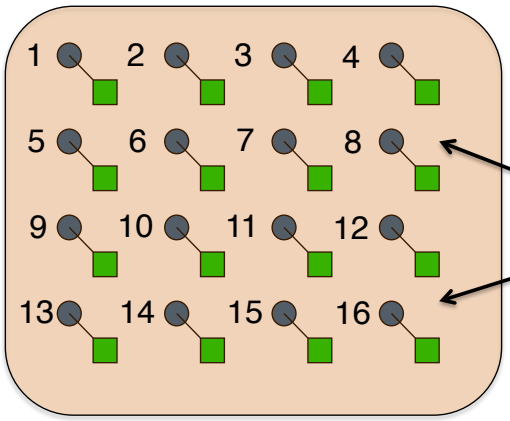
Lecture 26: Board Notes: On-chip IC NWs

PART A: Consider the following "sea" of processor cores and routers



(assume circles are processing nodes)
(assume squares are routers)

How should we connect different elements?
Can any topology be implemented on-chip?
What does a router do?
What's the overhead of traversing a router?
How do we calculate message latency?
…

Let's look inside of a router first…

Router has 2 main components:

1. Datapath:
   - Handles storage and movement of a packet's payload
   - Consists of input buffers, switch, & output buffers
2. Control
   - Logic to coordinate packet resource allocation

I'm going to talk about a **"Virtual Channel Router"** –not yet explicitly discussed…

- Virtual channel router requires extra resources (HW), but can help overcome blocking issues
  - (Remember blocking issues with wormhole routing)
  - (VC allows packets to *pass* a blocked packet and make better use of idle bandwidth)

Example:

1. Packet B enters node #1 from the network; B acquires channel *p* from node #1 → node #2
2. A 2nd packet A has entered node #1 from the wst and needs to be routed east to node #3
3. Meanwhile, B wants to leave node #2 and go south, but is blocked
4. Now channels p and q are idle .. but cannot be used
   a. Packet A is blocked in node #1
   b. It cannot acquire channel *p*
   c. B blocks

See figure:

Now, assume 2 VCs per physical channel:

1. B arrives at node #1 and acquires the bandwidth to go to channel *p*
2. A arrives from the east, B tries to leave node #2 and is blocked
3. A can use free bandwidth *p* and goto another VC on node #2
4. Can also proceed onto node #3

This is a better use of resources

- May have 1 physical channel, but more buffers

What happens during packet routing?

---

1. Let's start with a flit of a packet arriving at the input unit of a router

- Input unit consists of a flit buffers to hold arriving flits until they can be forwarded
- Input unit also maintains state of virtual channel
  - I: Idle
  - R: Routing
  - V: Waiting for virtual channel
  - A: Active
- Once packet in router, need to perform route computation to see where it goes
  - Can then go to VC for allocation

---

2. Each head flit must advance through 4 stages of routing computation

- It's pipelined! Assume…
  - RC: Routing Computation
  - VA: Virtual Channel Allocation
  - SA: Switch Allocation
  - ST: Switch Traversal

- Packet might move through like this:

	1	2	3	4	5	6	7
Head Flit	RC	VA	SA	ST			
Body Flit 1		**	####	SA	ST		
Body Flit 2			####	####	SA	ST	
Tail Flit						SA	ST

  - ** (second body flit arrives, waits its turn to traverse and leave the router…)

Important Points:

- t_r (time through a single router) does not equal 1!
  - (more like 5 or 6 at least)
- Routing and VC allocation are per packet functions
  - Nothing for body flits to do
  - With no stalls, need 3 input buffers (for 3 flits)
  - With stalls, need # of buffers = # of packets

Outlook:

- Ultimately, issues involved in routing process discussed above + router architecture + storage needed determine the bandwidth for the topology
    - Possibilities:
        - Even though you can devise a topology for ideal performance, it may not be feasible to implement
        - Or, 1 part may be technologically feasible (pitch) but another may not be (router or buffer)

Why can routers be hard to implement?

Consider the following picture:

For on-chip connections, must consider mapping network to on-chip metal stack

- How would a torus be implemented?
    - The "wrap around" could have a higher latency than other connections

- Looking at picture of metal routing…
    1. No lines of the same color can touch (it would be an electrical short)
    2. We draw 1 line, but really many (1 line for each bit)
    3. Router areas are by no means insignificant!

On-chip IC NW performance:

Want to know – for a given IC NW topology – how long it takes to send a message:

- Note → initial #s in the *absence* of contention → a bit more on this in just a bit

Time: [(# of hops) x (time in router)] +
[time required for packet to traverse *all* channels] +
[serialization latency]

(serialization latency = ceiling(length of message / bandwidth))

Assuming a 4x4 mesh network, how long does it take to send a message from node 10 to node 3?

- A flit spends in each router is 4 CCs
- It takes 1 CC for a flit to traverse a link between 2 routers
- Link bandwidth is 4 bytes
- We want to send a 50 byte message

Time:	(4 hops x 4 CCs / router) +	3 CCs (from links)	+	ceiling(50/4) CCs	
	16 +	3	+	13	= 32 CCs

see pipetrace…

Can calculate average time it takes to send a message too…

- Average # of hops = 6.25
- Average time for packet to traverse all channels = 5.3333
- Serialization latency = 3
- Time in router = 2
- Total time: = ~20.8

PART B: Example – estimating the impact of traffic…

- Assume 1 iteration of a task takes 500 CCs to complete
- The task requires 10000 independent iterations
- We have N cores at our disposal to parallelize computation if we so choose
- The overhead with a new instantiation on a different core is as follows:
    - The overhead – *per iteration* – is 64 CCs
        - (32 CCs to receive data and 32 CCs to get data back)
    - However, for every additional core used, there is an additional 4 CC overhead per iteration
        - Thus, if 2 cores are used, the overhead is 64 CCs per iteration, if 3 cores are used, the overhead is 68 CCs, etc.
            - (This extra overhead might come from increased network traffic.)
- What number of cores leads to the best overall performance?

We can write an expression to determine execution time.

Time	=	(10000 / n)(500)	+	(10000 / n)(n – 1)(64 + 4(n – 1))		
Time	=	$(4.4 \times 10^6 / n)$	+	(5.6×10^5)	+	$(4 \times 10^4 \times n)$

n = 10, 11 gives lowest time