Roca Pérez, A.; Flich Cardo, J.; Silla Jiménez, F.; Duato Marín, JF. (2011). A low-latency modular switch for CMP systems. Microprocessors and Microsystems. 35(8):742-754. doi:10.1016/j.micpro.2011.08.011.

A Low-Latency Modular Switch for CMP Systems

Antoni Roca, José Flich, Federico Silla, José Duato
Grupo de Arquitecturas Paralelas
Departamento de Informática de Sistemas y Computadores
Universidad Politécnica de Valencia
c/camino de vera s/n, 46022, Valencia
anrope2@gap.upv.es

Abstract

As technology advances, the number of cores in Chip MultiProcessor systems and MultiProcessor Systems-on-Chips keeps increasing. The network must provide sustained throughput and ultra-low latencies.

In this paper we propose new pipelined switch designs focused in reducing the switch latency. We identify the switch components that limit the switch frequency: the arbiter. Then, we simplify the arbiter logic by using multiple smaller arbiters, but increasing greatly the switch area. To solve this problem, a second design is presented where the routing traversal and arbitrations tasks are mixed. Results demonstrate a switch latency reduction ranging from 10 to 21%. Network latency is reduced in a range from 11 to 15%.

Keywords: Network-on-Chip, switch design, arbitration implementation

1. Introduction and Motivation

It is well-known that current Chip MultiProcessor (CMP) and high-end MultiProcessor System-on-Chip (MPSoC) designs are growing in their number of components. As technology advances, more and more transistors can be included in the same die. Rather than aggregating several complex processors in the die, the trend is to replicate and include many simpler ones. This is driven by the power consumption concern, as smaller devices are more power-efficient than complex ones.

This ever growing number of devices demands an efficient interconnect structure inside the chip. Initial implementations relied on buses or crossbars. However, both lack scalability in terms of network bandwidth and implementation cost, thus becoming either a bottleneck when the number of devices to interconnect increases or an unfeasible implementation option. As a solution, the network-on-chip (NoC) concept arose. The idea is quite simple, a point-to-point network inside the chip is implemented to connect all the devices. Current research prototypes by Intel deploy a 2D mesh topology built from switches and links. Such a network is used to interconnect 80 simple (2-FPU units) cores

in the Polaris chip[10], or 24 dual-core tiles each one being x86 compatible and able to run an operating system, in the case of the Single-chip Cloud Computing prototype [31]. Moreover, Tilera recently announced a 100 core chip that also includes a 2D mesh [32].

In this scenario, and with the expectations to reach hundreds of cores in the near future, an efficient implementation of the NoC becomes a challenge. The 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 that use wormhole switching in order to increase clock frequency and reduce buffer requirements. Moreover, it is common to use the Stop&Go flow control protocol to set the advance of data between switches and network interfaces and avoiding buffer overflows.

The basic pipelined wormhole switch design is made of four 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) may be required to arbitrate for the output virtual channel amongst the virtual channels of an input port.

Several techniques have been proposed to reduce the depth of the switch pipeline and thus providing lower latencies [8]. One method is to speculatively contend for the virtual channel (VA stage), in parallel with the SA stage. Another possibility is to speculatively forward the flit through the internal crossbar at the same time the VA and SA stages are performed, thus saving two cycles in total [8]. Finally, the RC stage can be removed from the critical path if the previous switch computes the output port at the next switch, and in parallel with the VA, SA, and ST stages, thus ending up in two stages.

The described efforts to reduce the switch pipeline show how critical is the switch delay in overall network performance. Actually, applications are specially sensitive to latency [27]. In this way, although achieving high throughput numbers in a NoC is important, it is much more challenging to achieve ultra low latencies. Indeed, as NoC bandwidth is not limited by pin count (as is the usual case for off-chip networks), large bandwidths can be, relatively easily, achieved by increasing link width. However, in order to get low latency, an important design effort must be done for the switch itself.

In this paper we aim to reduce switch latency. However, instead of reducing

the number of cycles of the basic pipelined switch design, which is orthogonal to our proposals, we rather identify the most consuming operations along the critical path of the switch, and propose three techniques to reduce such overhead. As on example of this rationale, Table 1 shows the delay of each stage for the basic 4-stage pipelined switch design previously described. We target a 2D mesh network connecting one end node per switch. Thus, a 5-port switch is assumed. As can be observed, the slowest stage is the SA stage which delay is 0.75ns. Such delay hinders the switch to operate at frequencies higher than 1.33 GHz. The delay of the slowest stage fixes the delay of the rest of stages because all of them function at the same operating frequency. Moreover, the total switch latency is 3 ns since the switch has four stages.

modules	area (um^2)	critical path (ns)
IB	1324.78	0.52
RC	124.26	0.32
SA	337.88	0.75
ST	1975.6	0.47

Table 1: Area and latency for the pipeline stages of the basic switch design.

Our first proposal aims to reduce the complexity of the SA stage in the switch. For that purpose, we reduce the complexity of the arbiter that comprises the SA stage. This is achieved by taking into account the routing algorithm implemented in the network (DOR routing). Notice that by using DOR, packets will never take some output ports from a given input port. For example, flits coming from X input ports may request any output port (or the local port). However, packets from Y input ports will only request Y output ports (or the local port). Based on this reasoning, the arbiter can be simplified for Y output ports. Arbiters for X output ports will be also simplified by providing parallel links, as will be explained later. The net result is a faster SA stage, what provides a reduced clock cycle, thus achieving lower latency. Similarly, a second proposal is presented where a high-performance switch is designed.

With the first proposal, the latency is significantly reduced at the expense of a non-negligible increase in area. Indeed, additional parallel ports will be provided to the switch. In order to overcome this issue, we further exploit the concept of simplifying the bottleneck stage. In this case, we propose a third approach where we partition two stages (SA and ST) into a tree of smaller and, thus, faster components. With this third proposal the switch is much faster without the area overhead of the initial approach.

The remaining of the paper is organized as follows. In Section 2 we describe the related work. Then, in Section 3 we describe the basic switch architecture we will use as the starting point for our proposals. In Section 4 we introduce the different switch architectures proposed in the paper where several small arbiters are used instead of a single and complex arbiter. In Section 5 we present an evolved architecture which minimize the area overhead required by the initial proposal. Then, in Section 6 and Section 7 we evaluate the new switch

architectures and conclude the paper in Section 8.

2. Related Work for CMP Switches

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 [7]. 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 [10]. In this paper we focus on pipelined switch designs for CMP systems.

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 [22]. 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 [21] different techniques are applied and a one-cycle switch is achieved, or in [17] the output port is predicted rather than computed in order to minimize latency.

One of the main contributors to latency in a switch is the arbiter algorithm that schedules how flits in the input buffers are dispatched to the output ports. Indeed, the design of a fast arbiter algorithm is key to achieve a high-performance low-latency switch. Several scheduling algorithms have been proposed, like PIM [1], PPA [4], DRRM [5], *i*SLIP [19], etc. These algorithms are iterative and approximate a maximum size matching by finding a maximal size matching, that means that the arbiter obtains the maximum performance – assuming the performance as the number of flits that could traverse the switch – not taking into account the delay introduced. However, as remarked in [29], these algorithms are slow and impractical for a high speed switch and, additionally, may cause unfairness. To overcome this issue, other non-iterative algorithms have been developed [25, 29]. The main property of these schemes is their speed and simplicity at the expense of some loss in performance (lower matching rates) when compared with previous iterative arbiter schemes. However, this lower matching capability is not a burden in real traffic conditions, as shown in [20].

In order to reduce the latency of the switch, the complexity of the arbiter can be further reduced by making use of a logic synthesis principle [18] 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. In fact, in [22], the authors remark the need of reducing the

complexity of the crossbar from previous works [6], arguing that smaller switch modules achieve faster switches. Similarly, Gilabert et al. [9] 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. In our case, we will relax area constraints in order to minimize latency.

Other interesting studies are those that discuss the possibility of replacing virtual channels by physical parallel ports. This has been inherited from the off-chip domain [26]. Carara et al. [2, 3] 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. [16] and Lane Division Multiplexing (LDM) technique introduced by Wolkotte et al. [28]. SDM and LDM basically increase the number of wires between switches to assign different bandwidth resources to each channel at the cost of increasing the critical path. Carara's contribution is to simplify SDM and LDM obtaining a reduction in the critical path and then obtaining better performance by replicating channels rather than using virtual channels.

Our switch is highly related with other low latency designs as those presented in [14]. In that paper, 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 [12], 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 [13]. Finally, another low latency switch highly related with our proposals is the one presented in [15] where an adaptive switch is presented. In this case, X direction is decoupled from the Y direction, thus reducing the latency of the whole switch.

3. Basic Switch Architecture

In this section we describe the switch design used throughout this paper. Figure 1 shows the main components of the switch. The switch is a pipelined input buffered wormhole switch with five stages: IB, RC, SA, ST, and link traversal (LT). Notice that the fifth stage does not belong to the switch itself.

We have designed a simple switch with five input and output ports. Thus, four ports are intended to provide connectivity with the neighbouring switches in the 2D mesh and the fifth port connects to the local computing core. Link width is set to 8 bytes. Flit size is also set to 8 bytes. Input buffers can store four flits. A Stop&Go flow control protocol has been deployed in order to control the advance of flits between adjacent switches. Additionally, the routing (RC) stage has been implemented to support the XY routing algorithm. Moreover, there is an RC module for each input port. Note that although the RC module has been designed to support the XY routing, each input port can forward packets to any output port including itself due to the complete crossbar implementation. Similarly, there is a switch allocator (SA) module for each output port. The

SA module determines when and which input port is connected to its requested output port. Finally, each SA module has been designed using a round-robin arbiter according to [25].

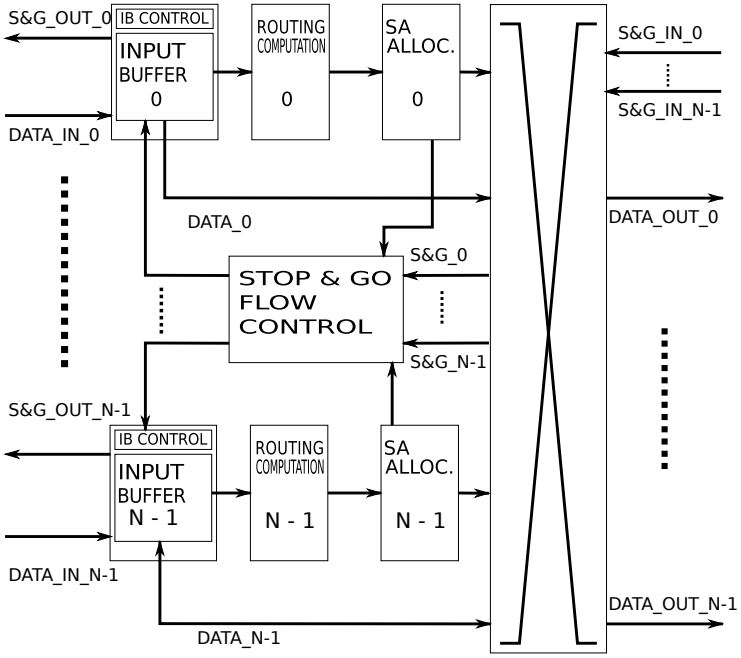


Figure 1: Switch schematic.

The switch has been implemented using the 45nm technology open source Nangate [30] with Synopsys DC. We have used M1-M3 metallization layers to perform the Place&Route with Cadence Encounter.

Using the same architecture of the basic switch presented above, we have designed an identical switch but with ten input/output ports. Thus, eight ports are intended to provide connectivity with the four neighbouring switches in a 2D mesh (two ports per neighbouring switch) and the last two ports connect the switch to the local computing core. In both switches, all input ports can be connected to all output ports, thus full connectivity is implemented. Table 2 shows the area and latency of the basic switch architectures presented in this section. The third column shows the latency of the slowest stage. This latency sets the operating frequency of the switch and hence each single stage of the switch is working at this frequency. Note that the 5-port switch is faster than the 10-port switch. The fourth column in Table 2 shows the switch delay, that is, the time that a flit remains in a switch when no contention is presented. This delay is equal to the delay of a single stage multiplied by the number of stages that has the switch pipeline.

In Table 1, it is shown that the SA stage is the slowest one. The SA stage determines when and which input port is connected to an output port. Then,

the latency of the SA stage is highly related with the number of input/output ports because as the number of input/output ports increases more complex is the task of interconnecting these input/output ports. Then, as the number of input/output ports increases the latency of the SA stage also increases, thus reducing the operating frequency of the switch.

switch	area (um^2)	slowest stage (ns)	switch delay (ns)
5 ports	12425.37	0.75	3.00
10 ports	27095.80	0.89	3.56

Table 2: Switch area and latency.

4. A Latency-Efficient Switch Architecture: Reducing the Arbiter Complexity

In this section we present two switch architectures intended to reduce message latency by reducing the critical path of the switch pipeline. Therefore, by shortening the longest stage of the pipeline, clock frequency will be increased, thus reducing the overall switch latency.

In our case, the switch allocator stage is the slowest stage. Thus, by reducing the latency of the switch allocator, the latency of the whole switch will be reduced because of a smaller clock cycle. Reducing the latency of a switch allocator is not easy [11, 25, 29]. One first attempt is to reduce the complexity of the algorithm implemented by the arbiter. The simplest arbiter algorithm is the round robin policy. More complex arbitration techniques [11, 20] may be used, although they are discarded due to their higher complexity, that translates into higher arbitration latencies. The second attempt is to reduce the complexity of the simplest round robin arbiter. Moreover, this is not trivial. Actually, a fast and simple arbiter implementation is presented in [25], which could be taken as an example of the fastest possible implementation that can be achieved because, although better implementations may be carried out, the difference in latency would not probably be significant.

Assuming that no faster implementation than the one presented in [25] can be carried out, the only parameter that could be modified in order to reduce the latency of this arbiter is the number of concurrent requests this arbiter deals with. Indeed, the complexity and delay of the arbiter is proportional to the number of simultaneous requests it can handle. In order to assess how the arbiter delay varies with the number of requests, we have evaluated the area and latency of the switch allocator with different number of requests. To do so, we have synthesized four different arbiters with 2, 3, 5, and 10 requests.

Table 3 shows the area and latency for the different switch allocators synthesized. As can be seen, the area and latency of the switch allocator increases with the number of requests. Additionally, the area required grows faster than the delay of the arbiter as shown in Figure 2. On the other hand, notice that the delay value for the 5-request arbiter in Table 3 is lower than the delay of the

switch allocator of the switch presented in the previous section. The difference in the latency is due to the extra control signals that compound the SA stage.

size	area (um^2)	critical path (ns)
2 request	95.00	0.344
3 request	142.13	0.383
5 request	348.89	0.433
10 request	564.37	0.525

Table 3: Area and latency of the arbiter for different number of requests.

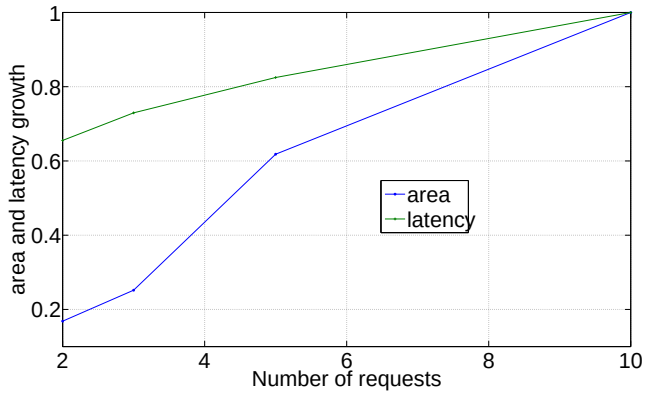
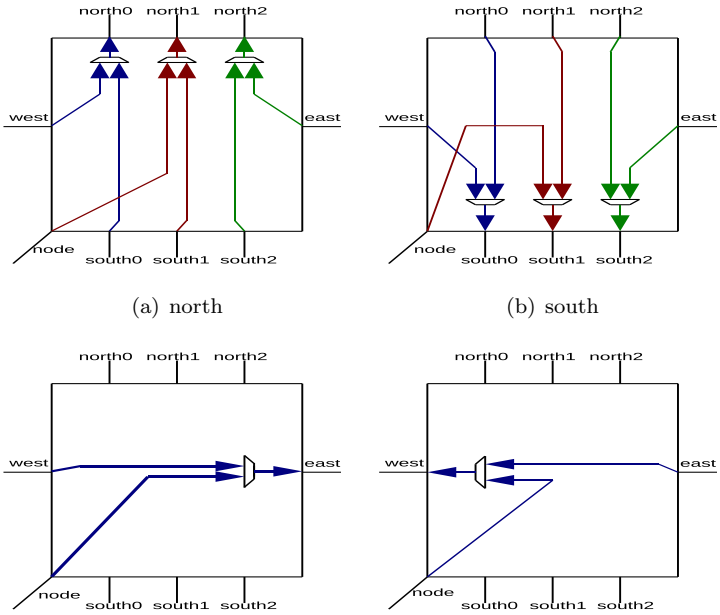


Figure 2: Area and latency of the arbiter growth with respect to the number of requests.

According to the delay numbers shown in Table 3, we could conclude that implementing the SA stage by using arbiters with a small number of requests would reduce the latency of that stage. However, that would also reduce the connectivity among ports. For example, if 2-request arbiters are used, then an output port could only receive requests from two different input ports. However, in a 2D mesh, each output port may receive requests from up to 4 input ports (3 ports connecting to other switches and one more port connecting to the local core). Therefore, if the SA stage is implemented by assuming 2-request arbiters, connectivity among ports will not be complete unless some additional changes are introduced. In order to keep using 2-request arbiters we replicate some of the output ports, thus guaranteeing that any input port can be connected to one of the replicas of the required output port.

Figure 3 shows a proposal for interconnecting ports. Y ports have been replicated three times in order to provide connectivity from both X ports and from the local port. Other schemes are feasible. However, as this switch assumes the use of the the DOR routing algorithm, replicating the Y port presents the additional advantage of featuring more bandwidth in the Y dimension, which usually gets congested when using DOR routing algorithm.



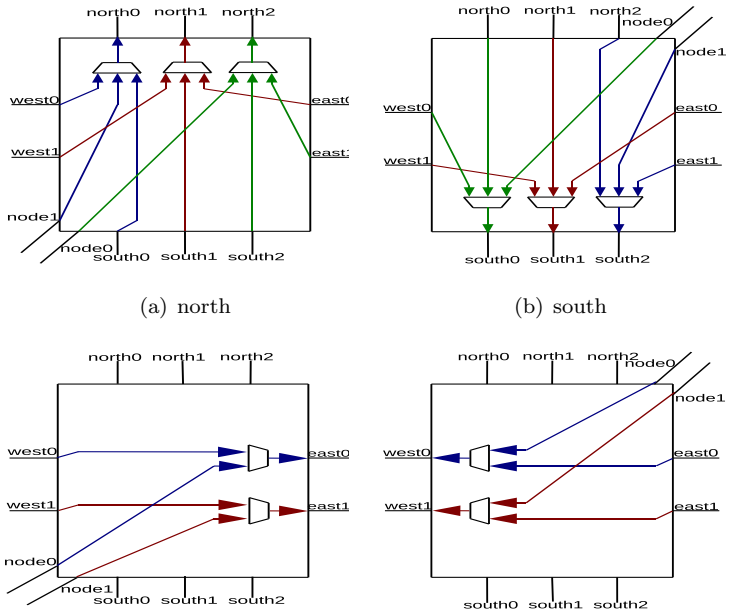
(a) north (b) south

(c) east (d) west

Figure 3: Ad-hoc switch connectivity proposal 1.

The connectivity scheme shown in Figure 3 guarantees that each output port arbiter receives requests from only 2 input ports, thus reducing the complexity of the switch allocator from a five-to-one configuration down to a two-to-one configuration. However, the number of ports in the switch has been increased from 5 to 9. Therefore, the complexity of the crossbar may considerably increase. However, an efficient crossbar implementation similar to the one presented in [9] can be deployed if each output port has its own independent crossbar. In this way, our switch proposal will include a 2×1 crossbar at each output port instead of a single 5×5 crossbar as shown in Section 3. Note that even in the case these smaller crossbars were used in the basic switch architecture, this would only save area, while no delay reduction would be achieved, as the bottleneck stage was the switch allocator. From now on, the switch proposal shown in Figure 3 will be referred to as *proposal 1*.

Our switch architecture can be extended if X ports are also replicated and 3-request arbiters instead of 2-request arbiters are used in the Y ports. In this way, the delay of the switch allocator stage will be slightly increased, but the bandwidth for connecting to other switches will be noticeably improved as it will be more balanced. Figure 4 shows the connectivity among ports for this option. Now we can include two local ports in the switch in order to match the number of input ports with the total number of requests the arbiters can deal with. From now on, the switch proposal shown in Figure 4 will be referred to as *proposal 2*.



(a) north (b) south

(c) east (d) west

Figure 4: Ad-hoc switch connectivity proposal 2.

It is noteworthy to mention that these proposals noticeably increase the number of ports of the switch. However, as this switch is intended to be used inside a chip, where interconnection bandwidth is not a major concern, the only disadvantage of an increased number of ports is the additional buffering required. Nevertheless, the philosophy of our switch is to trade transistors by latency, which is the real concern in current chips.

Finally, note that these proposals are independent of the arbiter scheme selected. That is, other arbiters can be used for deploying our proposal. In this way, if better arbiter implementations arise, then they can be incorporated to our architecture. Furthermore, our proposal is independent and orthogonal to other switch architectures, including unpipelined switches since the SA functionality is reduced. Moreover, other techniques as look-ahead routing or predictive routing are also compatible with our proposal.

5. A Latency-Efficient Switch Architecture: Reducing Area Overhead

As we have seen, the *proposal 1* and *proposal 2* approaches lead to an increase in the number of ports in the switch. In this new approach we embedded all the replicas for a given port into the switch, thus going back again to a 5-port switch architecture. In order to still rely on 2-request arbiter components, we

redesigned the crossbar and the arbiter together, and hence all the arbitration decisions are followed by a crossbar module. The main idea of the new switch architecture is to reduce almost all the switch operations to a sum of simple two-to-one arbitration-crossbar decisions. To do this, we developed a simple and fast two-to-one arbitration-crossbar (AC) module. Figure 5 shows the basic scheme of the AC module. The new module is a simple two-to-one round robin arbiter and a multiplexer used as a small crossbar. Furthermore, some extra control signals are needed. Note that AC module has its output signals registered by using flip-flops as it can be seen on the right part of the Figure 5, in order to keep pipelined the new switch architecture, and hence, to work similarly to the basic switch architecture previously shown. Keeping the functionality of both switches identical allows a fair performance comparison as it will be shown later. Table 4 shows the area and latency of the AC module when designed to obtain minimum latency. Comments that the latency of the AC module is 0.48 ns. This values is slower than the latency of a 5-request arbiter (which delay is 0.43 ns). However, these latencies are part of the whole latency of the respective switches. Then, the latency of the switch presented in this section is smaller than the latency of the basic 5-port switch, as it will be shown later.

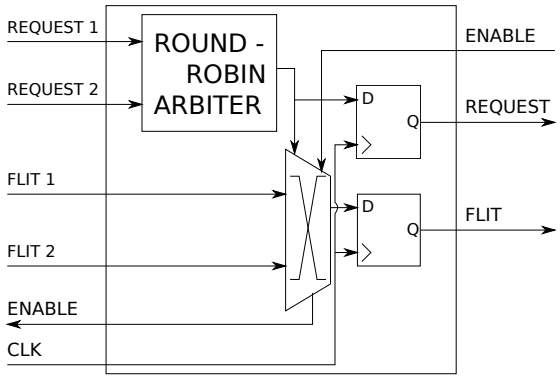


Figure 5: Schematic of the AC module used to implement our new switch architecture.

switch	area (um^2)	critical path (ns)
AC Module	974.05	0.48

Table 4: Area and latency of the AC module.

Figure 6 shows the basic scheme of the new switch architecture. This new switch architecture is referred as *proposal 3*. Note that the extra control signals are omitted to clarify the schematic. However, these extra control signals constitute a control module similar to the control module of the basic switch presented in Figure 1. In Figure 6, we can be seen that the new switch architecture is composed of four stages plus the link traversal stage, similarly to the basic switch architecture. The first and second stages of the new switch

architecture are identical to the basic switch, that is, the input buffer stage and the routing computation stage. However, the new switch architecture combines the SA and ST stages in the basic switch architecture and converts them into two new identical stages built on the simple AC module shown in Figure 5. Note, however, that the functionality of this new switch architecture is identical to the functionality of the basic switch architecture before introducing new input/output ports as it is was in the *proposals 1* and *proposal 2* in Section 4.

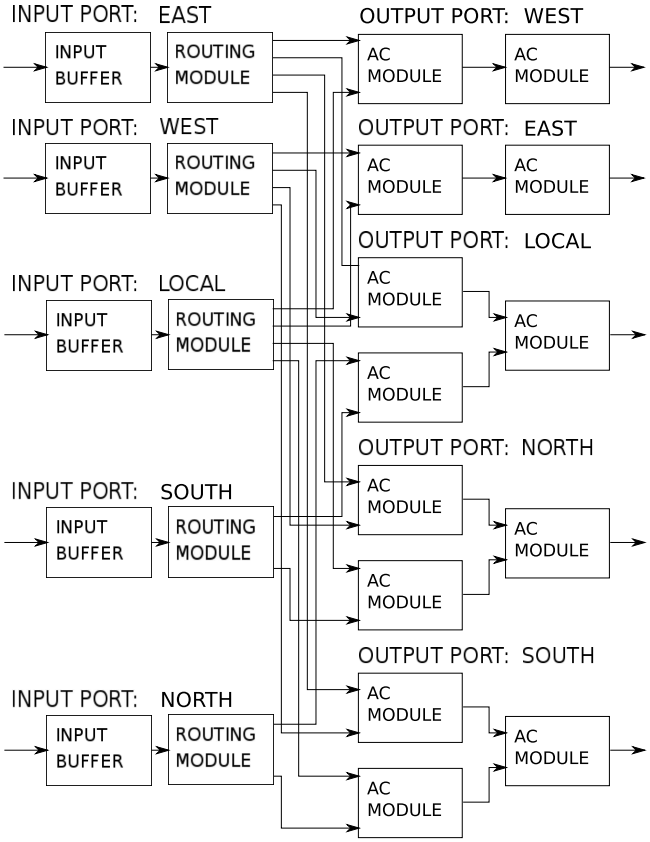


Figure 6: Schematic of the new switch architecture proposed.

As it can be seen in Figure 6, each request of each routing computation module is managed by a different AC module. That forces the routing computation stage to have its output signals registered in order to implement a pipelined switch. Notice that this means that each routing computation module is replicating the information as many times as requests it may forward to output ports. In order to minimize the impact of this redundancy, the switch has been optimized to implement only XY DOR routing. That means that the Y direction ports (north and south) input ports can only handle two output ports (local port or Y direction ports). However, the redundancy of data at the

output of the routing computation module is extra hardware resource when the communication between nodes is performed using unicast messages. But, this redundancy is necessary when transmitting broadcast/multicast messages since each request of an output port of a broadcast/multicast message can be dealt independently. Then, the redundancy of the information at routing computation output is optimum to perform broadcast/multicast techniques without introducing huge modifications in the switch. In fact, only a few changes to support broadcast/multicast in the RC stage are needed. In [23, 24] it is shown that these changes does not introduce any latency penalty in a switch.

To reduce the impact of the buffer redundancy at the output of the RC modules, the new switch architecture presents another important modification with respect to the basic switch. As is has been explained before, AC has registered output signals. Similarly, the output signals of the RC module are registered. That allows the input buffer to be reduced from five – as in the basic switch architecture – to 2, and the round trip time is still being fulfilled. The reduction of the size of the input buffer allows the new switch architecture to compensate the increase of the number of register – and hence an increase in area – produced by the necessity of register the outputs of each stage.

Finally, note in Figure 6 that the fourth stage of the output ports west and east, only handle one request form the previous stage, and hence when transmitting a message through the X direction only 3 stages are needed. It is maintained in order to keep the same functionality and behaviour in each output port, so how, to keep the same behaviour as the basic switch.

6. Evaluation of the Architecture Proposals 1 and 2

In this section we analyze the benefits of the proposed switch architecture over the basic ones presented in Section 3. To do so, we present the area and latency metrics as well as a thorough performance comparison of the different characteristics introduced by *proposal 1* and *proposal 2*.

6.1. Area and latency

Table 5 shows the area and latency of the switches proposed in Section 4. Remember that our proposals have nine and twelve ports rather than the five or ten that featured the basic architectures in Section 3. Obviously, as the number of ports increases (and hence the number of input buffers) the area of the proposed switches increases as well. However, note that both proposals have a shorter critical path than the basic switch architectures presented in Section 3. This means that both proposals have a higher operating frequency. Then, if we compare proposal 1 with a 5-port basic switch (as both of them have the same number of links in the X direction) we obtain that proposal 1 has a reduction of latency of 10.67% while there is an increase in area of 59.99%. Comparing our switch proposal 2 with a 10-port basic switch we obtain a reduction of latency of 20.22% with a small increase in area of 2.88%.

The huge difference when comparing the area increase in both cases is due to the fact that when comparing proposal 2 with a 10-port basic switch the number

switch	area (um^2)	critical path (ns)
proposal 1	19880.00	0.67
proposal 2	27876.33	0.71

Table 5: area and latency of the proposed switches.

of extra ports added is just two while in the first case the number of ports is almost doubled. Additionally, decoupling resources allows the synthesis tools to relax its constraints and then the increase in area produced by increasing the number of input buffers could be reduced by decreasing the area of the SA and ST stages. This behaviour is more evident when comparing proposal 2 with a 10-port basic switch. Nevertheless, remember that the purpose of our architecture is reducing latency at the expense of increasing area, if required.

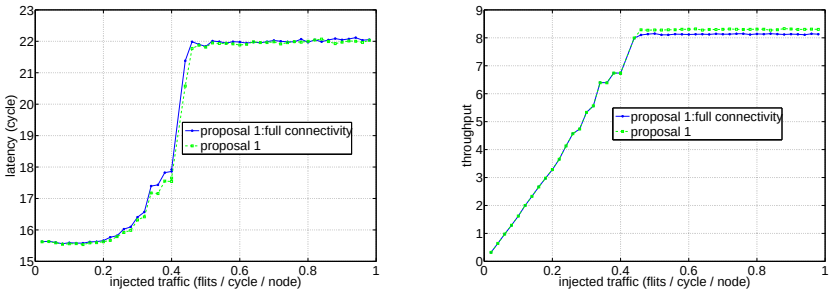
Also, note that the critical path of the proposed switches is not decreased according to the expected reduction of the critical path of the arbiter. This is due to the fact that in our proposed switches the critical path is now set by the IB stage instead of the SA stage. However, the critical path of the proposed switches is higher than the critical path of the IB stage of a basic 5-port switch – shown in Table 1. This is due to the higher number of ports and the control signals that conform the architecture of the IB stage in the proposed switches.

6.2. Effects of the partial connectivity on performance

In this section we analyze the performance penalty suffered when reducing the connectivity among the ports of a switch. For that purpose, we analyze two switches: the switch described in proposal 1 and the same switch but allowing that any input port can reach any output port, that is, full switch connectivity. Figure 7(a) and Figure 7(b) show the latency and throughput for a 4×4 network using both switches. Uniform traffic is used, and message size set to four flits. In both cases, switches are modelled with the same delay (one cycle per stage). As can be seen, there is a small loss in performance when full connectivity is not allowed. Notice, however, that this loss is small and it will be compensated when comparing both switches with its real frequency, as we will analyze later.

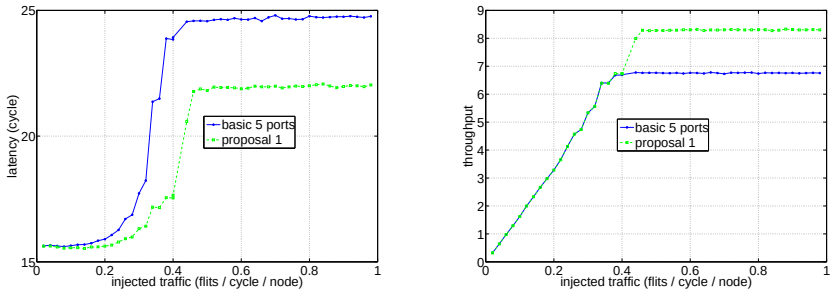
6.3. Effects on the increased number of ports

In this section we analyze variations in network performance when moving from the 5-port basic switch to the 9-port proposed architecture (proposal 1). The purpose of these experiments is to analyze the effect of a larger number of ports. Network size is 16 switches. These results do not take into account the operating frequency of each switch. Figure 8 shows the latency and the throughput for different injected uniform traffic rates. All the messages injected into the network are 4-flit long. Figure 8 shows that the performance of the ad-hoc switch is better than the basic switch. Our proposal obtains a higher throughput and a lower latency over the whole traffic range. This is due to the



(a) latency (b) throughput

Figure 7: Latency and throughput for proposal 1. Full connectivity for proposal 1 is also considered.



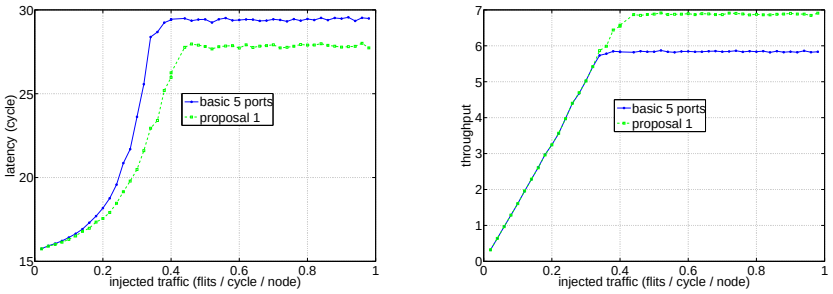
(a) latency (b) throughput

Figure 8: Performance comparison between a basic 5-port switch and proposal 1. All messages are 4-flits long.

fact that by increasing the number of ports we reduce the contention in each switch and increase the traffic bandwidth in the Y direction.

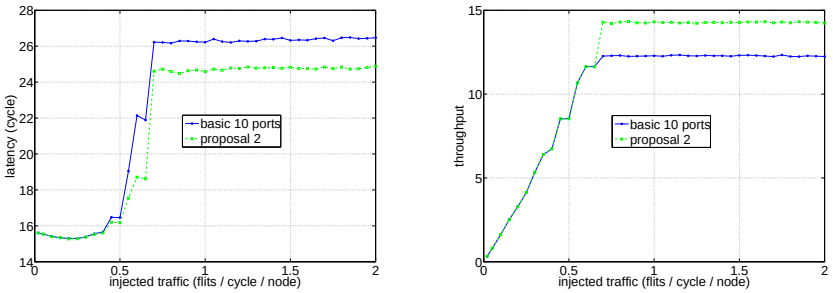
Figure 9 shows the latency and throughput of the same switches when the traffic injected into the network is made of 70% messages of 4 flits and 30% messages of 20 flits. Note that introducing longer messages makes the network performance worse (higher latency and lower throughput). Furthermore, differences between switches are reduced. However, despite the reduction of the difference in performance, the same conclusions as before can be obtained. From now on, all results presented will be taken when the traffic injected is made of 4-flit long messages.

Figure 10 shows the latency and the throughput for different injected uniform traffic rates for a 10-port basic switch and our switch proposal with twelve input/output ports (proposal 2). As before, the performance of the proposed switch is better than the basic switch for the same reasons explained above.



(a) latency (b) throughput

Figure 9: Performance comparison between a basic 5-port switch and proposal 1. Traffic is 70% 4-flit messages and 30% 20-flit messages.



(a) latency (b) throughput

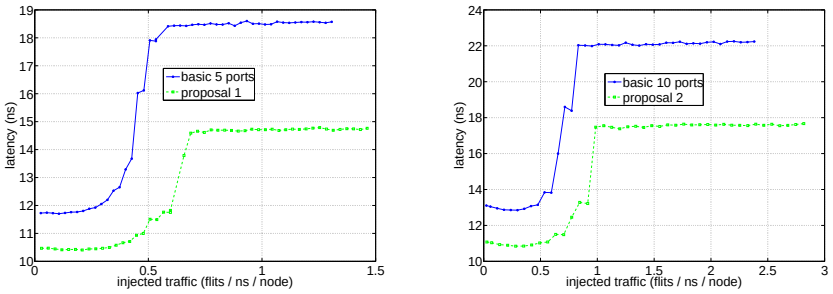
Figure 10: Performance comparison between a basic 10-port switch and proposal 2.

6.4. Effects of the increased switch frequency

The previous evaluations were focused on the impact of the connectivity pattern provided by the different switch architectures. Indeed all the switches were modelled at the same clock frequency. In this section we provide the real difference when switches are composed, each one using its maximum operating frequency. Figure 11 shows the latency in nanoseconds for different injection traffic rates for a 4×4 mesh network. Note that now the differences in performance described before are exacerbated as the proposed switches now work at higher operating frequencies. Note that differences are remarkable both for low injected traffic rates and for high injected traffic rates. For low injection traffic rates the reduction of the net latency is 11.37% and 15.41% as it can be seen in Figure 11(a) and Figure 11(b) respectively.

7. Evaluation of the Architecture Proposals 3

In this section we analyze the benefits of the proposed switch architecture in Section 5 over the basic ones presented in Section 3 and proposal 1 presented



(a) Basic 5-port architecture versus proposal 1

(b) Basic 10-port architecture versus proposal 2

Figure 11: Performance comparison for the different switch architectures when switch frequency is considered.

in Section 4. To do so, we present the area and latency metrics as well as a thorough performance comparison of the different characteristics introduced by the new architecture.

7.1. Area and latency

Table 6 shows the area and latency of the switch proposed in Section 5, *proposal 3*. Remark that the latency of the switch is 0.66 nanoseconds. This value is the sum of the latency of the AC module (see Table 4 in Section 5) with some extra control information generation.

When comparing *proposal 3* with the the basic architecture of five-ports presented in Section 3, remember that *proposal 3* maintains the number of input/output ports with respect to the basic architecture. However, despite the number of ports is not increased, the area of the switch increases. This is due to the redundancy of the data path inside the switch especially in stage 2. Remember that each routing computation implements a flit size buffer for each output port that can reach (see Figure 6. That means to increase the number buffers –with flit size– in 24%. This increment force that the area of the new switch architecture is incremented in a 22%. With respect to the operating frequency we observe that there is a reduction of 12%.

switch	area (um^2)	critical path (ns)
proposal 3	15163.28	0.66

Table 6: area and latency of the proposed switches.

Comparing *proposal 3* with *proposal 1*, we observe that both switches present almost the same operating frequency but *proposal 3* does not present a huge increment in area as presented in *proposal 1*. Furthermore, *proposal 3* does not increase the number of input/output ports and hence does not increase the complexity and the power consumption of the links.

7.2. Design Benefits of proposal 3

Another important benefit that the new switch architecture presents is its simplicity in the design. As it can be seen in Figure 6, only three kind of module are needed. Input buffers to perform the input ports, the routing computation modules and the basic arbitration-crossbar module. This simplicity allows the designer to have more control of the design constraint. First, the area of the switch is highly correlated with the number of registers used in the input ports and the rest of the modules. That number is fixed when the number of input ports and the flit size are determined. then, fixing the parameters that fix the area of the switch, the designer can only optimize the switch for power consumption or latency. As it is mentioned before, the basic arbitration-crossbar module with some extra control operations fix the critical path of the switch. That means that, both the input buffer and the routing computation modules can be designed for other purpose rather than minimum latency, as for example minimum power consumption [9]. Then, there is a high correlated dependency between the latency and power consumption of the basic arbitration-crossbar module and the latency and power consumption of the switch, and hence, optimizing the design of the switch for a concrete purpose means to design a simple arbitration-crossbar module. Table 7 shows the area, critical path, and power consumption for the *proposal 3* switch using different design constraints. Table 7 shows that the area of the switch has a fixed value (close to 14211 um^2) which is not possible to decrease. This value is highly related with the flit size, and the input buffer length that fix the number of registers which are the main contribution to the whole switch area. Another conclusion is that the power consumption is highly related with the clock frequency.

area (um^2)	critical path (ns)	power consumption (mW)
15163.28	0.66	32.07
14452.20	0.74	20.23
14258.58	0.83	16.12
14218.12	0.95	14.19
14211.58	1.02	12.56

Table 7: area and latency of the proposed switches for different design constraint.

Table 8 shows the critical path of the AC module when it is synthesized for different latency constraints. Furthermore, the second column shows the critical path of the switch created using the previous AC module previously independently synthesized. This switch is different for the switches synthesized which results are shown in Table 7. Those switches where globally synthesized. The third column shows the difference in latency between the switch latency and the AC module latency. It can be concluded that the difference between the switch latency and the AC module is almost constant and hence, designing and synthesising the AC module with a latency constraint is equivalent to design and synthesize the whole switch with the same latency constraint plus the difference constant defined. Note that designing and synthesising a simple and small

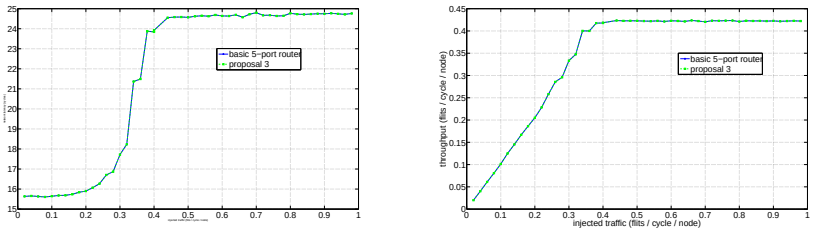
module is easier than designing and synthesising the whole switch, and the values obtained can be more close to the design constraint.

critical path(ns) AC module	critical path (ns) switch	difference
0.48	0.66	0.18
0.55	0.71	0.16
0.70	0.85	0.15
0.81	0.96	0.15
0.86	1.01	0.16

Table 8: area and latency of the proposed switches for different design constraint.

7.3. Simulation Results

We analyze the effect on the performance of *proposal 3* when it is compared to the five-port basic switch. Figure 12 shows the latency and the throughput of the basic 5-port switch and *proposal 3* switch. Uniform traffic is used, and message size set to four flits. Figure 12 shows that both switches have the same performance when the operating frequency is not considered (latency is measured in cycles). That means that both switches have the same functionality, and hence, any improvement in the operating frequency in the *proposal 3* switch will be reflected in a reduction in a message latency. This conclusion is identical when modifying the message size distribution of the simulation.



(a) latency (b) throughput

Figure 12: Performance comparison between a basic 5-port switch and proposal 3. All messages are 4-flits long.

In order to analyze the effect of the operating frequency when comparing the *proposal 3* switch with the five-port basic switch, we have run several applications in GEMS/SIMICS over a directory coherence protocol. The network is a 4x4 mesh network with two memory controllers. We have run SPLASH applications: Barnes, cholesky, fft, fmm , lu, lunc, ocean, oceanc, radiosity, radix, raytrace, volrend, watersnp, waterp. Alp benchmark applications: facerec and speechrec. And scientific applications: em3d, tomcatv, and unstructed. Figure 13 shows the execution time when running several application over a directory coherence protocol when using a basic five-port switch and *proposal 3*

switch. In all cases, *proposal 3* switch obtains better results, being all of them close to the difference in the operating frequency between switches: 12%.

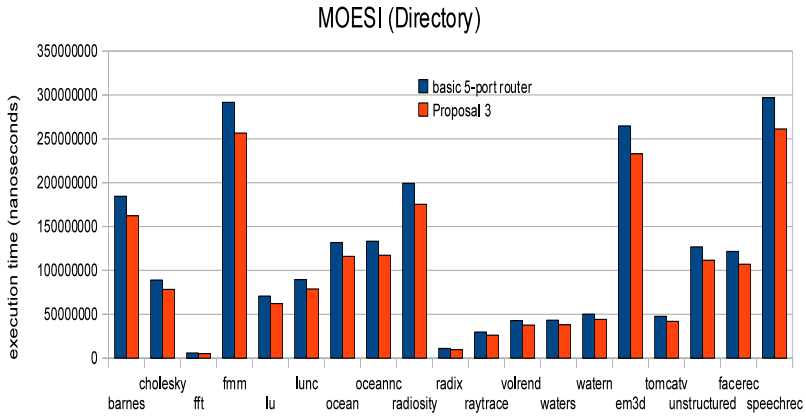


Figure 13: Performance comparison between a basic 5-port switch and proposal 3 when running different applications over a directory coherence protocol.

Similarly, Figure 14 shows the execution time when running several application over a directory coherence protocol when using a basic five-port switch and *proposal 3* switch. In all cases, *proposal 3* switch obtains better results, being all of them close to the difference in the operating frequency between switches: 12%.

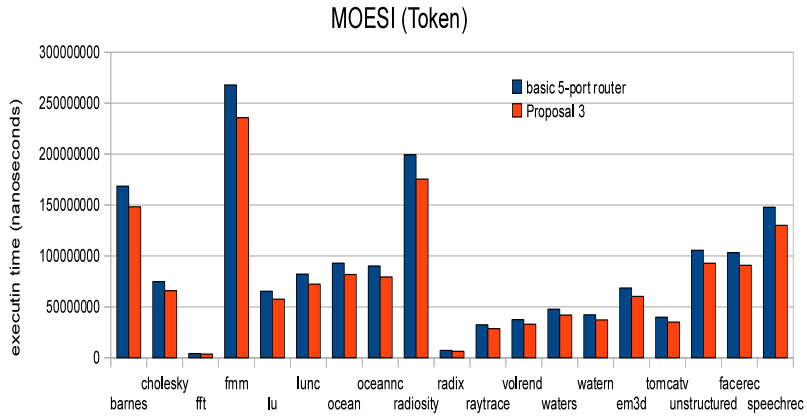


Figure 14: Performance comparison between a basic 5-port switch and proposal 3 when running different applications over a token coherence protocol.

7.4. Results compared With X improved

In Section 5 has been explained that *proposal 3* switch has an important benefit. Reconfigurating the operation tasks along the pipeline of the switch allows the switch to fulfil the routing tasks for the east and west ports in only 3 stages rather than 4 needed for the rest of ports. We implement this pipelined reduce switch in our network simulator – referred as *proposal 3 reduced pipeline*. Figure 15 shows the influence of reducing the pipeline depth in the X direction when running the applications with a directory coherence protocol. Note that, there are applications were the reduction in latency between *proposal 3* and *proposal 3 reduced pipeline* is considerable (up to 31% in Ocean). In average this reduction is up to 19%. When comparing the *proposal 3 reduced pipeline* with the basic 5-port switch this reduction increases up to 29% in average. In those cases where the improved switch is not better than the canonical switch, the differences are minimum.

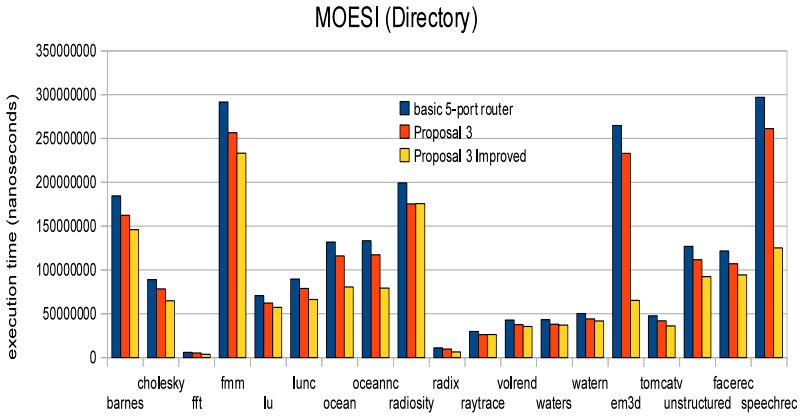


Figure 15: Performance comparison between a basic 5-port switch and proposal 3 when running different applications over a directory coherence protocol.

Similarly, Figure 16 shows the influence of reducing the pipeline depth in the X direction when running the applications with a token coherence protocol. Note that, the difference between switches are not as high as in the directory coherence protocol. In fact, the maximum difference between the *proposal 3* and *proposal 3 reduced pipeline* is 13%, but the mean reduction in latency is only 0.02%. The final reduction latency between *proposal 3 reduced pipeline* and the basic switch is in mean 14%.

8. Conclusions and Future Work

In this paper we have proposed various new switch designs to reduce the latency of the network. From the initial assumption of a 2D mesh topology and the use of the Dimension-Order Routing (DOR) algorithm, we have redesigned

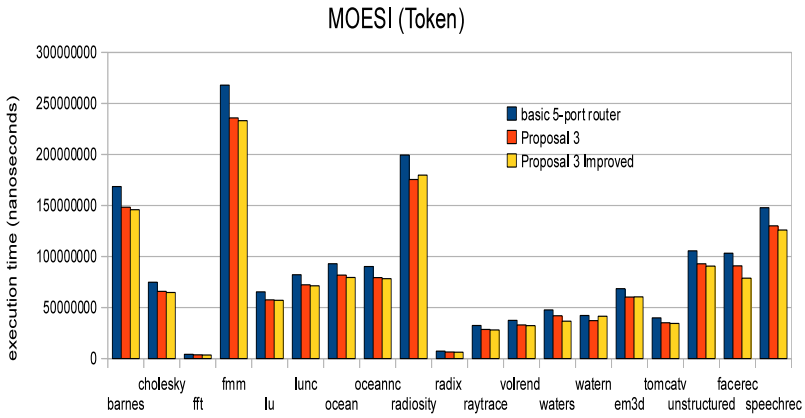


Figure 16: Performance comparison between a basic 5-port switch and proposal 3 when running different applications over a token coherence protocol.

a pipelined switch. In particular, at the first design presented the arbiter complexity has been reduced. Multiple smaller arbiters are used in parallel and thus exhibiting a lower latency. In order to build a full operational switch with such smaller arbiters new ports have been added to the switch, and different internal connections in the switch have been set. Based on such design style, different switch architectures (with different arbiter complexities) have been proposed.

Results show a clear reduction in switch latency. The new architecture with arbiters with two or three requests is able to reduce the delay of a canonical switch design by a range from 10% to 21%. This switch uses X ports, and thus, increases the switch area up to 59.99%. Network latency is reduced in a range from 11% to 15%.

Secondly, we have evolved this design in order to minimize the increment in area. From the fact that smaller arbiters are faster we have mixed operations tasks from the SA stage and the ST stage, creating a new switch that is mainly formed by addition of a simple and small module that performs arbitration-crossbar tasks. By introducing this module, we avoid the increment of new input/output ports and hence, minimize the increment of area. Results show that the increment of area is just 22% and there is a reduction in latency of 12%.

As future work we plan to analyze the impact of the new switch design when using virtual channels. Furthermore, the impact of the new switch design with other techniques as look-ahead routing should be done. Also, further evaluations with real applications will be analyzed. It is expected that applications will take profit of a faster switch.

Acknowledgement

This work was supported by the Spanish MEC and MICINN, as well as European Commission FEDER funds, under Grants CSD2006-00046 and TIN2009-14475-C04. It was also partly supported by the project NaNoC (project label 248972) which is funded by the European Commission within the Research Programme FP7.

References

[1] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, High-speed switch scheduling for local-area networks, ACM Trans. Computer Systems, vol. 1, no. 4 (1993) 319-352.

[2] E. Carara, N. Calazanz, F. Moraes, A new router architecture for High-Performance intrachip networks, Journal Integrated Circuits and Systems 3 (2008) 23-31.

[3] E. Carara, F. Moraes, N. Calazanz, Router architecture for high-performance NoCs, SBCCI (2007) 36.

[4] H.J. Chao, C.H. Lam, and X. Guo, A fast arbitration scheme for terabit packet switches", Proc. GLOBECOM (1999) 1236-1243.

[5] J. Chao, Saturn: a terabit packet switch using dual round-robin", IEEE Commun. Mag. 38 (2000) 78-84.

[6] A. chien, A cost and speed model for k-ary n-cube wormhole routers", Hot Interconnects (1993).

[7] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzim, L. Benini, Xpipes: a latency insensitive parameterized Network-on-chip architecture for Multi-Processor SoCs, Proceedings of the 21st International Conference on Computer Design (2003) 536–.

[8] W. J. Dally, B. Towles, Principles and practices of interconnection networks, Morgan Kauffman 2003.

[9] F. Gilabert, M.E. Gomez, S. Medardonni, D. Bertozzi, Improved utilization of NoC channel bandwidth by switch replication for cost-effective Multi-Processor Systems-on-Chip, NOCS (2010).

[10] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, S. Borkar, A 5-GHz mesh interconnect for a teraflops processor, in *IEEE Micro Magazine* (2007) 51-61.

[11] Y. Huang, C.M Ko, H.C. Teng, Design and performance analysis of a reconfigurable arbiter, in WSEAS Transactions on Electronics 4-5 (2008).

[12] J. kim, H. Kim, Router microarchitecture and scalability of ring topology in on-chip networks, Network on Chip Architectures NoCArc (2009).

[13] J. kim, Low-cost router microarchitecture for on-chip network, Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (2009).

[14] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, C. R. Das, A low latency router supporting adaptivity for on-chip interconnects, Design Automation Conference (2005) 559-564.

[15] S. Konstantinidou, L. Snyder, The Chaos Router, IEEE Transactions on Computers 43 (1994).

[16] A. Leroy, P. Marchal, A. Shickova, F. Catthoor, F. Robert, D. Verkest, Spatial division multiplexing: a novel approach for guaranteed throughput on NoCs, CODES-ISSS (2005) 81-86.

[17] H. Matsutani, M. Koibuchi, H. Amano, T. Yoshinaga, Prediction router: yet another low latency on-chip router architecture, HPCA (2009) 367-378.

[18] S. Medardoni, D. Bertozzi, E. Macii, Power-optimal RTL arithmetic unit soft-macro selection strategy for leakage-sensitive technologies, Proc. Int. Symp. on Low-Power Electronics and Design (2007) 159-164.

[19] N. McKeown, The iSLIP scheduling algorithm for input-queued switches, IEEE/ACM Trans. Networking 7 (1999) 188-201.

[20] S. S. Mukherjee, F. Silla, P. Bannon, J. Emer, S. Lang, D. Webb, A Comparative Study of Arbitration Algorithms for the Alpha 21364 Pipelined Router, in *ASPLOS X* (2002).

[21] Robert Mullins Andrew, The design and implementation of a low-latency on-chip network, Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA04) 2004.

[22] L.S. Peh, W.J. Dally, Low-latency virtual-channel routers for on-chip networks, In International Symposium on High-Performance Computer Architecture (2001) 255-266.

[23] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, J. Duato, Addressing manufacturing challenges with cost-efficient fault tolerant routing, ACM/IEEE International Symposium on Networks-on-Chip (2010).

[24] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, J. Duato, Fault-tolerant routing for next generation multicore chips, IEEE Transactions on Computers (2011).

[25] E. S. Shin, V.J.III Mooney, G.F. Riley, Round-robin arbiter design and generation, International Symposium on System Synthesis 15 (2002) 243-248.

[26] F. Silla, Routing and flow control in networks of workstations, PhD thesis (1999).

[27] J. C. Villanueva, J. Flich, J. Duato, H. Eberle, N. Gura, W. Olesinski, A performance evaluation of 2D-mesh, ring, and crossbar interconnects for chip multi-processors, International Symposium on Microarchitecture Proceedings of the 2nd International Workshop on Network on Chip Architectures (2009) 51-56.

[28] P. T. Wolkotte, G.J.M. Smit, G.K. Rauwerda, L.T. Smit, An energy-efficient reconfigurable circuit-switched Network-on-Chip, IEEE International Parallel and Distributed Processing Symposium 19 (2005) 155a. 2005.

[29] S. Q. Zheng, M. Yang, Algorithm-Hardware codesign of fast parallel round-robin arbiters, IEEE Transactions on Parallel and Distributed Systems 18 (2007) 84-95.

[30] The Nangate Open Cell Library, 45nm FreePDK, available online at https://www.si2.org/openeda.si2.org/projects/nangatelib/.

[31] J. Rattner, Single-chip Cloud Computer: An experimental many-core processor from Intel Labs, available online at http://download.intel.com/pressroom/pdf/rockcreek/SCCAnnouncement.

[32] TILE-Gx Processors Family, available at http://www.tilera.com/products/TILE-Gx.php.