C. As the maximum link length increases, the difference increases because long link critical path

[5]In theory, the critical path of both networks should be identical.



(a) Cmesh. (b) Flattened butterfly.

Figure 5.4: Critical path for high-radix topologies and the 2D mesh for a 64-node network when implemented using different switch architectures.

gets worse. In this sense, distributing the switch along the link has the same effect on the critical path than registering the links, but the network hop count is not increased as it will be shown later. Thus, the flattened-D has similar critical path than the flattened-CR for small inter-core distance. As the inter-core distance increases, the lower switch complexity allows the distributed switch to outperform the flattened-CR in terms of critical path.

Figure 5.5 shows the area of a switch used for the different network architectures. In any case, the larger switch area is shown[6]. As expected, the distributed switch for high-radix topologies occupies more area than the equivalent canonical switch. The Cmesh-C switch is a 65% larger than the 2D mesh-C switch. Remember that the Cmesh-C switch has three input/output extra ports which represents an increment of 60%. Registering the links – Cmesh-CR – implies to increase the resources used up to 24%. The Cmesh-D is a 23.4% larger than the Cmesh-C which is similar to the difference between the canonical switch and the distributed switch. Remember that, the Cmesh-D consumes less resources than the Cmesh-CR. The flattened-C switch is 121% larger than the canonical switch. Registering the links implies to increase the area by 24%. Finally, the flattened-D switch is a 42% larger than the flattened-C. Notice that, the difference between the distributed switch and

[6]Some area reduction is achieved for those switches placed at the network boundaries where the number of input/output ports is lower than the port of the switches placed at the center of the network.

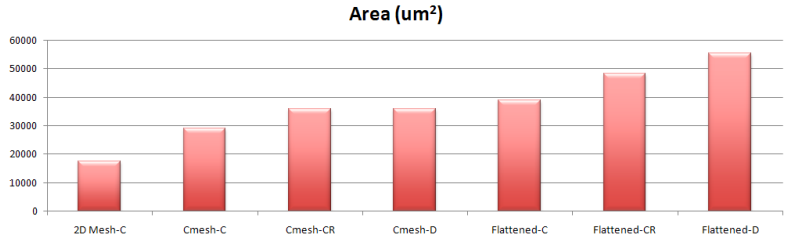


Figure 5.5: Canonical and distributed switch area for different architectures.

the canonical switch increases as the number of input/output ports increase due to the tree-based structure of the distributed switch.

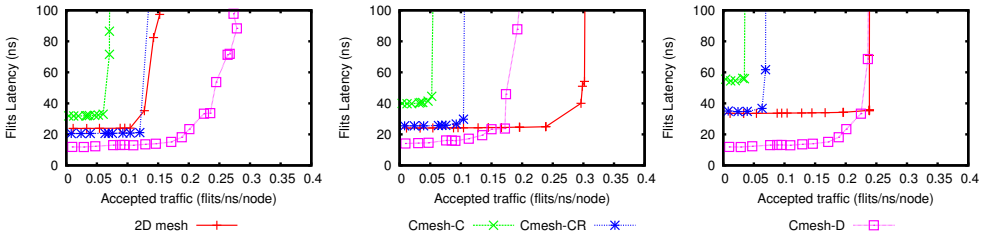
Then, referred to the implementation parameters – critical path and area – several conclusions can be obtained. First, the distributed switch has the shortest critical path. In fact, its critical path is close to the critical path to the simpler 2D mesh switch. That is, the distributed switch almost removes the dependency between the critical path with the switch radix and link length. In fact, the higher the switch radix the higher the benefits of distributing the switch. Notice that, this behaviour can be obtained in part by pipelining the link – as the elastic buffers do. However, this last solution implies to increase the network hop count. Thus, as analyzed in the previous chapter, the distributed switch can be seen as an elastic switch that reuses the buffering of the pipelined link to perform switching tasks. In contrast, switch area is increased. For large switch radix, the distributed switch area exceeds the area of other switch architectures.

5.4 Network Performance

In order to analyze the performance of high-radix topologies, we have leveraged our cycle-accurate network simulator to model these topologies. Same simulating parameters as those used in Chapter 3 have been chosen. Additionally, clock cycle has been adjusted to each topology implemented by using the results obtained in the previous section. The tile has been modelled as an ideal end node that injects messages into the network at a constant injection rate where each end node is made of a single queue that stores the flits that

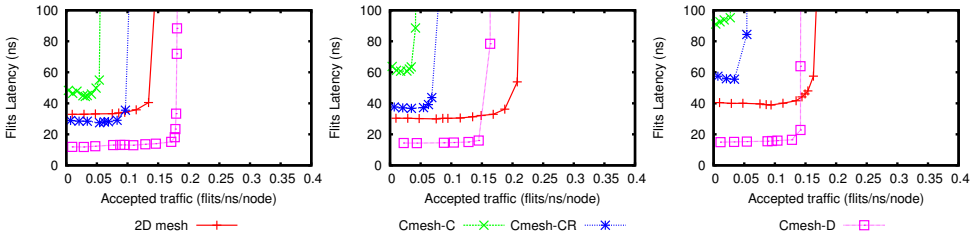
are waiting to be transmitted. Network size is set to 64 nodes. Three types of traffic distribution have been used: uniform, bit reversal, and bit complement. Flit size has been set to 64 bits. Two message generation scenarios are considered: messages that are 1-flit long, and bimodal traffic where there exist two messages sizes. Short messages 1-flit long, and long messages 9-flit long. Short messages represent 70% of messages injected. Network performance is measured as the accepted traffic rate (flits per nanosecond per node) and the average end-to-end flit latency measured in nanoseconds. Intercore distance is set to 1.2 mm. XY routing algorithm is used.

Figures 5.6 and 5.7 show the accepted traffic and the network latency for different Cmesh network implementations when compared with the 2D mesh implemented with the canonical switch – 2D mesh-C. Surprisingly, as it can be seen in both figures, despite the increment in resources of the Cmesh networks, those architectures do not clearly outperform the 2D mesh-C topology. The Cmesh-C is in fact the one with the poorest performance achieved as it has the highest latency and the lowest throughput. Registering the links – Cmesh-CR – reduces the latency with respect to the non-registered architecture – Cmesh-C. However, the Cmesh-CR network latency is almost equal than the 2D mesh-C meanwhile, the 2D mesh-C outperform Cmesh-CR in terms of throughput. There exist two combined reasons that explain this loss in performance. First, the increment in the network critical path that suffer Cmesh-C and Cmesh-CR makes the network latency to be higher when compared to the 2D mesh-C network latency. Additionally, the increment in the switch radix without increasing path diversity makes throughput to decrease due to traffic concentration at the switch inputs. Distributing the logic – Cmesh-D – along the links has two effects. First, network latency gets reduced due to the large critical path reduction. In this sense, Cmesh-D network latency clearly improves the 2D mesh-C latency. Additionally, the throughput of the Cmesh-D outperforms other Cmesh configurations due to the distributed buffering. However, it can not be concluded that Cmesh-D outperforms the 2D mesh-C in terms of throughput. For uniform traffic, the traffic concentration due to the increment in the switch radix makes the Cmesh-D throughput to be lower than the 2D mesh-C throughput. In contrast, for the bit reversal traffic pattern, Cmesh-D outperforms the 2D mesh-C, meanwhile for the bit complement



(a) Bit-reversal traffic. (b) Uniform traffic. (c) Bit-complement traffic.

Figure 5.6: Load-latency graph for a 64-node 2D mesh-C, Cmesh-C, Cmesh-CR and Cmesh-D networks. Messages are one flit long.



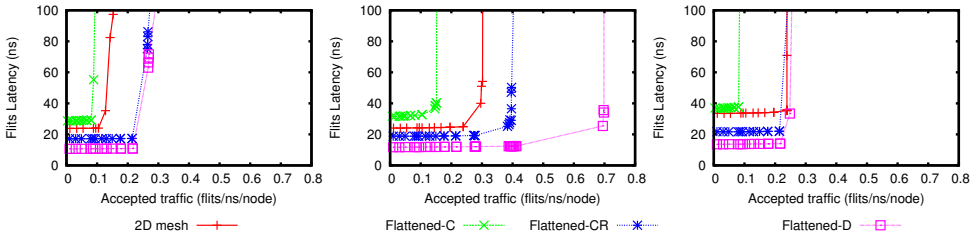
(a) Bit-reversal traffic. (b) Uniform traffic. (c) Bit-complement traffic.

Figure 5.7: Load-latency graph for a 64-node 2D mesh-C, Cmesh-C, Cmesh-CR and Cmesh-D networks. Bimodal traffic considered.

traffic pattern, both architectures behave similar[7].

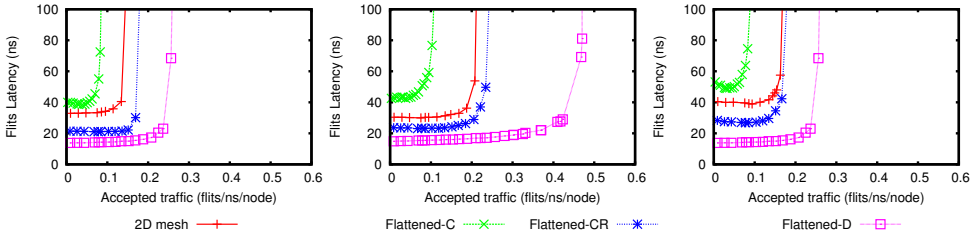
Figures 5.8 and 5.9 show the accepted traffic and the network latency for different FBF network implementations when compared with the 2D mesh implemented with the canonical switch – 2D mesh-C. As it can be seen, similar conclusions can be obtained. Despite the increment in resources the Flattened-C configuration presents the worst performance, due to its large critical path. For the same reason, the Flattened-C presents lower throughput than the 2D mesh-C. In this case, the FBF configuration also suffers from traffic concentration but is not as pronounced as the Cmesh architecture. Thus, traffic concentration is not the main cause of the lower performance of the Flattened-C configuration. It can be proved by analysing the Flattened-CR behaviour. As it can be seen, the Flattened-CR configuration mostly outperforms the 2D

[7]Cmesh throughput can be increased by using non-minimal or more complex routing algorithms. However, it is out of the scope of this dissertation.



(a) Bit-reversal traffic. (b) Uniform traffic. (c) Bit-complement traffic.

Figure 5.8: Load-latency graphs for a 64-node 2D mesh-C, Flattened-C, Flattened-CR and Flattened-D networks. Messages are one flit long.



(a) Bit-reversal traffic. (b) Uniform traffic. (c) Bit-complement traffic.

Figure 5.9: Load-latency graphs for a 64-node 2D mesh-C, Flattened-C, Flattened-CR and Flattened-D networks. Bimodal traffic considered.

mesh-C in terms of latency and throughput. In this case, distributing the logic presents visible benefits. Notice that, the Flattened-D is the architecture with the lowest latency and the highest throughput in all the cases analyzed. Distributing the logic over the link, allows to optimize the switch implementation and thus, enabling to get the most from the chosen topology[8].

When analysing both architectures – Cmesh and FBF – the benefits of using the distributed switch outperforms the benefits shown in the previous chapter. First, high-radix topologies are limited by its longer links. Additionally, increasing the switch radix means to increase the switch pipeline depth which allows the designer to spread the switch in a more optimized manner. As it can be seen, the distributed switch approach clearly outperforms other solutions like registering the links. This is due to the distributed switch reducing

[8]As before, more complex routing algorithms leverage the flattened architecture performance which is out of the scope of this dissertation.

the maximum link length without introducing extra cycles.

5.5 Conclusions

In this chapter, the distributed switch is leveraged to fulfill high-radix topology demands. As shown in this chapter, the benefits of the distributed switch become larger as the network architecture becomes complex. First, because more complex architectures require the use of longer links which are the main drawback of conventional designs. Second, they also require the use of high-radix switches which forces the switch pipeline to become deeper, allowing the designer to distribute the switch efficiently over the link.

Despite these improvements, the benefits of these architectures against a conventional 2D mesh are not spectacular if a simple routing algorithm is used. In the next chapter, we redefine the distributed switch to implement a distributed modular crossbar which clearly outperforms previous proposals.

Chapter 6

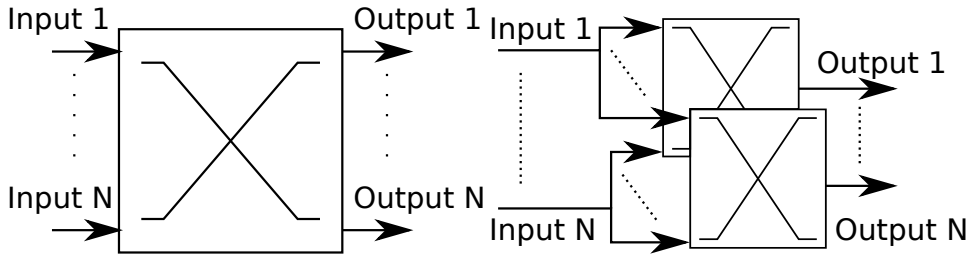
A Distributed Crossbar

In this chapter we reuse the modular design in order to develop a floorplan-aware crossbar NoC design. Basically, a large $N \times N$ crossbar is made of as set of AC modules. AC modules are properly placed along the chip in order to minimize link length, as done in the distributed switch approach in the previous chapter. Thus, large crossbar wiring, logic, and arbitration is split and spread along the chip. Thus, the contribution of this chapter is the most extreme design of the distributed switch approach, focusing on a single chip-level crossbar network.

This chapter is organized as follows. First, we describe the distributed crossbar NoC design. Then, we provide an evaluation of the distributed crossbar NoC compared to 2D architectures as the 2D mesh, the C-mesh, and the flattened butterfly network described in previous chapters.

6.1 A Distributed Pipelined Decoupled Crossbar

The *Distributed Crossbar* (DC) design is a *distributed pipelined decoupled crossbar* design made of AC modules (see Chapter 3). It is distributed as the crossbar logic is spread over the chip. It is pipelined as the flits will travel along the crossbar passing through different stages implemented with small multiplexers and registers. It is decoupled as each output of the crossbar will be totally decoupled from the rest (resembling a port slicing implementation approach).



(a) Conventional NxN crossbar. (b) Decoupled NxN crossbar.

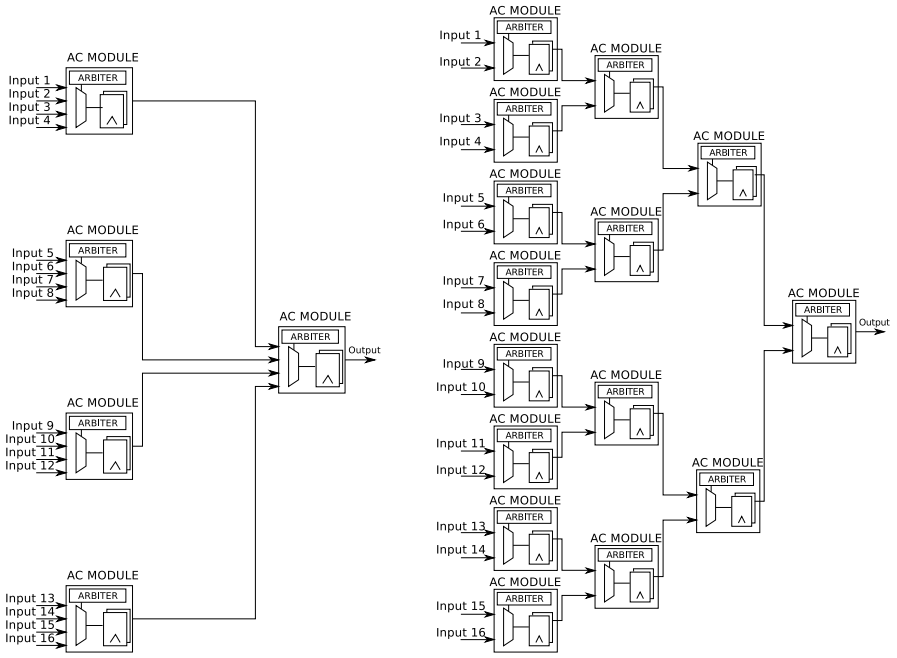
Figure 6.1: Conventional and decoupled NxN crossbars.

In a die with N nodes, a conventional $N \times N$ crossbar can be *decoupled* into N independent $N \times 1$ subnetworks, where each independent $N \times 1$ subnetwork connects any node to a given destination node (port slicing design) [48]. Figure 6.1 shows the decoupling process. Decoupling the crossbar allows the synthesis tool to improve its efficiency and, hence, to increase the maximum operating frequency of the design [47]. Furthermore, it allows to distribute the logic of the interconnect across the chip as it will be shown later. A crossbar implemented with decoupled $N \times 1$ subnetworks, theoretically allows achieving 1 flit/cycle/output. This can be achieved with low-complex, independent, and local arbiters.

Once the $N \times N$ crossbar is partitioned into N $N \times 1$ independent subnetworks, we implement each $N \times 1$ subnetwork with a set of AC modules of radix M. By using a simple and local decision, floorplan is simplified, and hence, a faster design can be obtained as shown later. Notice that, the RC module has been removed, as routing is performed at the NIC, once a flit gets injected into the proper subnetwork no routing is required. By leveraging the AC module, an $N \times 1$ network can easily be pipelined with S stages. Notice that N, M, and S are closely interrelated. Indeed, we can deduce

$$S = log_M N \tag{6.1}$$

Figure 6.2 shows a 16×1 network designed with AC modules and pipelined into 2 and 4 stages. As shown, the 16×1 network is designed as a tree of identical AC modules. N of these tree structures working in parallel are referred to as *DCx*, where x is the degree of the AC modules. Thus, Figure 6.2(a) shows



(a) S=2, M=4 (b) S=4, M=2

Figure 6.2: Pipelined decoupled 16×1 network (N=16) with different organizations.

part of a DC4 configuration whereas Figure 6.2(b) shows part of a DC2 one. Similarly, we also implement a crossbar with ACs assuming $M = 16$, referred to as *DC16* (not shown). In all the cases, 2-flit buffers are used in all the ACs in all the DC networks. Thus, a DC16 network will have lower buffering capacity than a DC2 network because DC16 has less AC modules.

6.1.1 DC Floorplan

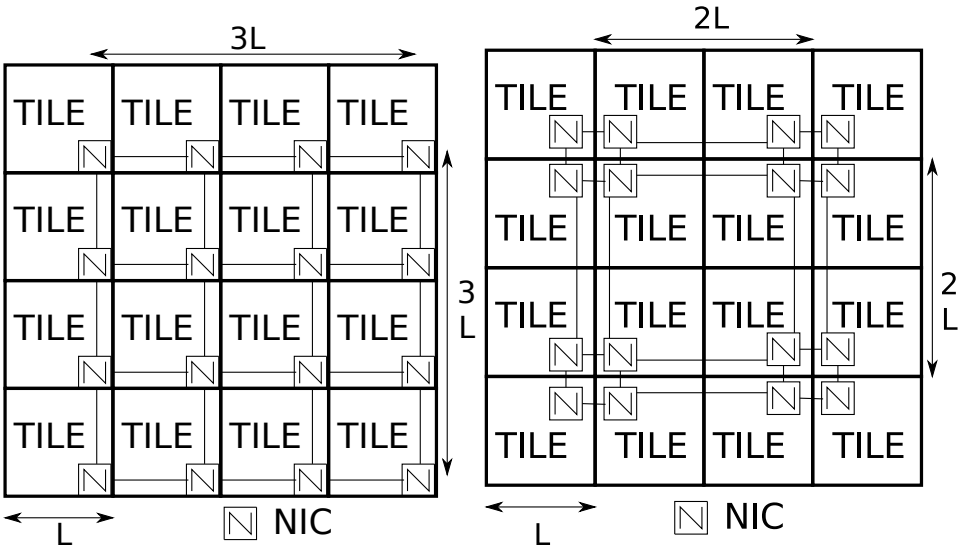
Once the $N \times N$ crossbar is designed as a set of independent AC-connected $N \times 1$ networks, the next step is to smartly distribute the AC modules across the chip in order to minimize the impact of long interconnects and, hence, optimizing the trade-off between maximum operating frequency and power consumption. This is the most critical step in the design (and the most novel one), since a proper distribution of AC modules will minimize the impact of long links on performance.

The critical path of our pipelined decoupled crossbar is made of the delay of the AC logic and the delay of data and control bits along the link between two adjacent AC modules, as can be derived from Figure 6.2. The delay of the AC module depends mainly on M (number of inputs) whereas the delay of the wire depends mainly on its length. Thus, the maximum operating frequency of the distributed crossbar is mainly set by M and the length of the links between ACs.

To minimize the impact of long wires and thus maximizing the operating frequency of the interconnect, the AC modules of a given $N \times 1$ network are physically distributed through the chip. In a conventional mesh-based CMP system, the distance between adjacent nodes is the inter core distance L (see Figure 6.3(a)). Inter core distance is set by the distance of two adjacent Network Interface Controllers (NIC). As it can be seen, the maximum distance between NICs in a regular floorplan for a 4×4 mesh is $3L$ in each direction, and a maximum distance of $6L$ (we assume horizontal and vertical links only). In this floorplan each switch is connected to a single NIC.

On the other hand, a balanced distribution of AC modules is achieved when the distance between two connected AC modules is bounded by D/S, where D is the maximum physical distance between two AC modules in an $N \times 1$ subnetwork, and S is the number of pipeline stages. In order to minimize D, we have reorganized the tiles in order to concentrate the injection/ejection ports to the network as shown in Figure 6.3(b). We call this mapping *concentrated floorplan*. D is now reduced down to $4L$ ($2L$ in each direction).

The *concentrated floorplan* approach is not suitable for a conventional 2D-mesh network. As shown in Figure 6.3, the distance between adjacent NICs is increased from L in a conventional regular floorplan to $2L$ in the *concentrated floorplan*. Therefore, the operating frequency of a 2D-mesh network would be impacted (or a pipelined link design would be required). Table 6.1 shows the maximum link length between adjacent AC modules (D/S) for different DC networks and different CMP sizes: 16 nodes and 64 nodes. Table 6.1 also shows the constant length (L) between NICs for a standard 2-D mesh. As can be seen, although network size increases quadratically, the distance between AC modules increases linearly. Also, AC modules with less number of ports (e.g. those used in DC2) are more effective (as the number of stages



(a) 4x4 2D mesh floorplan. (b) 16 nodes concentrated floorplan.

Figure 6.3: 2D mesh and a concentrated floorplan.

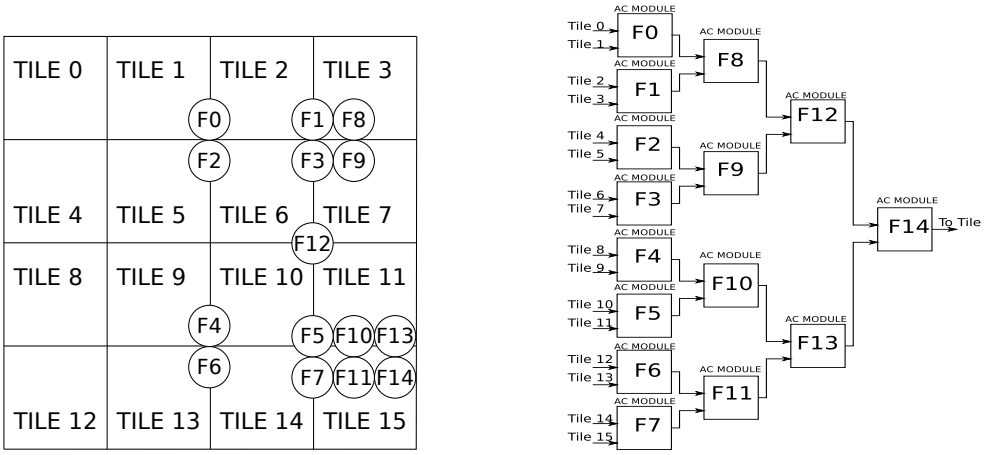
	NoC	DC2	DC4	DC16
4×4 network ($D = 4L$)	L	L	$2L$	$4L$
8×8 network ($D = 12L$)	L	$2L$	$4L$	$6L$
16×16 network ($D = 28L$)	L	$3.5L$	$7L$	$14L$

Table 6.1: Maximum link length between adjacent AC modules. In the DCx designs, the concentrated floorplan is used.

is increased).

Once the floorplan for the location of the injection/ejection ports is fixed, and the maximum distance between AC modules is set, then, a proper placement of the AC modules is needed for each $N \times 1$ subnetwork. Figure 6.4 shows the proposed placement scheme for a 16×1 network built from AC2 modules ($M = 2$, $S = 4$, $D = 4L$; DC2 network) where the output node is TILE 15. The remaining 15 16×1 networks have been manually distributed in a similar way, always guaranteeing that the maximum distance between AC modules is lower or equal to L ($D/S = 4L/4$).

Figure 6.5 shows the placement schematic of a 16-node DC2 network. Scale



(a) AC placement.

(b) 16×1 network.

Figure 6.4: AC module placement for a 16×1 subnetwork, corresponding to TILE15. $M = 2$, $S = 4$.

is set for a tile of 1.2mm×1.2mm. As each AC module is functionally independent, AC modules from different subnetworks can be grouped. As it can be seen, in a 4×4 DC2 networks, all AC modules are placed in just eight regions.

Our DC design philosophy described in this section can be applied to any NoC environment independently of the number of cores, their size, shape or distribution along the chip. Basically, the number of cores and its maximum distance determines the optimum placement of the AC modules, and hence, network performance. In contrast, a centralized crossbar design is largely influenced by the location, size, and shape of cores which could compromise the original centralized design. With our philosophy, AC modules are always appropriately spread over the chip, ensuring that maximum link length is D/S.

6.1.2 Flow Control

The flow control signalling that manages the communication between the distributed crossbar modules is identical to the flow control of the modular switch presented in Chapter 3. However, the distributed crossbar has an interesting property, an AC module is just connected to a single downstream AC mod-

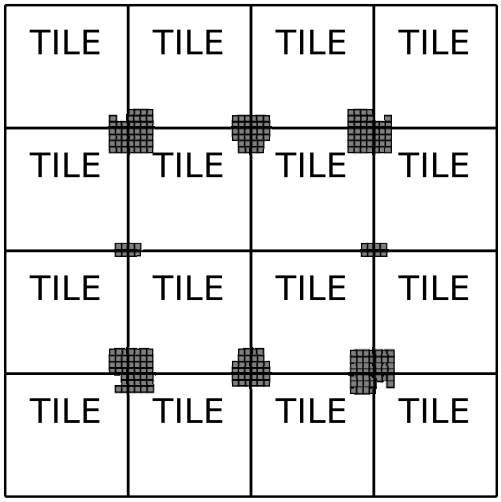


Figure 6.5: 16-node AC module placement.

ule[1]. By assuming this, it is possible to reduce the flow control critical path by replicating the arbitration logic of an AC module on its adjacent upstream AC modules. Thus, each upstream AC module has two arbiters. The main arbiter computes which input wins its buffer resources. The second arbiter obtains the same result as the main arbiter of the connected downstream AC module. By doing this, the upstream AC module knows if its flits will be accepted at the downstream AC module without needing to wait for the acknowledgement from the downstream AC module. Figure 6.6 shows the improved flow control mechanism used in this chapter. In this design, *downstreamFCi* signals are generated by using only the buffering state as it will be shown later. Additionally, the second arbiter must receive the *upstreamVALIDi* signal from the rest of upstream AC modules. As it can be seen from the figure, now the flow control signalling crosses at most one link length, thus, reducing the impact of the flow control on the critical path. On the contrary, the resources of the AC module increases, as the second arbiter mimics the behaviour of the upstream AC module. This optimization must be done in those scenarios where link length fixes the timing or power constraint of the network.

This optimization reduces the critical path of the distributed crossbar, but

[1]Remember that, the second stage AC module of the modular switch is connected to an AC module at each output port of the next downstream switch.

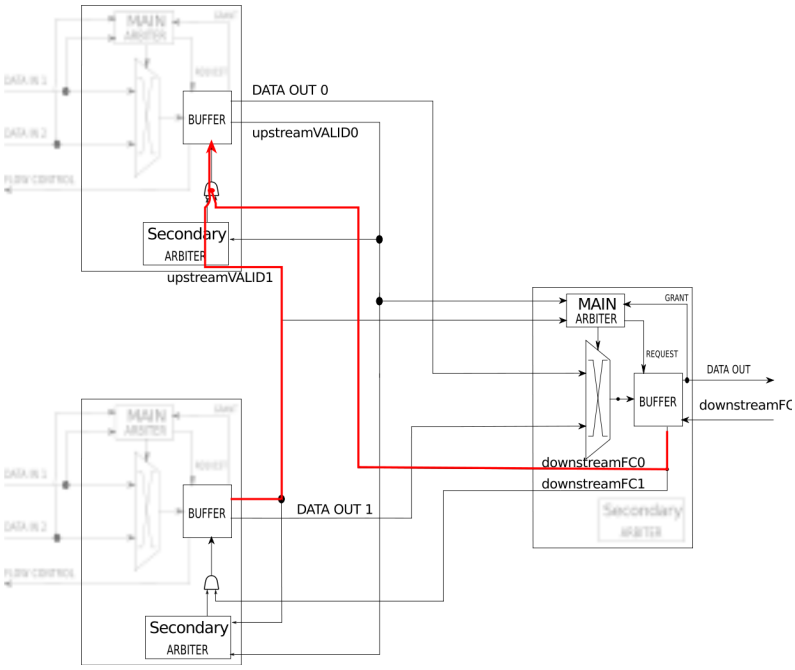


Figure 6.6: Improved flow control signalling generation.

can not be applied in a high-radix topology implemented with the distributed switch design. An example is provided in Figure 6.7. This figure shows the signals involved in an optimized AC module flow control environment. The last AC module in a switch output port is connected to one AC module at each output port at the next switch it is connected to. This means that all the arbiters at these AC modules would need to be replicated. In addition, the signalling between the upstream switches will need longer wires, as the AC modules will be spread over the links. Figure 6.7 shows the signalling involved for a single AC module in a 2D mesh where U-turns are not allowed. Notice that, *upstreamVALID* signal must traverse two links in a clock cycle. Thus, introducing the optimization described above does not reduce the critical path but even increases the critical path due to the use of longer signal wirings. This does not happen in the DC approach since AC modules are configured to build a tree-configuration, where AC modules connected to the same downstream AC module are at the same tree level and close enough in the concentrated floorplan.

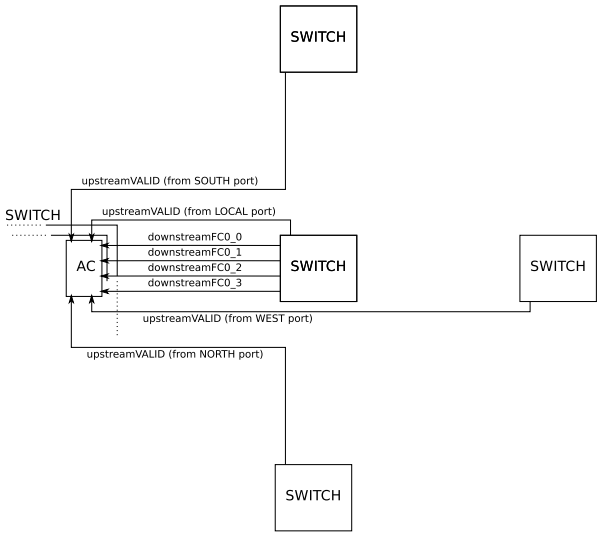
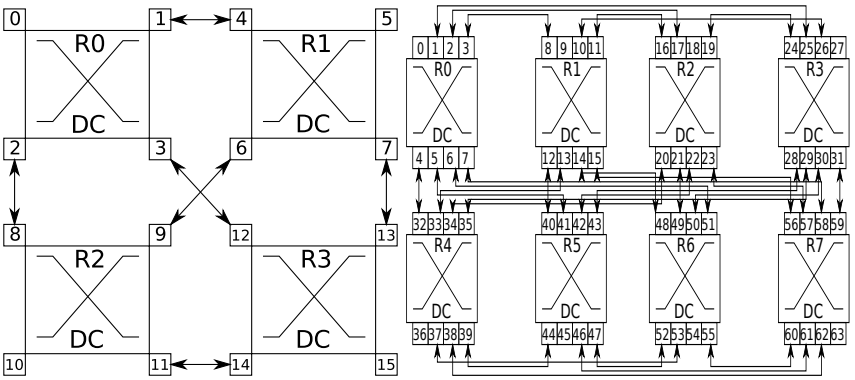


Figure 6.7: Unfeasible flow control optimization for the modular switch.

6.1.3 Hierarchical Distributed Crossbar

As it will be shown in next sections, the distributed crossbar design outperforms other NoC architectures at the expense of increasing the resources. For large systems it is possible to implement the distributed crossbar using a hierarchical approach, minimizing its costs. Thus, the *Hierarchical Distributed Crossbar* (HDC) NoC architecture will be able to retrieve the maximum performance of the *Distributed Crossbar* at the same time that area is bounded to reasonable limits.

In an HDC network sets of nodes are grouped in regions or subnetworks. Each region is *per se* a *DC* network. All DC subnetworks are then connected with each other using a fully connected network, with direct links between each DC network. Figure 6.8 shows the *Hierarchical Distributed Crossbar* scheme for a 16- and 64-node network, where each region has $\sqrt{N}$ nodes. Figure 6.8(a) shows a 16-node HDC network where 4-node DCs are used. Figure 6.8(b) shows a 64-node *HDC* network made of 8 regions with 8 nodes each. Both HDC networks are implemented with a hierarchical approach with two levels (at the first level the DC network and at the second level the fully connected network).



(a) 16-node network. (b) 64-node network.

Figure 6.8: 16-node and 64-node HDC networks.

Each region in the HDC network is directly connected to the rest of other regions. Thus, at each subregion there exists a special node called *gateway* to reach other subregions. Therefore, these nodes have to deal with both intra-region messages and inter-region messages. Meanwhile intra-region messages are directly sent to its destination by means of the local *DC* network, inter-region messages are forwarded through two DC networks, one in the source local region to reach the gateway, then it crosses a direct link to reach the next gateway, and finally, it crosses the destination region. Note that, each gateway receives messages from all the nodes in a region, and hence, a gateway may become a bottleneck. Although the HDC approach can use different hierarchy levels, we bound the analyzed cases to HDC networks with only two levels.

In order to enable inter-region communication the message injector module of the gateway has to be able to handle both inter- and intra-region messages. Figure 6.9 shows an schematic of how the interconnection of regions is performed. As shown in this figure, at the inputs of the regions, used to build the *DCs*, the messages may come from both local input queues (IN_i) and external region queues (REG_i). In this way the additional hardware resources required for implementing the HDC networks are the additional queuing resources required in the input gateways to store the flits coming from other regions, the multiplexors that enable to handle both inter- and intra-region messages, and the repeaters and wiring resources required to build the inter-region links.

Once the levels of hierarchy have been chosen, the parameter that defines

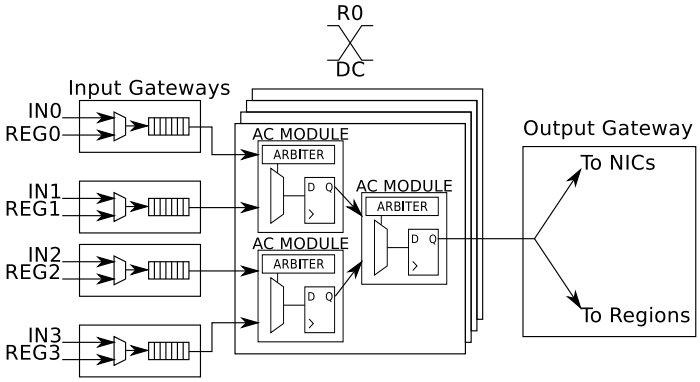


Figure 6.9: 16-node HDC region schematic.

the HDC network is the number of nodes in each region. The HDC network enforces the same number of nodes in a region and the number of regions, thus each region has $\sqrt{N}$ nodes. Notice that other unbalanced configurations are possible (e.g. 16 regions made of 4 nodes each or 4 regions made of 16 nodes each). However, in these cases the unbalance leads to worse performance and implementation results. We explore in Table 6.2 different configurations. The table shows basic metrics for different configurations of a 64-node network. An HDC_n network is defined as a *Hierarchical Distributed Crossbar* with n regions with the same number of nodes in each region. The balanced HDC configuration is, therefore, HDC_8. Thus, the HDC network with N nodes is made of $\sqrt{N}$ disjoint regions of $\sqrt{N}$ nodes.

Table 6.2 shows five metrics: area, Bisection Bandwidth (BB), BB traffic concentration, Effective BB, and Zero-load latency measured in number of hops. The BB traffic concentration measures the percentage of traffic that crosses each bisection channel, that is, the traffic demand over each bisection channel. The 2D mesh has been implemented by introducing a modular switch made of AC modules as described in Chapter 3. Area is computed as the number of AC modules to implement each architecture, and normalized to the 2D mesh case. As it can be seen, HDC_4 has the largest bisection bandwidth and the lowest zero-load latency. In contrast, HDC_4 has several drawbacks. First, the huge bisection bandwidth means that the wiring of this architecture is so large – in terms of number and length – that could not be affordable.

	Area	Bisection Bandwidth	BB traffic concentration	Effective BB	Zero-load latency
2D mesh	1	16	100	16	12
HDC_8 (HDC)	0.52	32	100	32	5.625
HDC_{16}	1.11	8	100	8	7
HDC_4	0.23	128	25	32	3.875

Table 6.2: High-level parameters for 64-end nodes for different HDC configurations and a 2D-mesh.

Secondly, the BB has a low traffic concentration that reduces the effective BB down to 32, which is identical to the effective BB of the HDC (with less wiring). Note that, without taking into account real implementation, HDC_4 has the lowest zero-load latency. However, the length of the links will drive a severe degradation when real implementation comes into play, making HDC_4 unfeasible. For the rest of the configurations affordable, HDC_8 and HDC_{16}, HDC has larger effective BB and lower zero-load latency. Similar conclusions can be obtained for larger networks.

Finally, two aspects arise on an HDC network. Routing and inter-region wires. As told before, in a DC network routing is performed at the source. On the contrary, in an HDC network, inter-region messages should be routed at the source but also at the destination gateway. Routing in the HDC network is based on node labelling and messages are routed using the final destination address included in the flit header. Nodes are labelled with increasing and consecutive order in each region. The most significant bits of the label identify the region and the least significant bits identify the node within the region. With this label strategy the gateway for a particular region is easily computed. By taking the destination bits of a message and the source node ID, the destination region is computed. Notice that routing is performed only in the source nodes and in the gateway nodes. In the source node the message is forwarded to the local region output gateway according to the destination region ID. In the destination region, input gateways use the final destination address to forward messages to the end node. In a balanced HDC network, the region ID can be used as the gateway ID to forward the message. That is, region 3 is reached through node 3 in the source region.

Another important aspect in the HDC proposal is the inter-region links that connect ACs at different regions (connect gateways). As mentioned before, inter-region links connecting the different regions have to be carefully

designed. If not, those paths will become the critical paths of the HDC network design. In order to avoid the penalization of the whole system, for the design of the inter-region links we apply a pipelined approach, as proposed in [73]. In this sense, if the propagation time of a link is higher than the clock cycle imposed by maximum achievable frequency of DCs, then the link is pipelined and several clock cycles are spent for crossing a region. In our case, the maximum link length will set the number of cycles required to travel from a region to another one. For the 64-node and 256-node network, inter-region link pipeline depth are set to 2 and 4 without reducing the overall operating frequency of the HDC network.

6.2 Implementation Analysis

In this section we perform an analysis of the critical path, area, and wiring of the DC network, compared to a traditional 2D mesh, C-mesh, and the flattened butterfly (FBF). The 2D mesh switch, the C-mesh, the FBF (see next section) and the AC modules have been synthesized using the 45nm technology open source Nangate [90] with Synopsys DC. We have used M1-M4 metallization layers to perform the Place&Route with Cadence Encounter of the AC modules. Once the modules of the switches are synthesized, they are imported as hard macro-blocks to perform the implementation of the whole network. In order to avoid designing the whole tile, which is out of the scope of this thesis, we used the MCPAT tool [105] to find out the area required by the tile when synthesized using a 45nm technology node. Two intercore distances for different NoC performance have been chosen, 1.2 and 2.4 millimetres. Interconnection links between switches in the 2D mesh, C-mesh and FBF have been forced to be routed by using the highest metallization layers available. Similarly, long links between AC modules in our DCx networks have been routed by using the same metallization layers. Notice that maximum size of repeaters available in the Nangate Library is not high (32X). By assuming higher performance repeaters or ad-hoc link designs, delays lower than the ones obtained in this thesis are possible, thus increasing the difference in performance of our DCx proposal over other topologies.

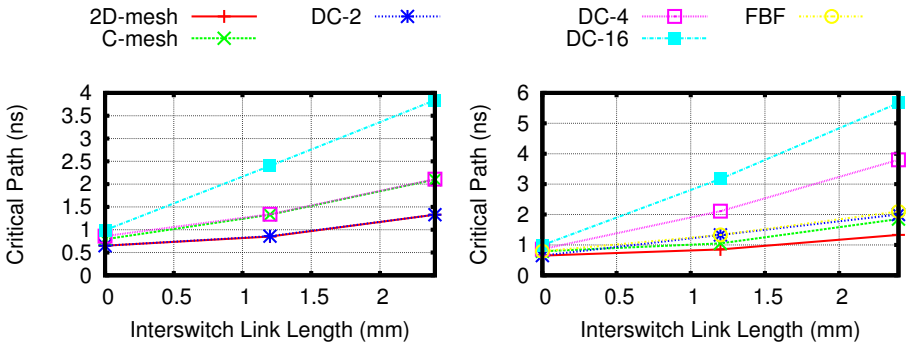
6.2.1 Critical Path

Figure 6.10 shows the critical path for the 2D mesh, the C-mesh, the flattened butterfly, and the different DCx networks. The critical path is set by the AC module plus the link delay between AC modules. Remember that the maximum link length depends on the intercore distance and the network architecture (see Table 6.1). As it can be seen, the critical path of each configuration strongly depends on the link length (fixed by the inter-core distance) and the AC degree. As the link length and the AC module degree increase (S lowers), the critical path increases[2]. As expected, a regular structure with the smallest link length as the 2D mesh or the C-mesh present the minimum critical path. Differences lay on the higher complexity of the C-mesh that forces the synthesis tool to fix a higher critical path than the 2D mesh despite that both architecture have the same maximum link length and AC degree. In contrast, the FBF architecture is highly sensitive to link length. As it can be seen, for a large network the critical path of the FBF is highly affected. Similar conclusions can be obtained when considering DCx architectures. As the degree of the AC module increases, the delay increases (see the ideal case when no inter-core distance is considered in Figure 6.10). Note that, DC2 presents the lowest critical path due to its lower AC degree, and its higher pipelining that reduces the maximum link length of the network. In contrast, the critical path of the DC16 is highly affected due its ultra low pipelining.

6.2.2 Area and Wiring Analysis

Table 6.3 shows the complete area needed for a 2D mesh, C-mesh, FBF, and the DC networks for two system sizes of 16 and 64 nodes. All the cases are implemented by using a modular design. Table 6.3 also shows the increment in area with respect to the FBF. In this chapter, we use the FBF as the baseline reference as it presents the lowest zero-load latency amongst the well-known 2D architectures analyzed, as it will be shown later. Results show how, surprisingly, the DC networks built out of AC modules with high degree require less area than a FBF for a 16-node system. The reason is that the

[2]Remember that different switch architectures have a different pipelining depth. The critical path defines the delay of each stage (see Chapter 3).



(a) 4x4 system. (b) 8x8 system.

Figure 6.10: Critical path measured in nanoseconds for different network architectures.

area of an ACx module strongly depends on its buffering. Notice that an ACx module has the same buffer slots regardless x. Thus, high degree AC modules are efficient in terms of area. Additionally, increasing the degree of the AC module means that the number of modules in the network reduces significantly.

For 8×8 systems, similar conclusions can be obtained. However, DCx networks do not scale as well as FBF as the number of cores increases. In this case, the larger wires required offset the savings in buffering achieved. Notice that this impact is alleviated as AC modules with higher degrees are used: DC2 takes a 109.9% area overhead meanwhile DC16 has an area reduction of 18.6% overhead. From the area results, it is clear that a trade-off between performance improvement and area cost is needed to really assess the potential of DC networks. We provide such a trade-off in the following section, however, remark that in a 64-node network scenario with a typical inter-core distance of 2.4mm, the whole area is $36864 * 10^4$, and hence, DC2 and DC4 only occupy a 2.41% and 1.05% of the whole chip, respectively, which is an affordable area when application latency and throughput requirements demands for a high-performance network.

Note that when comparing other high-radix topologies, similar results are obtained. When the network is small, the reduction in the number of switches allows to reduce the area with respect to a conventional 2D mesh. When

	4x4 Network		8x8 Network	
	Area(um^2)	Increment(%)	Area(um^2)	Increment(%)
2D mesh	$42.3 * 10^4$	33.4	$190.6 * 10^4$	-54.6
C-mesh	$31,7 * 10^4$	0	$173.0 * 10^4$	-58.8
FBF	$31,7 * 10^4$	0	$420,2 * 10^4$	0
DC2	$53.6 * 10^4$	69.1	$886.9 * 10^4$	109.9
DC4	$24.6 * 10^4$	-22.4	$387.1 * 10^4$	-7.7
DC16	$13.4 * 10^4$	-57.7	$342.0 * 10^4$	-18.6

Table 6.3: Network area for different architectures and system sizes.

the number of nodes increases, the C-mesh network reduces its differences with respect to the 2D mesh, but, it still has lower area. However, the main drawback of the C-mesh network is its poor performance as it we will shown later. The FBF network presents a similar trend as the DC4 or DC16 networks. When the number of nodes increases, the area of the FBF highly increases occupying more area than a conventional NoC.

Another concern is the wiring increment. However, technology wiring density allows the designer to obviate this problem. In a chip with a core size of 1.2mmx1.2mm, it is possible to route more than 7000 wires between two switches when applying a non-commercial technology library (45nm Nangate), fulfilling NoC wiring needs. For larger cores, the number of available wires between switches increases. Additionally, large links can be routed using high metallization layers. By using this kind of low-error layers, combinational logic can be placed below as the standard combinational cells use the lowest metallization layers. In this scenario, long links do not demand dedicated area, except for the area needed to implement the link drivers. Thus, area overhead introduced by links is minimum assuming the smaller inter-core distance considered. Finally, note in Figure 6.4(a) that the optimum placement of the AC modules in a DCx network enforces that several AC modules are placed close to each other. Thus, long wiring concern is further reduced.

6.3 Performance Analysis

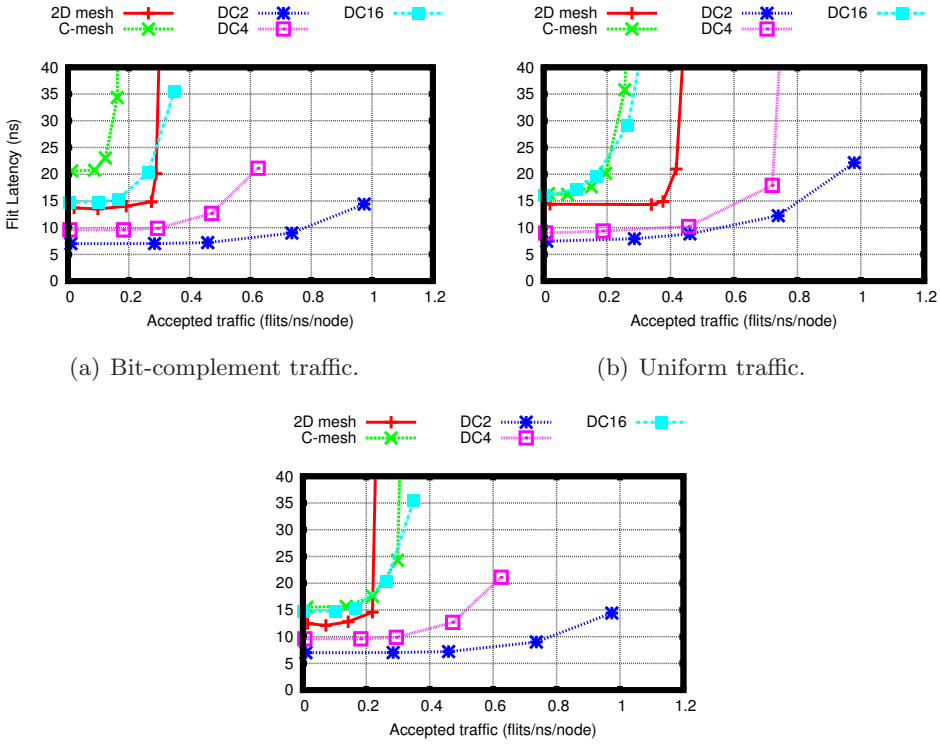
6.3.1 Synthetic traffic

In this section we analyze the performance of our DCx proposals in terms of average packet latency and network throughput, and compare it with the performance achieved by the 2D mesh NoC, C-mesh and the FBF networks. To do this, we use the simulation infrastructure described in Chapter 3, and the DCx fabrics described in Section 6.1. End nodes have been modelled as ideal NICs that inject messages into the network at a constant injection rate. Each NIC is made of a single queue per destination node where flits are stored before being injected. Despite in the next section real traffic will be analyzed, short and long messages have been used to model a real CMP traffic scenario. Short messages are composed of a single flit, while long messages are made of 9 flits. Flit size is set to 64 bits. Short messages represent 70% of the network load. Message destination follows three traffic distributions: uniform, bit-reversal, and bit-complement.

Figures 6.11, 6.12, 6.13, and 6.14 show the accepted traffic and the flit end-to-end network latency measured in flits/nanosecond/node and nanoseconds, respectively. Results shown in Figure 6.11 and 6.12 have been obtained when link length is fixed to 1.2mm. Results shown in Figure 6.13 and 6.14 have been obtained when link length is fixed to 2.4mm. In every case, clock period for each network architecture has been obtained from Figure 6.10.

The best network architecture in terms of performance is DC2, because DC2 is able to reach the highest throughput and the lowest latency. In contrast, as it has been shown in the previous section, it requires more resources than the rest of architectures analyzed. Note that DC2 is better than DC4 and DC16, despite that DC2 has a larger network pipelining. The reason is that DC2 presents the best trade-off between clock cycle and network pipelining. That means, the higher the pipelining – and hence intermediate buffering – the higher the overall performance.

Another noticeable fact is that DC2 and DC4 networks have the best response to any kind of traffic distribution. Note that as any output port has its own subnetwork, traffic to one node does not interfere to the rest of traffic. That property is desirable in those scenarios where traffic differs from



(a) Bit-complement traffic.

(b) Uniform traffic.

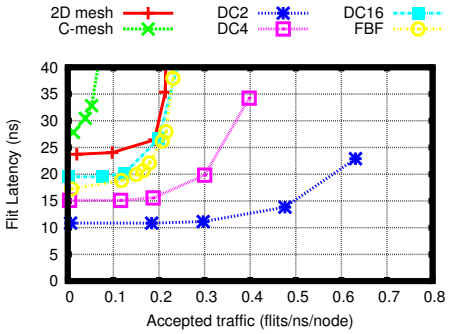
(c) Bit-reversal traffic.

Figure 6.11: Accepted traffic and end-to-end flit latency for a 16-node 2D mesh, C-mesh, FBF and DCx topologies. Link length equals to 1.2mm.

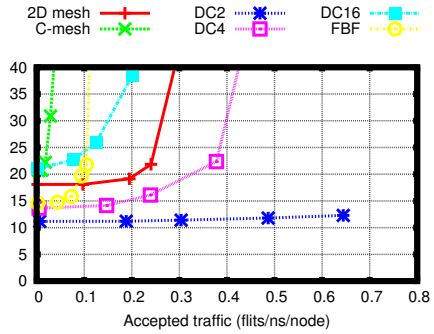
the uniform distribution. As it can be seen in Figures 6.11 and 6.12 DC2 and DC4 present small differences when considering uniform traffic distribution or other traffic distributions[3].

On the other hand, DC16 does not present as good performance as the rest of DCx configurations. This is due to the high degree of the AC modules that concentrates traffic, thus, leading to congestion. Additionally, its large links force the network to a high critical path degrading the DC16 performance – as it can be seen in the 64-node networks (Figures 6.12 and 6.14). Thus, in general, DC16 presents a bad response, similar or higher latency than a 2D

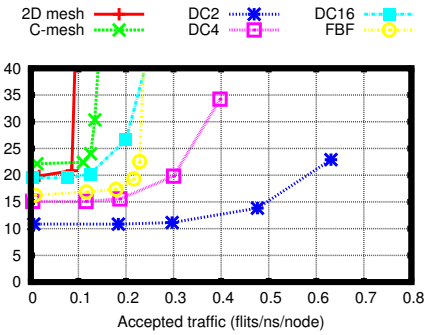
[3]Notice that, it is measured end-to-end flit latency in nanoseconds. This is the reason that DCx suffers saturation when introducing bit reversal and bit complement traffic distributions.



(a) Bit-complement traffic.



(b) Uniform traffic.

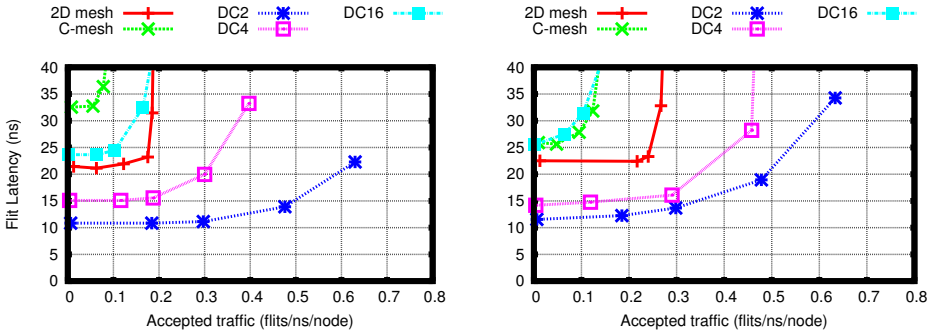


(c) Bit-reversal traffic.

Figure 6.12: Accepted traffic and end-to-end flit latency for a 64-node 2D mesh, C-mesh, FBF and DCx topologies. Link length equals to 1.2mm.

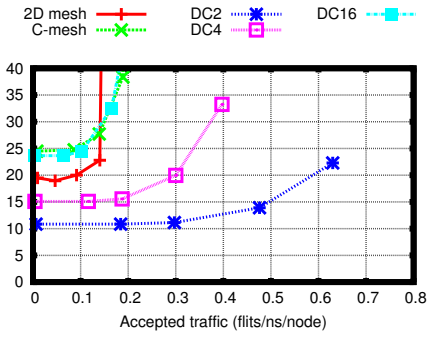
mesh and worse throughput.

With respect to other conventional 2D topologies, it can be confirmed in Figures 6.11 and 6.12 that 2D mesh performance does not scale well. Despite its optimal planar physical placement over silicon technology, as the network diameter increases, its bisection bandwidth does not scale well, reducing drastically its throughput and network latency. Note that the 2D mesh behaviour is highly sensitive to traffic distribution. In fact, when non-uniform traffic is considered, 2D mesh performance is severely degraded. As it can be seen in Figure 6.12, the FBF presents lower zero-load latency but poorer throughput than the 2D mesh. The reason is that concentration – high-radix switches due to its higher connectivity leads to congestion, reducing its throughput, but at the same time, the lower number of hops to traverse the network makes a



(a) Bit-complement traffic.

(b) Uniform traffic.



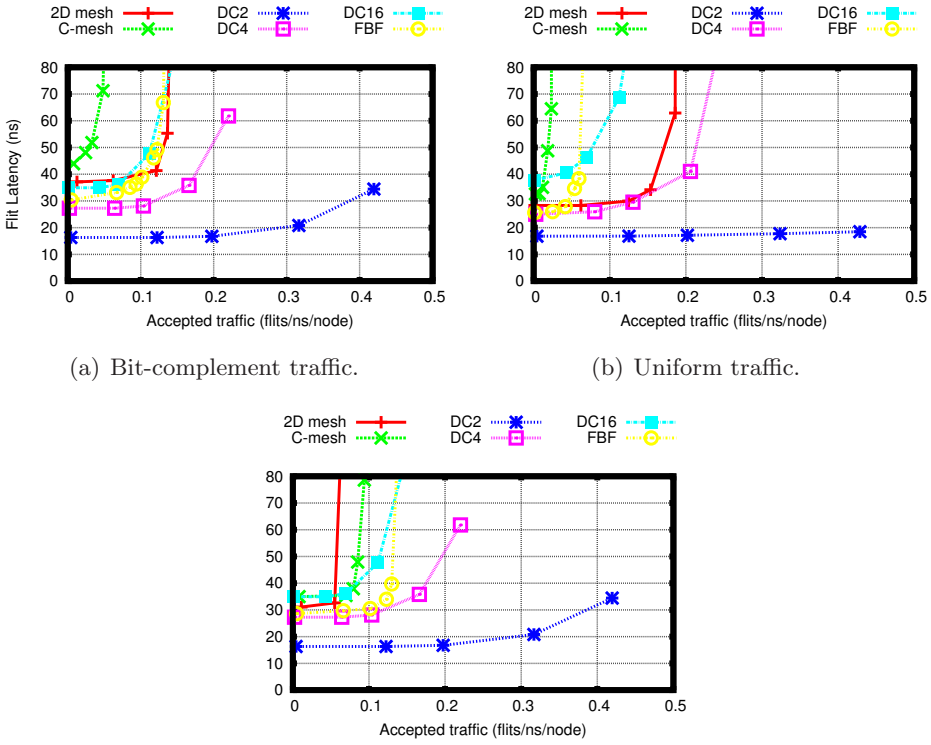
(c) Bit-reversal traffic.

Figure 6.13: Accepted traffic and end-to-end flit latency for a 16-node 2D mesh, C-mesh, FBF and DCx topologies. Link length equals to 2.4mm.

reduction in latency. In this sense, C-mesh suffers from traffic concentration that reduces the throughput. Additionally, longer link lengths than the 2D mesh degrades its performance minimizing or even removing the benefits of reducing the hop count.

When increasing the link length, similar conclusions can be obtained. As it can be seen in Figures 6.13, and 6.14, small pipelined networks with long links as FBF or DC16 do not scale well. In contrast, DC2 and DC4 still have the best performance due to its deeper pipeline. Note that, differences between both networks are increased due to the longer links used in the DC4 network.

From now on, we focus on DCx proposals as they provide the best performance. The best solution is DC2 which, when considering uniform traffic and the non-commercial 45nm Nangate Library, has a reduction in zero-load



(a) Bit-complement traffic.

(b) Uniform traffic.

(c) Bit-reversal traffic.

Figure 6.14: Accepted traffic and end-to-end flit latency for a 64-node 2D mesh, C-mesh, FBF and DCx topologies. Link length equals to 2.4mm.

latency of 48.2% when compared to the 2D mesh for a 16-node network and and intercore distance of 1.2mm (see Figure 6.11(b)). For larger intercore distances (see Figure 6.12(b) and 6.14(b)) the reduction grows up to 48.7% (see Figure 6.13(b)). In both cases, the increment of throughput is 125%. Similarly, for a 64-node network, DC2 has a reduction of latency of 36.2% and 37.8% for intercore distance of 1.2mm and 2.4mm, respectively. In those cases, the increment in throughput is 200% and 125%. Note that, those values are higher when considering traffic distribution different than the uniform distribution.

Table 6.4 shows the improvements in performance for medium size networks (64-node) of the DC2 and DC4 with respect to the FBF – which is the best 2D topology analyzed. Table 6.4 shows the increment in throughput

	Increment in Throughput (%)		Decrement in Latency (%)	
	1.2mm	2.4mm	1.2mm	2.4mm
DC2	447.0	544.7	42.1	34.4
DC4	276.1	274.8	29.7	2.3

Table 6.4: Improvements in performance of DC2 and DC4 over the FBF.

and the decrement in zero-load latency measured as a percentage. Results are provided for both cases, 1.2mm and 2.4mm, and uniform traffic distribution. For other traffic distributions similar conclusions will be obtained as it can inferred from previous figures. As it can be seen in Table 6.4, DC2 and DC4 outperform the FBF topology in throughput and latency. The main difference lies on the throughput. Both DC2 and DC4 present a huge increment in throughput. The increment for the DC2 grows up to 544.7% for long intercore distance (2.4mm). For the DC4, the increment is 274.8% in the same scenario. With respect to the zero-load latency, differences are not so large. The reason is that, the FBF is designed to minimize the network hop count, and hence, the network latency. Despite this, DC2 and DC4 outperforms the FBF in terms of latency. The difference is bigger for small chips, where DC2, and DC4 suits best. In this case, latency decrement is 42.1% and 29.7% for the DC2 and DC4 networks. For longer chips, differences decrease down to 34.4% and 2.3%.

6.3.2 Performance an Energy Results With Applications

Now we evaluate the behaviour of the DC2 network when running parallel applications on the CMP system. Different SPLASH-2 applications have been run in a 16-core modelled CMP system using the GRAPHITE simulator [93]. On the top of the simulator a detailed network simulator has been coupled to model the different network architectures. Additionally, a coherence layer models a regular MOESI invalidation-based coherence protocol. The configuration of the CMP system is summarized in Table 6.5.

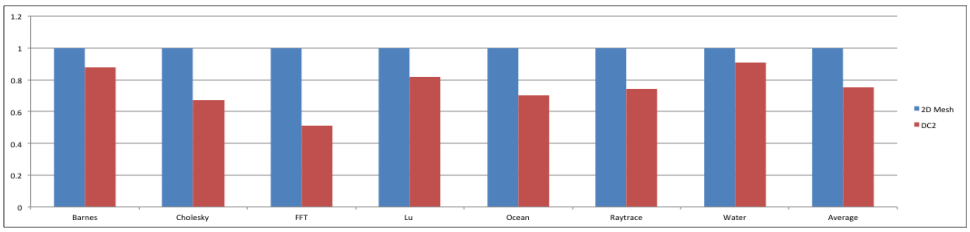
Figure 6.15 shows the applications execution time and the energy consumption when using the DC2, and a 2D mesh modular network. Results of this plot are normalized with respect to the 2D mesh. As shown in Fig-

Parameter	Values
CMP	16 cores, tiled organization, Distributed Shared Memory
Cores	x86 Architecture, Area $2.4mm^2$, in-order, single thread
L1 inst cache	private, 64KB, 4-way
L1 data cache	private, 64KB, 4-way
L2 cache	shared, 512KB, 16-way

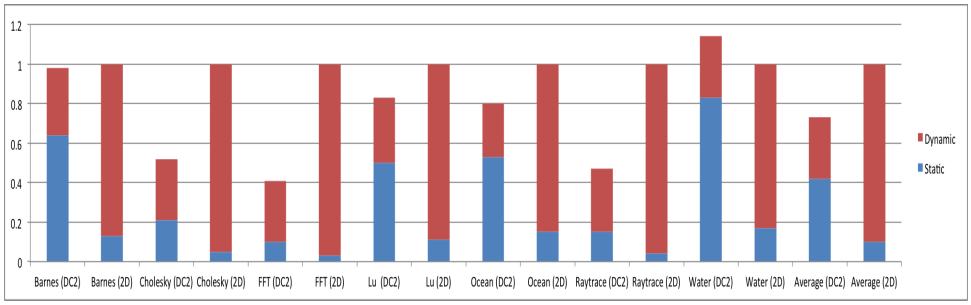
Table 6.5: CMP configuration.

ure 6.15(a), the DC2 network clearly outperforms the results of the 2D mesh, confirming the conclusions obtained in the previous section. The maximum improvements are achieved in the case of the *Ocean* benchmark where the DC2 network reduces execution time by almost 30%. On average, DC2 allows a speed-up of 14.8%.

More significant conclusions can be obtained from Figure 6.15(b). Figure 6.15(b) shows the results of energy consumption for the different benchmarks normalized to the energy consumption of a 2D mesh. In this plot the contributions of both static and dynamic energy are shown. The energy values of AC modules and links, for both static and dynamic energy, have been obtained after the synthesis of the networks. Notice that, DC2 is more energy-efficient than the 2D mesh, despite that, the DC2 network needs more resources (remember Table 6.3). On average, the energy reduction is up to 26%. The reason is that the static consumption depends on the resources, and hence, DC2 requires more static energy than the 2D mesh. On the other hand, dynamic consumption is highly related with zero-load latency, because it is made of the flits traversing the network. The higher the number of hops to traverse the network, the higher the energy. Then, as it can be seen in Figure 6.15, the main component in the DC2 network energy is the static component meanwhile the dynamic energy consumed is just a fraction of the energy consumed. This trade-off between static energy – resource-dependent – and dynamic energy – zero-load latency-dependent – makes the DC2 network to present lower energy consumption than the 2D mesh.



(a) Execution Time.



(b) Network Energy.

Figure 6.15: Normalized execution time and energy for the DC2, and a mesh-based network for real applications. Results normalized to the 2D-mesh case.

6.4 Hierarchical Distributed Crossbar Performance

In previous sections, it is shown how DCx networks outperform conventional topologies as C-mesh or FBF. However, DCx could lead to an increment in area which could be unaffordable. Thus, in this section, we analyze the performance and implementation analysis of a hierarchical distributed crossbar (HDC), and we compare it to the DC2 network which we proved that has the best results. The HDC network allows to implement a crossbar-based network beyond 64 nodes without needing an infeasible amount of resources. HDC has been built by using AC-2 modules. Thus an HDC N-node network is made by using $\sqrt{N}$ regions with $\sqrt{N}$ nodes each. In this way, the zero-load latency of each subregion is $\log_2(\sqrt{N})$ cycles. As inter-region links have been pipelined the critical path of the HDC is equal to the critical path of an $\sqrt{N}$ DC network.

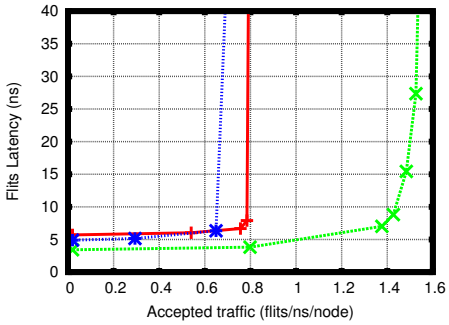
Table 6.6 shows the area and critical path of the 2D mesh, DC and HDC. Notice that, the area of the HDC network is not only the area of the DC network but is lower than the area of the 2D mesh for a 16- and 64-node

	Area (um^2)			Critical Path (ns)		
	4×4	8×8	16×16	4×4	8×8	16×16
2D mesh	$42 * 10^4$	$191 * 10^4$	$357 * 10^4$	0.65	0.65	0.65
HDC	$14 * 10^4$	$99 * 10^4$	$858 * 10^4$	0.65	0.65	0.65
DC2	$54 * 10^4$	$886 * 10^4$	$19176 * 10^4$	0.65	0.77	1.98

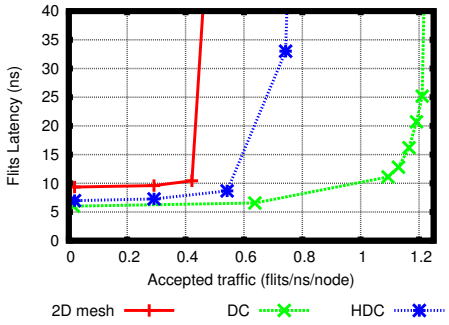
Table 6.6: Area and critical path of the 2D mesh, DC and HDC.

network. For larger networks, the HDC becomes larger than the 2D mesh but with a very significant performance boost. Notice that the area of the HDC network is much lower than the area of the DC2 network. Indeed, is less than 10% of the area of the DC2 network. On the other hand, the critical path is also reduced. As the number of nodes increases, the critical path of the DC2 network degrades. On the contrary, HDC critical path depends on the critical path of the subregions, and hence, it is kept low despite the huge increment in the number of nodes. Thus, for large networks, the HDC critical path is three times lower than the DC2 critical path.

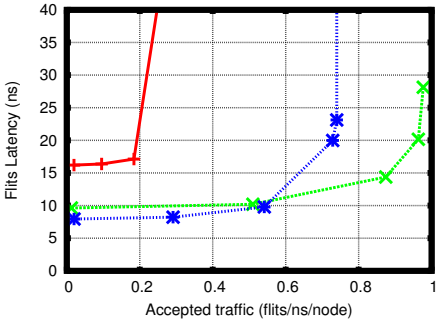
Figure 6.16 shows end-to-end latency measured in nanoseconds and accepted traffic measured in flits/nanosecond/node. As it can be seen in Figure 11(a), DC2 is still the best option for a small network. HDC does not work properly for small networks. First, because flits must traverse two regions of two stages which equals to the hop count in a 2D mesh. Additionally, traffic congestion suffered at the gateways reduces the HDC network throughput to values lower than the 2D mesh throughput. As the network size increases, HDC scales better in terms of performance and area. As it can be seen in Figure 6.16(b) the HDC network overcomes the 2D mesh and it gets closer to the DC2 architecture in terms of zero-load latency. The throughput of the HDC is lower than the throughput of the DC2 but clearly outperforms the 2D mesh. For larger networks (see Figure 6.16(c)), the HDC configuration has the lowest zero-load latency. This is due to the optimum AC module placement which reduces the critical path of the network. The throughput achieves by the HDC network remains lower than the DC2 throughput because of the traffic concentration produced at the gateways. Table 6.7 details the zero-load latency in nanoseconds and the maximum accepted traffic in flits/ns/node. Note



(a) 4x4 system.



(b) 8x8 system.



(c) 16x16 system.

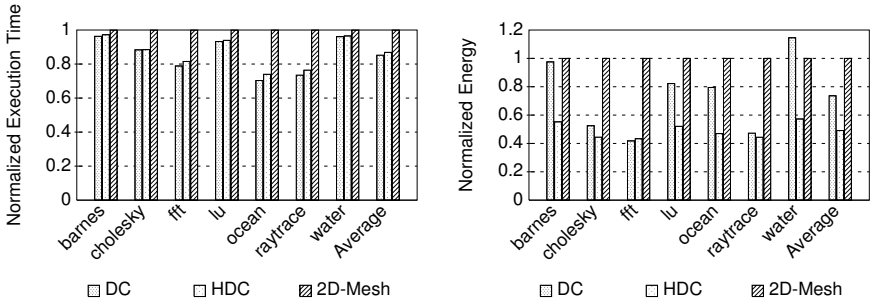
Figure 6.16: Accepted traffic and end-to-end latency for different network architectures with uniform traffic.

that, HDC improves a conventional 2D mesh by 25.5% in zero-load latency and up to 50% in throughput for a 64-node network. In a 256-node network, this improvement increases up to 50.8% and 185% in zero-load latency and throughput, respectively.

Figure 6.17 shows the applications execution time and the energy consumption when using the HDC, the DC, and a 2D mesh modular 16-node network. Results of this plot are normalized with respect to the 2D mesh. As shown in Figure 6.17, both the DC and the HDC networks clearly outperform the results of the 2D mesh. The maximum improvements are achieved in the case of the *Ocean* benchmark where the DC network reduces execution time by almost 30%, and HDC reduces execution time by 26%. On average, the DC network allows a speed-up of 14.8%, whereas the HDC network reaches 13.1%.

	Accepted Traffic (flits/ns/node)			Zero-Load Latency(ns)		
	4×4	8×8	16×16	4×4	8×8	16×16
2D mesh	0.72	0.38	0.20	5.8	9.4	16.3
HDC	0.57	0.57	0.57	4.9	7.0	7.9
DC2	1.18	0.99	0.74	3.4	5.7	9.7

Table 6.7: Accepted traffic and end-to-end latency for a HDC, DC, and a mesh-based network implementations.



(a) Execution Time. (b) Network Energy.

Figure 6.17: Normalized execution time and energy for a HDC, DC, and a 2D mesh network for real applications.

Specially noticeable is the fact that differences in performance between HDC and DC architectures are slim. This is a very important result as HDC area is 80% lower than the one required for the DC network. This demonstrates the HDC architecture presents the best performance-area trade-off among the different topologies considered, as the DC architecture was seen to be the one with the best network performance for a 16-node network.

6.5 Conclusions

In this chapter, we go one step beyond the previous proposals and present a new on-chip architecture based on an efficient floorplan-aware implementation of a distributed crossbar, where not only the crossbar is spread across the chip but also the buffering and arbitration is decentralized. As a result, our DC network architecture presents the minimum latency and the highest network

throughput among the proposed on-chip architectures analyzed, at the expense of increasing resources needed. Despite the resources increased, whole network power consumption is reduced due to the reduction in the power consumption when flits cross the network as the network latency – and hence number of hops – is reduced.

Our DC design philosophy can be implemented easily on any NoC environment, making our proposal appealing to NoC designers. It reduces the design and implementation complexity of a NoC centralized crossbar, meanwhile theoretical crossbar performance is maintained.

Chapter 7

Conclusions

In this final chapter, we present the conclusions of this dissertation, the contributions to the research domain and a brief list of research directions that will be addressed in the future. Finally, we also expose the results in terms of scientific publications and other contributions derived from the work presented in this dissertation.

7.1 Conclusions

The following is a list of conclusions extracted from the current dissertation. These conclusions helped to obtain the contributions and scientific publications that are at the core of the document.

- **Current high-performance multicore solutions pledge for tile-based designs.** Tile-based design is gaining momentum for newer Chips Multiprocessor (CMPs) and Multiprocessor System-on-Chips (MPSoCs) solutions. As the expected trend is to include more and more cores inside a chip, these solutions rely on networks-on-chip (NoCs) to handle all the communication traffic between cores.

- **2D meshes and rings have been established as the proper solution for current CMP and MPSoC.** Simple planar topologies as 2D meshes and rings have been chosen to implement current CMP and MPSoCs. Their acceptance relies on that they fit perfectly the chip lay-

out. Additionally, their simplicity removes several challenging issues to physical designers, which can handle those kind of topologies easily.

- **Performance of those simple topologies do not scale with the number of nodes.** Current fabricated CMP and MPSoC that rely on 2D meshes or rings manage a number of cores that could be considered small. As the number of cores increases, the performance of such kind of topologies degrades. That is, network latency increases due to the increment in the hop count and throughput decreases due to contention.

- **High-radix topologies could lead to the same problem.** One solution to outperform performance degradation of 2D meshes or rings when the number of cores increases is introducing high-radix switches, that is, to increase the network connectivity which in general means that they can handle with more cores and/or they can communicate with more and/or farther switches. The performance of such high-radix topologies such as *Concentrated mesh* or the *flattened butterfly* clearly outperforms the 2D mesh and ring performance for large systems. However, if no complex routing algorithm is applied, performance of high-radix architectures remains low due to switch contention, and link underutilization.

- **NoC implementation determines the network performance.** As important of the theoretical architecture behaviour is its implementation. That is, suboptimal NoC implementation leads to a decrement in the theoretical network performance or even makes a solution unfeasible. In this sense, high-radix architectures are more sensitive to the physical implementation, because they make use of complex switches, and longer links.

- **Crossbars provide low latency and high throughput.** Crossbars have been discarded due to the common belief that their implementation cost is unfeasible. However, their theoretical performance outperforms the rest of architectures. Recent works have shown that a detailed crossbar implementation is feasible, introducing this solution into NoC environment.

- **Simplifying logic allows to improve physical design.** Removing complex NoC modules, and splitting them into a sum of smaller and simpler modules simplifies the overall NoC design, and offers the designers the possibility to close the gap between theoretical design and final physical implementation.

- **Future challenges in CMPS and MPSoCs rely on a proper NoC physical design.** Nowadays, interconnection network knowledge clearly outperforms fabrication reality. Thus, next CMPs and MPSoCs advances could be related with physical design implementation which can get closer manufacturing in industry and research in academia.

7.2 Contributions

The overall contribution of the dissertation is to present a modularized design methodology that allows to implement a floorplan-aware NoC design that minimizes long link drawbacks as high power consumption and delay. Minimizing link impact reduces overall network critical path and power consumption. Additionally, the modularized design methodology allows to introduce large crossbars into the NoC scenario, thus, reducing network latency and increasing network throughput. This overall achievement has been obtained with a step by step procedure described next:

- **Modularizing the switch improves switch performance**. A modular switch is presented that clearly outperforms the canonical switch with identical functionality. Modularizing the switch introduces two main advantages. First, buffering is distributed among the stages increasing the switch throughput. Additionally, critical path is reduced as complex switch tasks as arbitration is simplified and spread over several stages.

- **Spreading the logic over the chip reduces link impact.** The modular switch can be spread over the link reducing the maximum link length. By distributing the logic link delay and power consumption is reduced, enhancing the overall network performance and power.

- **Introducing crossbars to NoC**. Recently some studies have tried to introduce crossbar in the NoC domain. However, crossbars are not widely accepted as a proper solution. In this dissertation a distributed crossbar has been designed that minimizes conventional crossbar designs, as a complex control logic and long links.

Analysing this dissertation step by step, more detailed contributions can be stated:

- We have designed a modular switch design (Chapter 3) able to increase throughout by 60% when compared to a 2D mesh case for a 64-node network.
- We have designed a distributed modular switch design (Chapter 4) able to reduce the network critical path up to 53% for large core size. In this case, a 13% energy savings can be obtained when compared to the functionally identical modular switch.
  - The distributed and modular switch design increases fault tolerance, when compared to the 2D mesh canonical-switch-based design due to their modular nature.
  - The distributed and modular switch can minimize easily variation process due to the relation between link and logic.
- We have adapted high-radix topologies with the modular distributed-switch based design (Chapter 5) where performance is increased and the critical path is reduced.
- We have proposed a distributed crossbar (Chapter 6) that reduces application execution time of 14.8% with an energy saving of 26% in a 64-node network over the 2D mesh.
- We have also proposed a hierarchical approach able to overcome scalability issues of crossbars. With the resulting design (HDC) we are able to reduce application execution time a 13% with an energy saving larger than 50% in a 64-node network. In this case, the area occupied by the hierarchical crossbar is smaller than the conventional 2D mesh.

7.3 Future Work

In this Section we highlight possible future work directions coming from this dissertation. The following is a list of possible directions:

- **Extend Network-on-FPGA contribution**. Network-on-FPGA designs are being considered valuable by NoC industry because, by using FPGAs, conventional NoC designs can be better analyzed and simulated, and hence, fabrication process can be properly modelled, reducing costs and fabrication time. In this sense, current FPGAs provide peculiar features that forces NoC designer to even redefine basic interconnection assumptions. In Chapter 4, the distributed switch has been enhanced to fulfil FPGA environment constraints. It has been shown that a modularized switch clearly suits a FPGA architecture. Thus, a first attempt to get introduced onto the FPGA world has been made but FPGA world provides a huge range of opportunities to be taken into account.

- **Analyze system requirements to improve hierarchical distributed crossbar.** This dissertation has been focused to implement a floorplan-aware NoC. However, system requirement as, number of applications to be run, type and behaviour of traffic, communication pattern between nodes, etc. can be used to even outperform the hierarchical network. A complete study of how system requirements can be properly defined to match a hierarchical network must be done.

- **Introduce snoopy coherence protocol into NoC.** Snoopy protocols have typically been discarded due to the difficulty of implementing broadcast messages in a 2D mesh NoC. Notice that, the distributed crossbar NoC is suitable to support broadcast messages, and hence, a snoopy protocol can be easily implemented. Simplifying the coherence protocol could lead to an overall network simplification which can lead to minimize the impact of the distributed crossbar, reducing area overhead and power consumption. On the other hand, a significant reduction of the application execution time could be achieved.

7.4 Publications

The following list enumerates the papers related with this dissertation that have been published, or are under review process, in specialized conferences or journals. We outline for each contribution the novelties that are part of this dissertation.

- Antoni Roca, José Flich, Federico Silla, and José Duato. "A Latency-Efficient Router Architecture for CMP Systems", 13th Euromicro Conference on Digital System Design, Architectures, Methods and Tools (DSD), 2010.

This publication is the first attempt to build a modular switch. In this paper, the switch allocator is modularized meanwhile the rest of blocks of the canonical switch – input buffer and crossbar – are not modified.

- Antoni Roca, José Flich, Federico Silla, and José Duato. "A low-latency modular switch for CMP systems." Journal of Microprocessors and Microsystems Embedded Hardware Design, 35(8):742754, 2011.

In this publication we further develop the modular switch design. In fact, the modular switch presented in this dissertation has minor changes with respect to the one presented in this paper.

- Antoni Roca, Carles Hernández, José Flich, Federico Silla, and José Duato. "A Distributed Switch Architecture for On-Chip Networks." International Conference on Parallel Processing, ICPP 2011.

In this paper the distributed switch design is presented, and how moving the switch onto the links outperforms conventional switch design philosophy in terms of performance and power consumption.

- Antoni Roca, José Flich , Giorgos Dimitrakopulos. "DESA: Distributed Elastic Switch Architecture for efficient Networks-on-FPGAs." International Conference on FPGA, FPL 2012.

In this paper, the distributed switch has been modified to fulfil FPGA constraints. It is shown that the distributed switch matches perfectly the FPGA architecture.

- Antoni Roca, Carles Hernández, José Flich, Federico Silla, and José Duato. "Enabling High-Performance Crossbars through a Floorplan-Aware Design." International Conference on Parallel Processing, ICPP 2012.

In this paper, the distributed crossbar is implemented and it is shown how a proper placement allows designers to use crossbars in larger networks.

- Antoni Roca, José Flich, Federico Silla, and J. Duato. "A low-Latency Router Architecture for CMP Systems." in Actas XXI Jornadas de Paralelismo (JP2010), pages 1-9.

- Antoni Roca, Carles Hernández, José Flich, Federico Silla, and J. Duato. "Modular Distributed Switch: Spreading the Switch along the Link." in Actas XXII Jornadas de Paralelismo (JP2011), pages 221226.

- Antoni Roca, Federico Silla, José Duato José Flich. "A full custom modular switch for CMP systems." In 7th HiPEAC Summer School on Advanced Computer Architecture and Compilation for Embedded Systems (ACACES).

The first two papers are summaries of the work done in the dissertation. The third one is an efficient full custom implementation of the modular switch. These papers belong to conferences with non peer-review systems.

- Carles Hernández, Antoni Roca, Federico Silla, José Flich, and José Duato. "On the Impact of Within-Die Process Variation in GALS-Based NoC Performance", IEEE Trans. on CAD of Integrated Circuits and Systems, vol. 31, number 2. 2012.

- Jesus Camacho, José Flich, Antoni Roca, José Duato. "PC-Mesh: A Dynamic Parallel Concentrated Mesh", ICPP, 2011.

- Samuel Rodrigo, José Flich, Antoni Roca, Simone Medardoni, Davide Bertozzi, Jesus Camacho, Federico Silla, José Duato. "Cost-Efficient On-Chip Routing Implementations for CMP and MPSoC Systems", IEEE Trans. on CAD of Integrated Circuits and Systems, vol. 30, number 4, 2011.

- Carles Hernández, Antoni Roca, Federico Silla, José Flich, and José Duato. "Characterizing the impact of process variation on 45 nm NoC-based CMPs", J. Parallel Distrib. Comput., vol. 71, num. 5, 2011.

Finally, several papers collect the main research done on the canonical switch that contribute to define the starting point of this dissertation.

- Antoni Roca, José Flich, Federico Silla, and José Duato. "VCTlite: Towards an efficient implementation of virtual cut-through switching in on-chip networks", International Conference on High Performance Computing (HiPC), 2010.

In this paper, the flow control mechanism between switches has been analyzed. In the paper, mainly two kind of contributions have been made. First, how to improve the routing algorithm implementation. Second, how the variation process affects the switch performance. The participation on these papers is focused only on the switch design. Amongst others,

All the studies made over the canonical switch were summarized in two chapters of the next book.

- J. Flich and D. Bertozzi, Designing Network On-Chip Architectures in the Nanoscale Era. CRC Press, 2010.

Appendix A

Canonic Switch Model

A.1 gNoC: A Switch Design for CMP Systems

gNoC is a baseline switch design targeted for CMP systems, where high frequencies are usually required. However, it is not the intention of gNoC to reflect current state-of-the-art switches for CMPs. Instead, gNoC must be seen as a canonical academic implementation of a basic switch design. The real purpose of gNoC is to help in the evaluation of different techniques proposed and presented in the book. We consider gNoC as the baseline switch which the *modular switch* presented in Section 4.

The main features of the switch is its pipelined design and its support for virtual channels. Furthermore, gNoC has been built as a modular switch where the main parameters that define the switch can be tuned at design time. Parameters such as flit width, link width, number of input/output ports, and number of virtual channels, can be set at synthesis time. However, the gNoC switch instantiation used through this book is designed with five input and five output ports, so that four ports are intended to provide connectivity with the neighbouring switches in a 2D mesh and the fifth port connects to the local computing core.

The baseline design relies on wormhole switching but virtual cut-through switching has also been implemented to support broadcast. The Stop&Go flow control protocol has been deployed in order to control the advance of flits between adjacent switches.

In the following sections we describe the gNoC baseline switch design. Figure A.1 shows the main components of the baseline switch design. In this Figure, gNoC does not implement virtual channels (we later enhance the switch with support for VCs).

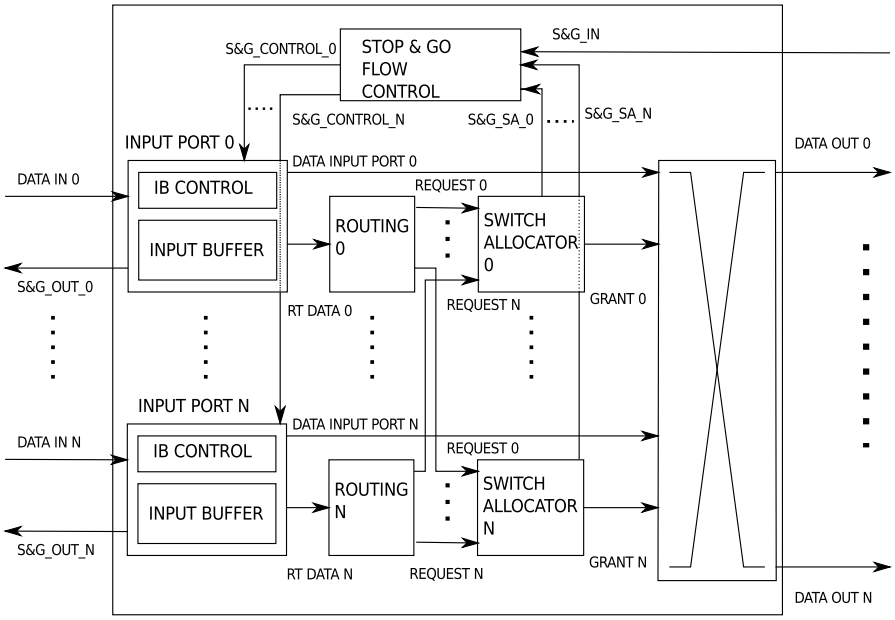


Figure A.1: Switch schematic.

A.1.1 Pipeline Organization

The gNoC switch follows the standard pipelined design with five stages: input buffer (IB), routing (RT), switch allocator (SA), crossbar (XB), and link traversal (LT). Note, however, that as the last stage does not belong to the switch, the gNoC design is actually a four-stage pipelined switch design.

A standard pipelined design as gNoC is a non-optimized design from a point of view of latency because no stages are performed in parallel and therefore the latency for traversing the pipeline is five cycles. However, a pipelined switch design like gNoC allows each stage to be almost independent from the others, thus allowing for a simpler analysis of the impact each component has over the entire switch. This feature is very interesting from an academic point of view. Figure A.2 shows the pipeline when different flits of a single packet

are processed by the switch. Notice the first flit is a header flit followed by payload flits of the same message. The header flit crosses all the stages while the remaining flits cross only the IB, XB, and LT stages.

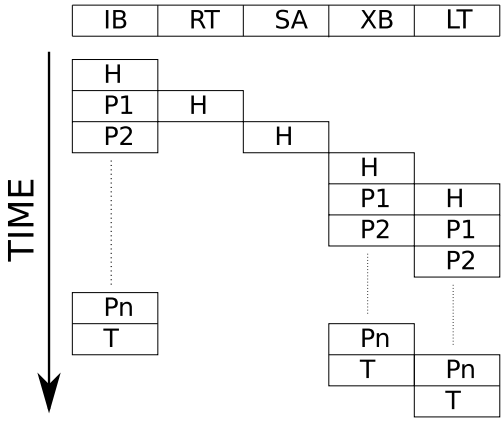


Figure A.2: Pipeline stages when forwarding a message through a gNoC switch.

The gNoC implementation used throughout this book implements buffering of flits at the input ports only. In this implementation, the IB stage can store up to four flits at each input port. This number has been chosen in order to guarantee the round-trip time of the link and the delay in the flow control (Stop&Go) mechanism. The link delay is set to one cycle. Therefore, bubbles between flits are avoided and thus, maximum throughput per flow is achieved. Furthermore, notice that the IB stage not only stores the incoming flits but also the flits that are in transit across the switch (e.g. flits crossing the switch), which are not removed from the input buffer until they completely leave the switch. Therefore, those flits which are at different stages of the switch such as the RT, SA or XB stages are kept in the IB buffer, thus being unnecessary to replicate those flits across the switch. For this purpose, control signals distribute the basic control information required by each module. In this way, there is no need to add pipeline registers within the switch for storing the message flits, but only control pointers. More specifically, a set of control read/write pointers at each input buffer guarantees the proper operation of the switch.

Instead of using memory macros, buffers in the IB stage have been designed

using FIFOs. More specifically, they have been implemented using a Push-In-Mux-Out register [8] as its power consumption is lower than that of a conventional shift register. Implementing a Push-In-Mux-Out register forces the IB stage to implement a write pointer and several read pointers used to store the incoming flits and later retrieve the stored flits. The write and read pointers also make the implementation of the Stop&Go flow control easier, since those pointers reflect the input buffer occupancy, as it will be explained later.

The routing stage (RT) has been designed as a modular block, in order to support, with no major modifications, the several routing algorithms described in [7]. Nevertheless, the DOR algorithm is the one used in the baseline design. Additionally, distributed routing has been supported in gNoC, instead of using the source-based routing approach. Notice that the RT stage is replicated on every input port. The reason is that the routing algorithm depends only on the destination information which is independent for the different packets arriving at the router. Additionally, a global routing unit for the whole switch would increase the wiring complexity of the switch. Nevertheless, the DOR algorithm (and other options like LBDR) are small enough to allow replication through all the input ports, thus being a more cost-effective option than the increased wiring.

The RT stage is fed by the flit header of the message that includes its destination (in coordinates of the 2D mesh). The switch has an internal register with its coordinates and a small logic is used to implement DOR routing. Figure A.3 shows the logic for the DOR routing implementation in the RT stage. The routing proposals from [7] also rely on current and destination switch coordinates.

The SA (switch allocator) stage is in charge of deciding the flits that get access (through the crossbar) to a given output port. A switch allocator module has been implemented for each output port. The allocators are independent from each other (arbitration decisions from one allocator do not influence another allocator's decision). The SA module receives from each RT module (one per input port) the requests for the output port. The SA module determines which input port will be connected to its own output port. As a result, the SA module generates control signals that program the crossbar and the FIFO of

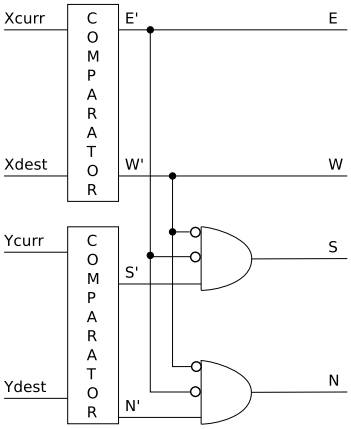


Figure A.3: Simple DOR implementation in gNoC switch design.

an IB stage. Each SA module has been designed using a round-robin arbiter according to [106]. The decision of designing a round-robin arbiter has been made due to its simplicity, that allows the SA stage to have a low delay [106]. Note that there exist more advanced arbiter designs [7]. However in a switch with no virtual channels and with deterministic routing, the round-robin arbiter guarantees maximal matching. If an output port receives at least one request, that output port will grant one of them. Also, an input port will only raise one request.

In [17] there exists a discussion on the convenience of performing a flit-level arbiter or a packet-level arbiter. In a flit-level arbiter each flit of a packet has to compete for an output port. In contrast, in a packet-level arbiter, when the header of a packet obtains an output port, the connection remains set for the whole message. Hence only the header of a packet will compete for an output port. In the first case, a fair arbitration can be achieved [17] meaning that all packets of all input ports have the same chances to be selected at every arbitration cycle. As a disadvantage, flits from different packets will share the same buffer at the IB stage in the downstream switch, thus increasing the complexity of the control logic that determines the proper operation of the input buffer. In contrast, a packet-level arbiter may lead to unfair situations [17], but allows a simpler arbiter design and an easier IB stage design as flits from different packets are not mixed in the same downstream buffer.

In addition, in a packet-level arbiter power consumption will be much lower since arbitration is performed packet by packet rather than flit by flit. gNoC implements a packet-level arbiter.

A.1.2 Flow Control

The flow control mechanism is an important part of a router. The gNoC switch implements a simple Stop&Go flow control. The operation of the Stop&Go algorithm is simple: when the input buffer of the IB stage is going to be full, a Stop signal is sent back to the previous switch, which is injecting flits to this IB. When the IB reaches a level where new flits can be stored, a Go signal is sent back. Notice that both signals can be seen as a physical signal where the value of this physical signal determines the Stop or Go status of an IB. Thus, the IB status –determined by the write/read pointers– is enough to generate the flow control information. However, the Go signal generation and the round trip time of a link determine the minimum IB size needed in order to not introduce any bubble between data flits. In order to minimize this dependency between the flow control mechanism and the IB size, the gNoC switch generates the Stop&Go signal using the current IB and the SA status. In this way, once the SA determines that a flit is winning an output resource, as this flit will leave the router in the next cycle, then we can anticipate the IB status, thus anticipating one cycle the generation of the Go signal.

A.1.3 Virtual Channel Support

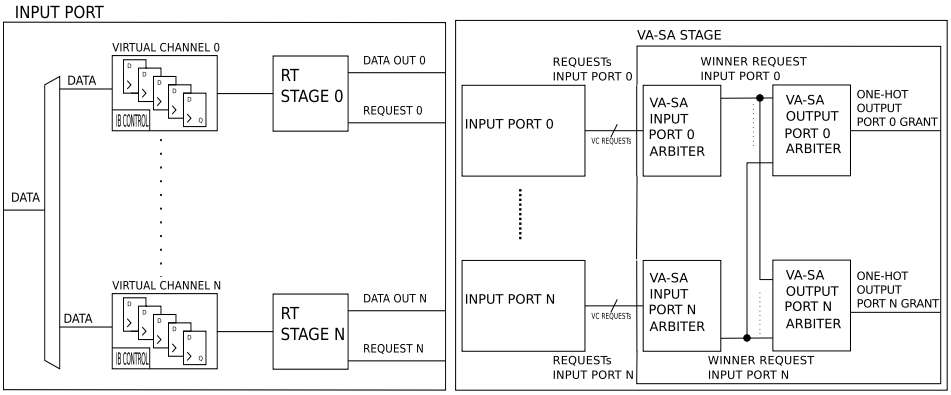
Some changes have been introduced to the baseline router presented above in order to support virtual channels. At each input port the number of input buffers has been incremented, thus providing buffering support for V virtual channels. As all the virtual channels are devoted the same numbers of resources, a static partition of the overall buffer at the input port is feasible, thus simplifying the design. Note that a multiplexer has to be introduced at the input port to map an incoming flit into the proper buffer (associated to the virtual channel the flit is using). In order to reduce the complexity of the control logic associated to each input port, each virtual channel has its own RT module as can be seen in Figure A.4(a). Notice that an alternative could

be to have only one RT module for each input port.

The basic functionality of the switch with virtual channel support determines that a message mapped to a virtual channel of an input port can access any virtual channel at any output port. That forces the resource allocator of the switch to be complex. To deal with this issue, in the gNoC switch, the SA stage is extended to support the virtual channel allocation stage (VA-SA). Indeed, we may think of this stage performing both actions in parallel. The first action deals with assigning a free virtual channel to a header flit that requests a free virtual channel of an output port. The second action deals with each output port to arbitrate amongst all the virtual channels of all the input ports that request that output port. The way to implement the VA-SA stage has been widely studied [7]. gNoC implements a VA-SA stage that is divided into two simple arbiters. Firstly, there is an arbiter at each input port that arbitrates amongst all the virtual channels of that input port. That functionality has been implemented with a round-robin arbiter identical to the one implemented in the SA stage previously described. After performing this arbitration, only a single flit from each input port is competing for an output port. The second task of the VA-SA implemented in the gNoC switch design is to arbitrate amongst all the input ports for an output port resource. Note that this second task is identical to the SA stage described above. As a side effect, by implementing the VA-SA stage in two parts reduces the matching capabilities of the VA-SA stage as only one flit from an input port may compete for an output port, thus not achieving maximal matching. However, splitting the VA-SA stage into two simple arbitration processes reduces its latency with respect to more complex arbiter designs that perform maximal matching amongst all the virtual channels of all input ports. An scheme of the VA-SA stage can be seen in Figure A.4(b).

A.1.4 Switch Implementation

Table A.1 shows the area and latency (critical path) of the gNoC switch design without virtual channels and five input and outputs ports. No link delay has been computed. Flit size has been set to 8 bytes. Flit size is identical to the short message size in a CMP scenario. Input buffer size is set to four flits. The router has been implemented using the 45nm technology open source



(a) Input port with virtual channels. (b) VA-SA stage.

Figure A.4: gNoC switch support for virtual channels.

Nangate [90] with Synopsys DC. The wire model used is 5K-hvratio-1-1. We have used M1-M4 metallization layers to perform the Place and Route with Soc Encounter.

module	area (um^2)	critical path (ns)
IB	3113.45	0.55
RT	124.26	0.32
SA	337.88	0.75
XB	1975.6	0.52
gNoC	17651	0.75

Table A.1: Area and delay the router modules and gNoC

Notice that the SA stage is the one with the highest latency, thus becoming the bottleneck of the switch. This large latency is due to the complexity of the arbiter (although being a round-robin arbiter) when compared to the other stages, and the management of flow control signalling in the same stage. Figure A.5 shows the critical path of the canonic switch when assuming different intercore distances. As it can be seen, the critical path increases as the link length increases. From now on, we consider the canonic switch critical path as:

$$T = 0.75ns + \text{link delay} \tag{A.1}$$

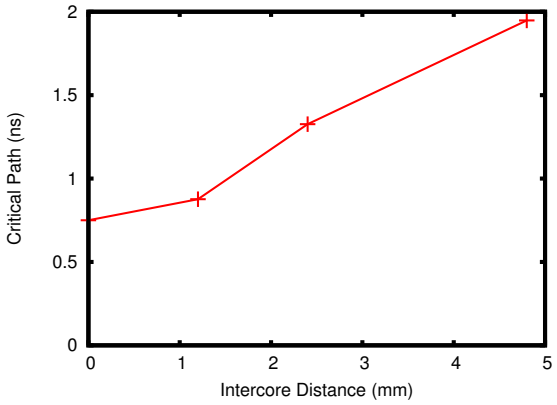


Figure A.5: gNoC critical path.

For the area results note that the IB stage is the stage with major area requirements. The area computed for the IB and RT stages must be replicated for each input port, while the area computed for the SA stage must be replicated by the number of output ports. The final entry in the table reflects the total area and delay of a 5-port gNoC switch design. Figure A.6 summarizes the area of each stage (computing all the modules) of a gNoC switch. Figure A.6 also shows the power consumption of each stage. Note that the IB stage consumes almost 70% of the overall power budget of the router.

When introducing virtual channels the area and latency of the router are, obviously, incremented. Each new IB and RT module will be similar to the modules in a switch with no virtual channel. However, these two modules must be replicated by the number of virtual channels, thus increasing the area of the entire switch. Note that the increment in area of the VA-SA stage with respect to the SA stage, or the area increment due to the increased amount of RT stages, are secondary. That can be inferred from Table A.1, where it can be seen that the RT stage and the SA modules do not highly contribute to the whole switch area. The crossbar remains identical to the case where no virtual channels are used as the policy used in the VA-SA stage is to reduce the requests from an input port down to one (independently from the number of virtual channels implemented). Finally, the latency of the switch is affected by the increased complexity of the VA-SA stage. Remember that the VA-SA stage is divided into two phases. Table A.2 shows the difference in area and

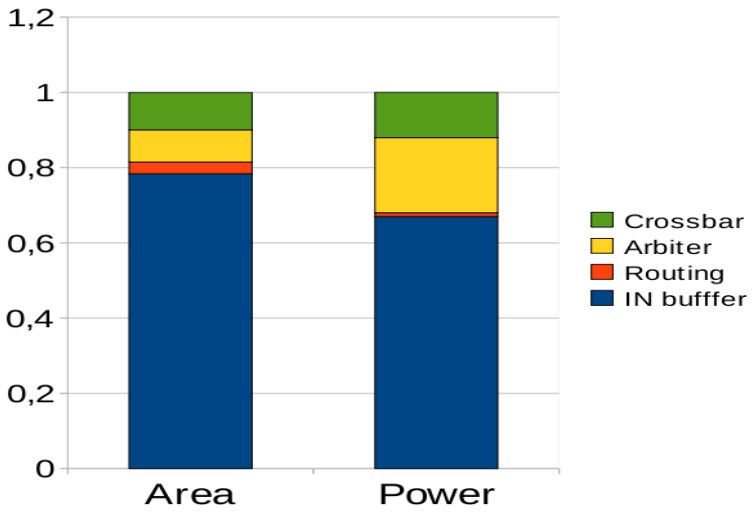


Figure A.6: Area and power of the gNoC switch.

latency of the gNoC switch design with no virtual channels and with 5 virtual channels. Finally, Figure A.7 shows the floorplan of the core of the canonic switch.

gNoC	area (um^2)	critical path (ns)
no VC	17651	0.75
5 VC	89976	1.01

Table A.2: Area and delay of the gNoC with no VC and five VC

A.1.5 Virtual Cut-Through and Tree-Based Broadcast Support

The pipelined wormhole switch described above has been modified to support virtual cut-through (VCT) switching and tree-based broadcast support. Notice that support for the fork operations described in that chapter is also provided. In order to migrate a wormhole switch to a VCT switch, some changes at different levels must be applied. The most important consideration is that packets must fit into switch buffers, because VCT is leveraged. Thus, the changes required are: (1) packetization at the network interfaces; (2) ad-

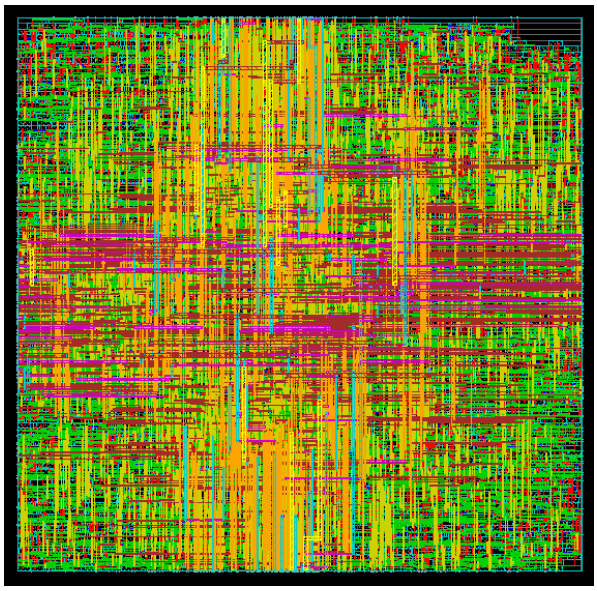


Figure A.7: Core floorplan of the canonic switch.

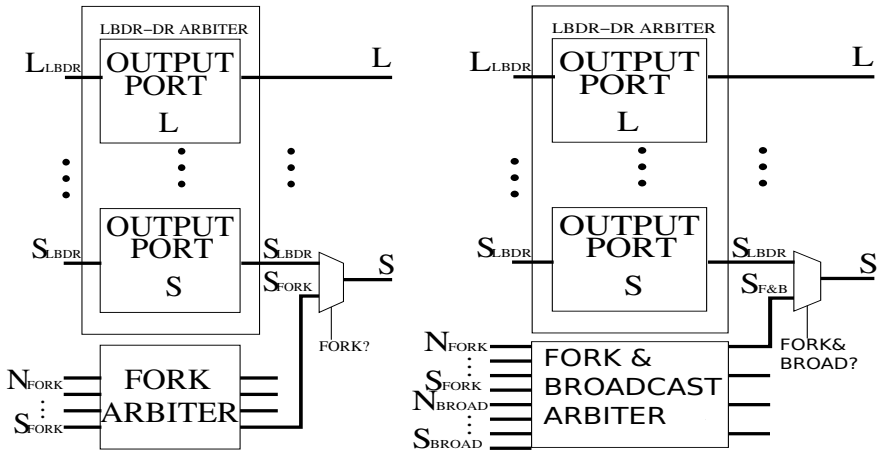
justing buffer size to packet size; (3) changes in flow control, in order to adapt it to the nature of VCT. In addition, to support a tree-based broadcast and fork operations, we need also to: (4) change the arbiter logic to support replicating packets; (5) remove stale copies of packets (being replicated/forked).

he main changes between a wormhole switch with packet-level arbitration and a VCT switch relies on the flow control mechanism, as a VCT switch must guarantee that a packet that is crossing the switch will be completely transmitted without interruption. Furthermore, the changes in the flow control mechanism applied to the VCT switch will incur in modifications in the structure of both the IB and the SA stages. In addition, packetization is performed at the node interfaces when required (thus not covered by this discussion on switch design). Nevertheless, some short explanation on packet size is worth. Messages in CMPs (using a coherence protocol) can be either short messages containing a coherency command and a memory address or long messages containing a cache line. In both cases, the length of those messages is known. Additionally, the percentage of short messages is usually much larger than that for large ones. Thus, the maximum packet size can be seized to that of short messages. In that case, long and less frequent messages should be packetized.

On the other hand, the Stop&Go flow control protocol implemented in the wormhole switch can be relaxed in order to make it more efficient. Instead of leveraging a flow control at the flit level, the nature of VCT allows implementing it at the packet level. The reason is that when a packet is granted an output port in VCT, all the flits of that packet will be forwarded. Therefore, a stop or go signal may be asserted per packet. Note that we assume links with one cycle delay, and thus round trip time is set to three cycles. In this case buffers must be set to four flits to avoid introducing bubbles. However, for messages with sizes shorter than packet size and round-trip time (e.g. one-flit packets), bubbles between packets are generated if the buffer size does not fulfil the round trip-time restriction. To avoid bubbles we decided to pad short packets to the input buffer depth. Obviously, this may affect performance.

Changes in the SA stage should be minimum since the SA stage is the most critical stage in our design (explained previously). To address this concern we have implemented the arbiters shown in Figure A.8. Figure A.8(a) describes the arbiter in charge for unicast fork operations, while Figure A.8(b) shows the arbiter for broadcast operations. Both arbiters follow the same design strategy used in the wormhole switch. Both arbiters add a new module performed in parallel with the unicast arbiter previously described (see Figure A.8). This new module arbitrates between fork and broadcast requests similarly to the unicast requests. The grant signals of this module enable (or disable) the unicast grants. Higher priority is given to fork/broadcast requests. Finally a multiplexer decides if a unicast request or a broadcast/fork request is winning the crossbar. By doing this, minimum impact on the SA stage latency is expected.

Finally, Table A.3 summarizes the frequency and area of the wormhole and VCT implementations with no virtual channels. Both switches have the same input buffer size and flit width. Note that VCT is faster than the wormhole switch. This is due to the fact that VCT switching allows the switch to relax the creation of the Stop&Go signal because in VCT the Stop&Go flow control is performed per-packet rather than per-flit as in the wormhole switching. Then, relaxing the Stop&Go signal reduces the critical path of the VCT switch. Furthermore, VCT is designed with an IB size equal to the packet size. This design constraint, in addition to the flow control decreasing complexity, allow



(a) New arbiter for the VCT switch with fork requests.

(b) New arbiter for the VCT switch with fork and broadcast requests.

Figure A.8: Arbiter for the VCT switch with fork or broadcast requests.

the IB to be simpler than the IB of the wormhole switch. For that reason, the area of the VCT is smaller than the wormhole switch for the same size of the IB in both cases.

area / Freq	Wormhole	VCT
area ($um2$)	17651	13643
freq (GHz)	1.33	1.54

Table A.3: Area and frequency for the wormhole and VCT switches

A.1.6 High-radix Switches

In this subsection, we analyze the area and critical path of the canonical switch with respect to the number of input/output ports. Note that, high-radix switches (more than 5 ports) are required to implement high-radix topologies as C-mesh or flattened butterflies analyzed in Chapter 5. Table A.4 shows the area and the critical path for a canonical switch with different number of input/output ports. As it can be seen, the area, and the critical path increases as the number of input/output ports increases, as expected.

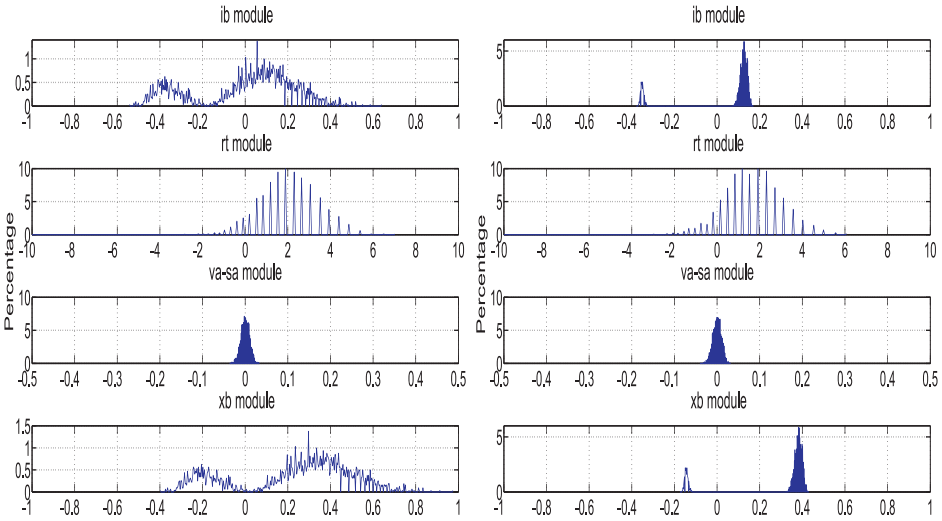
Ports	Area (um^2)	Critical Path (ns)
3 Ports	10108	0.69
4 Ports	14083	0.70
5 Ports	18630	0.75
6 Ports	22952	0.83
8 Ports	35154	0.89
10 Ports	46870	0.96

Table A.4: Area and frequency for the wormhole and VCT switches

A.1.7 Variability in the Router

In order to analyze the effects of variability in the switch presented in the previous section, we applied the variability model to 100 instances of our 8x8 mesh NoC synthesized using 45nm technology and studied how variability modifies the operating frequency of each of the switches of the network and also each of their modules. From the 100 NoC instances analyzed, 50 of them were produced using a value equal to 1 for the ρ correlation parameter while the other 50 were produced with ρ set to 0.5. We analyzed the influence in the switch of random and systematic variations. Figures A.9(a) and A.9(b) show the probability distribution function (*pdf*) of the relative operating frequency with respect to their own nominal frequency (Table A.1) of each stage of the switch in two scenarios: when only systematic variation with correlation 1 is considered(A.9(a)) and when only random variation is taken into account(A.9(b)).

Figures A.9(a) and A.9(b) show that systematic variation has a larger influence in the operating frequency of the switch than random variability. This is due to the fact that random variability differently affects two adjacent components. Thus, random variability , as will be shown later, may even be reduced or canceled as the number of gates in a chain of logic increases [107], as it can be seen for the IB and the XB module. Notice that despite the differences in the *pdf* of the different stages, all the *pdf* have the same nature. They can be seen as an addition of different peaks which are spread out due to the variability . Each peak represents different paths in a stage. These paths become alternatively the critical path in their respective stage. The fact



(a) Only systematic L_{eff} variations considered($\rho = 1$). (b) Only random V_{th} variations considered.

Figure A.9: Frequency variation in the stages of a switch pipelined

that the critical path changes due to the variability increases the difficulty to estimate the influence of the variability in a pre-synthesis design, because not only the nominal critical path has to be taken into account.

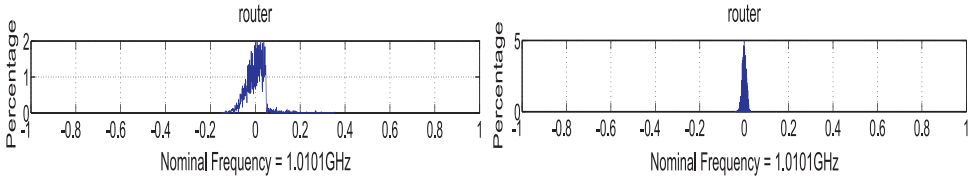
Furthermore, Figures A.9(a) and A.9(b) show that variability (both systematic and random) influence differents stages in different ways. However, the different kinds of variability affects the same stage in the same manner. Then, the properties of the implementation of each stage is critical in order to understand how the variability affects it. For a small and fast module (like the RT module) with a small number of gates in the different paths, variability affects the most. Small changes due to the variability have great impact. On the opposite extreme, a slow module with a big number of gates in their paths (VA-SA module) is slighty affected. Notice that these two modules as their number of paths is small have only one main peak. In the middle of these two extremes are the IB and the XB module. Their main characteristic is that as their number of paths is bigger and these path are similar (they are regular modules), the number of paths that becomes the critical path increases.

Figures A.10(a) and A.10(b) show the probability distribution function

(*pdf*) of the relative operating frequency with respect to their own nominal frequency of the switch when only systematic variation with correlation 1 is considered(A.10(a)) and when only random variation is taken into account(A.10(b)). When only random variability is considered, the influence of this varibility is smaller than when only systematic variability is considered. Then, when only random variability is considered, it is the VA-SA stage the one that determines the operating frequency in all cases since this stage is slower than the others. On the other hand, when only systematic variability is considered the operating frequency can be determined for different stages despite the VA-SA module is still being the main stage that fixed the operating frequency of the switch. This can be seen in Table A.5. Table A.5 shows the correlation between the operating frequency of the switch and the operating frequency of the stages of this switch. Note that the correlation of the VA-SA stage is the higest in all case because this stage is the stage that fixed the operating frequency of the switch in almost all the cases. Note however, that the correlation when only systematic (sys) variability is considered is higher than when only random (rnd) variability is taken into account in all the stages. When only systematic variability is considered, this correlation is high even for those modules that do not determined the operating frequency of the switch. This is due to the fact that all the components in the switch are affected by variability in a similar way. This mean that if the frequency of one stage is reduced because systematic variability, then the frequency of the other stage will probably be also reduced. therefore, it can be seen as a biased variability that causes that the critical path does not change among the switches so often. This effect is not present when only random variability is considered due to the nature of the random variability that affects adjacents components differently.

Figures A.11(a),Figures A.11(b), A.12(a) and A.12(b) show the probability distribution function (*pdf*) of the operating frequency of each of the stages of a switch and this switch when systematic and random sources of variation are simultaneously considered. These figures and Table A.5 show that there exist small difference in frequency, between high and low correlation.

Table A.6 shows the main parameters of each configuration described above. It shows the nominal, maximum, mean and minimum frequencies and



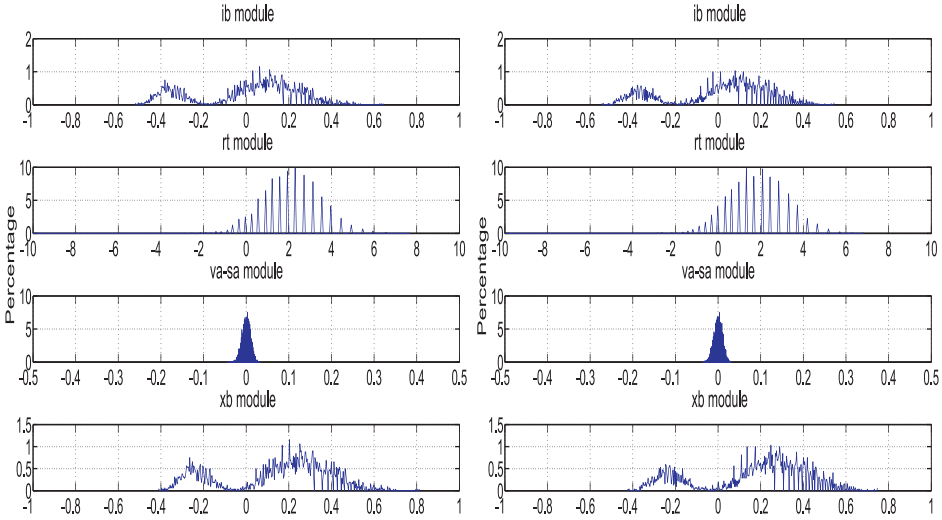
(a) Only systematic L_{eff} variations considered($\rho = 1$).

(b) Only random V_{th} variations considered.

Figure A.10: Frequency variation in switch pipeline

	variability				
stage	sys($\rho = 1$)	sys($\rho = 0.5$)	rnd	sys($\rho = 1$)+rnd	sys($\rho = 0.5$)+rnd
IB	0.9955	0.8789	0.1178	0.9732	0.8745
RT	0.9922	0.7644	0.2354	0.9549	0.6879
VA-SA	1.0000	0.9998	1.0000	0.9788	0.9902
XB	0.9933	0.9765	0.1715	0.9644	0.9712

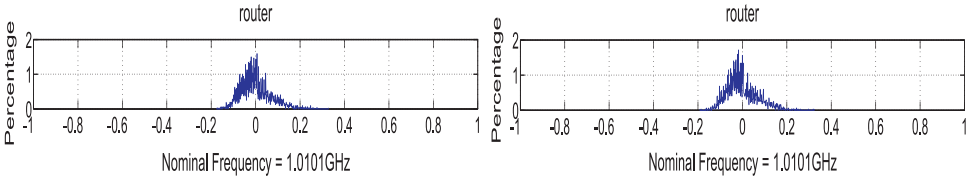
Table A.5: Correlation between stage delay and switch delay



(a) Systematic correlation of variability set to 1.

(b) Systematic correlation of variability set to 0.5.

Figure A.11: Frequency variation in the stages of a switch pipelined as a consequence of both systematic and random variations.



(a) Systematic correlation of variability set to 1.

(b) Systematic correlation of variability set to 0.5.

Figure A.12: Frequency variation in switch pipeline as a consequence of both systematic and random variations.

Param.	sys($\rho = 1$)	sys($\rho = 0.5$)	rnd	sys($\rho = 1$)+rnd	sys($\rho = 0.5$)+rnd
Nom. Freq.	1.0101	1.0101	1.0101	1.0101	1.0101
Max. Freq.	1.3158	1.1891	1.0373	1.3316	1.3298
Mean Freq.	0.9624	0.9635	1.0091	1.0023	1.0041
Min. Freq.	0.8123	0.8143	0.9681	0.8258	0.8110
σ/μ	0.0642	0.0657	0.0094	0.0695	0.0703

Table A.6: Nominal, maximum, mean and minimum frequencies and frequency variation of a switch

the frequency variation of each *pdf*. Frequency variation is computed as (σ/μ) where σ is the standard deviation and μ is the mean of the *pdf*. Data in Table A.6 confirm that the exact value of th ρ parameter introduces very small differences. Moreover, random variability moves the mean frequency more than the systematic one. As mentioned before, this is due to the fact that random variability makes that the critical path changes from one instance of the switch to another more often than systematic variability.

Bibliography

[1] Giorgos Passas, Manolis Katevenis, and Dionisios N. Pnevmatikatos. A 128 x 128 x 24gb/s crossbar interconnecting 128 tiles in a single hop and occupying 6% of their area. In *NOCS*, pages 87–95, 2010.

[2] Yatin Hoskote, Sriram R. Vangal, Arvind Singh, Nitin Borkar, and Shekhar Borkar. A 5-ghz mesh interconnect for a teraflops processor. *IEEE Micro*, 27(5):51–61, 2007.

[3] J. Rattner. Single-chip cloud computer: An experimental many-core processor from intel labs.

[4] Tile-gx processors family.

[5] ARM Ltd. Cortex-a5 processor, 2010.

[6] Intel Corp. Intel atom processor for nettop platforms, 2008.

[7] Jose Flich and Davide Bertozzi. *Designing Network On-Chip Architectures in the Nanoscale Era*. Computational Science Series. Chapman and Hall, 2010.

[8] Giovanii de Micheli and Luca Benini. *Networks on Chips: Technology And Tools*. In Systems on Silicon. Morgan Kaufmann Pub, July 2006.

[9] D.C. Pham, T. Aipperspach, D. Boerstler, M. Bolliger, R. Chaudhry, D. Cox, P. Harvey, P.M. Harvey, H.P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Pham, J. Pille, S. Posluszny, M. Riley, D.L. Stasiak, M. Suzuoki, O. Takahashi, J. Warnock, S. Weitzel, D. Wendel, and K. Yazawa. Overview of the architecture,

circuit design, and physical implementation of a first-generation cell processor. *Solid-State Circuits, IEEE Journal of*, 41(1):179 – 196, jan. 2006.

[10] Amit Kumar, Partha Kundu, Arvind P. Singh, Li-Shiuan Peh, and Niraj K. Jha. A 4.6tbits/s 3.6ghz single-cycle noc router with a novel switch allocator in 65nm cmos. In *ICCD*, pages 63–70, 2007.

[11] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar. An 80-tile 1.28tflops network-on-chip in 65nm cmos. In *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pages 98 –589, feb. 2007.

[12] C. Park, R. Badeau, L. Biro, J. Chang, T. Singh, J. Vash, B. Wang, and T. Wang. A 1.2 tb/s on-chip ring interconnect for 45nm 8-core enterprise Xeon processor. In *International Solid State Circuits Conference*, 2010.

[13] Robert D. Mullins, Andrew West, and Simon W. Moore. The design and implementation of a low-latency on-chip network. In *ASP-DAC*, pages 164–169, 2006.

[14] Hiroki Matsutani, Michihiro Koibuchi, Hideharu Amano, and Tsutomu Yoshinaga. Prediction router: A low-latency on-chip router architecture with multiple predictors. *IEEE Trans. Computers*, 60(6):783–799, 2011.

[15] Ron Ho, Kenneth W. Mai, Student Member, and Mark A. Horowitz. The future of wires. In *Proceedings of the IEEE*, pages 490–504, 2001.

[16] Antonio Flores, Juan L. Aragón, and Manuel E. Acacio. An energy consumption characterization of on-chip interconnection networks for tiled cmp architectures. *J. Supercomput.*, 45:341–364, September 2008.

[17] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Pub, San Francisco, CA, 2003.

[18] Boris Grot and Stephen W. Keckler. Scalable on-chip interconnect topologies. In *2nd Workshop on Chip Multiprocessor Memory Systems and Interconnects*, 2008.

[19] R.S. Patti. Three-dimensional integrated circuits and the future of system-on-chip designs. *Proceedings of the IEEE*, 94(6):1214 –1224, june 2006.

[20] A. W. Topol, D. C. La Tulipe, Jr., L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K. W. Guarini, and M. Ieong. Three-dimensional integrated circuits. *IBM J. Res. Dev.*, 50:491–506, July 2006.

[21] John Kim, James D. Balfour, and William J. Dally. Flattened butterfly topology for on-chip networks. In *MICRO*, pages 172–182, 2007.

[22] John Kim, William J. Dally, and Dennis Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. In *ISCA*, pages 126–137, 2007.

[23] Yu-Hsiang Kao, Najla Alfaraj, Ming Yang, and H. Jonathan Chao. Design of high-radix clos network-on-chip. In *Proceedings of the 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '10, pages 181–188, Washington, DC, USA, 2010. IEEE Computer Society.

[24] Crispin Gomez Requena, Francisco Gilabert Villamon, Maria Engracia Gomez Requena, Pedro Juan Lopez Rodriguez, and Jose Duato Mar. Ruft: Simplifying the fat-tree topology. *Parallel and Distributed Systems, International Conference on*, 0:153–160, 2008.

[25] Daniele Ludovici, Francisco Gilabert Villamón, Simone Medardoni, Crispín Gómez Requena, María Engracia Gómez, Pedro López, Georgi Nedeltchev Gaydadjiev, and Davide Bertozzi. Assessing fat-tree topologies for regular network-on-chip design under nanoscale technology constraints. In *DATE*, pages 562–565, 2009.

[26] James Balfour and William J. Dally. Design tradeoffs for tiled CMP on-chip networks. In *Proceedings of the 20th annual international conference on Supercomputing*, ICS '06, pages 187–198, New York, NY, USA, 2006. ACM.

[27] Reetuparna Das, Soumya Eachempati, Asit K. Mishra, Narayanan Vijaykrishnan, and Chita R. Das. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. In *HPCA*, pages 175–186, 2009.

[28] B. Grot, J. Hestness, S.W. Keckler, and O. Mutlu. Express cube topologies for on-chip interconnects. In *High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on*, pages 163 –174, feb. 2009.

[29] M. Karol, M. Hluchyj, and S. Morgan. Input versus output queueing on a space-division packet switch. *Communications, IEEE Transactions on*, 35(12):1347 – 1356, dec 1987.

[30] Mingjie Lin and Nick McKeown. The throughput of a buffered crossbar switch. *IEEE Communications Letters*, 9(5):465–467, 2005.

[31] Giorgos Passas, Manolis Katevenis, and Dionisios N. Pnevmatikatos. *VLSI Micro-Architectures for High-Radix Crossbar Schedulers.* NOCS, 2011.

[32] George Kornaros. BCB: A buffered crossbar switch fabric utilizing shared memory. In *DSD*, pages 180–188, 2006.

[33] S. Yalamanchili J. Duato and Ni. L. M. *Interconnection Networks: An Engineering Approach.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[34] James A. Kahle, Michael N. Day, H. Peter Hofstee, Charles R. Johns, Theodore R. Maeurer, and David J. Shippy. Introduction to the cell multiprocessor. *IBM Journal of Research and Development*, 49(4-5):589–604, 2005.

[35] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The splash-2 programs: characterization and methodological considerations. *SIGARCH Comput. Archit. News*, 23:24–36, May 1995.

[36] William J. Dally. Virtual-channel flow control. *IEEE Trans. Parallel Distrib. Syst.*, 3(2):194–205, 1992.

[37] Matteo Dall'Osso, Gianluca Biccari, Luca Giovannini, Davide Bertozzi, and Luca Benini. xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor socs. In *ICCD*, pages 536–, 2003.

[38] Li-Shiuan Peh and William J. Dally. A delay model and speculative architecture for pipelined routers. In *HPCA*, pages 255–266, 2001.

[39] Li-Shiuan Peh and William J. Dally. A delay model for router microarchitectures. *IEEE Micro*, 21(1):26–34, 2001.

[40] Jongman Kim, Dongkook Park, Theo Theocharides, Narayanan Vijaykrishnan, and Chita R. Das. A low latency router supporting adaptivity for on-chip interconnects. In *DAC*, pages 559–564, 2005.

[41] John Kim and Hanjoon Kim. Router microarchitecture and scalability of ring topology in on-chip networks. In *Proceedings of the 2nd International Workshop on Network on Chip Architectures*, NoCArc '09, pages 5–10, New York, NY, USA, 2009. ACM.

[42] John Kim. Low-cost router microarchitecture for on-chip networks. In *MICRO*, pages 255–266, 2009.

[43] Antoni Roca, José Flich, Federico Silla, and José Duato. Vctlite: Towards an efficient implementation of virtual cut-through switching in on-chip networks. In *HiPC*, pages 1–12, 2010.

[44] Pablo Abad, Valentin Puente, José Angel Gregorio, and Pablo Prieto. Rotary router: an efficient architecture for cmp interconnection networks. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 116–125, 2007.

[45] Simone Medardoni, Davide Bertozzi, and Enrico Macii. Power-optimal rtl arithmetic unit soft-macro selection strategy for leakage-sensitive technologies. In *ISLPED*, pages 159–164, 2007.

[46] Andrew A. Chien. A cost and speed model for k-ary n-cube wormhole routers. *IEEE Trans. Parallel Distrib. Syst.*, 9(2):150–162, 1998.

[47] Francisco Gilabert Villamón, María Engracia Gómez, Simone Medardoni, and Davide Bertozzi. Improved utilization of noc channel bandwidth by switch replication for cost-effective multi-processor systems-on-chip. In *NOCS*, pages 165–172, 2010.

[48] John Kim, William J. Dally, Brian Towles, and Amit K. Gupta. Microarchitecture of a high-radix router. *SIGARCH Comput. Archit. News*, 33:420–431, May 2005.

[49] Federico Silla. *Routing and flow control in networks of workstations*. PhD thesis, 1999.

[50] Everton Carara, Fernando Moraes, and Ney Calazans. Router architecture for high-performance nocs. In *SBCCI*, pages 111–116, 2007.

[51] Anthony Leroy, Paul Marchal, Adelina Shickova, Francky Catthoor, Frédéric Robert, and Diederik Verkest. Spatial division multiplexing: a novel approach for guaranteed throughput on nocs. In *CODES+ISSS*, pages 81–86, 2005.

[52] Pascal T. Wolkotte, Gerard J. M. Smit, Gerard K. Rauwerda, and Lodewijk T. Smit. An energy-efficient reconfigurable circuit-switched network-on-chip. In *IPDPS*, 2005.

[53] Jongman Kim, Chrysostomos Nicopoulos, and Dongkook Park. A gracefully degrading and energy-efficient modular router architecture for on-chip networks. In *Proceedings of the 33rd annual international symposium on Computer Architecture*, ISCA '06, pages 4–15, 2006.

[54] Jongman Kim, Chrysostomos Nicopoulos, Dongkook Park, Reetuparna Das, Yuan Xie, N. Vijaykrishnan, Mazin S. Yousif, and Chita R. Das. A novel dimensionally-decomposed router for on-chip communication in 3D architectures. In *In Proc. of ISCA*, pages 138–149. ACM Press, 2007.

[55] Smaragda Konstantinidou and Lawrence Snyder. The chaos router. *IEEE Trans. Computers*, 43(12):1386–1397, 1994.

[56] Gaspar Mora, Jose Flich, José Duato, Pedro López, Elvira Baydal, and Olav Lysne. Towards an efficient switch architecture for high-radix switches. In *ANCS*, pages 11–20, 2006.

[57] Nick McKeown, Martin Izzard, Adisak Mekkittikul, Bill Ellersick, and Mark Horowitz. The tiny tera: A packet switch core. *CoRR*, cs.NI/9810006, 1998.

[58] Thomas E. Anderson, Susan S. Owicki, James B. Saxe, and Charles P. Thacker. High speed switch scheduling for local area networks. *ACM Trans. Comput. Syst.*, 11(4):319–352, 1993.

[59] Yuval Tamir and Gregory L. Frazier. High-performance multi-queue buffers for vlsi communication switches. In *ISCA*, pages 343–354, 1988.

[60] Roberto Rojas-Cessa and Ziqian Dong. Load-balanced combined input-crosspoint buffered packet switches. *IEEE Transactions on Communications*, 59(5):1421–1433, 2011.

[61] John Kim, William J. Dally, Brian Towles, and Amit K. Gupta. Microarchitecture of a high-radix router. In *ISCA*, pages 420–431, 2005.

[62] Michael Butts, Anthony Mark Jones, and Paul Wasson. A structural object programming model, architecture, chip and tools for reconfigurable computing. In *FCCM*, pages 55–64, 2007.

[63] Hans M. Jacobson, Prabhakar Kudva, Pradip Bose, Peter W. Cook, and Stanley Schuster. Synchronous interlocked pipelines. In *ASYNC*, pages 3–12, 2002.

[64] Josep Carmona, Jordi Cortadella, Michael Kishinevsky, and Alexander Taubin. Elastic circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 28(10):1437–1455, 2009.

[65] Marc Galceran Oms, Alexander Gotmanov, Jordi Cortadella, and Michael Kishinevsky. Microarchitectural transformations using elasticity. *JETC*, 7(4):18, 2011.

[66] Jordi Cortadella, Marc Galceran Oms, and Michael Kishinevsky. Elastic systems. In *MEMOCODE*, pages 149–158, 2010.

[67] Charles Clos. A study of non-blocking switching networks. *Technical Journal*, pages 406–424, 1953.

[68] Jesus Camacho, José Flich, Antoni Roca, and José Duato. Pc-mesh: A dynamic parallel concentrated mesh. In *ICPP*, pages 642–651, 2011.

[69] Ümit Y. Ogras and Radu Marculescu. It's a small world after all: NoC performance optimization via long-range link insertion. *IEEE Trans. VLSI Syst.*, 14(7):693–706, 2006.

[70] Fabrizio Petrini, Marco Vanneschi, and Petrini Fabrizio Vanneschi Marco. K-ary n-trees: High performance networks for massively parallel architectures, 1995.

[71] Yu-Hsiang Kao, Ming Yang, N.S. Artan, and H.J. Chao. Cnoc: High-radix clos network-on-chip. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(12):1897 –1910, dec. 2011.

[72] George Michelogiannakis, James D. Balfour, and William J. Dally. Elastic-buffer flow control for on-chip networks. In *HPCA*, pages 151–162, 2009.

[73] George Michelogiannakis and William J. Dally. Router designs for elastic buffer on-chip networks. In *SC*, 2009.

[74] Masayuki Mizuno, William J. Dally, and Hideaki Onishi. Elastic interconnects: Repeater-inserted long wiring of compressing and decompressing data. In *Solid-State Circuits Conference, 2001. IEEE International*, 2001.

[75] Garrett Rose Aamir Zia, Sachhidh Kannan and H. Jonathan kao. Highly-scalable 3d clos noc for many-core cmps. In *NEWCAS*, pages 229–232, 2010.

[76] Guoqing Chen and Eby G. Friedman. Low-power repeaters driving RC and RLC interconnects with delay and bandwidth constraints. *IEEE Trans. VLSI Syst.*, 14(2):161–172, 2006.

[77] Dinesh Pamunuwa, Johnny Öberg, Li-Rong Zheng, Mikael Millberg, and Axel Jantsch. Layout, performance and power trade-offs in mesh-based network-on-chip architectures. In *VLSI-SOC*, pages 362–, 2003.

[78] Luciano Bononi, Nicola Concer, Miltos D. Grammatikakis, Marcello Coppola, and Riccardo Locatelli. Noc topologies exploration based on mapping and simulation models. In *DSD*, pages 543–546, 2007.

[79] Daniel Sanchez, George Michelogiannakis, and Christos Kozyrakis. An analysis of on-chip interconnection networks for large-scale chip multiprocessors. *TACO*, 7(1), 2010.

[80] Roberto Rojas-Cessa, Eiji Oki, and H. Jonathan Chao. On the combined input-crosspoint buffered switch with round-robin arbitration. *IEEE Transactions on Communications*, 53(11):1945–1951, 2005.

[81] Roberto Rojas-Cessa, Zhen Guo, and Nirwan Ansari. On the maximum throughput of a combined input-crosspoint queued packet switch. *IEICE Transactions*, 89-B(11):3120–3123, 2006.

[82] M. Katevenis and G. Passas. Variable-size multipacket segments in buffered crossbar (cicq) architectures. In *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, volume 2, pages 999 – 1004 Vol. 2, may 2005.

[83] Ioannis Papaefstathiou, George Kornaros, and Nikolaos Chrysos. A buffered crossbar-based chip interconnection framework supporting quality of service. In *ACM Great Lakes Symposium on VLSI*, pages 90–95, 2007.

[84] Aydin O. Balkan, Gang Qu, and Uzi Vishkin. A mesh-of-trees interconnection network for single-chip parallel processing. In *ASAP*, pages 73–80, 2006.

[85] Aydin O. Balkan, Gang Qu, and Uzi Vishkin. Mesh-of-trees and alternative interconnection networks for single-chip parallelism. *IEEE Trans. VLSI Syst.*, 17(10):1419–1432, 2009.

[86] Samuel Rodrigo, José Flich, Antoni Roca, Simone Medardoni, Davide Bertozzi, Jesus Camacho, Federico Silla, and José Duato. Cost-efficient on-chip routing implementations for cmp and mpsoc systems. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 30(4), 2011.

[87] Jih-Kwon Peir and Yann-Hang Lee. Look-ahead routing switches for multistage interconnection networks. *J. Parallel Distrib. Comput.*, 19(1):1–10, 1993.

[88] Aydin O. Balkan, Gang Qu, and Uzi Vishkin. Arbitrate-and-move primitives for high throughput on-chip interconnection networks. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS), Volume II*, pages 441–444.

[89] L.P. Carloni, K.L. McMillan, A. Saldanha, and A.L. Sangiovanni-Vincentelli. A methodology for correct-by-construction latency insensitive design. In *Computer-Aided Design, 1999. Digest of Technical Papers. 1999 IEEE/ACM International Conference on*, pages 309 –315, 1999.

[90] 45nm FreePDK. The Nangate open cell library.

[91] Design Compiler User Guide. Technical report, 20011.

[92] Encounter User Guide. Technical report, 20011.

[93] J.E. Miller, H. Kasture, G. Kurian, C. Gruenwald, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal. Graphite: A distributed parallel simulator for multicores. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1 –12, jan. 2010.

[94] Power Compiler User Guide. Technical report, 20011.

[95] Simone Medardoni, Marcello Lajolo, and Davide Bertozzi. Variation tolerant NoC design by means of self-calibrating links. In *DATE*, pages 1402–1407, 2008.

[96] Carles Hernandez, Antoni Roca, Federico Silla, Jose Flich, and José Duato. Improving the performance of GALS-based NoCs in the presence of process variation. In *NOCS*, pages 35–42, 2010.

[97] Ibis Benito. *Global Interconnects in the Presence of Uncertainty.* PhD thesis, University of massachusetts, 2008.

[98] Predictive Technology Model.

[99] A.B. Kahng, Bin Li, Li-Shiuan Peh, and K. Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 423 –428, 2009.

[100] The International Technology Roadmap for Semiconductors. 2009 edition. Technical report, 2009.

[101] Radu Teodorescu, Jun Nakano, Abhishek Tiwari, and Josep Torrellas. Mitigating parameter variation with dynamic fine-grain body biasing. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 40, pages 27–42, Washington, DC, USA, 2007. IEEE Computer Society.

[102] Rajamohana Hegde and Naresh R. Shanbhag. Toward achieving energy efficiency in presence of deep submicron noise. *IEEE TRANSACTIONS ON VLSI SYSTEMS*, 8(4):379–391, 2000.

[103] Jason Cong, David Zhigang Pan, and Prasanna V. Srinivas. Improved crosstalk modeling for noise constrained interconnect optimization. In *IN PROC. ASIA SOUTH PACIFIC DESIGN AUTOMATION CONF*, pages 373–378, 2001.

[104] Clint Hilton and Brent E. Nelson. A flexible circuit-switched noc for fpga-based systems. In *FPL*, pages 191–196, 2005.

[105] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *MICRO*, pages 469–480, 2009.

[106] Eung S. Shin, Vincent J. Mooney, III, and George F. Riley. Round-robin arbiter design and generation. In *Proceedings of the 15th international symposium on System Synthesis*, ISSS '02, pages 243–248, New York, NY, USA, 2002. ACM.

[107] Carles Hernández, Federico Silla, and José Duato. A methodology for the characterization of process variation in noc links. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, pages 685–690. European Design and Automation Association, 2010.