CHAPTER 6

EFFICIENT INTERCONNECTS FOR CLUSTERED MICROARCHITECTURES

In this chapter, we investigate the design of on-chip interconnection networks for clustered microarchitectures. This new class of interconnects have demands and characteristics different to traditional multiprocessor networks. In a clustered microarchitecture, a low inter-cluster communication latency is essential for high performance.

We propose some point-to-point interconnects and an improved instruction steering scheme, and show that they achieve much better performance than bus-based interconnects. The results show that the connectivity of the network together with effective steering schemes are key for high performance. We also show that these interconnects can be built with simple hardware and achieve a performance close to that of an idealized contention-free model.

6.1 Introduction

Previous work showed that the performance of a clustered superscalar architecture is highly sensitive to the latency of the inter-cluster communication network [16, 73]. Many steering heuristics have been studied to reduce the required communications [10, 16, 17, 70], and value prediction has been proposed to hide the communication latency [73]. The alternative approach proposed in this chapter consists of reducing the communication latency, by designing networks that reduce the contention delays and proposing effective improvements to the instruction steering scheme that minimize both the communication rate and the communication distance. Moreover, the proposed interconnects also reduce capacitance, thus speeding up signal propagation.

For a 2-cluster architecture it may be feasible to implement an efficient and contention-free cluster interconnect by directly connecting each functional unit output to a register file write port in the other cluster. However, as the number of clusters increases, the completely connected network may be very costly or unfeasible due to its complexity. On the other hand, a simple shared bus requires lower complexity but it has high contention. Therefore, a particular design needs to trade complexity for latency to find the optimal configuration.

Previous works on clustered microarchitectures have assumed interconnection networks that are either an idealized model ignoring complexity issues [10, 73], or they consider only 2 clusters (Multicluster [29], Alpha 21264 [53]), or they assume a simple but long-latency ring [3, 4, 52]. In this chapter, we explore several alternative interconnection networks with the goal of minimizing latency while keeping the cluster complexity low. To the best of our knowledge no other work has addressed this issue on dynamically scheduled processors. Sankaralingam et al. [86] analyzes several point to point interconnects for VLIW and Grid Processor architectures in a recent work, after our proposal [75]. We have studied two different technology scenarios: one with four 2-way issue clusters, the other with eight 2-way issue clusters. In both cases, we propose different point-to-point network topologies that can be implemented with low complexity and achieve performance close to those of idealized models without contention.

The rest of this chapter is organized as follows. In section 6.2 two new improvements to the steering scheme are proposed, section 6.3 discusses several design issues regarding the interconnection network, sections 6.4 and 6.5 describe the interconnect models proposed for four and eight clusters respectively, and section 6.6 analyzes the experimental results. Finally, section 6.7 summarizes the main conclusions of this chapter.

6.2 Improved Steering Schemes

The clustered architecture presented in this chapter assumes the best steering heuristic presented in section 4.2.7, i.e. the Priority RMB scheme with Accurate Rebalancing (AR-PRMB). As it was shown, the AR technique reduces communications during the periods when the workload imbalance exceeds the threshold, because it does not completely disable the communication criterion, but it just excludes the overloaded clusters from the choice of candidates. Since in this chapter we are going to explore architectures with up to 8 clusters, and communication latencies as high as 4 cycles, we expect that this technique proves even more effective than shown before. In more detail, this scheme works as follows:

1. To minimize communication penalties:
    1.1. If there is any unavailable operand, choose its producer cluster
    1.2. Else, select clusters with highest number of source registers mapped
2. Choose the least loaded among the above selected clusters

Except: If imbalance > threshold, exclude clusters with workload ≥ 0, prior to applying rules 1 and 2

6.2.1 A Topology-Aware Steering

For many of the interconnect topologies we study in this chapter, the latency of the communications depends on the distance between source and destination clusters. A topology-aware (TA) steering heuristic can take advantage of this knowledge to minimize the distance - and thus the latency - of the communications. Therefore, we have refined the primary criterion of the AR-PRMB algorithm to take the distance into account, in such a way that when all source operands are available (rule 1.2), it chooses the clusters that minimize the longest communication distance (the one that is in the critical path). To illustrate this feature, let us suppose that an instruction has two source operands, which are both available, and the left one is mapped to cluster 1, while the right one is mapped to clusters 2 and 3. In this case, the original primary criterion would select clusters 1, 2 and 3, since all of them have one operand mapped. Whatever is chosen, one copy would be needed, either between clusters 1 and 2 or between clusters 1 and 3. If we assume that cluster 1 is closer to cluster 2 than to cluster 3, then the topology-aware heuristic will consider only clusters 1 and 2.

6.3 The Interconnection Network

In this section we discuss several design trade-offs and constraints regarding the interconnection network, prior to describing in detail, in the following two sections, the schemes that have been experimentally analyzed for architectures with four and eight clusters.

6.3.1 Routing Algorithms

Interconnection networks have been widely studied in the literature for different computer areas such as multicomputers and network of workstations (NOWs) [23]. In these contexts, communication latencies may be thousands of processor cycles long, and routing decisions take several cycles. In contrast, for clustered microarchitectures performance is highly sensitive to the communication latency and just one cycle is a precious time, as shown by the results in section 6.6, and also by other previous works [16, 73]. Thus, in this context, networks must use simple routing schemes that carefully minimize communication latency (instead of maximizing throughput, like in other contexts). We assume that all routing decisions are locally taken at issue time (source routing), by choosing the shortest path to the destination cluster. If there is more than one minimal route, the issue logic chooses the first one that it finds available.

6.3.2 Register File Write Ports

Each cluster can inject copies to the network, which is connected to the cluster register files through a number of dedicated write ports where copies are delivered. From the point of view of the network design, including as many ports as required by its peak delivery bandwidth is the most straightforward alternative, but the number of write ports has a high impact on cluster complexity. First, each additional write port requires an additional result tag to be broadcast to the instruction issue queue, and the wakeup delay increases by a quadratic factor with respect

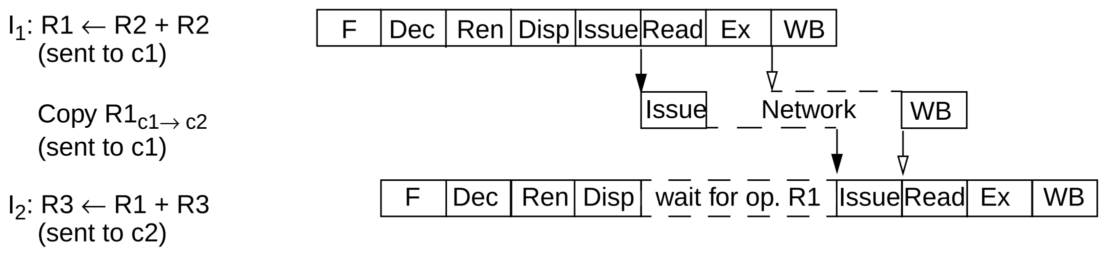


Figure 6-1: Sample timing of a communication between two dependent instructions I_1 and I_2, steered to clusters c1 and c2 respectively (solid arrows mean wakeup signals, hollow arrows mean data signals, and transmission time is 2 cycles in both cases).

to the number of broadcast tags [70]. Second, the register file access time increases linearly with the number of ports. Third, the register file area grows quadratically with the number of ports, which in turn makes the length and delay of the bypass wires to increase.

Moreover, previous studies (as well as our results in chapter 4) showed that, with adequate steering heuristics, the required average communication bandwidth is quite low (0.20 communications per instruction for 4 clusters [73]), and thus it is unlikely that having more than one write port per cluster connected to the network can significantly improve performance. Therefore, for all the analyzed networks we assume that they are connected to a single write port per cluster, except for the idealized models.

6.3.3 Communication Timing

In our distributed register file architecture, the access to remote operands is done exclusively through copy instructions, which are inserted into the instruction queues as normal instructions (see more details in chapter 2). A copy is issued when its source register is ready and it secures a slot of the network. Then, it reads the operand either from the register file or from the bypass, sends the value through the interconnection network, and delivers it to the consumer's cluster bypass network and register file.

The copy also sends through the network, along with the value, the tag of the destination physical register, in order to wake-up the dependent instructions. We assumed for simplicity that the tag forwarding delay is the same as the data forwarding delay. Consequently, the tag forwarding stays in the critical path of execution of the dependent instruction, which also includes issuing the copy instruction (see figure 6-1). Therefore, the total minimum issue distance between the producer and the consumer instructions equals the communication latency plus one cycle. However, a particular VLSI implementation could attempt to reduce this issue distance by optimizing the tag forwarding paths, which would leave it equal to the data communication latency.

6.3.4 Transmission Time

The total latency of a communication has two main components: the contention delays caused by a limited bandwidth, and the transmission time caused by wire delays. For a given network design, the first component varies subject to unpredictable hazards, and we evaluate it through simulation. On the other hand, the second component is a fixed parameter that depends on the propagation speed and length of the interconnection wires, which are low-level circuit design parameters bound to each specific circuit technology and design. To help narrowing this complex design space, we have taken two reasonable assumptions for point-to-point networks.

First, the minimum inter-cluster communication latency is one cycle. This clock cycle includes wire delay and switch logic delay. Note that, with current technology, most of the communication latency is wire delay. Second, only neighbor clusters (those at one-cycle distance) are directly connected with a pair of links, one in each direction. As a consequence, the communication between two non-neighbor clusters takes as many cycles as the number of links it crosses.

With these two assumptions, the space defined by different propagation speeds and wire lengths is discretized and reduces to the one defined by a single variable: the number of clusters that are at one-cycle distance from a given cluster (which is an upper bound of the connectivity degree of the network). Our analysis covers a small range of this design space by considering the connectivity degrees of several typical regular topologies.

Consistent with these long wire delays, the centralized L1 data cache is assumed to have a 3-cycle pipelined hit latency (address to cache, cache access and data back).

6.3.5 Router Structures

We assume a very simple router attached to each cluster for point-to-point interconnects. The router enables communication pipelining by implementing stage registers (buffers) in each output link (R_{right}, R_{left} and R_{up}, in figure 6-2). To reduce the complexity, the router does not include any other buffering storage for in-transit messages, but it rather guarantees that after receiving an in-transit message, it will be forwarded in the next cycle. This requirement is fulfilled by giving priority to in-transit messages over newly injected ones, and by structurally preventing that two in-transit messages compete for the same output link. Such a competence between in-transit messages never occurs on nodes with two neighbors, like those in a ring (figures 6-2a and 6-2b), but may happen in a mesh or a torus, where each node may have up to 4 neighbors (note, however, that since we have considered only small meshes with 4 and 8 clusters, each node has never more than 3 neighbors). For nodes with three neighbors, the router constrains the connectivity of in-transit messages by connecting every stage register (output link) to a single input link (see figure 6-2c), in the following way: in-transit messages can only traverse the router from the left to the right link, from the right to the left link or from the right to the upper link. Note that messages arriving from the upper link have no other connection than the input queue, thus this link is only available for messages doing their last hop.

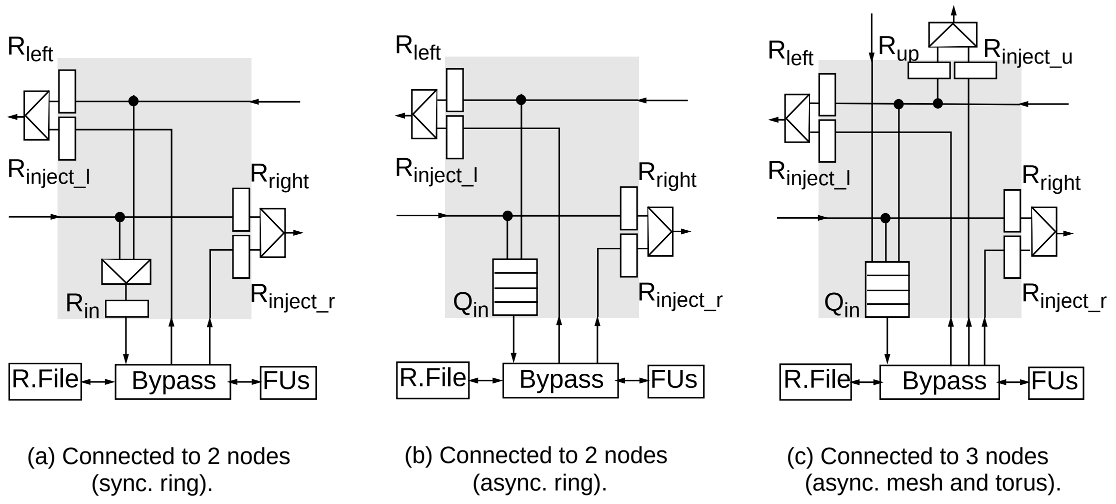


Figure 6-2: Router schemes for synchronous and asynchronous point-to-point interconnects

A *copy* instruction is kept in the issue queue until both its source operand is available and it secures the required injection register (R_{inject}, in figure 6-2), so no other buffering storage is required. That is, the scheduler handles the router injection registers as any other resource. However, while access requests for a bus-based network are sent to a distant centralized arbiter, the arbitration of each link in a point-to-point network is done locally at the source cluster, by simply choosing between one injection register and one stage register (priority is given to the latter one, as mentioned above). Eventually, the *copy* is issued and the outgoing message stays in one of the R_{inject} output registers while it is being transmitted.

The router also interfaces with the cluster datapath. For partially asynchronous networks, the router includes an input FIFO buffer (Q_{in}, in figures 6-2b and 6-2c) where all incoming messages are queued. Each cycle, only the message at the queue head is delivered to the cluster datapath, the others stay in the queue. For synchronous networks, the router is still less complex. By appropriately scheduling the injection of messages at the source cluster (more details are given later), the proposed scheme guarantees that a given router does not receive more than one input message per cycle. Therefore, the router requires just a single register (R_{in}, in figure 6-2a), instead of the FIFO buffer.

6.3.6 Bus versus Point-to-Point Interconnects

Although our analysis mainly focuses on point-to-point networks, we also study a bus interconnect, for comparison purposes. It is made up of as many buses as clusters, each bus being connected to a write port in one cluster, and each cluster being able to send data to any bus (figure 6-3a). Although this is a conceptually simple model, it has several drawbacks that make it little scalable. First, since buses are shared among all clusters, their access must be arbitrated, which makes the communication latency longer, although bandwidth is not affected as long as arbitration and transmission use different physical wires. Second, a large portion of

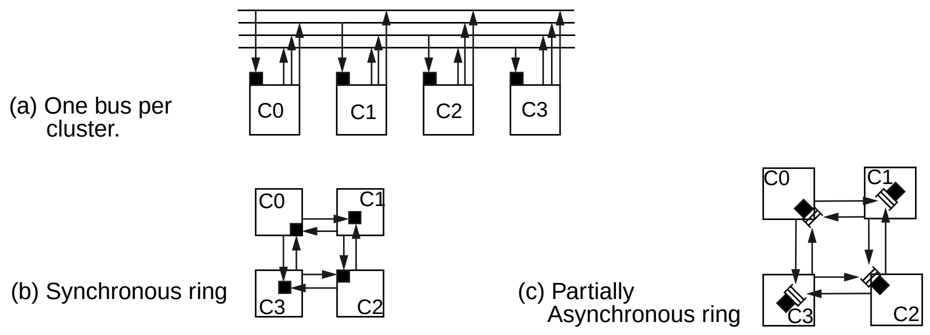


Figure 6-3: Four-cluster topologies

the total available bandwidth, which is proportional to the number of clusters, is wasted due to the low bandwidth requirements of the system. However, if the number of buses was reduced, then the number of conflicts would increase, and hence the communication latency. Third, each bus must reach all clusters, which implies long wires and long transmission times, which can drastically reduce the bandwidth if the bus transmission time is not pipelined[1].

Compared to the above bus interconnect, a point-to-point interconnect (a ring, a mesh, a torus, etc.) has the following advantages. First, the access to a link can be arbitrated locally at each cluster. Second, communications can be more easily and effectively pipelined. Third, delays are shorter, due to shorter wires and smaller parasitic capacitance (there are less devices attached to a point-to-point link than to a bus). Fourth, network cost is lower than a configuration with as many buses as clusters. Finally, it is more scalable: when the number of clusters increases, its cost, bandwidth and ease of routing scales better than for the bus-based configuration.

6.4 Four-Cluster Network Topologies

For four clusters, we propose two alternative point-to-point networks based on a ring topology, and compare them to a realistic bus-based network (see figure 6-3). We also compare their performance to that of an idealized ring, which represents an upper bound for ring networks. Below we describe these topologies.

6.4.1 Bus2

This is a realistic bus interconnect with a 2-cycle transmission time (hence its name). It has as many buses as clusters, each one connected to a single write port (see figure 6-3a), and a very simple centralized bus arbiter. The total communication latency is 4 cycles because bus

1. Note that it is difficult to pipeline bus communications, but it is easy to pipeline communication through point-to-point links (although the latter case is not needed with current VLSI technology, so we assume a transmission time of 1 cycle per hop), it clearly indicates that point-to-point links are much more scalable than buses.

arbitration, including the propagation of the request and grant signals, takes 2 additional cycles. We assume that the arbitration time may overlap with the transmission time of a previously arbitrated communication, so each single bus bandwidth is 0.5 communications per cycle.

6.4.2 Synchronous Ring

This interconnect is one of the contributions of this work, since previously proposed rings work in asynchronous mode. This topology assumes no queues in the routers, neither to store in-transit messages nor to store messages arriving at their destination clusters.

Since no queues are included at the destination clusters, when a message arrives it must be immediately written into the register file. The router arbitration logic injects copy instructions with an algorithm (summarized in table 6-1) that ensures that no more than one message arrives at a time at a given node. During odd cycles, a source cluster *src* is allowed to send a short-distance message (D=1) to its adjacent cluster in the clockwise direction (*(src +1) mod 4*), and a long-distance message (D=2) in the counter-clockwise direction (*(src + 2) mod 4*). During an even cycle, the allowed directions are reversed. Since in-transit messages are given priority over

Table 6-1: Rules to secure a link in the source cluster *src* (D refers to distance in cycles)

Direction	Odd Cycle		Even Cycle	
	D	Target Cluster	D	Target Cluster
Clockwise	1	*src* → *(src+1) mod 4*	2	*src* → *(src+2) mod 4*
Counter-clockwise	2	*src* → *(src+2) mod 4*	1	*src* → *(src+3) mod 4*

newly injected ones (see section 6.3.5), an issued copy instruction may have to wait in the injection register until the cycle parity is appropriate.

Despite the fact that there are cyclic dependencies between links [23], deadlocks are avoided by synchronously transmitting messages through all the links in the ring, even if the stage buffer at the next router is busy (it will be free when the message arrives). This is possible thanks to using the same clock signal for all the routers and giving a higher priority to in-transit messages.

6.4.3 Partially Asynchronous Ring

Typical asynchronous networks include buffers both in the intermediate routers, to store in-transit messages, and in the destination routers, to store messages that are waiting for a write port (in our case to the register file) [23]. The former are removed in our design, like in the synchronous ring. However, we still need the latter, since two messages can arrive at the same time to the same destination cluster and there is only one write port in each cluster. In this case, the message whose data cannot be written is delayed until it has a port available. Note that the system must implement an end-to-end flow control mechanism in order not to lose messages when a queue is full. This is an additional cost of the asynchronous schemes, which is discussed in more detail in section 6.6.3. In this network, routers use the same clock signal. Therefore, it is only partially asynchronous. A fully asynchronous network has not been considered because its cost would be much higher (larger buffers, link-level flow control, extra buffers to avoid deadlocks, etc.). Deadlocks are avoided as in the synchronous ring.

6.4.4 Ideal Ring

For comparison purposes, we consider an idealized ring whose inter-cluster distances are the same as those of the realistic ring (as discussed previously), but an unlimited bandwidth is assumed, which makes it contention-free (i.e., it has an unlimited number of links between each pair of nodes and an unbounded number of register file write ports for incoming messages in each cluster). The performance of this ideal ring lets us estimate how much performance is lost due to interconnect bandwidth constraints.

6.4.5 Ideal Crossbar

We consider also an idealized crossbar with unlimited bandwidth and 1-cycle distance between any pair of clusters. Its performance is an upper bound for all other models, and it lets us estimate how much performance is lost due to constraining the connectivity degree - and hence the complexity - to 2 adjacent nodes per cluster.

6.5 Eight-Cluster Network Topologies

For eight-cluster architectures, we first consider two ring-based interconnects, synchronous and partially asynchronous, similar to those proposed for 4-cluster architectures, and also two versions of a realistic bus-based network, having transmission times of 2 and 4 cycles, respectively.

In addition to the ring, we also analyze mesh and torus topologies, both of them partially asynchronous, since they feature lower average communication distances. Figure 6-4 shows these two new schemes. Below, we describe each scheme in detail.

6.5.1 Bus2 and Bus4

The bus required to connect 8 clusters is likely to be slower than that required by the 4-cluster configuration due to longer wires and higher capacitance. To account for this, we consider two bus-based configurations: the Bus2, which optimistically assumes the same latencies as those of the 4-cluster configuration (i.e., a transmission time of 2 cycles), and the Bus4, which more realistically assumes twice this latency (i.e., a transmission time of 4 cycles).

6.5.2 Synchronous and Partially Asynchronous Rings

For this interconnect, the scheduling algorithm of the synchronous ring discussed in section 6.4.2 for 4 clusters is extrapolated to 8 clusters. Like in the 4-cluster configuration, at issue time the scheduler of copy instructions (shown in table 6-2) must ensure that only one message arrives at a time at a given cluster, because there is only one write port to the register file. Note that distances in an 8-cluster ring topology range from 1 (D=1) to 4 (D=4) cycles. The partially asynchronous ring model is identical to that described for four clusters in section 6.4.3.

Table 6-2: Rules to secure a link in the source cluster *src* (D refers to distance in cycles)

Direction	Odd Cycle		Even Cycle	
	D	Target Cluster	D	Target Cluster
Clockwise	1	*src → (src+1) mod 8*	2	*src → (src+2) mod 8*
	3	*src → (src+3) mod 8*	4	*src → (src+4) mod 8*
Counter-clockwise	4	*src → (src+4) mod 8*	3	*src → (src+5) mod 8*
	2	*src → (src+6) mod 8*	1	*src → (src+7) mod 8*

6.5.3 Mesh

A mesh topology (see figure 6-4a) reduces some distances with respect to a ring. The average distance in a ring is 2.29 hops, while in a mesh it is 2 hops; however, the maximum distance is still 4 hops. The dashed lines in the figure show the links added to the ring topology to convert it into a mesh.

Due to the increased connectivity, this topology introduces a new problem to the design of the routers with respect to a ring, because at central nodes (labelled C2, C3, C6 and C7) more than one in-transit message at the router input links could compete to access the same output link. As discussed in section 6.3.5, our approach is to constrain the connectivity of in-transit messages within the router. On the one hand, an *upper* router input (refer to figure 6-2c) is only connected to the input queue, so the routing algorithm must ensure that this link is used only for the last hop of a transmission. The four links between C2-C3 and C6-C7 in our mesh (shown with dashed arrows in figure 6-4a) are connected to *upper* router inputs and outputs. Thus, if one message is sent, for instance, from cluster C2 to C7, it must be routed through C6 because the link C2-C3 is not associated to the link C3-C7. On the other hand, in-transit messages arriving to a *right* router input have no constraints, so they may be routed either to the *left* or to the *upper* output. The four links between C3-C7 and C2-C6 in our mesh (figure 6-4a) are connected to *right* router inputs and outputs. Finally, the rest of links connected to the four central nodes of the mesh are connected to *left* router inputs. In-transit messages arriving to a *left* router input can only be routed to the *right* output. Thus, for instance, a message from C0 to C3 must be routed through C1 because the link C0-C2 is not associated to the link C2-C3.

Again, deadlocks are avoided by using the same clock signal for all the routers and transmitting messages synchronously.

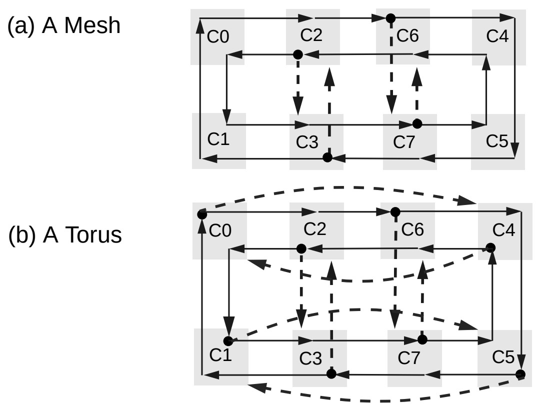


Figure 6-4: Additional topologies for 8 clusters. A black dot at the end of a link means that there is more than one link that can be followed for the next hop if the corresponding node is not the destination. Messages can always be routed through solid links but dashed links are only used for their last hop.

6.5.4 Torus

A torus has smaller average distance than a mesh (see figure 6-4b). In all nodes, more than one in-transit message at the router input links could compete to access the same output link. Like for the mesh, this problem is solved without including intermediate buffers, by constraining the connectivity of several links. The solution is outlined in the figure, where dashed arcs indicate links with a limited connectivity (see also router details in figure 6-2c).

Note that this constraint does not change the minimal distance between every pair of nodes, but for some pairs it does reduce the number of alternative routes. For example, there is only one 2-hop route from C4 to C1. However, due to the poor utilization of the network (as we will show later) this is a minor drawback.

Again, deadlocks are avoided as indicated above. On another issue, when mapping torus links on silicon, some links may be longer than the rest (e.g., links between C0 and C4). This may introduce delays in those particular links. For the sake of simplicity, we did not consider that additional delay.

6.5.5 Ideal Torus

For comparison purposes, we also consider an idealized torus model, with distances identical to those of the realistic torus but with unlimited bandwidth, which makes the network to be contention-free. In other words, it is assumed that the network has an unlimited number of links between any pair of adjacent nodes and an unbounded number of register file write ports connected to the network in each cluster. The performance of this model is an upper bound on the performance of the realistic torus.

6.5.6 Ideal Crossbar

We consider also an idealized crossbar with unlimited bandwidth and 1-cycle distance between any pair of clusters. Its performance is an upper bound for all other models, and it lets us estimate how much performance is lost due to constraining the connectivity degree.

6.6 Experimental Evaluation

In this section the different network architectures proposed above are evaluated. For these experiments, we have used the Mediabench benchmark suite [56, 62] and assume all the experimental setup described in section 2.3, including the simulator and the architectural parameters in table 2-1. The eight-clusters architecture assumed an identical cluster model as those of the four-clusters architecture, which means doubling the total effective issue width of the processor. Accordingly, the eight-cluster architecture also assumes twice the fetch/decode bandwidth, number of data cache ports and number of entries in the reorder buffer and in the load/store queue.

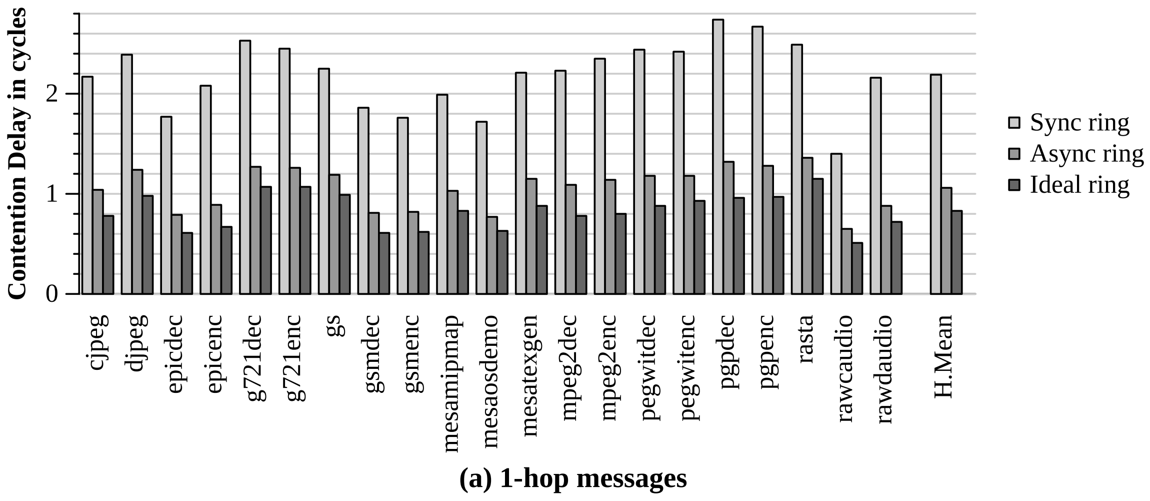
6.6.1 Network Latency Analysis

To gain some insight on the different behavior of synchronous and partially asynchronous rings for a 4-cluster architecture, we analyze their average communication latency. In particular, since the transmission time component of the latency is the same for both interconnects, we only analyze the contention delay component.

Figure 6-5 compares the communications contention delay for each of the two ring interconnects, and it also includes an ideal ring for comparison. For each of the former two interconnects, the contention delay has two components: it may be caused by an insufficient issue width or an insufficient interconnect bandwidth. In contrast, the contention delay of the ideal ring is exclusively due to the limited issue width, and it is on average 0.8 cycles. Therefore, comparing the delays of the first two interconnects to that of the ideal-ring, the difference gives an estimation of the contention caused by the insufficient interconnect bandwidth.

As shown in figure 6-5, short-distance messages (graph a) wait for longer than long-distance ones (graph b). The main reason is the available bandwidth for each type of message: the latter have two alternative minimal-distance routes, while the former have only one. However, since the routing algorithm is the same for both ring interconnects, it does not explain the differences observed between the two rings.

Figure 6-5a shows that the contention delay of short-distance messages for a synchronous ring (2.19 cycles) is two times longer than for a partially asynchronous one (1.06 cycles). In contrast, the contention caused to long-distance messages for a synchronous ring (0.71 cycles) is just slightly lower than for a partially asynchronous one (0.87 cycles). These differences are



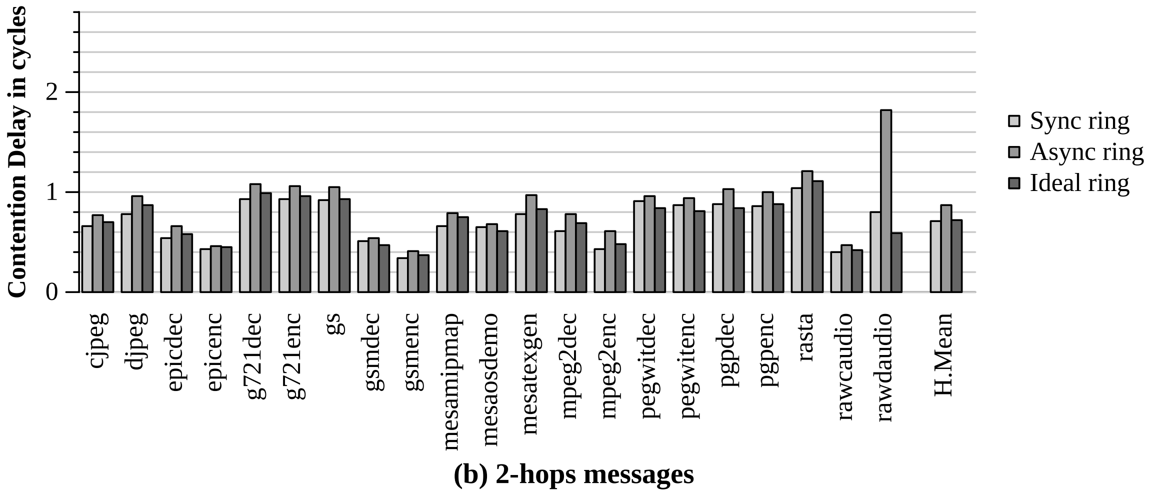


Figure 6-5: Average contention delays of 1-hop and 2-hops messages, with synchronous, partially asynchronous and ideal ring interconnects

due to the different ways each interconnect avoids conflicts between messages that require access to the same register file write port: in a synchronous ring, these conflicts are prevented by ensuring that a short-distance message is not issued if the parity of the cycle is not the appropriate one (see table 6-1), regardless of whether the link is busy or not. For example, a long-distance message from C1 to C3 (see figure 6-3b) will be injected in an even cycle, thus reaching the router at C2 and requesting the C2-C3 link during the next odd cycle. As messages from C2 to C3 must be injected during odd cycles, these short-distance messages will be delayed if there are in-transit long-distance messages. In a partially asynchronous ring, a message of any kind can be issued as soon as the required output link is available, although it may have to wait in the destination cluster router until it gains access to the register file write port. For an asynchronous ring, the network contention causes a delay between 0.15 and 0.23 cycles, while the rest of the contention delay of the communications is due to the issue width.

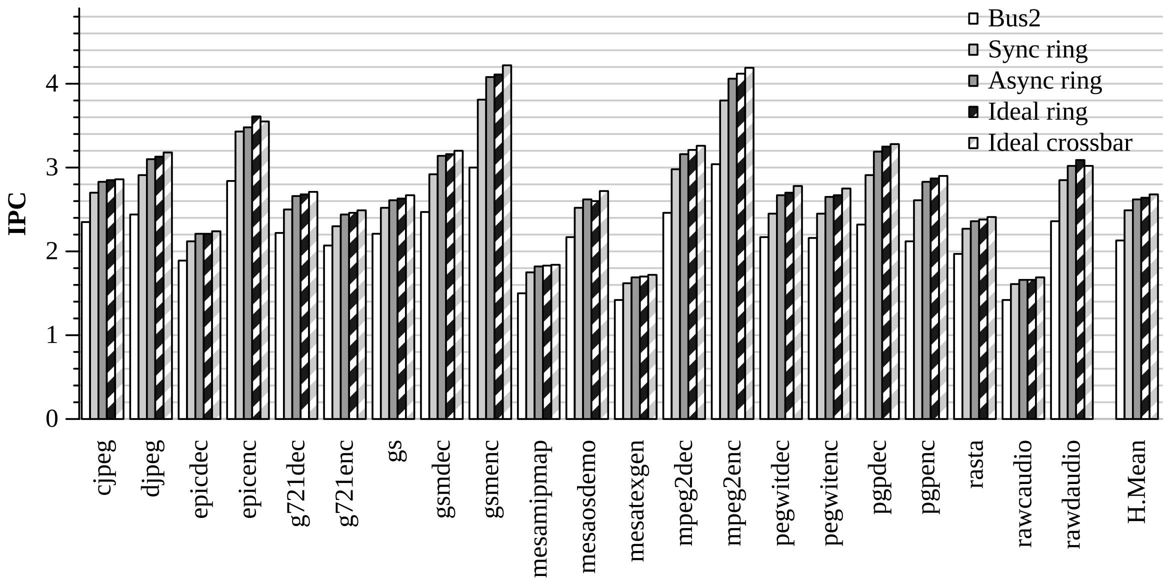


Figure 6-6: Comparing the IPC of 4-cluster interconnects

To summarize, the long delays caused to short-distance messages by the scheduling constraints of the synchronous ring make its overall contention delay be higher than for a partially asynchronous one. In addition, since there are about twice as many short-distance messages than long-distance ones, they have a high impact on the overall contention delay. As a consequence, the partially asynchronous ring performs better than the synchronous one, as it is shown below.

6.6.2 Performance of Four-Cluster Interconnects

Figure 6-6 compares the performance, reported as number of committed instructions per cycle (IPC), of a four-cluster architecture for all the proposed interconnects.

The two ring interconnects consistently achieve better performance than the bus topology, for all benchmarks. This is mainly because short-distance messages have a shorter transmission time in point-to-point interconnects, and because the steering heuristic exploits it to keep close instructions that have to communicate. Besides, the ring topology offers a higher bandwidth, although in this scenario bandwidth is not critical for performance due to the low traffic generated by the steering scheme [73] (e.g. it is on average 0.20 communications per instruction, with a partially asynchronous ring).

The IPC achieved by the synchronous ring is, on average, 16.8% higher than that achieved by the bus, while the partially asynchronous ring performance is 23.2% higher than that of the bus, because the contention delays of short-distance messages are lower for the asynchronous ring, as discussed in section 6.6.1.

The performance of the partially asynchronous ring is very close to that of the ideal ring (less than 1% difference), which shows that increasing the number of links or the number of register file write ports would hardly improve performance. In other words, due to the effectiveness of the steering logic to keep the traffic low, a simple configuration with two links between adjacent clusters (one in each direction) and a single register file write port for incoming messages is clearly the most cost-effective design.

Finally, we found that a ring performs very close to a complex fully connected crossbar. The performance lost by reducing the connectivity degree from 3 to 2 adjacent nodes per cluster (corresponding to the ideal crossbar and the ideal ring, respectively) is on average just 1.3%. In other words, a ring interconnect is a cost-effective topology for a four cluster interconnect, since having a higher connectivity - hence complexity - returns very small performance gains.

6.6.3 Queue Length

Typical partially asynchronous rings need specific mechanisms to prevent (or to recover from) potential overflows of the network buffers. In our partially asynchronous interconnect, this problem occurs only in the queues for incoming messages at each cluster.

In order to adequately dimension these queues, we first assumed unbounded size queues and measured the number of occupied entries each time a new message arrives at its destination cluster. Note that with FIFO queues and a single write port, this number is equal to the number of cycles a message stays in the queue. We found that for any benchmark, more than 85% of the messages do not have to wait because they find the queue empty (92.1%, on average), and the maximum observed number of occupied entries was 11. For instance, table 6-3 shows a typical queue length distribution (for benchmark *djpeg*).

Table 6-3: Queue length distribution (for *djpeg*)

# occupied entries	# messages	Distribution (% times)	Cumulative Distribution (%)
0	1263899	90.25	90.25
1	122802	8.77	99.02
2	12078	0.86	99.88
3	1524	0.11	99.99
4	105	0.01	100.00
5	18	0.00	100.00
6	4	0.00	100.00
>= 6	0	0.00	100.00

Although 11-entry queues are long enough in our experiments, the model should ensure that data is never lost, in order to guarantee execution correctness. Two approaches are possible: first, to implement a flow control protocol that prevents FIFO queue overflows; and second, to implement a recovery mechanism for these events. Flow control can be based on credits. In this case, each cluster would contain a credit counter for each destination cluster. Every time a message is transmitted to a cluster, the corresponding credit counter would be decreased. If the counter is equal to zero, the message would not be transmitted because the FIFO queue may be full. When a message is removed from the queue, a credit is returned to the sender of that message, thus, consuming link bandwidth. Upon reception of the credit, the corresponding

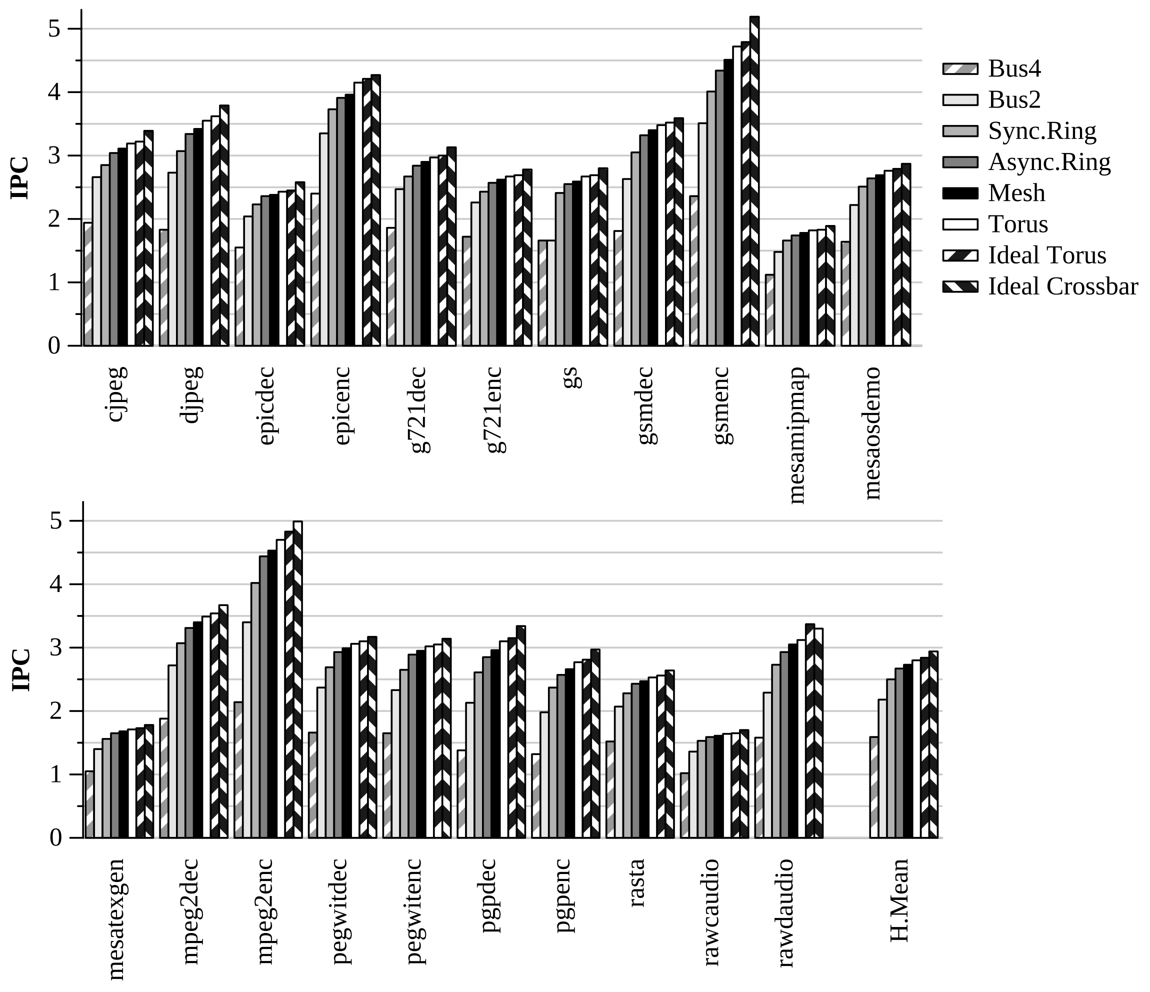


Figure 6-7: Comparing the IPC of 8-cluster interconnects

credit counter is increased. However, since overflows are so infrequent, the most cost-effective solution in case of an overflow is to squash the instruction that generated the message that caused the overflow, as well as all other younger instructions, very much like in the case of exceptions or branch mispredictions, and to restart again execution at this instruction. This approach requires minimal additional hardware and it produces negligible performance penalties (for 11-entry queues there is no penalty at all for our benchmarks).

6.6.4 Performance of Eight-Cluster Interconnects

In this section, we evaluate the eight-cluster network interconnects described in section 6.5, for a 16-way issue architecture (as described at the beginning of section 6.6). Figure 6-7 shows the IPC for the different schemes. The point-to-point ring achieves a significant speed-up even over the optimistic bus architecture denoted as bus2. The average speed-up of the synchronous ring over bus2 is 14.6% whereas the partially asynchronous ring outperforms bus2 by 22.3%.

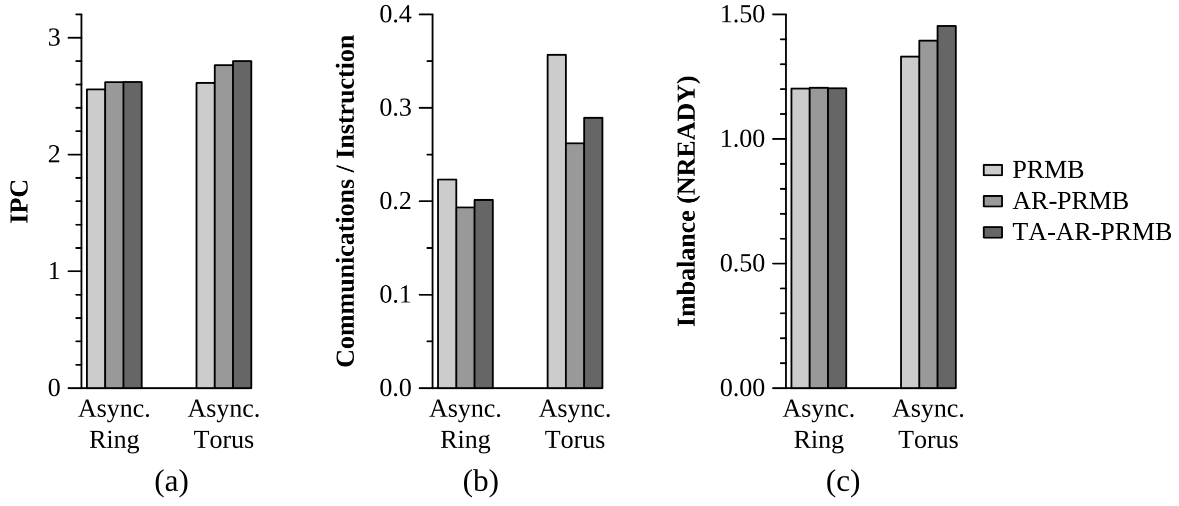


Figure 6-8: Effectiveness of the Accurate Rebalancing (AR) and Topology Aware (TA) techniques applied to the baseline PRMB steering scheme, for a four-cluster ring and an eight-cluster torus: (a) IPC, (b) Communications rate and (c) Workload imbalance

Comparing the partially asynchronous topologies, the mesh achieves an IPC 2.2% higher than that of the ring, while the IPC of the torus is 4.9% higher than that of the ring. On the other hand, the partially asynchronous torus performance is very close to that of the ideal torus configuration with unlimited bandwidth (just 1.4% difference). The performance of the ideal torus is just 3.4% below that of the ideal crossbar that has unlimited bandwidth and all nodes at a 1-cycle distance.

6.6.5 Effectiveness of the Accurate-Rebalancing and Topology-Aware Steering

In all the previous experiments it was assumed the PRMB steering scheme, including both the Accurate-Rebalancing (AR) and the Topology-Aware (TA) improvements described in section 6.2. In this section, the effectiveness of these two techniques are analyzed. They are evaluated for a four-cluster architecture with an asynchronous ring and for an eight-cluster architecture with a torus interconnect. Figure 6-8 compares the PRMB steering with and without these two techniques. It is shown the average IPC (a), the average communications rate (b), and the average workload imbalance (c).

The AR technique is aimed at reducing the communications generated during strong imbalance situations, when the PRMB steering is mainly concerned on rebalancing the workload. Instead of totally ignoring dependences, it just excludes the overloaded clusters. As shown in graph (a), AR improves the performance of PRMB by 2.4% and 5.8% with four and eight clusters respectively, because it significantly reduces the amount of communications. As shown in graph (b), AR reduces the communication rate by 13% and 27% with four and eight clusters respectively. Not surprisingly, the effectiveness of the AR technique grows with the number of clusters, because the likelihood of causing a communication when operand locality is ignored increases with the number of clusters.

The TA technique is aimed at minimizing communication distances - hence latencies - for point-to-point interconnects. As shown in figure 6-8a, it produces a small 1.2% IPC improvement over the AR-PRMB scheme for eight clusters and almost no effect for four clusters (the total improvement using both AR and TA techniques is 7.2% and 2.5% respectively). TA produces a small impact on performance because there are actually few instructions that offer the chance to reduce the communication distance, and because in some of these cases the distance is reduced at the expense of generating one extra communication.

With our AR-PRMB steering scheme, the opportunity to reduce the communication distance occurs only when an instruction has two register operands, and both are available, and they are mapped in two disjoint subsets of clusters. When this happens, at least one communication is required, and the TA technique chooses the cluster that minimizes the longest communication distance to the source operands instead of choosing one of the clusters with a source register mapped. We found that the TA reduces the average communication distance from 1.32 to 1.20 hops for four clusters, and from 1.70 to 1.37 hops for eight clusters. However, it often occurs that the TA chooses a cluster where none of the operands is mapped, which forces generating a second communication. We found that the TA increases the number of communications per instruction from 0.193 to 0.201 for four clusters, and from 0.262 to 0.289 for eight clusters, as shown in figure 6-8b. Therefore, the added communication overhead offsets the expected improvements of reducing communication latency.

6.6.6 Experiments with the SpecInt95

In previous sections, the Mediabench [56, 62] benchmark suite was used for all the experiments. For the sake of higher generality of our conclusions, we run an identical set of experiments with the SpecInt95 benchmark suite [97].

For 4 clusters, we found that the synchronous ring performs on average 12.6% better than the bus2, while the partially asynchronous ring outperforms bus2 by 14.9%. The performance of the partially asynchronous ring is very close to that of the ideal ring with unlimited bandwidth (1.0% difference), and it is just 1.7% below that of the ideal crossbar, with unlimited bandwidth and 1-cycle latency between any pair of nodes.

For 8 clusters, the average speed-up of the synchronous ring over bus2 is 9.2% whereas the partially asynchronous ring outperforms bus2 by 13.5%. Comparing the partially asynchronous interconnects, the mesh achieves an IPC 1.5% higher than that of the ring, while the IPC of the torus is 3.3% higher than that of the ring. On the other hand, the partially asynchronous torus performance is just 0.4% below that of the ideal torus configuration with unlimited bandwidth, and 4% below that of the ideal crossbar with unlimited bandwidth and 1-cycle latency between any pair of nodes.

Finally we found that the AR technique improves the performance of the PRMB steering scheme for an 8-cluster torus and a 4-cluster ring by 2.5% and 1.1% respectively. The TA technique produces a 1.4% speedup for 8 clusters and almost no impact for 4 clusters. The total performance improvement using both the AR and TA techniques for an 8-cluster torus is 4%.

Compared to the results with the Mediabench suite shown in previous sections, these results show similar trends, although in some cases the differences among the various configurations may vary. However, the same overall conclusions hold for both benchmark suites.

6.7 Summary and Conclusions of this Chapter

In this chapter we have investigated the design of on-chip interconnection networks for clustered microarchitectures. This new class of interconnects have demands and characteristics different to traditional multiprocessor networks, since a low communication latency is essential for high performance. We have shown that simple point-to-point interconnects together with effective steering schemes achieve much better performance than bus-based interconnects. Besides, the former do not require a centralized arbitration to access the transmission medium.

In particular, we have proposed a very simple synchronous ring interconnect that only requires five registers and three multiplexers per cluster and substantially improves the performance of a bus-based scheme.

We have also shown that a partially asynchronous ring performs better than the synchronous one at the expense of some additional cost/complexity due to the additional queue required per cluster. However, we have found that a tiny queue will practically never overflow. Thus, instead of using complex flow control protocols, it is much more cost-effective to handle overflows by flushing the processor pipeline, which is a mechanism that current microprocessors already implement for other purposes (e.g., branch misprediction).

We have explored other synchronous and partially asynchronous interconnects such as a mesh and a torus, in addition to rings. These three topologies basically differ in their connectivity degree, and consequently, in the average inter-cluster distances. From our study we extract two main conclusions. First, the interconnects with higher connectivity perform better because they have shorter communication latency. However, point-to-point partially asynchronous interconnects with moderate connectivity/complexity perform close to an idealized crossbar with unlimited bandwidth and all nodes at one-cycle distance: a four-cluster ring performs within 2% of the ideal, and an eight-cluster torus performs within 4% of the ideal. Second, despite the low hardware requirements of partially asynchronous interconnects, they achieve a performance close (within 1%) to an equivalent idealized interconnect with unlimited bandwidth and number of write ports to the register files.

To conclude, the choice of an effective interconnection network architecture together with an efficient steering scheme is a key to high performance in clustered microarchitectures. Simple implementations of point-to-point interconnects such as those proposed in this chapter are quite effective and scalable.