Architecture of a wormhole router with virtual channels

In general a router can be divided into a datapath and the control section. Blocks descriptions below:

1. Input units: buffer incoming flits and store the state information for each input channel
2. Output units: outgoing flits are received from the switch and then forwarded to the downstream router. Contains state information indicating status of the output channels
3. VC allocator: arbitrates between bids for output VCs by the incoming head flits. Can place input port state registers here.
4. Switch allocator: arbitrates switching requests to connect input ports to output ports and schedules the Switch
5. Switch: typically an NxN crossbar. Connects the inputs to the outputs based on the configuration determined by the switch allocator

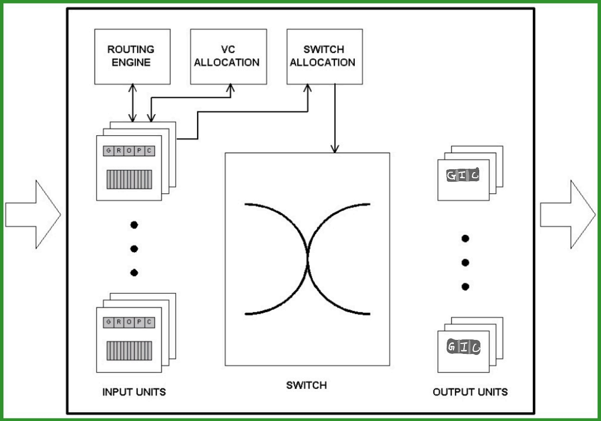


Figure 1: Block Diagram of Wormhole Router

Functionality Overview

Consider the processing of a 3-flit packet arriving at above router input.

[G] => (RT)
Next Buffer Pointer
[P] => Buffer{H}
Buffer = {H}

Cycle 0: head flit arrives, routing engine checks the header to obtain a set of virtual channels that can be used for this packet based on its internal routing relations, and the global state [G] is set to routing (RT) and the Next Buffer Pointer is updated to point to this flit

[G] => (VC)
[R] => {R}
[P] => Buffer{H}
Buffer = {H, B}

Cycle 1: head flit completes routing stage, set of VC's obtained and stored in the Routes [R] field, VC allocator arbitrates for an available and matching VC. Body flit arrives at the cycle and is stalled because VC allocation is pending - VC needs to be allocated before head can bid for switch allocation. Global state is set to VC allocation (VC)

[G] => (A)
[R] => {R}
[O] => VID_N
[P] => Buffer{H}
[C] => non-zero
Buffer = {H, B, T}
—outputs units—
[O] => (B)

Cycle 2:
- output VC allocated to packet - ID stored in actual output (O) VC field
- head flit bids for switch allocation: if the credit field is non-zero, switch allocator performs scheduling for switch bandwidth to transfer the flits
- body flit continues to stall waiting for the header flit to complete switch allocation. Tail flit arrives and is also stalled. Stalls are needed to prevent misordered flits
- Global state is set to active (A), and output unit state is set to busy (B)

[G] => (A)
[R] => {R}
[O] => VID_N
[P] => Buffer{B}
[C] => decrement 1
Buffer = {B, T}
—output unit—
[O] => (B)

Cycle 3:
- path has been allocated through the switch and head flit is transferred to output port
- next buffer pointer is updated to point to the body flit and credit count is decremented
- body flit bids for switch bandwidth and tail continues to wait

[G] => (A)
[R] => {R}
[O] => VID_N
[P] => Buffer{T}
[C] => decrement 1
Buffer = {T}
—output unit—
[O] => (B)

Cycle 4:
- path has been allocated through the switch and the body flit is transferred to the output port
- next buffer pointer is updated to point to the tail flit and credit count is decremented
- tail flit bids for switch bandwidth

[G] => (I or RT)
[R] => {R}
[O] => VID_N
[P]. => Buffer{null}
[C] => decrement 1
Buffer = {}
—output unit—
[O] => (B)

Cycle 5:
- path has been allocated through the switch and the tail flit is transferred to the output port
- VC can be deallocated at this point and global state returns to Idle (I); or a more conservative approach would be to hold the contents and VID until downstream router sends an ACK
  - if this was the approach, then the buffer would be Buffer = {H, B, T} until the ACK
  - additionally, global state would be [G] = (RT) until the ACK
- If the VC is deallocated, then the credit count returns to max

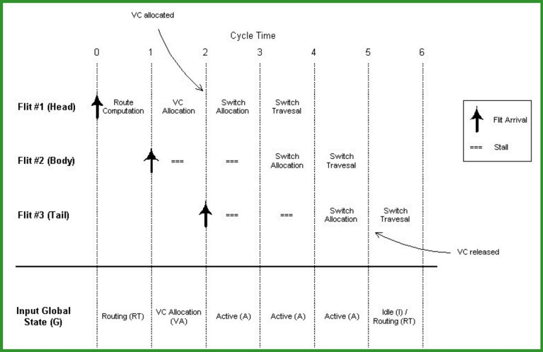


Figure 2: Router Pipeline Diagram

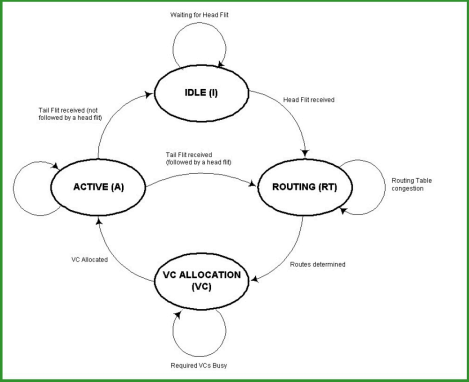


Figure 3: State Diagram for G

Router Structure

Input unit:

- Router contains a series of input ports for which multiple virtual channels can be associated
- Each VC has a corresponding buffer unit and state table to indicate its status
- State table contains the following fields
  - [G] = global state. Cycles between 4 states: Idle, Routing, VC allocation, and Active
    - Idle : no new flits
    - Routing : route computation. Destination is determined form the head flit by the router to produce a route (set of routes for adaptive routing). There can be stalls at this stage if the routing table is shared and >1 head arrives simultaneously
    - VC allocation : virtual channels is being allocated. Stalling occurs if VC's are not available
    - Active : VC bids for the switch. Remains in this stage until the tail flit is done with switch allocation
  - [R] = route storage register(s). Stores VC or VC's if adaptive routing is used.
  - [O] = actual output VC that the buffer has been granted
  - [P] = contains next buffer pointer and last buffer pointer. Next buffer pointer is used to retrieve next flit for transmission through the switch to the output port. Also used to determine buffer occupancy
    e.g. Next Buffer Ptr - Last Buffer Ptr = buffer occupancy
  - [C] = contains the count of empty buffers on the downstream router for the VC. Switch bidding only occurs if the credits > 0.

Output unit:

- Each VC has a corresponding output unit and a state table to indicate its status
- State table contains the following fields
  - [G] = global state. Cycles between 2 states: Idle and Busy. Used by the VC allocator to know which channels are free
  - [I] = input channel matched to the respective output virtual channel. This connection is needed so the credits from the destination router can be by directed to the appropriate input unit. Output unit controls the switching/forwarding of the credit value to the input unit
  - [C] = contains the count of empty buffers on the downstream router for the VC of the respective output port.

- In order to control credit forwarding, there is another smaller crossbar switch in the reverse direction separate from the main crossbar; credits are buffered at the output and then switched to the respective input unit
- General architectural note here is that delays in forwarding the credit value back to the input unit is more costly to the router latency compared to overprovisioning the reverse switch's bandwidth/performance is to area/cost

Stalls

- Depending on the architecture of the router (whether or not it employs node based routing, virtual channels, etc. the routing pipeline may have fewer or greater stages)
- Many stages may be subject to stalling
- Header stalls
  - Routing stall: if each input does not have dedicated routing hardware, and more than one head flit arrives simultaneously, this could result in a structural conflict or hazard which will cause the losing flit to stall equal to the time until it gets served
  - VA stall: if all of the VC's corresponding to the desired output port are occupied, a stall will occur until it gets served
- General stalls
  - SA stall: stalls when the flit does not succeed in switch bandwidth allocation
  - Credit stall: occurs if the buffer space at the input is underprovisioned

Credit Stall Dynamics

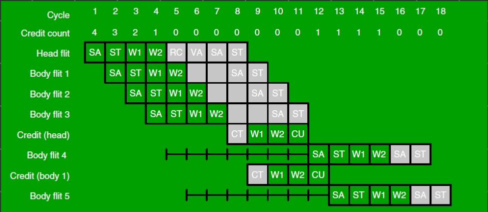
Credits are sent from the destination router to the source router output unit, but at what point are credits returned to the destination router? And how is this timing consider such that the buffer is sufficiently sized and credit stalls are avoided?

The buffer occupied by a certain flit will be freed in the next clock cycle after the switch has been allocated for that flit. Only until then can we send the credit back. This delay must be incorporated into buffer sizing.

Let's examine penalties in a case where there are no pipeline delays (i.e. streamlined service of the flit). Pipeline diagram shown below.

=> Here the buffer is given baseline sizing of 4 based on the RT delay of 2 cycles. As a result the input credits go to 0 after the first 4 flits are sent.
=> It takes another 4 cycles until the flit makes it to switch traversal (ST) in the downstream router and then a credit is finally issued.
=> At cycle 11, the credit finally reaches the upstream input unit (CU), and the next flit can be served on the next cycle.
=> So the number of cycles with optimistic assumptions for no pipeline delays, minor RT delay, credit generation, and traversal to upstream input unit is 11
E.g. T-cycles = 4(pipeline) + 4(RTT) + 2(CT,U) + 1(startNextSA) = 11

Duty factor is this example is 4/11 active cycles



Cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Credit count	4	3	2	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0
Head flit	SA	ST	W1	W2	RC	VA	SA	ST										
Body flit 1		SA	ST	W1	W2			SA	ST									
Body flit 2			SA	ST	W1	W2			SA	ST								
Body flit 3				SA	ST	W1	W2			SA	ST							
Credit (head)								CT	W1	W2	CU							
Body flit 4												SA	ST	W1	W2	SA	ST	
Credit (body 1)									CT	W1	W2	CU						
Body flit 5													SA	ST	W1	W2	SA	ST

Figure 16.6 Credit stall: a buffer is allocated to the head flit when it is in the upstream SA stage in cycle 1. This buffer cannot be reassigned to another flit until after the head flit leaves the downstream SA stage, freeing the buffer, *and* a credit reflecting the free buffer propagates back to the upstream credit update (CU) stage in cycle 11. Body flit 4 uses this credit to enter the SA stage in cycle 12.

Some designs find this acceptable as the output bandwidth can still be utilized by saturating with traffic from other VC's.

But this can be credit stall can be avoided by appropriately right sizing the input unit buffers to 11.

Finally even after avoiding credit stalls, we can still incur stalls if the body flits following the header flit simply do not arrive in time to be served after the header due to congestion from other traffic, i.e. just not enough data in the VC to be readily served after resources have already been allocated

Flit Format

Flit format specification is motivation by how crediting can be piggybacked on flits.

One approach would be to have the flit span multiple clock cycles such that a VID and other fields don't take up too much flit overhead.

Consider a router with 15 virtual channels. VID field needs to be 4 bits. This allots one VID address to be used for an idle flit with which credits can be sent standalone. Allot another 4 bits for the credit field for clog2(buffer size =11).

Also need a CRC in both the flit header and for the entire flit. CRC for the head flit is to ensure integrity in credit tracking + the portion of the payload embedded in the header.

Finally, we need a NACK field that would be sent with the corresponding VID to indicate to the sender to retransmit the respective flit.

Flit sizing: too small of a flit would lead to increased percentage of header overhead. Too large would lead to fragmentation overhead. Fragmentation overhead is the extra work required to recreate and assemble a, in this case, flit.

Regarding sideband signals for crediting, the author considers this to be a waste of resources as this would also incur a serialization latency in addition to adding wiring overhead per port.

Tradeoff on speculative flow control vs buffer storage:
=> speculative transmission and crediting, e.g. sending flits prematurely assuming credits will be received - this method would cost a lot of storage overhead at the output unit buffers; typically this storage is reserved for rollbacks for error correction and retransmission

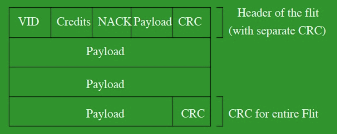


Figure 6: A sample flit format for a 4 cycle long flit

Allocation and Deallocation of Resources (Speculative and Non-Speculative Routing for pipeline improvements)

Previous discussions of allocation and deallocation have focused on the input unit, however the output unit is also subject to allocation/ deallocation processes.

=> Input VC may be deallocated as soon as the tail flit passes switch allocation.
=> Output virtual channel deallocation occurs when all credits have been received.

Consider a case in which 2 packets arrive back to back in the same input VC but are to be mapped to different output VCs. Instead of VC deallocation occurring as soon as the tail flit passes switch allocation, we wait until the credit has been received by the output channel. Pipeline diagram of the transaction shown below. We can see here there is an 11 cycle latency between the 1st packet being deallocated and subsequently the 2nd packet being allocated.

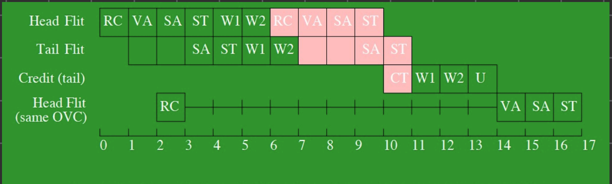


Figure 7: Virtual channel allocation after recieving credit for tail flit

In this design, it would follow that in order to keep the output port busy, we would need at least 11 VC's per output port in order to ensure one flit per buffer at a time as the design prevents more than one flit per buffer being served in a timely manner by the routing engine. This is a problem.

One modification would be to deallocate the virtual channel as soon as the tail flit passes switch allocation as discussed in the beginning of this section. This is doable because we can save the uncredited flits in the output buffer unit and rollback if a transmit is needed. This saves 10 cycles. Below shows the resulting pipeline diagram.

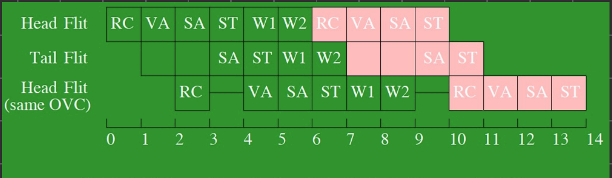


Figure 8: Virtual channel allocation after tail flit completes SA

Next we want to eliminate the idle cycle at time = 9. RC for packet 2's head flit is pending SA completion for packet 1's tail flit, but this can be pipelined because route information, e.g. output port assignment, has already been consumed by the switch allocation module, and only VID information is needed to allow the flit to traverse the switch to the respective VID output buffer. Hence packet 2's head flit can run through RC and overwrite the route information [R] of packet 1. Resulting pipeline diagram below.

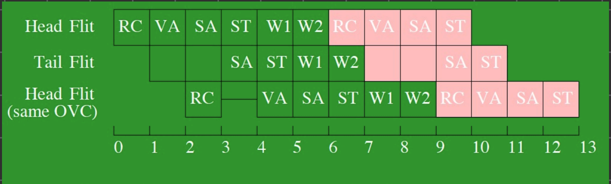


Figure 9: Route computation in parallel with switch allocation

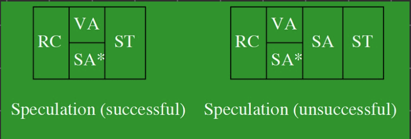


Figure 10: Speculative switch allocation

Finally we still have a stall at time = 3. This can be achieved by performing virtual channel speculation, e.g. we try to allocate bandwidth with the switch for all VID's and then we just traverse the switch through to the assigned VID the next cycle after VA has been performed simultaneously with SA. Now all stalls have been eliminated. Resulting pipeline diagram above.

In [SA], the requests will be categorized as speculative vs non-speculative. So for non-speculative requests for switch allocation, they will be given priority. For speculative requests which are an attempt to decouple the dependency between consecutive packets, they will be non-interfering with the preceding packet's switch allocation and subsequent traversal.

One more note: it is very critical that virtual channel allocation be consolidated into one cycle. This is because a successive packet will not be able to make any decisions in its early stages of VA as the previous packet is still wrapping up its VA assignment. This will render any gains for speculation useless.

Buffer Organization

Buffer organization can be done with FIFOs for each port or shared memory for the entire input controller.

Shared memory requires high bandwidth (see advancements in high bandwidth memory). But buffers can be dynamically allocated more freely either across ports or with static port allocations and dynamic virtual channel allocations.

Additionally, buffers can be organized as being input queued or output queued, but it is more practical to be input queued as state information from the next router must be maintained for virtual channel allocations (since there is no switch logic effectively decoupling these two routers).

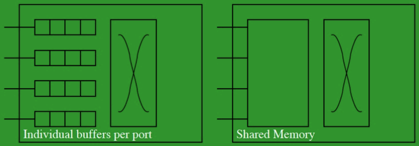


Figure 11: Buffer Architectures

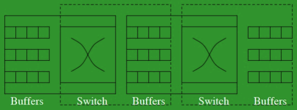


Figure 12: Possible chip boundaries