Chapter Six

Arbiter Support for Adaptive Routing and for High-Priority Traffic

This chapter describes the arbiter designs that support other types of traffic in the interconnection network. Arbiters for supporting adaptive routing and high-priority traffic are presented. Adaptive routing can be used to improve the performance of the interconnection network and provide fault tolerance. In a communication switch with adaptive routing capability, a packet can have more than one destined output. Packets arriving at the inputs can adapt to the congested or faulty area in the network and decide which output they should be destined to in order to avoid the area. In Section 6.1, we discuss arbiter designs which can be used in adaptive routing switches. We describe how our symmetric crossbar arbiters can be applied to the arbiter design for adaptive routing switches with conventional FIFO input buffers. In some multiprocessors and multicomputers, high-priority traffic is used to support real-time applications. Section 6.2 discusses how to design a crossbar arbiter which can support high-priority traffic. Arbiter designs with two levels of packet priorities are described.

6.1. Arbiter Design Supporting Adaptive Routing

Many interconnection networks use *deterministic routing* to transmit packets from the source node to the destination node. With deterministic routing, a packet can only follow the pre-determined path in the network to get to the destination. If some portion of the path is heavily loaded or faulty, the packet cannot circumvent the congested or faulty area. This can cause poor network performance or failure to deliver messages. Figure 6.1 shows an example of the deterministic dimension-order routing from the source node (5,1) to the destination node (2,5) in a mesh network. The packets sent from the source node to the destination node follow a fixed path by reducing the relative address in X dimension to 0 and then reducing the relative address in Y dimension to 0.

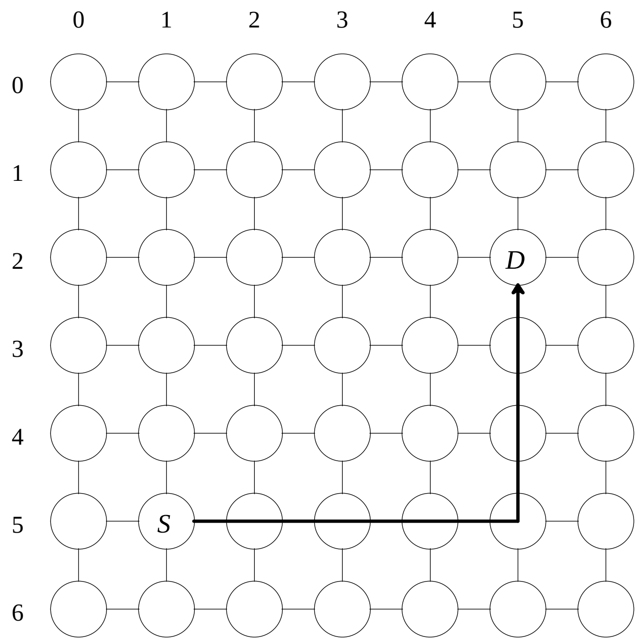

Figure 6.1: A path using deterministic dimension-order routing. The path is fixed and determined by the addresses of the source node and the destination node.

Adaptive routing has been proposed to overcome the problems with deterministic routing and achieve performance improvement and fault tolerance of interconnection networks [Ngai89, Dall93, Chow87, Chen90, Kons90b]. In general, adaptive routing can be classified into *minimal* routing and *non-minimal* routing. With minimal adaptive routing, packets can only be routed to switches which are one hop closer to the destination node. Figure 6.2 shows an example of the minimal adaptive routing from the source node (5,1) to the destination node (2,5) in a mesh network. Along the path, the link connecting nodes (5,3) and (5,4) and the link connecting nodes (4,4) and (4,5) are heavily loaded or faulty. With non-minimal adaptive routing, packets can be misrouted farther away from their destination in the presence of congestion or faulty components. Figure 6.3 shows an example of the non-minimal adaptive routing in a mesh network. Along the path, the link connecting nodes (5,4) and (4,4) and the link connecting nodes (5,4) and (5,5) are heavily loaded or faulty. In this example, packets are misrouted at node (5,4) to avoid the problem links.

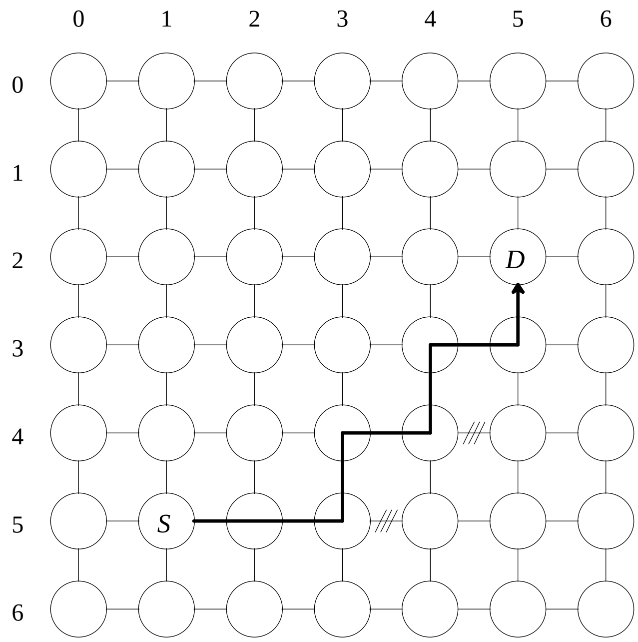

Figure 6.2: A path using minimal adaptive routing. The link connecting nodes (5,3) and (5,4) and the link between nodes (4,4) and (4,5) are heavily loaded or faulty.

Symmetric crossbar arbiters can be applied to the switch design supporting

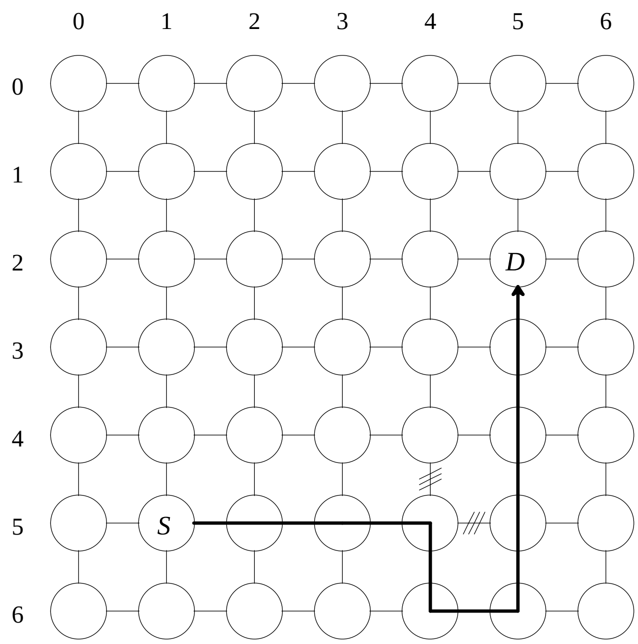

Figure 6.3: A path using non-minimal adaptive routing. The link connecting nodes (5,4) and (4,4) and the link connecting nodes (5,4) and (5,5) are heavily loaded or faulty.

adaptive routing. Since each packet can have more than one destined output, the multi-queue buffers which partition the buffer space based on the single destined output of each packet are inappropriate here. Hence, FIFO buffers are used at the input ports and only packets at the heads of the FIFO buffers can submit requests to the crossbar arbiter. Figure 6.4 shows an example of arbitration for an adaptive routing switch. The routing decision circuit at each input determines the destined outputs for each packet arriving in the input port. The arbitration decision is made after the routing decision. The arbiter gathers the information of the destined outputs generated by all the routing decision circuits at the input ports and globally resolves the contention of the requests.

Figure 6.5 shows the arbitration model of the example in Figure 6.4. In the

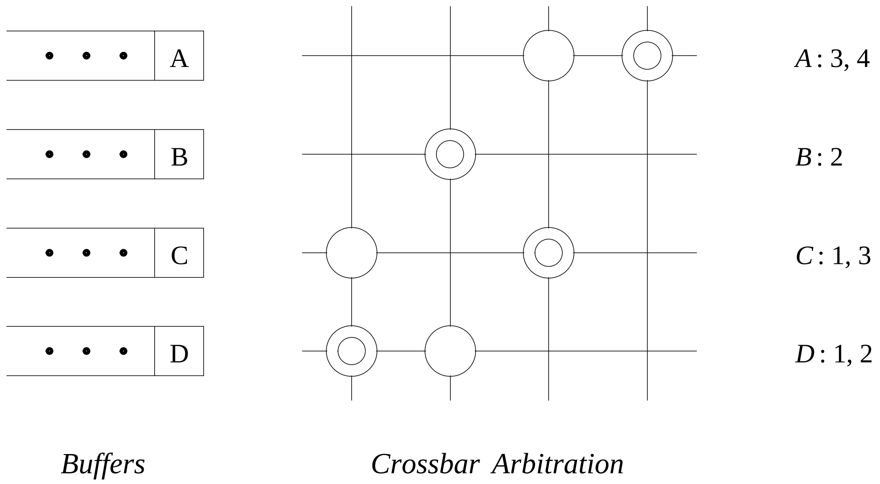

Figure 6.4: An example of arbitration for an adaptive routing switch. The destined outputs of packets A, B, C, and D are {3,4}, {2}, {1,3}, and {1,2}, respectively. Single circles are denied requests after the arbitration. Double circles are granted requests.

model, each input buffer has only one requesting packet, which is at the head of the queue, but this packet can have more than one candidate output. Only one candidate output at most can be granted to each requesting packet. Furthermore, each output port can receive more than one request from the buffers, but only one request at most can be granted. This model is exactly the same model as the symmetric crossbar arbitration we described in Chapter 3. Hence, the symmetric crossbar arbiters we proposed can be used here.

In some systems, the adaptive routing strategy employed provides cost information about each route. In order to optimize performance for a particular packet, at each switch that packet should be routed to the most ‘‘profitable’’ output, so that the packet will end up traversing the least cost route to its destination. However, routing packets only through the most profitable outputs, based on static cost information, does not result in optimal system performance.

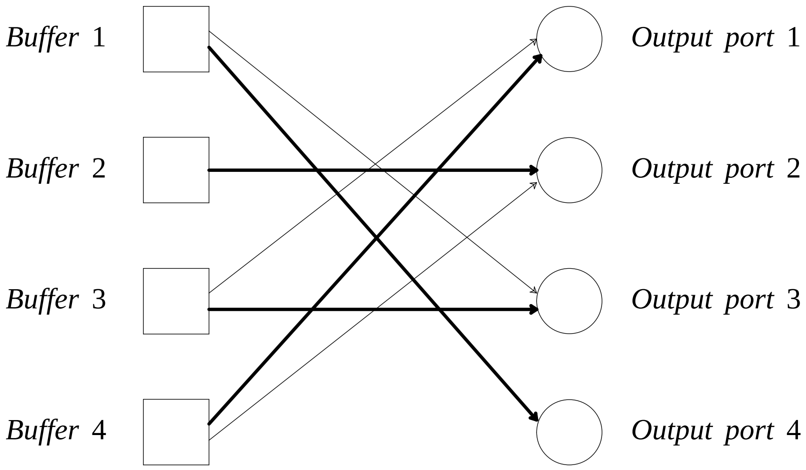

Figure 6.5: The arbitration model for the example in Figure 6.4. Output ports 1, 2, 3, and 4 are granted to buffers 4, 2, 3, and 1, respectively.

Specifically, if only the most profitable outputs are considered, there may be contention for output ports that will result in blocking of some packets which could have been routed towards their destinations via less profitable outputs. System performance can be maximized if all the profitable outputs for each requesting packet are considered in the arbitration process. Ideally, the arbiter should arbitrate requests in a sequence of phases, incrementally trying less desirable outputs if the packet does not get access to the most profitable (desirable) switch outputs. However, such an arbiter would be too slow and complex. A practical alternative is an arbiter that divides all the possible outputs for a packet into only two classes: *preferred* outputs and *acceptable* outputs. The arbiter arbitrates for the preferred outputs in the first phase and then arbitrates for the acceptable outputs in the second phase.

The symmetric crossbar arbiters described in Chapter 3 can be used for the two-phase scheme described above. Both phases are performed by the same circuit. Figure 6.6 shows an example of the two-phase arbitration in an adaptive routing switch. For any design of a crossbar arbiter, there is an interface circuit between the buffers and the arbiter. The interface circuit keeps track of which rows and columns are busy (granted) and blocks additional requests for busy rows and columns. In the two-phase arbiter design, this interface circuit must maintain the busy-row and busy-column information between the two phases. After grants in the first phase have been given to the preferred requests in the first phase, the acceptable requests for the rows and columns granted in the first phase are blocked by the interface circuit.

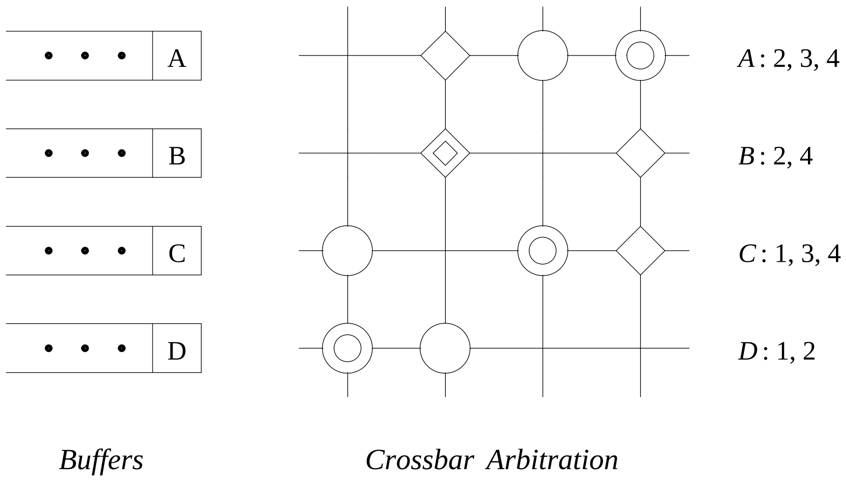

Figure 6.6: An example of two-phase arbitration in an adaptive routing switch. Single circles are denied preferred requests after the arbitration. Double circles are granted preferred requests. Single diamonds are denied acceptable requests. Double diamonds are granted acceptable requests.

The two-phase arbitration can be completed in either one clock cycle or two clock cycles. With the two clock cycles design, each arbitration phase corresponds to a separate clock cycle. The interface circuit must ensure that in the first cycle only the preferred requests of every packet are considered. The disadvantage of this design is that it takes two cycles to grant an acceptable request for which the resources are available in the first cycle. Alternatively, the one clock cycle design can be used, where each arbitration phase is completed in one clock phase of a clock cycle. With this one clock cycle design, the circuit of the two-phase arbiter has to meet the timing constraints of the switch. The interface circuit which keeps track of the busy-row and busy-column information must be fast enough to latch the grant signals generated in the first phase.

The two-phase arbiter design can be used to support non-minimal routing. For example, in a switch with a non-minimal routing scheme the preferred outputs can be those which are connected to the switches that are one hop closer to the destination. For packets at the heads of the full buffers, the acceptable outputs are those connected to switches that are farther away from the destination. In such a switch design, the packet at the head of a buffer is misrouted only if all the preferred outputs are busy and the buffer is full. With this scheme, no packets can be blocked and deadlock is prevented [Kons90a]. The same two-phase arbiter design can also be used to support minimal routing if there is a significant difference in the cost of multiple outputs connected to switches that are one hop closer to the destination.

6.2. Arbiter Design Supporting High-Priority Traffic

As discussed in Section 2.3.2, communication switches for interconnection networks may have to support high-priority traffic for real-time applications. Various buffer designs supporting high-priority traffic have been studied, and it has been shown that a separate queue for high-priority packets in each input buffer is the most cost-effective scheme [Tami88b, Tami92b]. To support this buffering scheme, the arbiter must be able to arbitrate high-priority requests as well as regular requests. This section describes arbiter designs which support high-priority traffic.

The two-phase arbiter design we described in Section 6.1 can be used to support high-priority traffic. In the first phase, only the requests from high-priority packets are arbitrated, and in the second phase the requests from regular packets are arbitrated. The operation of the arbitration for the high priority requests in the first phase is the conventional FIFO arbitration. In each row, only one high-priority request is asserted since there is only one high-priority queue in each buffer. The second phase is the symmetric crossbar arbitration. The wrapped wave front arbiter can be used here for both phases since the conventional FIFO arbitration is a special case of the symmetric crossbar arbitration. The interface circuit we described in Section 6.1 keeps track of the row-busy and column-busy information. This is to ensure that the regular requests that are in the same rows and columns with those high-priority requests granted in the first phase cannot join in the second phase.

An alternative to the two-phase arbitration is to overlap the arbitration of the high-priority requests and the regular requests in a single phase. This can be done in an arbiter where each arbitration cell performs two independent functions: it is a cell in a FIFO arbitration operation and it is a cell in a wave front arbitration operation. Figure 6.7 shows such an arbiter. The arbitration wave for the high-priority requests starts from the top row moving vertically toward the bottom row. The arbitration wave for the regular requests starts from the cell at the left top

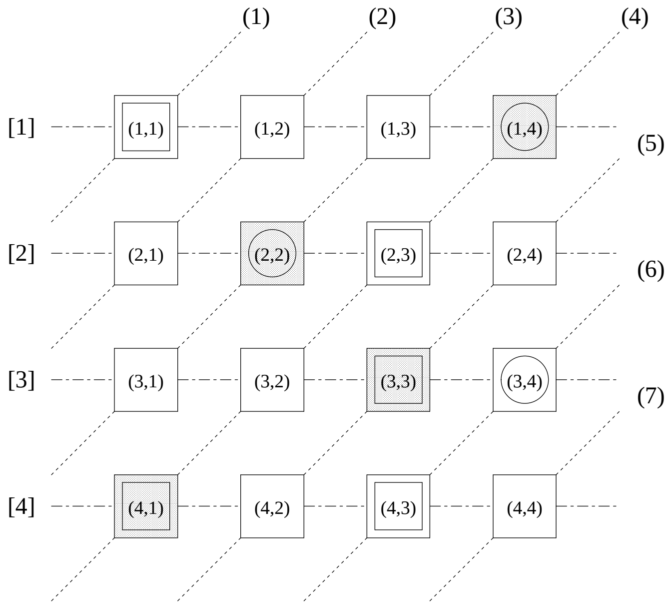

Figure 6.7: An arbiter supporting high-priority traffic. A cell with a circle and a square indicates that the high-priority request for the corresponding crosspoint is asserted. A cell with double squares indicates that the regular request for the corresponding crosspoint is asserted. There is no request asserted for the cell with one square. The grants are given to those shaded cells after the arbitration is complete.

corner moving diagonally toward the cell at the right bottom corner. For each cell, the arbitration wave for the high-priority requests must traverse the cell before the arbitration wave for the regular requests.

The arbiter that performs the combined high-priority and regular arbitration is a combinational circuit consisting of an $n \times n$ array of arbitration cells. Each cell receives two request signals from the corresponding buffer. One is the regular request and the other is the high-priority request. Since there is only one high-priority queue from each buffer, no more than one high-priority request is asserted in each row. Interface circuitry between the buffers and the arbiter must block any regular request requiring a column for which there are are high-priority requests.

This can be implemented by a wired-OR line, one per column, in the interface circuit. For each column, the interface circuit blocks the regular requests if the wired-OR line is pulled down by any asserted high-priority request signal. This wired-OR line is a dynamic circuit, and thus the delay should be much smaller than the delay for arbitrating a cell. In the arbiter, a broadcast line for each row is used to prevent the regular requests from being granted if the high-priority request in the same row is granted. This broadcast line is the output of a NOR gate whose inputs are the high-priority grant signals from all the cells in the row. If this line is 0, it indicates the row has been granted to a high-priority request and is not available to any regular requests.

The influence of the high-priority arbitration on the regular arbitration is through the wired-OR lines of the interface circuit and the row-busy broadcast lines. Assuming the cell delay is T time units, the arbitration result for the high-priority requests for the top row is stable after the first T time units. Since the delay of the wired-OR line of the interface circuits is smaller than the cell delay, the regular requests to the cells are all stable after the first T time units. If the delay for the row-busy broadcast line is ignored, all the inputs to the regular arbitration of the first diagonal are stable after the first T time units. Hence, the arbitration of the first diagonal is complete after the first $2T$ time units. The high-priority arbitration result of row i is stable after the first iT time units, and thus the arbitration of diagonal i is complete after $(i+1)T$ time units. Since there are $2n-1$ diagonals in the array, the arbitration delay of the entire array is $2nT$ time units. If the delay of the row-busy broadcast line is taken into account and this delay is δ time units, then the arbitration of the first diagonal is complete after the first $2T+\delta$ time units. The arbitration of diagonal i is complete after the first $(i+1)T+\delta$ time units. Hence, the arbitration delay of the entire array becomes $2nT+\delta$ time units.

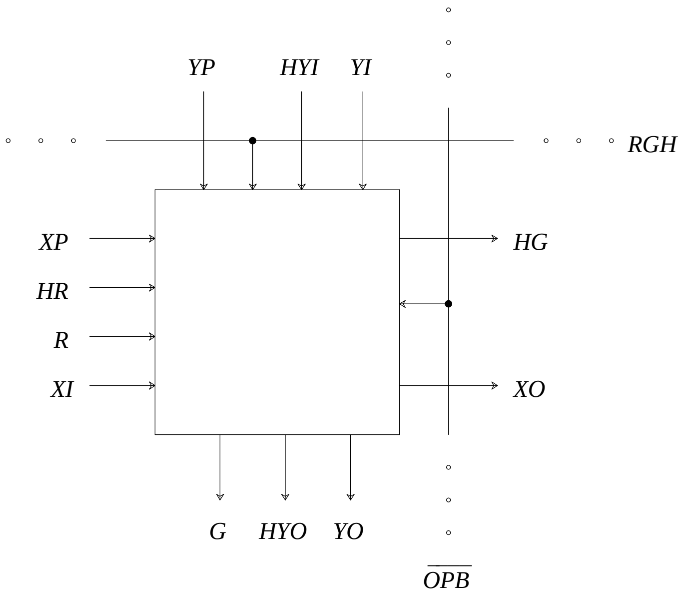

Figure 6.8: The arbitration cell supporting high-priority traffic.

Figure 6.8 shows the configuration of an arbitration cell for this design. R is the regular request signal from the corresponding regular queue of the buffer. HR is the request signal from the high-priority queue of the the buffer. If HR is 1, then R has to be 0. If HR is 0, then R can be either 1 or 0. There is only one grant signal G needed since the buffer is able to determine which request the grant is for. When HR is 1, G indicates if the high priority request is granted. When R is 1, G indicates if the regular request is granted. XP and YP indicate if the cell has the top priority. HYI and HYO are the propagation input and output signals in the high-priority arbitration. XI and XO are the propagation input and output signals for the row in the regular arbitration. YI and YO are the propagation input and output signals for the column in the regular arbitration. $\overline{OPB}$ is the signal that indicates if the output port requested is blocked. HG is the high-priority grant which is fed to a NOR gate for generating the broadcast signal. RGH is the broadcast signal that indicates if the row has been granted to a high-priority request. If RGH is 1, the high-priority request in the row is not granted and the row is available to the regular requests.

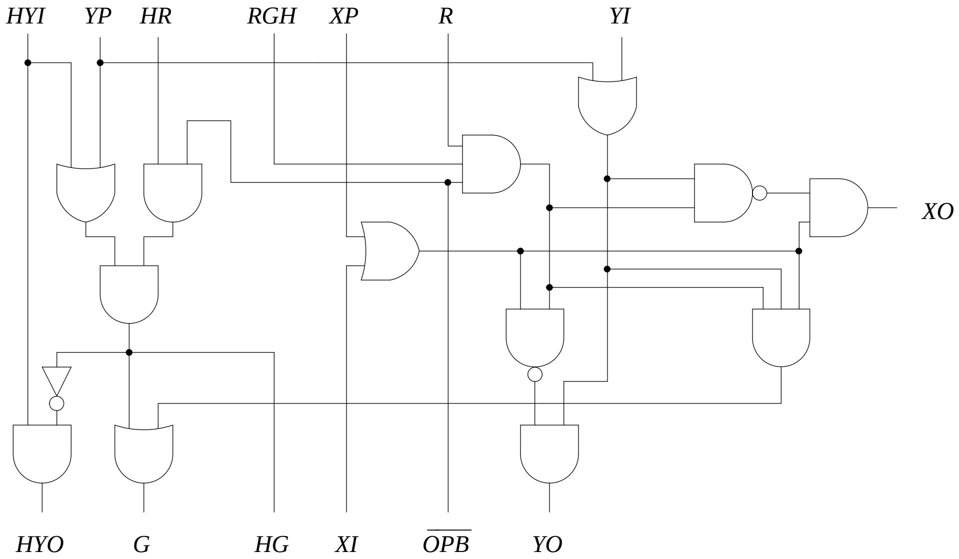

Figure 6.9: Logic design of the arbitration cell supporting high-priority traffic.

Figure 6.9 shows the logic design of the arbitration cell. The function of the arbitration cell can be expressed in the following logic equations:

$$HG = (HYI \vee YP) \wedge HR \wedge \overline{OPB}$$

$$HYO = HYI \wedge \overline{HG}$$

$$G = ((R \wedge RGH \wedge \overline{OPB}) \wedge (YI \vee YP) \wedge (XI \vee XP)) \vee HG$$

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

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

While the two-phase arbiter in Section 6.1 can be used here for supporting high-priority traffic, the single-phase arbiter design described above cannot be used to support the adaptive routing with two classes of requests in Section 6.1. It is possible to design a single-phase arbiter where each arbitration cell can arbitrate both preferred requests and acceptable requests in the same phase. However, unlike the single-phase arbiter for supporting high-priority traffic, the acceptable requests should not be blocked just because there are preferred requests for the same column. In the single-phase arbiter for supporting high-priority traffic, the high-priority arbitration is a FIFO arbitration. If there is any high-priority request for a particular column, it is guaranteed that there is a high-priority grant for that column. Hence, if there are any high-priority requests for a column, any regular requests for that column should be blocked. For the arbitration for adaptive routing with two classes of requests, the arbitration of the preferred requests is a symmetric crossbar arbitration. Hence, there is the possibility that there are several requests for a particular column but none of them are granted. Thus, despite the fact that there are preferred requests for a column, it ends up being available for acceptable requests. Hence, the arbitration consists of two phases, and the phase for the preferred requests has to be complete before the phase for the acceptable requests starts.

The circuit speeds for the two-phase arbiter and the single-phase arbiter are approximately the same. We assume that the arbitration cell delay in both arbiters is the same (T time units). The wired-OR delay in the interface circuit and the delay for the row-busy broadcast signal in the single-phase arbiter are small compared to the cell delay. Hence, the arbiter delay of both designs is $2nT$.

The circuit of the two-phase arbiter is simpler than that of the single-phase arbiter. The two-phase arbiter can use the same arbitration cell of the symmetric crossbar arbiter, while the arbitration cell in the single-phase arbiter is different and has more circuitry. A key disadvantage of the two-phase arbiter is the need to latch the grant signals generated in the first phase for use in the second phase. If each phase is performed in a different clock cycle, this is not a problem. In this case, the two-phase arbiter simply uses the same interface circuit that the single-level single-phase arbiter (Chapter 3) uses for maintaining the row-busy and column-busy information between cycles. If both phases of the arbiter are performed in the same cycle, the requirement to latch the results of the first phase may reduce the arbitration speed since the setup and hold time requirements of the latch must be met. The first-order approximation presented earlier indicated that the arbitration delays of the single-phase and two-phase arbiters are the same. However, the single-phase arbiter avoids the latching of signals in the middle of its operation. Hence, it is expected that the single phase design will be faster.