Router Architecture and Allocators

Lecture #9: Tuesday, April 27, 1999
Lecturer: Bill Dally
Scribe: Shu-Yi Yu, Robert Wei-Je Huang, Pranab Bhooma

Topics for this lecture:

1. Router Architecture Review
2. Acceleration
3. Engineering Issues
4. Arbiters
5. Allocators

1 Router Architecture Review

Figure 1 shows the architectural block diagram of a router, which contains the following components:

Input Ports A flit buffer and a state vector are associated with each input channel. A state vector is composed of five fields: G for global state (Routing, Active, or Idle), R for routing information, O for the assigned output port, P for the input buffer pointer indicating where the flit is stored, and C for available credit remained in the next-hop input port.

Output Ports Outgoing flits are latched and transmitted to the next-hop routers. A state vector indicating the global state (active/idle), the corresponding input channel number, and the credit of the next router can also be used for each output port.

Switch This connects the input ports to the output ports under the control of the switch allocator.

Virtual Channel Allocator VC allocator arbitrates the requests from each input packet competing for the output channels. The requests from each packet contain a set of virtual channels that are allowed to use for the adaptive routing. The VC allocation is done once for each packet.

Switch Allocator SW allocator arbitrates the requests from each input port bidding for the connections to the output ports and controls the state of the switch. Unlike the VC allocation, the state of the switch is set for the duration of a single flit.

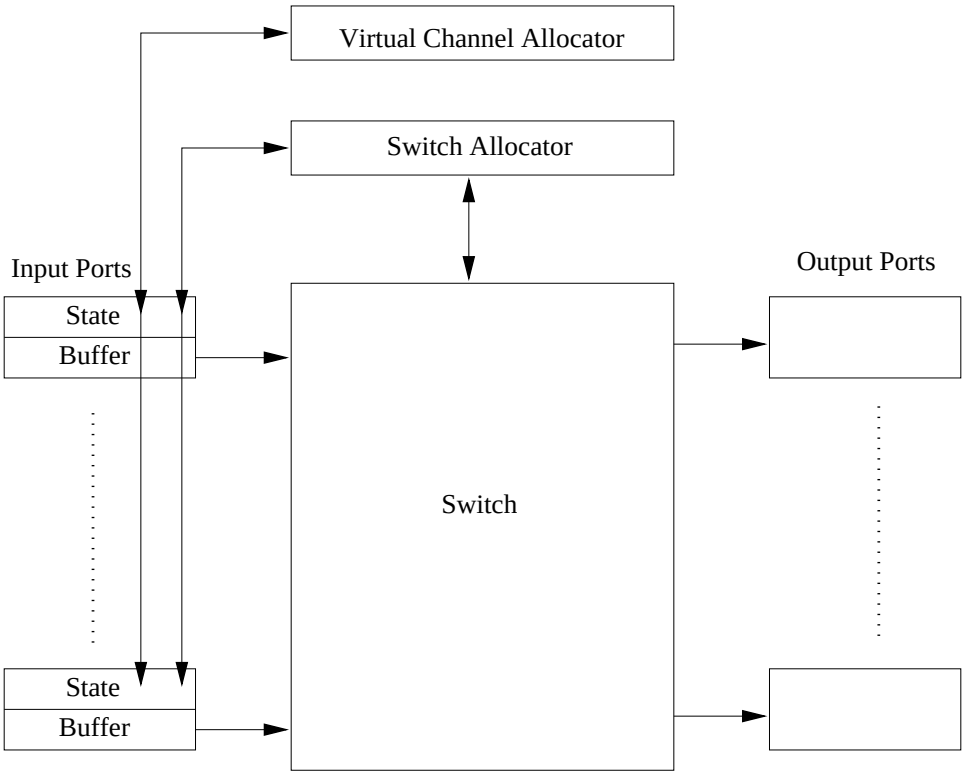

Figure 1: Block diagram of the router architecture

1.1 Pipeline Stages

Figure 2 illustrates the timing diagram of a router, assuming that flit 1(the header flit) arrives at the beginning of the first cycle, followed by flit 2(a payload flit) one cycle later. We can pipeline the flow of the header flit into 4 cycles as follows:

VC Allocation (VC) At this stage, the input packet sends a request bidding the set of output channels for the purpose of adaptive routing. The VC allocator then arbitrates the requests and assigned a output VC for the requestors. At the end of this cycle, if a channel is assigned to the requesting packet successfully, the G field of the input state vector will switch from R(routing) to A(active), and the O field will get the assigned output channel number.

SW Allocation (SA) The state of the switch is scheduled in this cycle according to the requests from input ports bidding for output ports.

Switching After the 2 resources are properly assigned, the flit can now pass through the switch and go to the corresponding output port.

Transmission Now the flit is in the output port. It can be transmitted through the outgoing wire of the port to the next-hop router.

For the payload flits arriving sequentially in order, VC allocation is not needed because the output channel is assigned for the whole packet according to the request of the header flit. However, we have to keep the serial order of the flits in a packet during transmission for the purpose of routing in next hops. Therefore, in figure 2, the second flit has to wait for the completion of the SA stage of the first flit before bidding for the switch allocation.

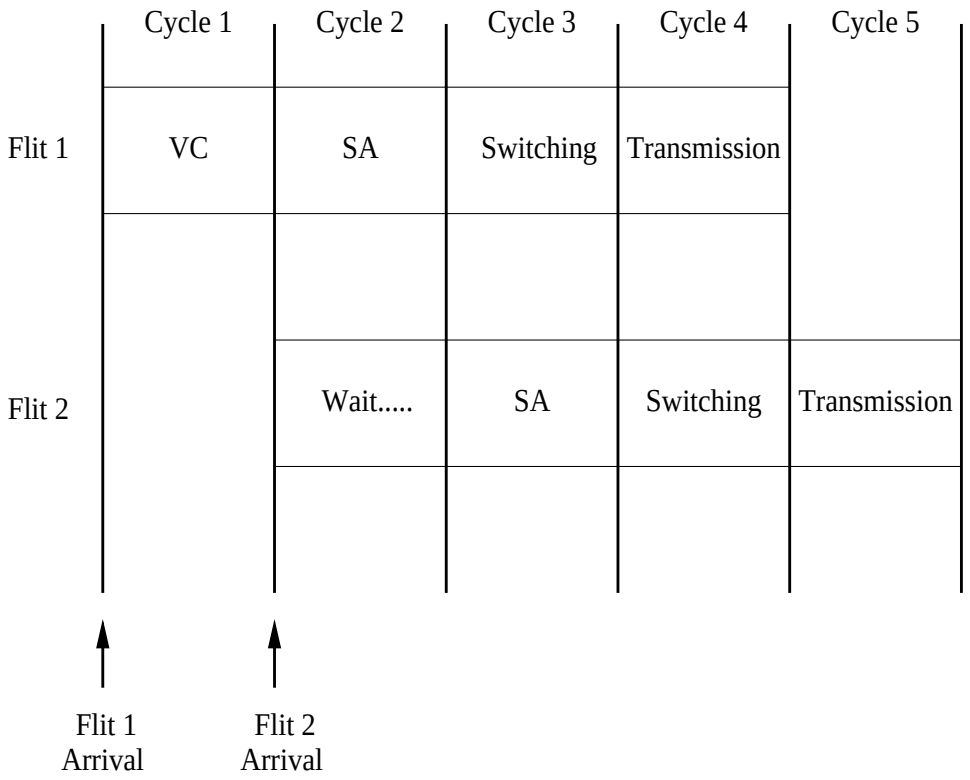

Figure 2: Timing diagram of the router

1.2 Credit Issues

In order to guarantee the sufficiency of the input buffer space of the next hop when a flit is routed, the VC allocator has to act as a gate keeper to control the flow of the flit according to the credit information sourced from the next-hop routers. Since the P field of the input state vector indicates the number of flits remaining at the input port and the C field shows the availability of the next-hop input buffer, it is required that none of the two fields are 0 before switch allocation is performed.

Once a router receives a credit from the next hop, it looks for the corresponding input channel state vector according to the record in the output state and then increments the C field. If the current flit under switching is the tail flit, the G field of the input state goes from A(active) to I(idle) at the next cycle, and the input channel field of the output state becomes null since the input/output connection is now broken. However, the G field in

the output state vector remains at A until it receives the credit indicating the emptiness of the input buffer at the next hop. The router can thus perform the credit-based flow control under this mechanism.

2 Acceleration: Optimistic Approach

The pipeline of the router shown in the previous section has a latency of at least 4 cycles for each flit. For the high-speed routers, it is necessary to use aggressive pipelining methods to reduce the latency as much as possible. An optimistic approach is to speculatively execute the SW allocation (denoted as SA*) simultaneously with VC allocation.

The idea of this optimistic control can be best described as a "don't worry, be happy" approach. Since the duty factors of the ports are low for most of the multiprocessor applications, this speculative method can keep the switch busier under light traffic and improve the router latency. Note that the throughput is unchanged, and this approach can burn more power.

In order to avoid the undesired waste of resource in the case of inconsistent allocations of VC and SA* during heavy traffic, a lower priority is assigned to the speculative switch allocation result. In this way, if a flit gets the output port 2 from the speculative SA* but is not assigned to any channel of that port by VC at the same cycle, this flit will follow the result of VC and perform the SA again at the next cycle. Also, the discarded SA* result will not occupy the resource, and the output port 2 can be used for other higher-priority allocations if someone else is also bidding for this port.

For an even faster design, we can also move the actual switching stage to the same cycle with VC and SA*. Again, this speculative execution should be assigned a lower priority to keep the resource utilization high.

3 Engineering Issues

The critical design issues that affect the router performance are,

1. Buffers
2. Queues vs.Virtual Channels
3. Switches
4. State Logic

3.1 Buffers

Packets that cannot be immediately forwarded to an output port need to be buffered within the switch. There are three basic conditions where buffering is neccesary:

1. The output port through which the packet needs to be routed is blocked by the next stage of the network.

2. Two packets destined for the same output port arrive simultaneously at different input ports but the output port can accept only one packet at a time.

3. The packet needs to be held while the router in the switch determines the output port through which the packet is to be routed.

Buffering has two architectural parameters relating to the placement and organization of buffers (queues). Input buffering and output buffering are two alternatives where queues are placed at the input and output buffers respectively.

In general, buffer organization can be either *central buffer* (one big buffer) or *separate buffers*:

Central Buffer There is one big buffer and all the channels are connected to this single buffer as shown in figure 3. It is good in the management of dynamically allocating the available buffer space to arrving packets. But the disadvantage is that a single congested channel may hog all the available buffer space.

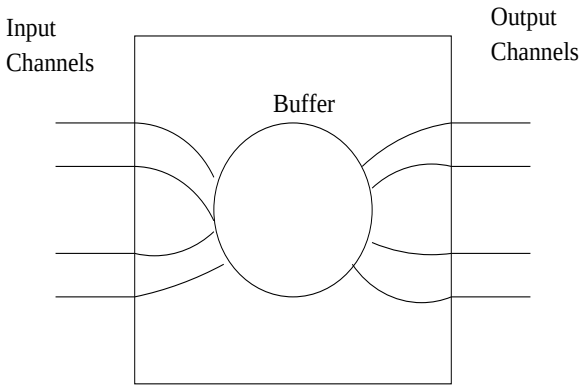

Figure 3: Central buffer

The bandwidth of the interconnection between the buffer and the channels should be sum of the bandwidth all the channels, else an overhead can be experienced.

Separate Buffers There are separate buffers to each separate Channel as shown in figure 4. The advantage of the Central Buffer is reduction in space, but it has lot of disadvantages like increase in the Bandwidth, Overhead, Speed.

3.2 Queues vs Virtual Channels

Virtual Channels can be organized as *One Big Queue*, *Static Queues*, or *Dynamic Queues*.

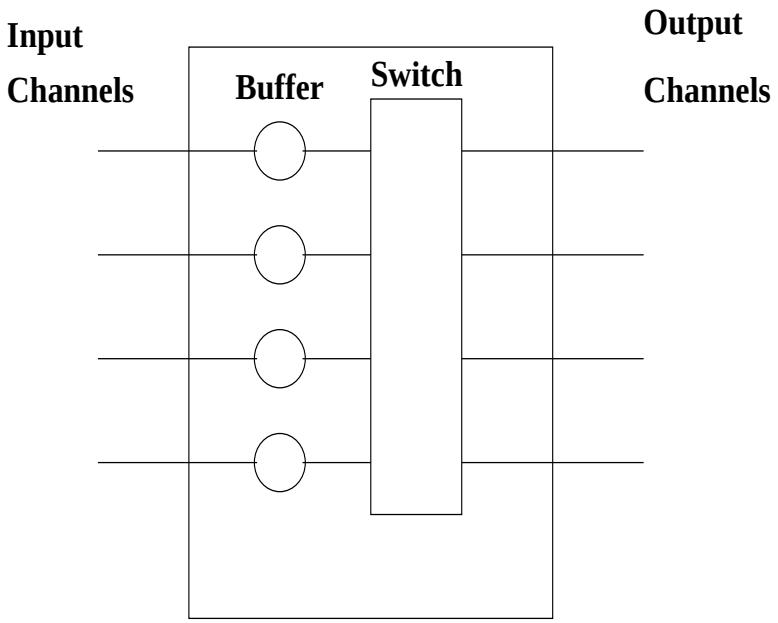

Figure 4: Separate buffers

One Big Queue In this structure all the Queues are placed in Fifo fashion. So any Queue which comes first is the first to get routed. (See figure 5)

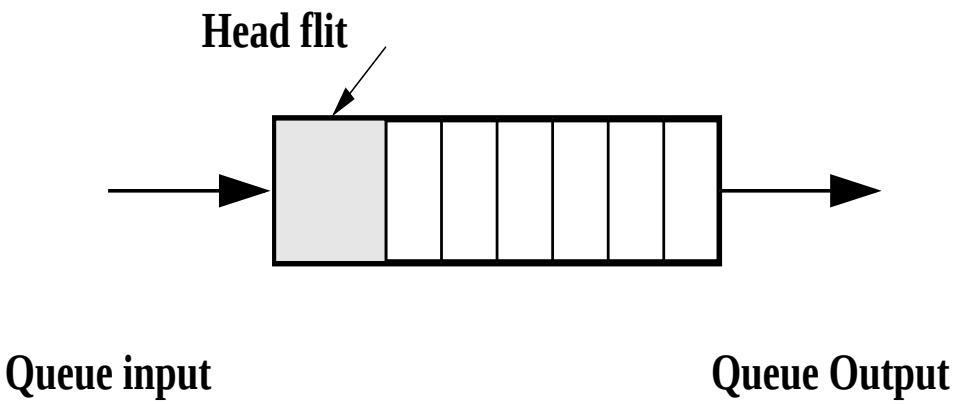

Figure 5: One big queue organization for VCs

The advantage of one big buffer is that only one packet at a time arrives at the input port so that the buffer needs only a single write port. Furthermore, if the buffer is managed as a FIFO queue, it is ideal to deal with variable length packets and avoid the memory management problems. The disadvantage with the FIFO buffers is that a packet that does not get its desired output port can block the all the other Packets.

Static Queues In this structure all the Queues have fixed length buffer allocated for

them(See figure 6).The advantage is uses less resource management as all the buffers are of equal length, and it is also faster as the credits are not required to be passed back.

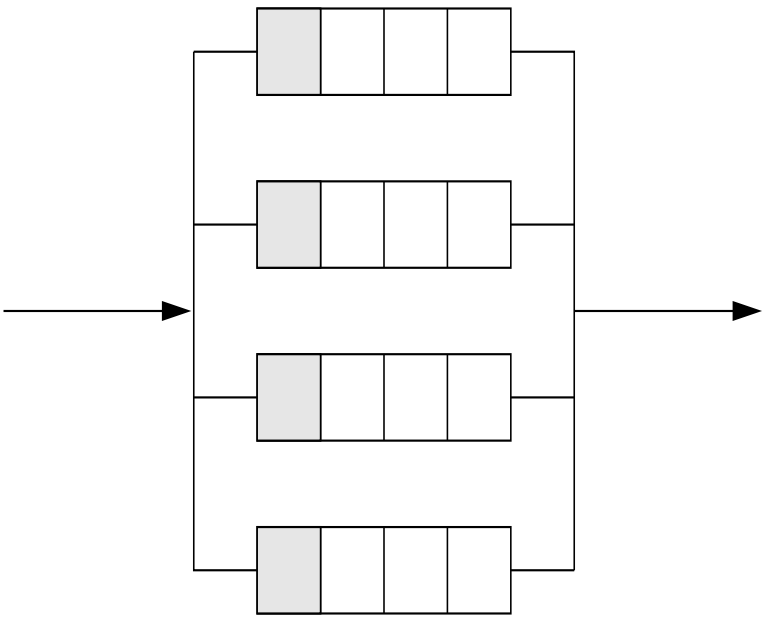

Figure 6: Static queue organization for VCs

Dynamic Queues In this structure all the Queues have variable length Queues(see figure 7). The advantage is it has a very good buffer management, although it comes at the cost of speed due to credit management and needs extra logic to implement the buffer management.

3.3 Switches

Cross Bars vs Buses: Switch is needed in a router so that input channels can be routed to the desired output channels.

CrossBar Switch All the input Channels have a point to point connection to the output channel(see figure 8). Advantage is fast, and reducing latency. Disadvantage is it needs more wires to implement.

Bus Switch A Bus can imitate a Cross Bar Switch by buffering all the input channels (N bit wide) with 3N bit wide Registers and the output of the Registers arbitrate the Bus with 3N bit wide Ports(see figure 9). It needs extra buffer space, increases latency, and requires 3N Bandwidth. Practically it is not feasible for the router switch.

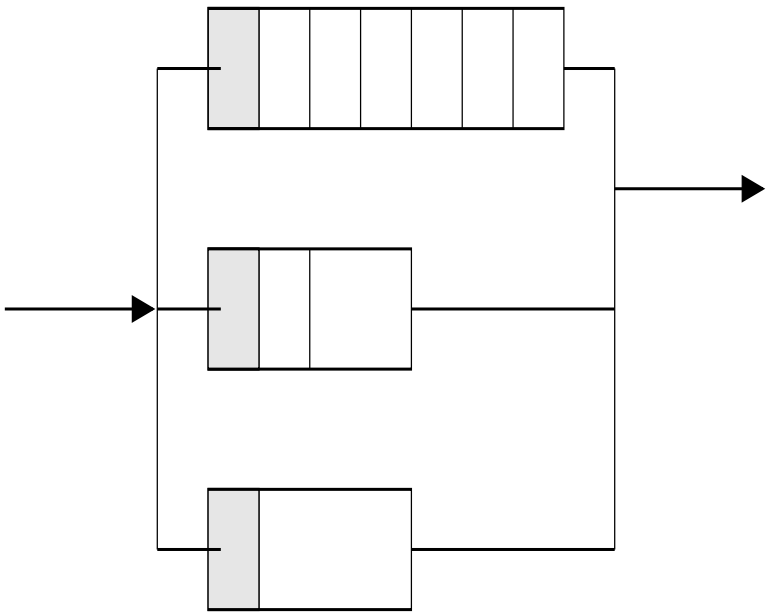

Figure 7: Dynamic queue organization for VCs

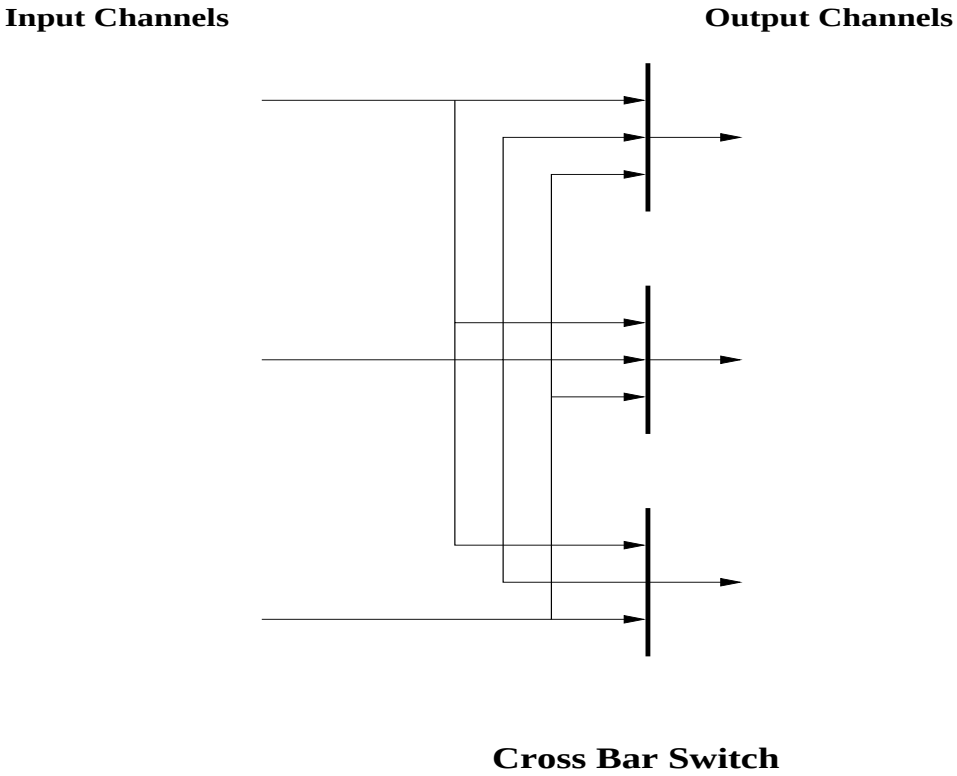

Figure 8: 3x3 Crossbar Switch

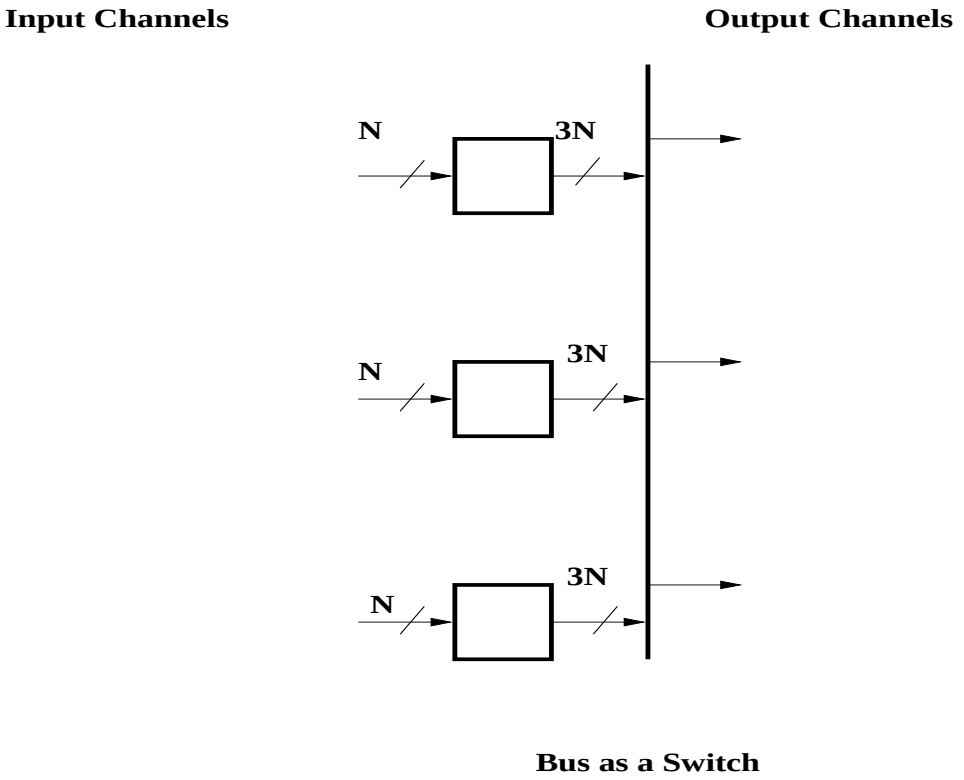

Figure 9: 3x3 Bus Switch

3.4 State Logic

The various methods by which a Routing function can be used to get Virtual Channels are: *Table*, *Algorithm*, or *One Virtual Channel*.

Table In the Table method, a lookup table is being used to get a set of Virtual Channels.

Issues: Lot of buffer space, Slow Speed, Easy to Change the table.

Algorithm In this method, a routing function is implemented in a circuit to allocate the Virtual Channels.

Issues: Less space, Fast Speed, Tough to change.

One Virtual Channel This true only for source routing, as the entire route is predetermined at the Source Node.

Issues: Waste of Resources if the Virtual Channel is not being used quite often.

4 Arbiter

Suppose there are N requestors and only one resource. The question of arbitration is about how to decide which requestor gets the resource. An example of arbitration is

shown at figure 10. (Also see Figure 2.5 on Prof. Dally's lecture note chapter 2.) In this arbitrator, each requestor has different priority, and requestor 0 gets the highest one. The problem of the arbitrator is that it lacks fairness. Fairness could mean that each request will eventually be served; each requestor gets equal chance; first come first served; bounded wait time; and last used last served.

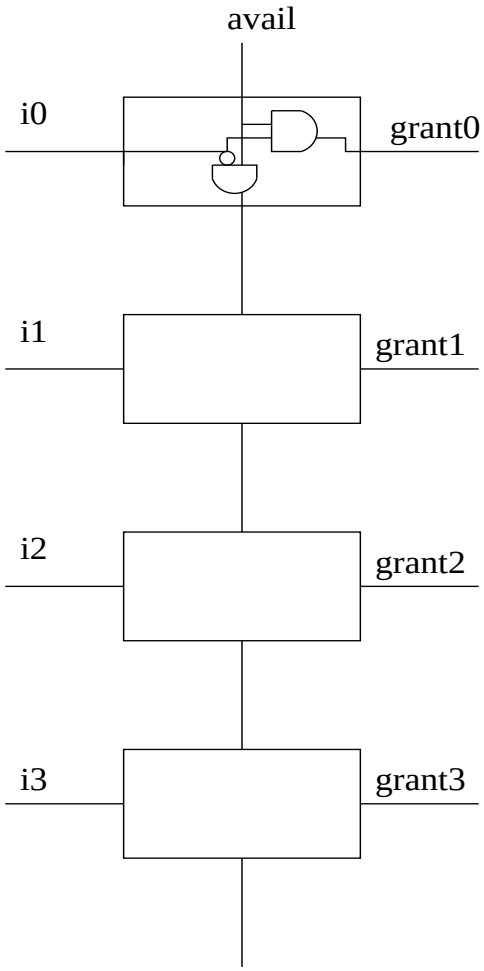

Figure 10: A simple arbiter

4.1 Round-Robin Arbiter

A simple way to make the arbitration fair is shown as figure 11. Similar to figure 10, but now we add switches between the stages. When requestor i gets the resource, requestor i+1 gets the token for priority. For example, after requestor 2 gets the resource, the original opened switch is closed and mf2 is open, then requestor 3 gets the highest priority.

The drawback of Round-Robin arbiter can be shown in the following example. Suppose there is a series of requests 0, 2, 1, and 0. Requestor 2 doesn't release the resource

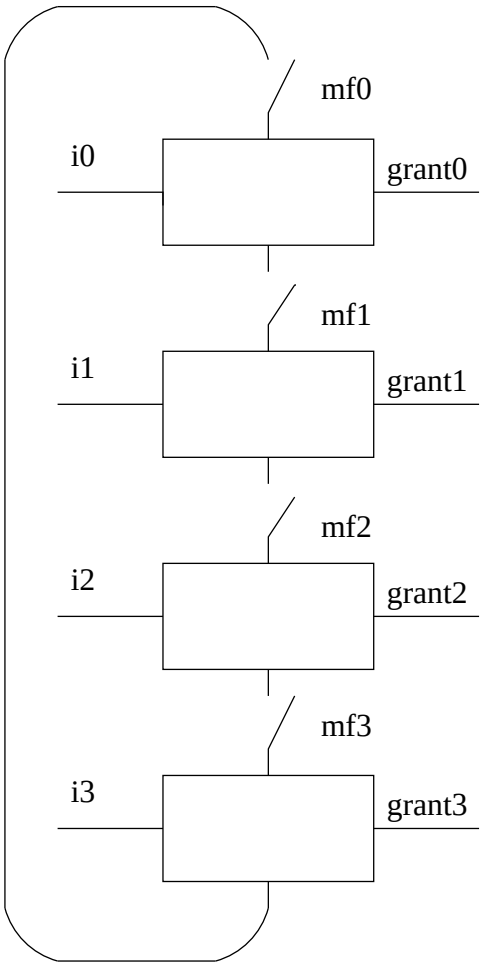

Figure 11: A round-robin arbiter

until the requests of requestor 1 and 0 come. For the first-come-first-served rule, requestor 1 asks first therefore it needs to be served first. For the last-used-last-served rule, requestor 0 asks last therefore it will be served last. However, after the use of requestor 2, mf2 is open. Therefore the order of priority is 3, 0, 1, and 2. Requestor 0 will be served first.

4.2 Matrix Arbiter

A fairer arbiter is called Matrix Arbiter. The matrix is shown as figure 12 when there are four requestors 0, 1, 2, and 3. The element at ath row and bth column means requestor a beats requestor b. Each element can actually be stored in a flip-flop. In this case we only need 6 flip-flops. Suppose initial pattern is composed of all ones. After requestor 1 gets the resource, it will clear its row and sets its column. Then the requestor will be put at the end of the lists. Now if requestor 2 and 3 request, from the matrix we know

that requestor 2 beats requestor 3. Therefore requestor 2 gets the resource.

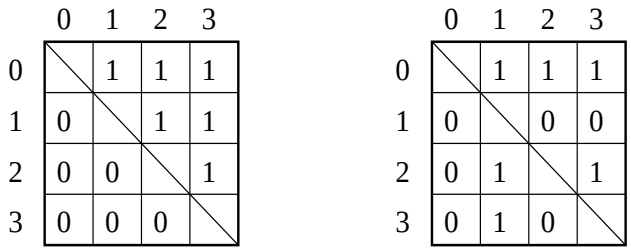

Figure 12: Matrix arbiter

This arbiter is based on the last-used-last-served rule. Let's consider the example in the previous section. The series of requests is 0, 2, 1, and 0. After requestor 2 finishes the task, because requestor 0 used the resource last, requestor 1 will be served next.

4.3 Queuing Arbiter

Queuing arbiter is based on the first-come-first-served rule. In the queuing arbiter, there is a ticket counter and a served counter. The ticket implies the position of the request in the queue. The ticket counter stored the ticket number, and when one request comes, it distributes the ticket to the requestor and increments its stored number. The served counter stores the ticket number of the requestor that is being served. When the service is finished, the served counter increments its value, and then the requestor which holds the ticket of that value gets the service.

For example, in figure 13, requestor 3 is being served and the served number is 2. Requestor 0 asks for the service, then the ticket counter gives it 3, and the ticket counter is incremented into 4. When the service is finished, the served number is incremented into 3. Then requestor 0, which holds the ticket number 3, will be served.

5 Allocation

The allocation problem can be in two forms. One is that each requestor asks for one of several singleton subsets of resources. The other is that the requestor asks for a coupled set of resources, such as a set of input and output virtual channels. We are going to show that these two problems are actually the same.

As shown in the figure 14, the allocator has VC1-8 in the direction +x, -y, -x, and +y respectively. Suppose input from VC1 wants to go to VC1, 2, and 4; input from VC2 wants to go to VC3 and 4; etc.

The problem can be solved in two ways. One way is to solve it by the greedy allocation method. One example of the bipartite matching problem is shown in figure 15. The greedy allocation gives the input VC (the requestor) with the minimal number the output

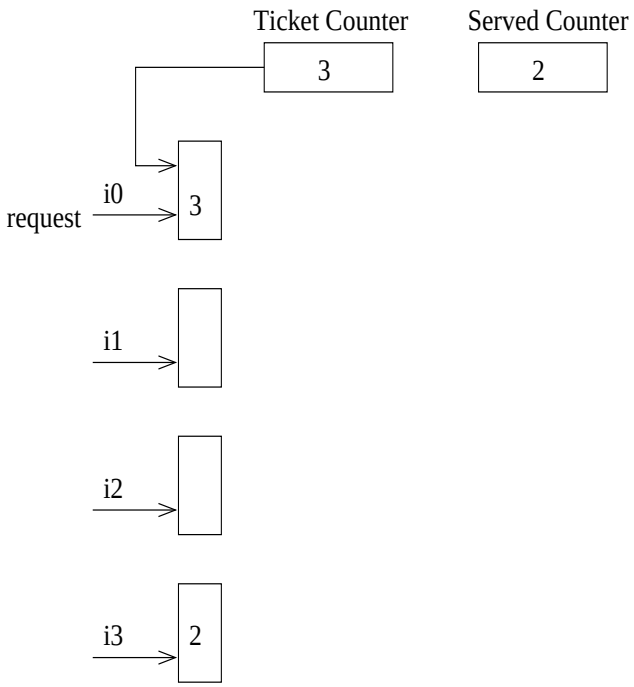

Figure 13: Queuing arbiter

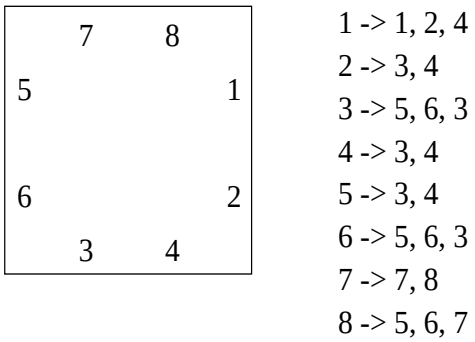

Figure 14: An allocation problem

VC (the resource) with the minimal number. In the shown figure, input VC1 gets output VC1, input VC2 gets output VC3, and so on.

The other way is to solve it by the exhaustively searching method. As shown in figure 16, each row represents an input virtual channel and each column represents an output virtual channel. An X in the ith row and the jth column represents the case when VCi makes a request for VCj. The exhaustive searching method is to find every X which is only owned a row or a column first. When row i only has one X,) it means that VCi only sends request to a single VC. (Let's call it VCj.) To achieve maximal match, this X has to be selected. After selecting the X, since the problem of allocating input from VCi is solved, row i can be removed from the matrix. In addition, since VCj is allocated

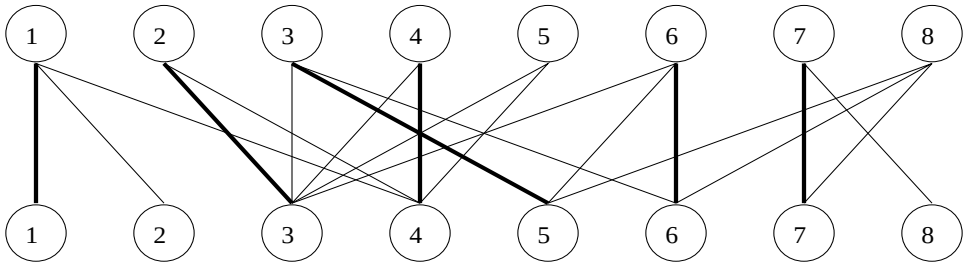

Figure 15: Bipartite matching problem

for VCi, column j can also be removed. By reducing the matrix into a smaller one and repeating the algorithm stated above, the problem can be solved.

For example, column8 only has an X, which is at the cross with row7. This X has to be selected. Then the problem of allocating VC8 to VC7 is solved. Now row7 and column8 can be removed and the matrix can be reduced into a smaller one.

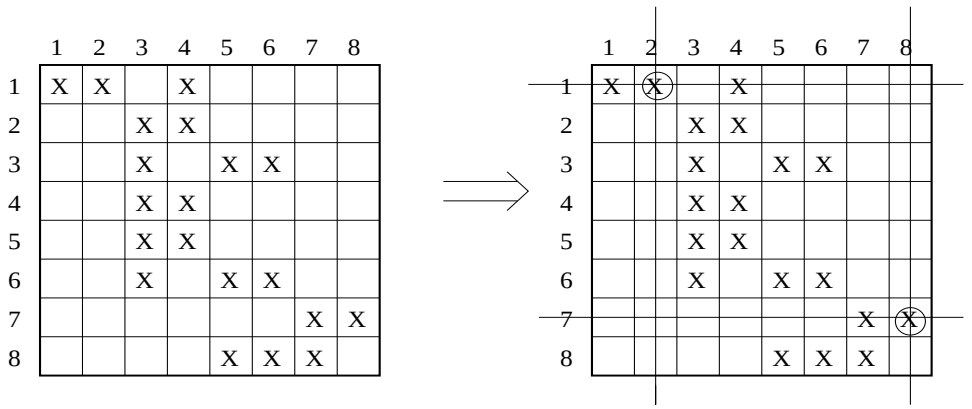

Figure 16: Exhaustive search matrix