Chapter Four

Decomposed Arbiters for Large Crossbars

Crossbars are key components of communication switches used to construct multiprocessor interconnection networks. Small networks can be implemented as a single crossbar while large networks are composed of many small crossbars [Dias81a, Dias81b, Kuma84, Dall86, Bhuy87, Lang82a]. Theoretically, throughput is maximized and latency is minimized if the entire network is one large crossbar. In general, given a choice between a small number of large crossbars or a large number of small crossbars, larger crossbar switches help lower the probability of conflicts and allow packets to traverse the network in fewer hops. With advances in VLSI fabrication and packaging technology, larger crossbars are becoming practical (e.g. the Inmos C104 packet-routing switch has 32 inputs and 32 outputs). A key factor in determining the performance of crossbars for communication is the mechanism used to arbitrate conflicting requests.

Several packets arriving at different input ports of the switch simultaneously may be destined to the same output port. However, only one packet at a time may be forwarded through each output port. Due to the resulting contention for output ports, packets may have to be buffered at the inputs of the crossbar while awaiting service. Recent communication switch designs utilize *multi-queue* buffers, which maximize performance by allowing packets at an input port to be processed in *non-FIFO* order [McMi80, Kuma84, Tami88a, Dall90]. Packets at each input port, which are destined to different output ports, may be forwarded through the switch in any order. Hence, each input port contends for *multiple* output ports but needs only one for full utilization. Similarly, each output port contends for multiple input ports and needs one for full utilization. The arbitration task is thus *symmetrical* with respect to inputs and outputs. Since the arbitration result for each port is dependent on the arbitration for other ports, the crossbar arbitration for switches with multi-queue buffers is more complicated than for switches with FIFO buffers. We have described in Chapter 3 high-speed *symmetric crossbar arbiters*, which efficiently solve the arbitration problem for multi-queue buffers.

One of the major challenges with large crossbar design is the long arbitration delay. Since the arbitration delay grows with the crossbar size, the arbitration speed of a large crossbar can become a performance bottleneck. The design of symmetric arbiters for large crossbars is the topic of this chapter. It will be shown that high performance can be achieved by decomposing the arbitration process so that the arbitration of a large crossbar is performed using an array of smaller arbiters. Furthermore, in a large crossbar it is not necessary for the number of queues at each input port to be equal to the number of output ports. A small number of queues reduces the cost of buffer management and is sufficient for achieving high performance.

The next section discusses the possible impact of increasing the crossbar size on the network performance. Section 4.2 describes how to decompose the arbitration for large crossbars in order to improve the network performance. The analysis results for the network latency in an unloaded network are presented. In Section 4.3, simulations are used to evaluate different arbitration schemes for large crossbars in loaded networks. Section 4.4 describes the logic and circuit design of the proposed decomposed arbiter. The performance impact of varying the number of queues per input buffer is discussed in Section 4.5.

4.1. Performance Impact of Large Crossbars

For interconnecting a fixed number of nodes, larger crossbars lower the probability of conflicts and reduce the number of hops for a packet to traverse the network. Hence, theoretically, larger crossbars improve the performance of the interconnection network. However, the crossbar arbitration delay increases as the crossbar size increases. Thus, if large crossbars are used, arbitration delays can become a critical performance bottleneck.

To study the speed of crossbar arbitration, the CMOS implementation of 2×2, 4×4, and 8×8 crossbars and their wave front arbiters and wrapped wave front arbiters have been laid out using the MOSIS scalable design rules. With 2μ technology, circuit simulations indicate that the worst case delays for reaching an arbitration result for 2×2, 4×4, and 8×8 wave front arbiters are 7.2 ns, 15.5 ns, and 32 ns respectively. For the wrapped wave front arbiters, the worse case delays are 4.5 ns, 10 ns, and 20 ns respectively. In general, for the WFA and WWFA arbiters, the arbitration delay grows linearly with the crossbar size.

If the arbitration must complete in one clock cycle, long arbitration delays for large crossbars may force the use of a slower clock. In order to overcome this problem, the arbiter may be allowed several clock cycles to complete the arbitration. In this scheme, new requests are blocked during the arbitration. In the following, the network latency is analyzed for this multi-cycle arbitration when the network is unloaded. We discuss how the arbitration delay can affect the network performance. This multi-cycle arbitration scheme is called the *nondecomposed* *arbitration* (as opposed to the *decomposed arbitration* to be presented in Section 4.2). For comparison, an unrealistic ideal arbitration scheme which can arbitrate a crossbar of arbitrary size in one cycle is also analyzed.

We analyze the performance of an arbitration scheme in terms of *base network latency*, which is the latency for a packet traversing an unloaded (or empty) network [Agar91]. The latency for a packet going through an unloaded crossbar switch is called *base switch latency*. In the analysis, the base switch latency consists of two components: the arbitration-related delay and the non-arbitration-related delay. The arbitration-related delay includes the wait a packet has to stay in the buffer before entering the arbitration, and the arbitration time. The non-arbitration-related delay includes the time for receiving the packet, making routing decision, and transmitting the packet. Typically, the non-arbitration-related delay is approximately independent of the crossbar size, and thus a constant will be assumed in our analysis.

Consider a $2^w \times 2^w$ multi-stage interconnection network (e.g. an Omega network) composed of $2^s \times 2^s$ crossbar switches. Suppose that one cycle is needed to arbitrate a $2^u \times 2^u$ array of crosspoints for the nondecomposed scheme. This array of crosspoints is called the *1-cycle subarray*. Since the switch arbitration delay is proportional to the switch size, the switch arbitration delay is $d = 2^s / 2^u = 2^{s-u}$. When a packet arrives in an unloaded crossbar switch, in the best case the wait for the packet to enter the arbitration is 0 cycle (the d-cycle arbitration is going to start). In the worst case, the packet has to wait $d-1$ cycles in order to enter the arbitration (the d-cycle arbitration started a cycle ago). On the average, a packet waits $\frac{1}{2}(d-1)$ cycles. Assume the non-arbitration-related delay of the switch latency is k cycles. The base switch latency is thus $k + ½(d-1) + d$. Since it takes w/s hops for a packet to traverse the network, the base network latency is $[k + ½(d-1) + d] \cdot (w/s)$, namely, $[k + ½(3 \cdot 2^{s-u} - 1)] \cdot (w/s)$.

For the ideal scheme, the arbitration takes one cycle regardless of the crossbar size. Hence, the base switch latency is $k+1$, and the base network latency is $(k+1) \cdot (w/s)$.

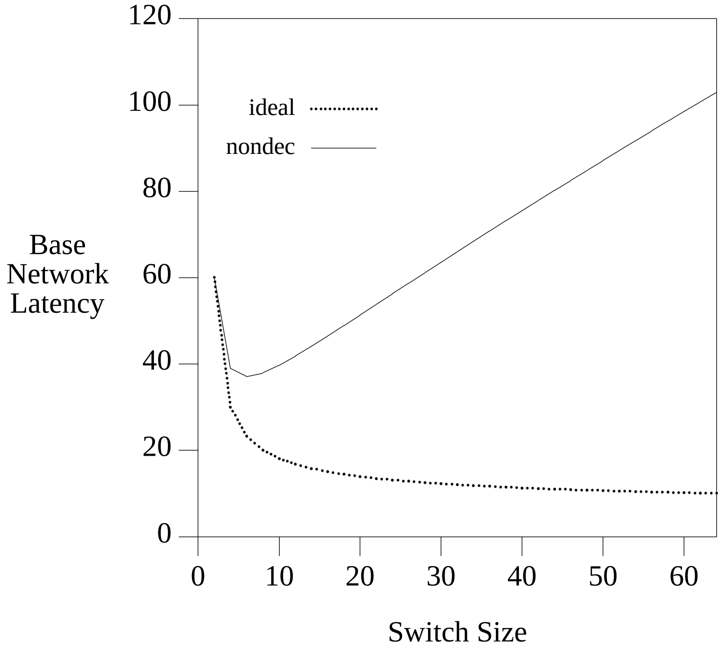

Figure 4.1: Base network latency vs. switch size of a 4096×4096 network for the ideal arbitration and the nondecomposed arbitration. 1-cycle subarray size 2×2. The non-arbitration-related component of the switch latency is 4 cycles.

The results of the above analysis of the base network latency for the ideal arbitration scheme and the nondecomposed arbitration scheme are shown in Figure 4.1 and Table 4.1. The network size is 4096×4096. It is assumed that the 1-cycle subarray size is 2×2. The non-arbitration-related delay of the switch

Switch Size	Base Network Latency (4096×4096)	
	ideal	nondec
2	60.0	60.0
4	30.0	39.0
8	20.0	38.0
16	15.0	46.5
64	10.0	103.0
4096	5.0	3075.5

Table 4.1: Analysis of the base network latency of a 4096×4096 network for the ideal arbitration and the nondecomposed arbitration. 1-cycle subarray size 2×2. The non-arbitration-related component of the switch latency is 4 cycles.

latency is 4 cycles [Tami88a, Fraz89]. Since in practice the switch size is usually a power of 2 for the efficiency of addressing and routing in the network, only these switch sizes are listed in Table 4.1. The figure and the table show that unlike the unrealistic ideal scheme, the practical nondecomposed scheme does not always reduce the base network latency as we increase the crossbar size. In Table 4.1, when the crossbar size is larger than 8×8 for the nondecomposed scheme, the base network latency starts to increase. It validates the assertion that the arbitration delay for a large crossbar can become a performance bottleneck. It can be derived that the optimal switch size for the nondecomposed scheme is 6×6 (the base network latency is minimum in the graph when the switch size equals 5.95). Note that different network sizes lead to the same conclusions since only w in the equation is affected.

4.2. Decomposed Arbitration for Large Crossbars

In the multi-cycle nondecomposed scheme, new requests are blocked during the arbitration. Hence, it reduces throughput and increases latency. In order to improve the performance, the crossbar can be logically partitioned into subarrays, each of which is arbitrated in one cycle. Requests can be granted as soon as the arbitration of the relevant subarray is completed. This leads to a significant reduction in the average delay to granting a request. We call this scheme the *decomposed arbitration*.

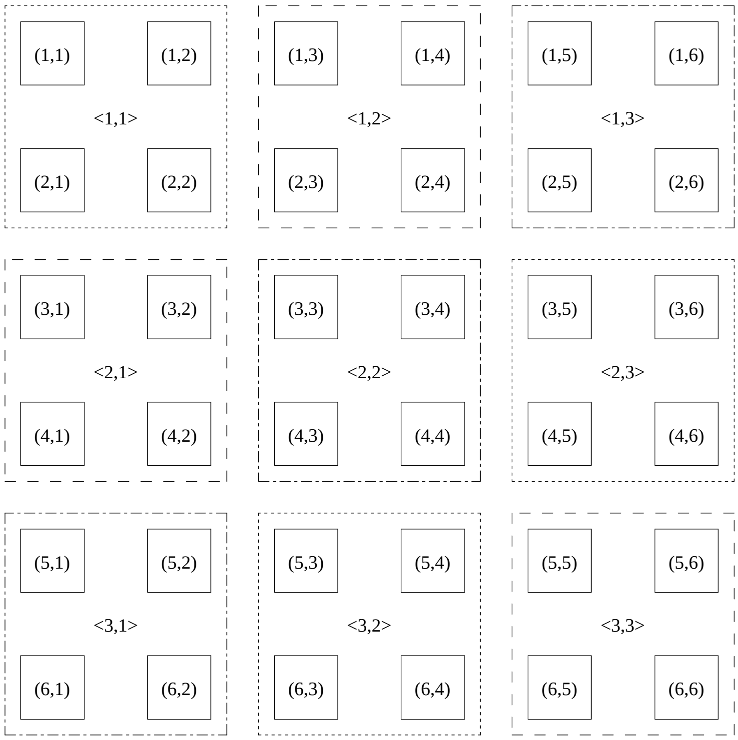

Figure 4.2: A 6×6 decomposed arbiter with 2×2 subarrays.

Figure 4.2 shows the operation of the decomposed arbitration for a 6×6 crossbar. In the figure, the whole arbitration array is partitioned into 9 subarrays where each subarray is a 2×2 WWFA. With the simple WFA and WWFA, conflicting requests are guaranteed to be on different ‘‘wave fronts’’ [Tami93]. The same principle is the key to the operation of the decomposed arbiter. Subarrays which are arbitrated in parallel, in the same clock cycle, have no inputs or outputs in common. Hence, it is guaranteed that requests which are arbitrated in different subarrays during each cycle cannot conflict. In the arbiter shown in Figure 4.2, three subarrays can be arbitrated during each cycle. For instance, subarrays <1,1>, <2,3>, and <3,2> may be arbitrated in the first cycle. Then subarrays <1,2>, <2,1>, and <3,3> are arbitrated in the second cycle, and <1,3>, <2,2>, and <3,1> are arbitrated in the third cycle. This allows requests to <1,1>, <2,3>, and <3,2> to be granted in the first cycle so that transmission can begin without waiting for the second cycle and the third cycle. Similarly, granted requests for <1,2>, <2,1>, and <3,3> can start transmitting after the second cycle is complete. A local circular shift register, which shifts every three cycles, is used for fairness within each subarray. Subarray arbitration is controlled by a global circular shift register which shifts every cycle. This global circular shift register indicates which ‘‘wrapped wave front’’ of the subarrays can be arbitrated in the current clock cycle. As we mentioned in Section 3.2, WWFA has a lower arbitration delay than WFA. Hence, WWFA is used for each arbitration subarray throughout this chapter.

To understand more precisely why decomposed arbitration outperforms nondecomposed arbitration, we consider a crossbar where nondecomposed arbitration takes d cycles. Two situations are discussed in the following: very light network load and heavy network load. When the network load is very light, with the nondecomposed scheme an arriving packet first has to wait, on the average,

½$(d-1)$ cycles in order to enter the arbitration and then has to wait for another d cycles for the arbitration to complete. On the other hand, with the decomposed scheme under very light network load, the average wait is ½$(d-1)$ cycles to enter the arbitration and then one cycle for the arbitration to complete. When the network load is heavy, with the nondecomposed scheme, once a packet leaves an input port buffer, the port has to wait ½$(d-1)$ cycles on the average before entering the arbitration and then wait for another d cycles for the arbitration to complete. On the other hand, with the decomposed scheme under heavy load, once a packet leaves an input port buffer, some of the queues at the buffer may enter the arbitration immediately in the following cycle and may begin transmission after the one cycle arbitration of their subarray is complete. Hence, using the decomposed arbitration lowers the latency for a packet going through the crossbar switch under the above two scenarios, compared to the nondecomposed arbitration. Note that the performance difference between the decomposed arbitration and the nondecomposed arbitration is more significant with larger crossbars, since d increases as the crossbar size increases.

4.2.1. *Analysis of Base Network Latency*

The base network latency of the decomposed arbitration can be derived by using analysis similar to that in Section 4.1. Assume again that the network size is $2^w \times 2^w$, and the switch size is $2^s \times 2^s$. The number of hops for a packet to traverse the network is $\frac{w}{s}$. Suppose $2^u \times 2^u$ is the 1-cycle subarray size. On the average, a packet waits ½$(d-1)$ cycles in order to enter the arbitration, where d equals 2^{s-u}. After the wait for entering the arbitration, it takes one cycle for the arbitration of the subarray to complete. Assuming k is the non-arbitration-related delay of the switch latency, the base network latency is thus $[k + \frac{1}{2}(d-1) + 1] \cdot (w/s)$, which equals $[k + \frac{1}{2}(2^{s-u}+1)] \cdot (w/s)$.

Long arbitration delay when the number of ‘‘agents’’ is high can be a problem even with conventional bus arbiters. A possible solution in this case is the use of tree-structured arbiters, whose delay grows logarithmically with the number of agents [Mudg87, Lang82a]. The existence of these logarithmic bus arbiters motivated us to compare our crossbar arbiter designs to a *hypothetical* crossbar arbiter whose delay grows logarithmically with the number of rows and columns. Although we use this logarithmic arbiter for comparison, we are not aware of any fast practical circuit that implements it. To analyze the base network latency of the hypothetical logarithmic arbitration, assume again that the switch size is $2^s \times 2^s$ and the 1-cycle subarray size is $2^u \times 2^u$. The arbitration delay for the hypothetical logarithmic scheme is then $\frac{s}{u}$. The average wait for a packet to enter the arbitration is $\frac{1}{2}(\frac{s}{u} - 1)$. The base network latency for the hypothetical logarithmic scheme is $[k + \frac{1}{2}(\frac{s}{u} - 1) + \frac{s}{u}] \cdot \frac{w}{s}$, which equals $[k + \frac{1}{2}(\frac{3s}{u} - 1)] \cdot \frac{w}{s}$. This arbitration scheme is very expensive in terms of hardware cost if it is at all possible to implement.

The results of the analysis for the ideal, the nondecomposed, the decomposed, and the hypothetical logarithmic schemes are shown in Figure 4.3 and Table 4.2. It shows that like the nondecomposed scheme, the decomposed scheme does not always reduce the base network latency as we increase the crossbar size either. However, unlike the nondecomposed scheme, when the crossbar size increases from 8×8 to 16×16, the base network latency for the decomposed scheme still improves. Furthermore, the decomposed scheme consistently outperforms the nondecomposed scheme. Their difference is especially significant when the switch size is large. Figure 4.3 also shows that unless the switch size is greater than 22, the decomposed scheme is superior to the hypothetical logarithmic scheme.

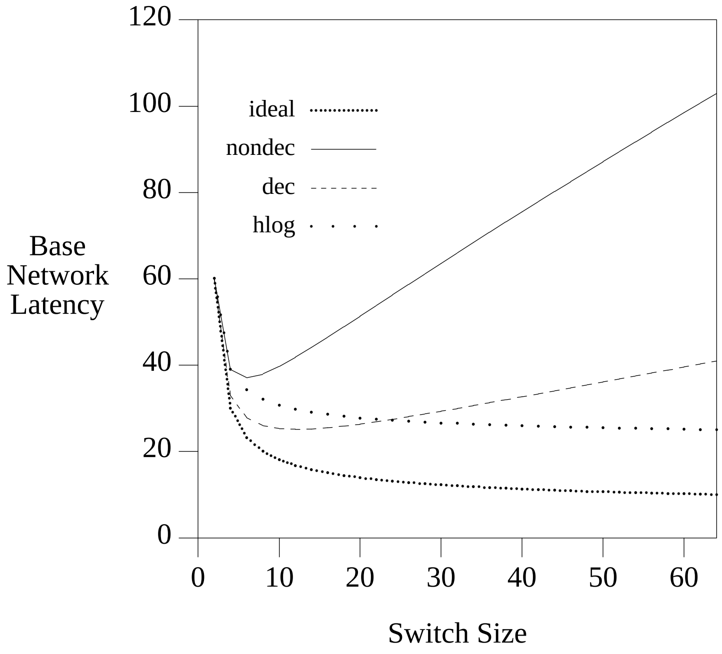

Figure 4.3: Base network latency vs. switch size of a 4096×4096 network for the four arbitration schemes. 1-cycle subarray size 2×2. The non-arbitration-related component of the switch latency is 4 cycles.

4.2.2. Switch Design Space for Optimal Networks

The decomposed arbitration scheme can be generalized such that each subarray can consist of more than one 1-cycle subarrays. This subarray is called the *arbitration subarray*. Assuming the 1-cycle subarray size is $2^u \times 2^u$ and the arbitration subarray size is $2^r \times 2^r$, it takes 2^{r-u} cycles to complete the arbitration of

Switch Size	Base Network Latency (4096×4096)			
	ideal	nondec	dec	hlog
2	60.0	60.0	60.0	60.0
4	30.0	39.0	33.0	39.0
8	20.0	38.0	26.0	32.0
16	15.0	46.5	25.5	28.5
64	10.0	103.0	41.0	25.0
4096	5.0	3075.5	1028.5	21.5

Table 4.2: Analysis of the base network latency of a 4096×4096 network for the four arbitration schemes. 1-cycle subarray size 2×2. The non-arbitration-related component of the switch latency is 4 cycles.

an arbitration subarray. The arbitration delay for the whole switch is $d = 2^{s-u}$. On the average a packet waits ½$(d-1)$ cycles in order to enter the arbitration, and the arbitration takes 2^{r-u} cycles to complete. Hence, the base network latency is $[k + ½(2^{s-u} - 1) + 2^{r-u}] \cdot (w/s)$. Note that with $r = s$, the scheme becomes the nondecomposed arbitration, while with $r = u$, the scheme becomes the decomposed arbitration scheme we analyzed in Section 4.2.1, where the arbitration subarray is the 1-cycle subarray.

Figure 4.4 shows the analysis results of the generalized decomposed arbitration based on the above equation, assuming that the 1-cycle subarray is 2×2, and the network size 4096×4096. The switch size and the arbitration subarray size vary between 2×2 and 64×64, but the arbitration subarray size is always smaller than or equal to the switch size. The base network latency in the figure is an approximation. We assume the switch size and the arbitration subarray size can be real numbers. In reality, they have to be integers, and a typical practical network is usually composed of switches whose size is a power of 2 for the ease of

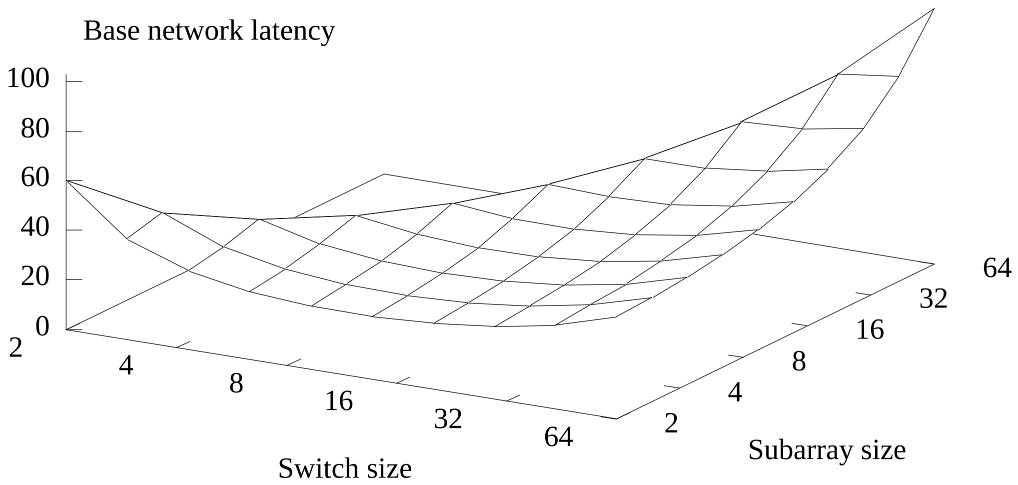

Figure 4.4: Analysis of base network latency for the generalized decomposed arbitration. Network size 4096×4096. Switch size $2^s \times 2^s$, $1 \le s \le 6$. Arbitration subarray size $2^r \times 2^r$, $1 \le r \le s$. The non-arbitration-related component of the switch latency is 4 cycles.

construction of the network. Furthermore, we also assume that the number of hops is $\frac{w}{s}$ for any switch size, including sizes for which s is not an integer.

Basically, Figure 4.4 illustrates the design space of a given network, with the switch size and the arbitration subarray size as the design parameters. The curve above the diagonal represents the base network latency for the nondecomposed arbiter, as shown in Figure 4.1. The base network latency is the product of the switch latency ($[\,k + \frac{1}{2}(2^{s-u}-1)+2^{r-u}\,]$) and the number of hops ($\frac{w}{s}$). The switch latency includes the *average arbitration wait*, which is the time a packet has to wait before entering arbitration (ˆ$\frac{1}{2}(2^{s-u}-1)$ˆ), and the subarray arbitration delay (2^{r-u}). For a fixed switch size, the base network latency decreases monotonically

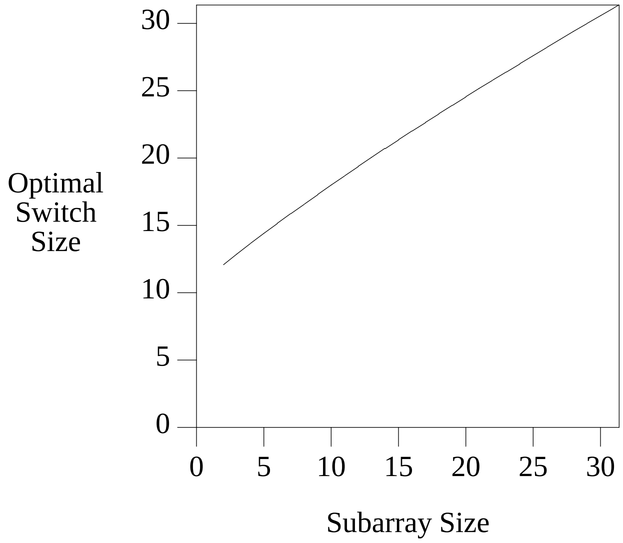

Figure 4.5: Optimal switch size vs. arbitration subarray size. 1-cycle subarray size 2×2. The non-arbitration-related component of the switch latency is 4 cycles.

with decreasing subarray size since the subarray arbitration delay is decreased and the other factors (arbitration wait time and number of hops) are not affected. For a fixed subarray size, increasing the switch size decreases the number of hops, which reduces the network latency, while increasing the single switch latency, which increases the network latency. Hence, for each subarray size, there is a switch size for which the network latency is minimized.

Figure 4.5 shows the optimal switch size for each subarray size. The optimal switch size increases with increasing arbitration subarray size. The reason for this behavior is that the subarray arbitration delay component of the switch delay increases with increasing subarray size and is independent of the switch size. The switch size only impacts the arbitration wait delay, which is independent of subarray size. Hence, for a ‘‘large’’ arbitration subarray, the *relative* increase in switch delay for a ‘‘unit’’ increase in switch size is lower than for a ‘‘small’’ arbitration subarray. On the other hand, the relative decrease in the number of hops for a ‘‘unit’’ increase in switch size is independent of the arbitration subarray size. This means that with increasing subarray size the negative impact of increasing the switch size is reduced while the positive impact is unchanged. The result is that a larger switch size is optimal. Note that when the arbitration subarray is greater than 31.4, the optimal switch size is equal to the arbitration subarray size since the arbitration subarray cannot be larger than the switch.

Based on the assumptions in the analysis above, reducing the arbitration subarray size always reduces the total switch arbitration delay. Hence, the smallest possible arbitration subarray is optimal. However, in reality, reducing the subarray size can have negative implications. In particular, for a fixed switch size, reducing the subarray size implies that there are more arbitration subarray in the switch. This can lead to an increase in the complexity and size (chip area) of the arbitration circuitry. As a result, the total switch arbitration delay may increase as the subarray size is decreased. Hence, the smallest arbitration subarray may not be optimal. For example, consider an 8×8 crossbar. With 4×4 arbitration subarrays, there will be four circular shift registers, with a total of 16 shift register cells, needed for the priority rotation within the arbitration subarrays. With 2×2 arbitration subarrays, there will be 16 circular shift registers, with a total of 32 shift register cells, needed for the priority rotation within the arbitration subarrays. These additional circular shift register cells require extra chip area and thus additional delay for switch arbitration due to the longer wires. An analysis of the negative impact of reducing the subarray size for our decomposed arbiter design is presented in Section 4.4.

4.3. Performance Evaluation

The analysis of network latency in Section 4.2 is only valid when the network has no traffic. In order to evaluate the various arbitration schemes for networks under different traffic loads, an even-driven simulator is also used. Simulations were performed for single switches and multistage networks. The interconnection network is an Omega network. The arbitration schemes simulated include the ideal arbitration, the nondecomposed arbitration, the decomposed arbitration, and the hypothetical logarithmic arbitration. The size of both the 1-cycle subarray and the arbitration subarray is 2×2. The links and buffers are byte-wide and every cycle one byte can be read from a buffer, written to a buffer, or transferred through a link.

The simulations are based on traffic with the following properties: 1) packet size is evenly distributed between 8 and 32 bytes. 2) during each cycle there is an equal probability of generating a packet at each of the inputs. 3) packet destinations are uniformly distributed over the outputs. We again assume that the non-arbitration-related delay of the switch latency is 4 cycles. A packet requests the use of the crossbar three cycles after arriving at the switch. After the arbitration, one cycle is spent before the packet starts to leave the switch.

The performance measures used are the average latency, the normalized throughput, and the 99^{th} percentile latency. The latency of a packet is the number of cycles that elapse from when the first byte of a packet generated at an input of the switch or network to when it leaves the switch or network. The normalized throughput is the average number of bytes received by each output per clock cycle. The 99^{th} percentile latency is the minimum of the latencies of the 1% of the packets that received the poorest service (longest latencies). It is reported here as a measure of the *fairness* of the arbitration. Lack of fairness in the arbitration results in increasing the 99^{th} percentile latency and increasing the difference between the 99^{th} percentile and average latencies [Tami93]. Each simulation run is terminated after 70,000 packets have arrived at their destination outputs. Statistics were gathered only after 7,000 packets in order to remove start-up effects.

Single switch simulations were performed for 8×8, 16×16, and 32×32 switches. The buffer size for each input port is 128 bytes for all switch sizes. Figure 4.6 and Figure 4.7 show the performance impact of various arbitration schemes for an 8×8 switch, in terms of the average latency vs. the normalized throughput and the 99^{th} percentile latency vs. the normalized throughput, respectively. Figures 4.8, 4.9, 4.10, and 4.11 show the results for the 16×16, and 32×32 switches. For all these switches, the decomposed arbitration outperforms the nondecomposed arbitration by achieving higher maximum throughput, as well as lower average latency and 99^{th} percentile latency. The performance difference between the decomposed arbitration and the nondecomposed arbitration increases as the crossbar size increases, as we discussed in the analysis in Section 4.2. Furthermore, the performance difference between the decomposed arbitration and the ideal arbitration is limited, compared to the difference between the decomposed arbitration and the nondecomposed arbitration. The decomposed arbitration also performs better than the hypothetical logarithmic arbitration for 8×8 and 16×16 switches. For the 32×32 switch, although the decomposed arbitration has a higher

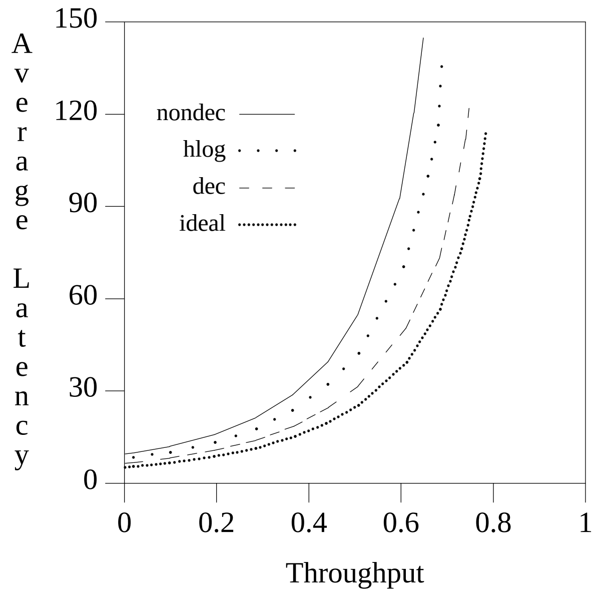

Figure 4.6: Average latency vs. normalized throughput of an 8×8 switch for various arbitration schemes. The subarray size is 2×2. The buffer size for each input is 128 bytes.

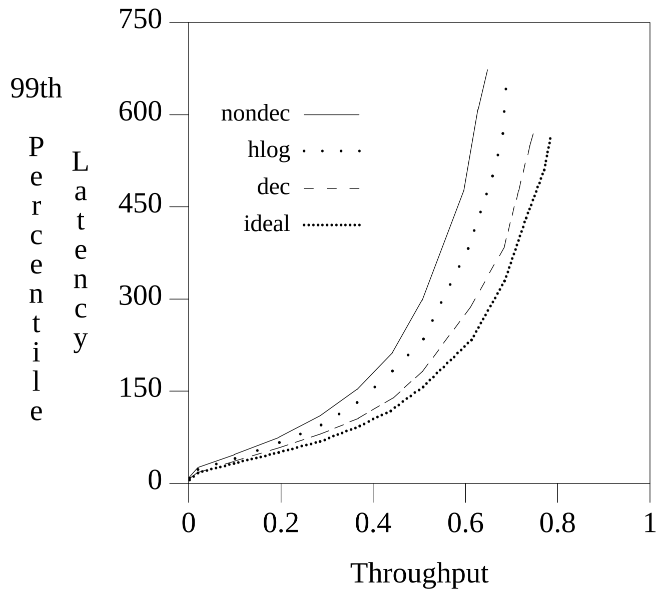

Figure 4.7: 99^{th} percentile latency vs. normalized throughput of an 8×8 switch for various arbitration schemes. The subarray size is 2×2. The buffer size for each input is 128 bytes.

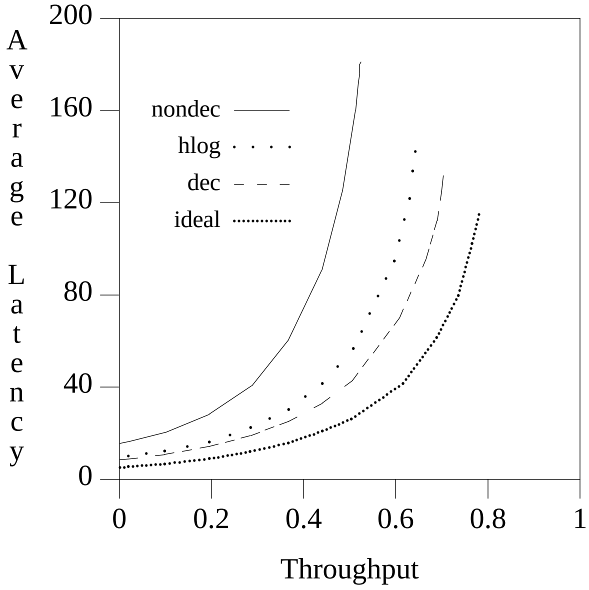

Figure 4.8: Average latency vs. normalized throughput of a 16×16 switch for various arbitration schemes. The subarray size is 2×2. The buffer size for each input is 128 bytes.

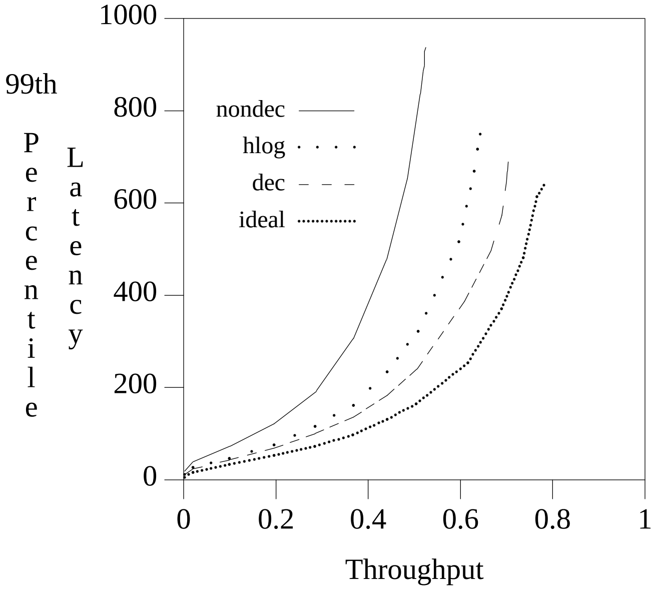

Figure 4.9: 99th percentile latency vs. normalized throughput of a 16×16 switch for various arbitration schemes. The subarray size is 2×2. The buffer size for each input is 128 bytes.

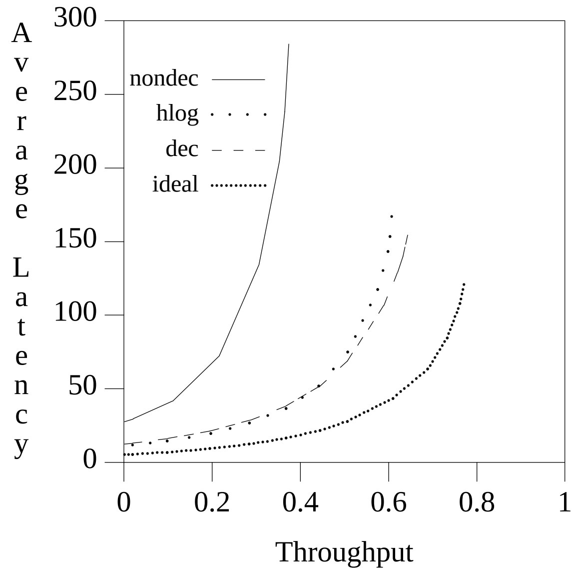

Figure 4.10: Average latency vs. normalized throughput of a 32×32 switch for various arbitration schemes. The subarray size is 2×2. The buffer size for each input is 128 bytes.

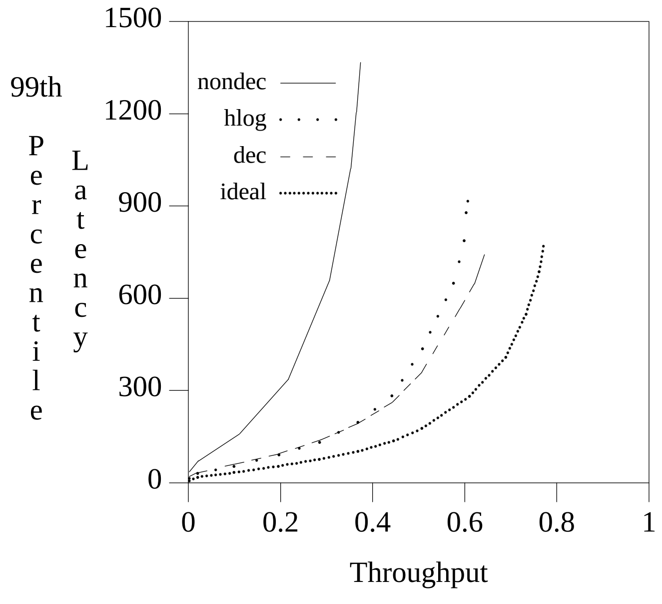

Figure 4.11: 99th percentile latency vs. normalized throughput of a 32×32 switch for various arbitration schemes. The subarray size is 2×2. The buffer size for each input is 128 bytes.

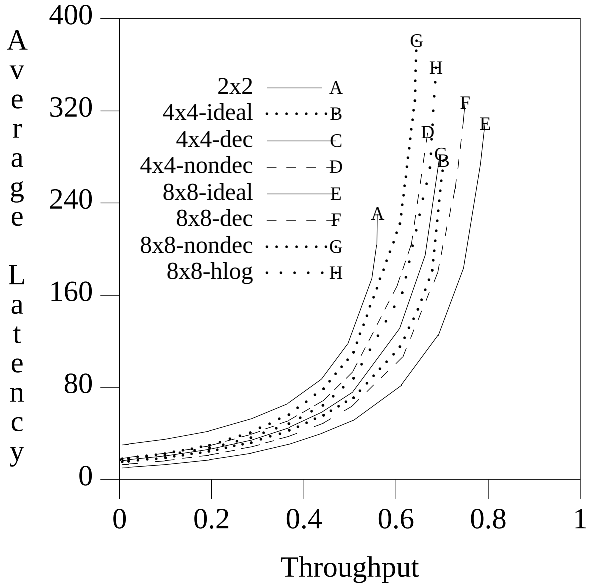

Figure 4.12: Average latency vs. normalized throughput of a 64×64 network for various switch sizes and various arbitration schemes. The subarray size is 2×2. The buffer size for each input is 64, 128, and 192 bytes for 2×2, 4×4, and 8×8 respectively.

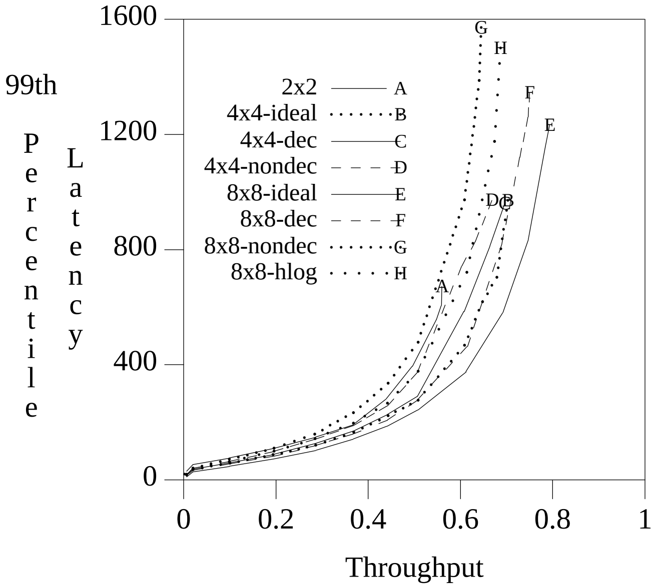

Figure 4.13: 99th percentile latency vs. normalized throughput of a 64×64 network for various switch sizes and various arbitration schemes. The subarray size is 2×2. The buffer size for each input is 64, 128, and 192 bytes for 2×2, 4×4, and 8×8 respectively.

base network latency than the hypothetical logarithmic arbitration (12.5 vs. 11.0), the decomposed arbitration starts to perform better when the throughput is moderate to high.

Multistage network simulations were performed for 64×64, and 256×256 networks. In the multistage network simulations, different input buffer sizes are used for different switch sizes, such that we can compare the performance of the networks with the same amount of buffer space in the whole network. The input buffer sizes for 2×2, 4×4, 8×8, and 16×16 switches are 64, 128, 192, and 256 bytes, respectively. The simulation results for a 64×64 network are shown in Figure 4.12 and Figure 4.13. Three different switch sizes (2×2, 4×4, and 8×8) and four arbitration schemes (ideal, nondecomposed, decomposed, and hypothetical logarithmic) are evaluated. The 64×64 large crossbar switch is not included here due to the lack of enough computing power for its simulation. The figure shows that, given the same switch size, the decomposed arbitration outperforms both the nondecomposed arbitration and the hypothetical logarithmic arbitration, and is close to the ideal arbitration. With the ideal arbitration, the performance steadily improves as the switch size increases. However, with the nondecomposed arbitration, the performance of the 8×8 switch is even worse than that of the 4×4 switch. On the other hand, when we increase the switch size from 4×4 to 8×8 for the decomposed arbitration, the performance still improves.

Figure 4.14 and Figure 4.15 show the simulation results for a 256×256 network. The switch size varies from 2×2, 4×4, to 16×16. Again, the decomposed arbitration outperforms the nondecomposed arbitration, and the performance difference is particularly significant for using 16×16 switches. The performance of

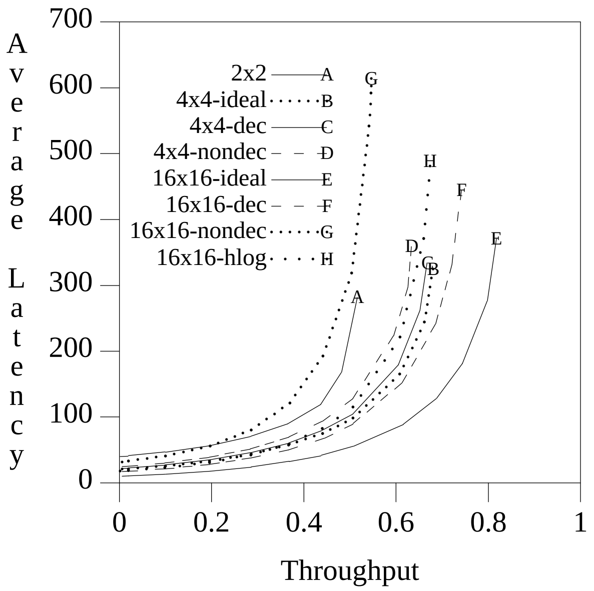

Figure 4.14: Average latency vs. normalized throughput of a 256×256 network for various switch sizes and various arbitration schemes. The subarray size is 2×2. The buffer size for each input is 64, 128, and 192 bytes for 2×2, 4×4, and 8×8 respectively.

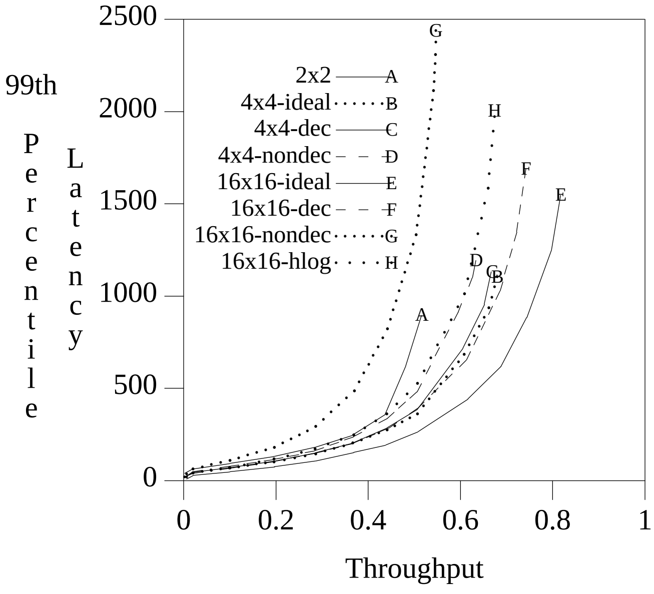

Figure 4.15: 99th percentile latency vs. normalized throughput of a 256×256 network for various switch sizes and various arbitration schemes. The subarray size is 2×2. The buffer size for each input is 64, 128, and 192 bytes for 2×2, 4×4, and 8×8 respectively.

the hypothetical arbitration is also not as good as the decomposed arbitration for 16×16 switches. With the ideal arbitration and the decomposed arbitration, the performance improves as we increase the switch size. In contrast, the performance of the nondecomposed arbitration for the 16×16 switch is worse than that for the 4×4 switch. The nondecomposed arbitration for the 16×16 switch even has higher average latency and 99th latency than the 2×2 switch case unless the throughput is low.

It should be noted that for both the decomposed and nondecomposed schemes, performance under heavy load may be improved by using the length of the packet currently being forwarded from an input port to predict when the input port will be free and begin arbitration while the current transmission is still in progress. This technique will involve significant increase in hardware complexity and will not improve performance under light load.

4.4. Design of Decomposed Arbiters

The decomposed arbiter is composed of several subarrays. Each subarray is implemented as a wrapped wave front arbiter. Figure 4.16 shows the organization of a 2×2 WWFA [Tami93]. For each arbitration cell, the P (priority) input indicates whether the cell has the top priority. The XI, XO, YI, and YO signals correspond to the W, E, N, and S signals, respectively, of Figure 3.8. The circular shift register is used to rotate top priority among the wrapped diagonals of arbitration cells.

Figure 4.17 shows the organization of an example decomposed arbiter for a 4×4 crossbar. In this example the whole arbitration array is partitioned into four

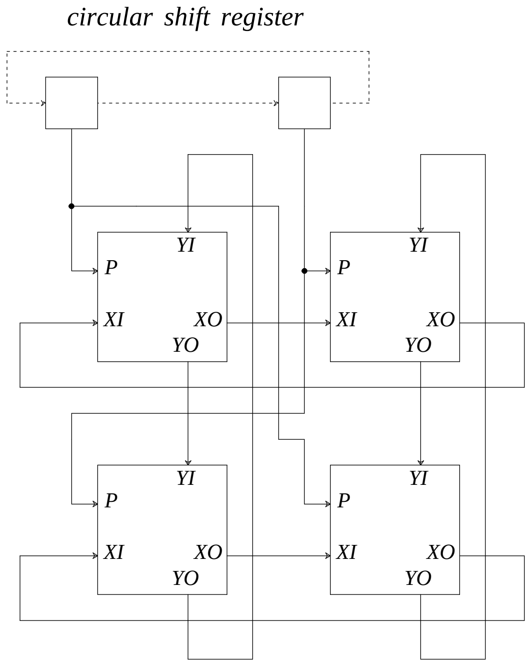

Figure 4.16: Organization of a 2×2 wrapped wave front arbiter. This can be used as a subarray of a decomposed arbiter.

subarrays, and each subarray is a 2×2 WWFA. The detailed design of an arbitration cell is shown in Figure 4.18 and Figure 4.19. The $\overline{OPB}$ (output port blocked) line is 0 when the output port is blocked. SAE (subarray enable) is 1 when the associated subarray is enabled by a signal from the global circular shift register for subarray arbitration (Figure 4.17). R is the request line from a queue of the buffer and G is the grant line. P (priority) indicates if the cell gets the top priority. BT (buffer transmitting) signal is asserted by the buffer in order to reserve the crosspoint it is using while the packet is being transmitted. When BT is 1, only one R line from the buffer is asserted. $\overline{CB}$ (column busy) is a wired-OR line which connects all the cells in a column, indicating whether some buffer is using the column for an ongoing transmission and it should not be interrupted by

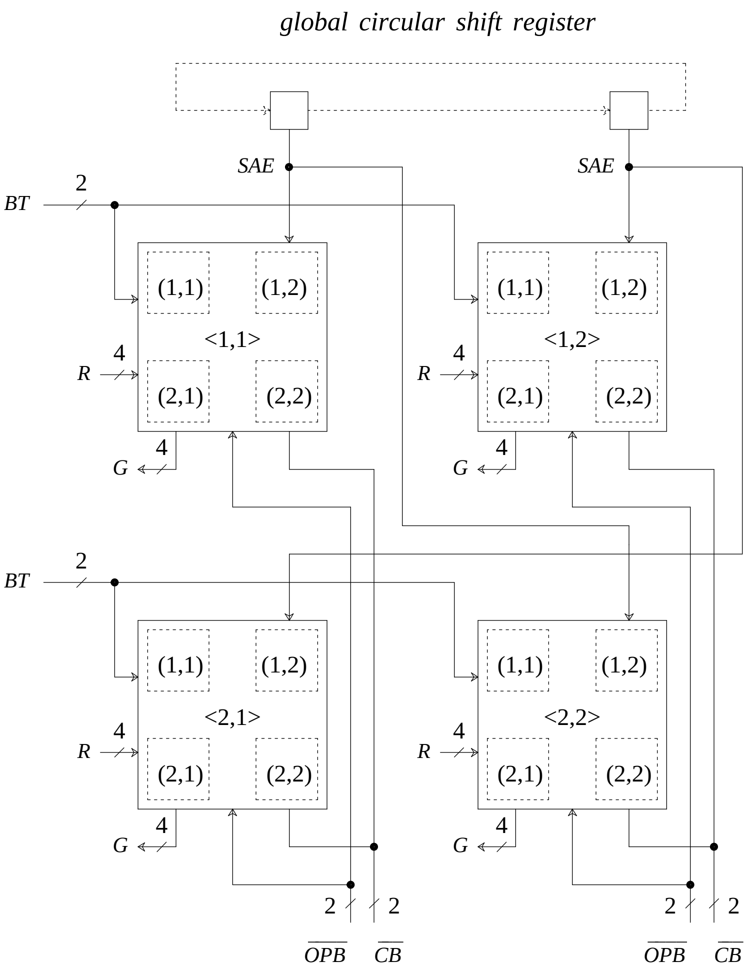

Figure 4.17: A 4×4 decomposed arbiter with 2×2 subarrays.

other buffer requests.

The circuitry in the right part of Figure 4.19 is identical to the circuitry of the arbitration cell described in Chapter 3 and implements the following Boolean equations:

$$G = (RR \wedge \overline{OPB}) \wedge (YI \vee P) \wedge (XI \vee P)$$

$$YO = (YI \vee P) \wedge \overline{G}$$

$$XO = (XI \vee P) \wedge \overline{G}$$

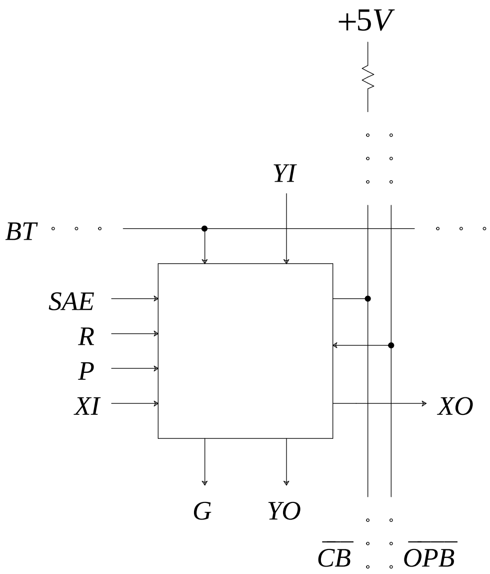

Figure 4.18: An arbitration cell.

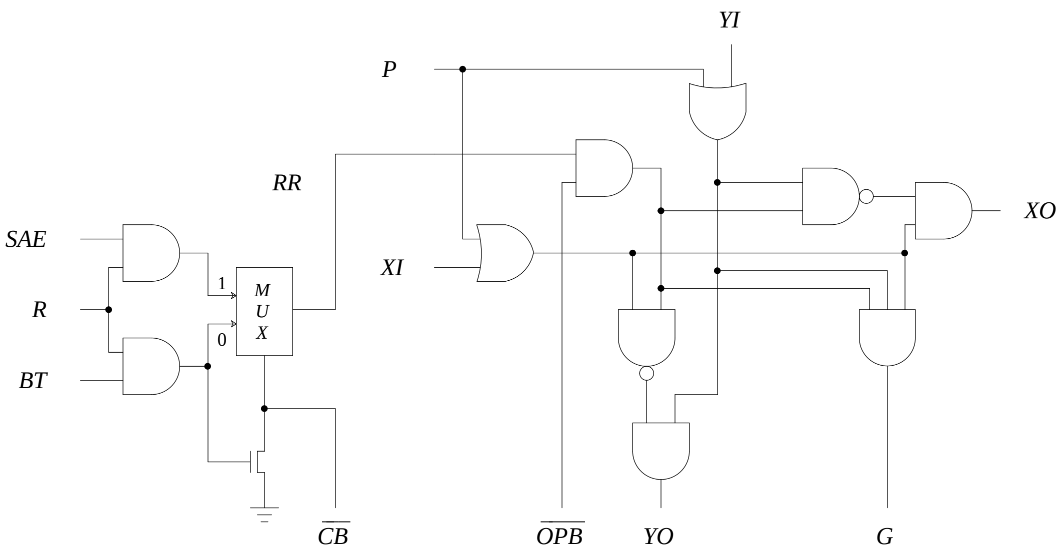

Figure 4.19: Logic design of the arbitration cell for subarrays of decomposed arbiters.

The left part of the circuit is unique to the decomposed arbiter. The RR line is asserted when either the particular queue is currently forwarding a packet (R and BT are asserted and $\overline{CB}$ is pulled down), or when the column is idle, the subarray is enabled, and the queue has a packet ready for transmission to the column.

The design presented in this section is amenable for modular implementation where several arbiter subarrays are implemented on a single chip and multiple copies of this chip can be used to construct a large arbiter. A single chip with arbiter subarrays as well as their corresponding data part of the crossbar can be used as a flexible building block for crossbar switches.

To construct a nondecomposed crossbar switch whose size is multiple of the building block chip size, several chips can be connected in an array. The SAE inputs of each chip are set to 1. A decomposed arbiter can be constructed similarly but the SAE inputs of each chip are connected to their corresponding state latches of the global circular shift register. To provide a decomposed crossbar with the global circular shift register, latches can be included in each chip. Several latches in different chips are linked together to form the global circular shift register. The state of a latch is fed to the associated group of subarrays to indicate ‘‘the turn’’ of the group. Assume the crossbar size is $2^s \times 2^s$ and the subarray size is $2^u \times 2^u$. Then the switch arbitration delay is $d = 2^s / 2^u = 2^{s-u}$. Since the local circular shift register shifts every d cycles the clock signal for the local circular shift register can be obtained from the latch output of one latch of the global circular shift register.

Switch Design Space

In Section 4.2.2 we indicate that with the smallest arbitration subarray size, the base network latency may not be optimal. In the following, we discuss how the arbiter circuit design discussed in this section affects the switch design space described in Section 4.2.2. The equation for calculating the base network latency for the generalized decomposed arbiter in Section 4.2.2 is modified to reflect the impact of the implementation of the arbiter.

In the design of our decomposed arbiter, each column has a wired-OR CB line connected to every arbitration cell in the column. The CB line indicates if any arbitration cell is using the column in the current cycle. After an arbitration subarray (size $2^r \times 2^r$) is arbitrated, the CB line has to be set to reflect the results of this arbitration, which will be used by the next arbitration. Hence, the time allotted for a complete $2^r \times 2^r$ subarray arbitration must include the time to set the CB line. The delay for setting the CB line depends on the number of the arbitration cells (2^s) and the number of the circular shift register cells (2^{s-r}) in a column. For each arbitration cell, the gate inputs driven by the CB line and the length of the CB line through the arbitration cell add capacitances to the CB line. For each circular shift register cell, the length of the CB line through the register cell also adds capacitances to the CB line. These capacitances result in extra delay for the arbitration of a $2^r \times 2^r$ subarray. If the switch size is fixed, this extra delay increases as the arbitration subarray size decreases, since the number of the circular shift register cells (2^{s-r}) increases.

As we decrease the arbitration subarray size, the CB line is set (pulled down) more times during the arbitration of the whole switch. Since the average arbitration wait is half the total delay for arbitrating the whole switch, the switch latency increases due to more times for setting the CB line. Furthermore, with the arbitration subarray size decreasing, the extra CB line capacitance increases due to more shift register cells and thus causes longer delay for setting the CB line. With both the effect of more times for setting the CB line and the effect of longer delay for each CB setting, the base network latency may no longer be the lowest with the smallest subarray size.

Assume the extra delay of setting the CB line due to each arbitration cell is α and the extra delay due to each circular shift register cell is $\beta \cdot \alpha$, where α and β are implementation-dependent parameters. The extra delay of setting the CB line for each arbitration subarray is then $\alpha \cdot 2^s + \alpha \cdot \beta \cdot 2^{s-r}$. Hence, the delay for arbitrating a $2^r \times 2^r$ subarray is $h = 2^{r-u} + \alpha \cdot 2^s + \alpha \cdot \beta \cdot 2^{s-r} = 2^{r-u} + \alpha \cdot 2^s \cdot (1 + \beta \cdot 2^{-r})$. Since there are 2^{s-r} arbitration subarrays in each dimension of the arbiter, the average time a packet has to wait before entering the arbitration is $½ \cdot 2^{s-r} \cdot h$. The resulting base network latency is thus $[k + ½ \cdot 2^{s-r} \cdot h + h] \cdot (w/s)$, which equals $[k + (2^{s-r-1} + 1) \cdot [2^{r-u} + \alpha \cdot 2^s \cdot (1 + \beta \cdot 2^{-r})]] \cdot (w/s)$. Note that the above derivation ignores the length of the clock cycle, and it is only an approximate delay calculation.

Based on the equation of the base network latency in the above, we can derive the optimal switch size r from a given set of values for k, w, s, u, α, and β. The reasonable range of the values for α and β can be estimated as follows. The delay of each arbitration cell is approximately 3 to 4 gate delays according to the design in Figure 4.19. Assuming the 1-cycle subarray size is 2×2, the delay of each

1-cycle subarray is 6 to 8 gate delays. This delay corresponds to one time unit in the equation of the base network latency. Hence, the minimum size gate delay is on the order of 0.1 to 0.2 time units. As stated earlier, the number of arbitration cells and circular shift register cells per column determine the ''fan-out'' for the CB line pull-down transistor. Each arbitration cell is a load of approximately one gate plus the capacitance of the CB line segment across it. Hence, reasonable values for α are in the range of 0.01 to 0.2. The circular shift register cell is smaller than the arbitration cell and is not driven by the CB line. Hence, the extra delay for each shift register cell is smaller than for each arbitration cell and the value of β is expected to be in the range of 0.1 to 0.5. As an example, we consider a network with the following parameters: $k = 4$, $w = 12$, $s = 4$, $u = 1$. For $\alpha \leq 0.02$ and $0.1 \leq \beta \leq 0.5$, the optimal arbitration subarray size r is 1. However, for $\alpha \geq 0.04$ and $0.1 \leq \beta \leq 0.5$, the the base network latency with $r = 2$ is lower than with $r = 1$. Hence, in the latter case, a network with arbitration subarray size of 4×4 will perform better than a network with subarray size of 2×2.

4.5. Reducing the Number of Queues

For a large crossbar, the cost of managing, at each input port, a separate queue for each output port may be prohibitive. The control registers can require significant chip area and the large number of these register will reduce circuit performance (larger decoders, longer buses, additional loads on buses, etc) [Fraz89]. This problem can be alleviated by reducing the number of queues in each buffer. The output ports are partitioned into several groups, and all the packets destined to the output ports in the same group share the same queue. For example, consider an $n \times n$ switch, with q queues in each input buffer. The output ports are partitioned into q groups. A packet destined to output port t is stored in queue $\lfloor qt/n \rfloor$. The crossbar arbiter design described in Section 4.4 can still be used here even though fewer queues are implemented in each buffer.

Reducing the number of queues in a buffer may allow a reduction in the number of wires for *request* and *grant* signals between the buffers and the arbiter. For example, if there are eight queues in a buffer for a 32×32 crossbar, then we need only three wires for the requests from each queue. One wire is used to indicate if there is a request from this queue, and the other two specify which crosspoint is requested. Only one grant wire is needed per queue. Hence, the total number of wires for *request* and *grant* signals between each buffer and the arbiter is 32. This is a significant reduction compared to the 64 wires that would be required if 32 queues were maintained in each buffer. With this scheme there is additional hardware required in the arbiter for decoding the lines carrying the number of the output port requested by each queue.

In order to determine the impact of varying the number of queues per input port, we have simulated a 32×32 switch and a 256×256 network composed of 16×16 switches. The arbitration is performed by a decomposed arbiter with 2×2 subarrays. The simulation parameters and performance measures are as described in Section 4.3. Figure 4.20 and Figure 4.21 show the simulation results for the 32×32 switch, and Figure 4.22 and Figure 4.23 show the simulation results for the 256×256 network. In the 32×32 single switch simulation, the number of queues per buffer can be reduced to 16 or even 8 while the performance is only degraded a little. In the simulation for the 256×256 network with 16×16 switches, when there

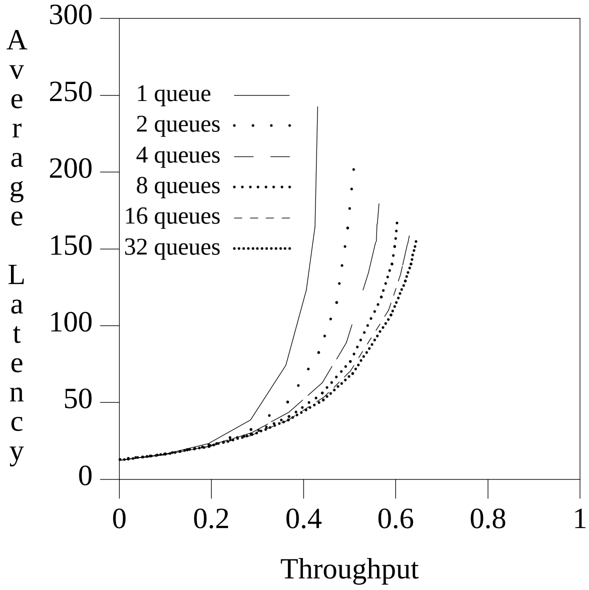

Figure 4.20: The impact of varying the number of queues per input for a 32×32 switch. Average latency vs. normalized throughput. The subarray size is 2×2. The buffer size for each input is 128 bytes.

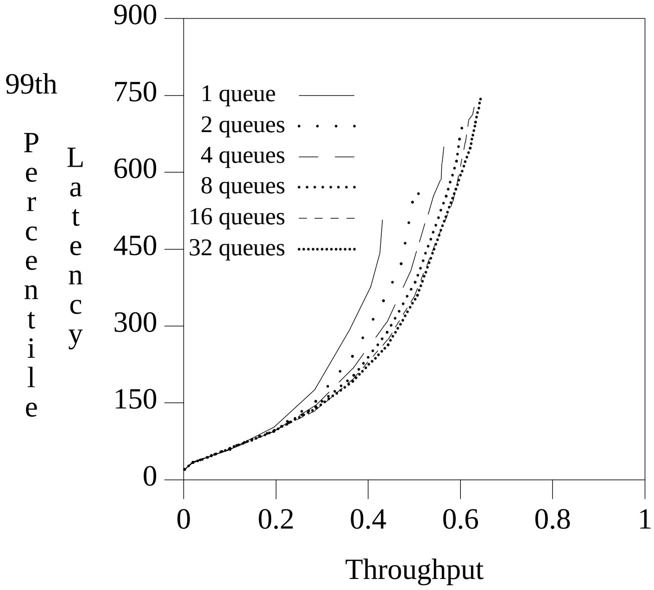

Figure 4.21: The impact of varying the number of queues per input for a 32×32 switch. 99th percentile latency vs. normalized throughput. The subarray size is 2×2. The buffer size for each input is 128 bytes.

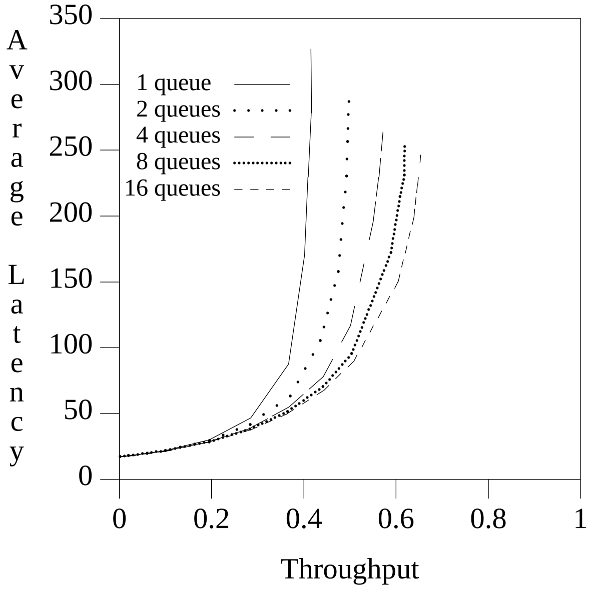

Figure 4.22: The impact of varying the number of queues per input for 16×16 switches in a 256×256 network. Average latency vs. normalized throughput. The subarray size is 2×2. The buffer size for each input is 128 bytes.

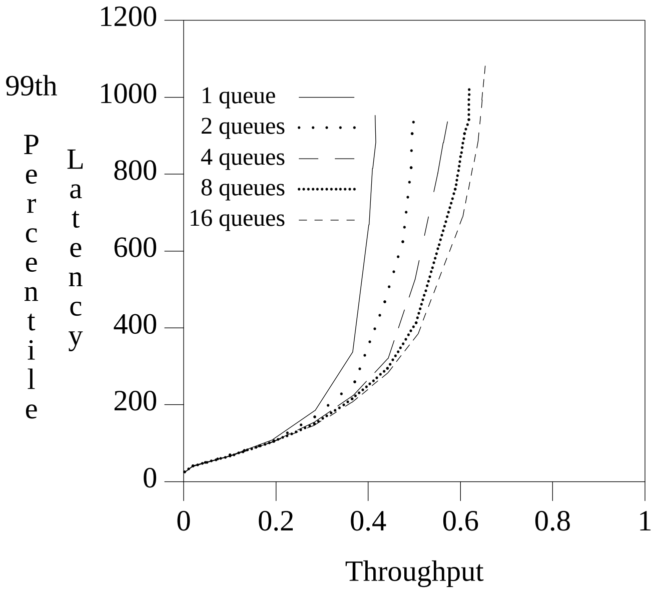

Figure 4.23: The impact of varying the number of queues per input for 16×16 switches in a 256×256 network. 99th percentile latency vs. normalized throughput. The subarray size is 2×2. The buffer size for each input is 128 bytes.

are 8 queues per buffer, the performance is still close to that with 16 queues. Note that with only one queue, the buffer degenerates into a FIFO buffer.

4.6. Summary and Conclusions

Theoretically, larger crossbars improve the network performance by lowering the probability of conflicts and reducing the number of hops for a packet to traverse the network. However, the time required to arbitrate conflicting requests to crossbar switches increases with the size of the crossbar. In reality, this long arbitration delay can become a critical performance bottleneck. We have proposed an arbiter design, which is called the decomposed arbiter, to solve the arbitration delay problem. By decomposing the arbitration process, some requests can be granted before the arbitration of the entire crossbar is complete. This leads to a significant reduction in the average delay to granting a request. Our analysis and simulation results show that decomposed arbiters effectively increase the bandwidth and reduce the latency for the interconnection network. The additional implementation cost compared to a nondecomposed scheme is minimal. This design lends itself to efficient modular implementation of large crossbars using small crossbar building blocks.

As originally designed, switches based on DAMQ buffers manage, at each input port, a separate queue for each output port. The circuit complexity for managing multiple queues in an input buffer increases as the number of queues is increased. Furthermore, with the original DAMQ design, the number of *request* and *grant* lines between each buffer and the arbiter increases linearly with the size of the switch. Based on these considerations, for a large crossbar, it is undesirable to have at each input port a separate queue for each output port. We have shown that this problem can be alleviated by reducing the number of queues per input buffer. For large switches, even if the number of queues is reduced significantly, there is only a small reduction in performance compared to switches that use full DAMQ buffers.