ECE 1749H: Interconnection Networks for Parallel Computer Architectures:

Router Microarchitecture

Prof. Natalie Enright Jerger

Introduction

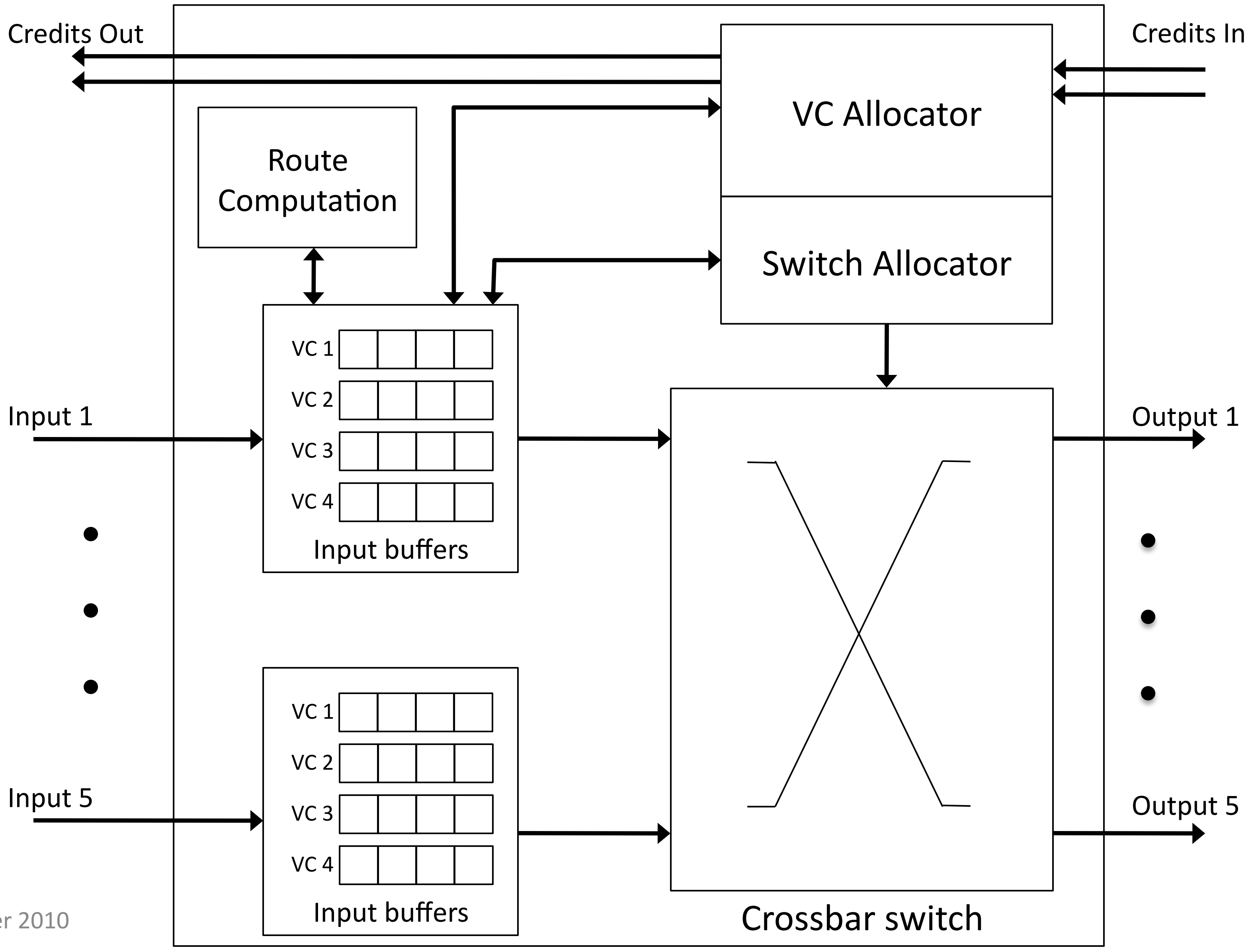
- Topology: connectivity
- Routing: paths
- Flow control: resource allocation

- Router Microarchitecture: implementation of routing, flow control and router pipeline
  – Impacts per-hop delay and energy

Router Microarchitecture Overview

- Focus on microarchitecture of Virtual Channel router
  - Router complexity increase with bandwidth demands
  - Simple routers built when high throughput is not needed
    - Wormhole flow control, unpipelined, limited buffer

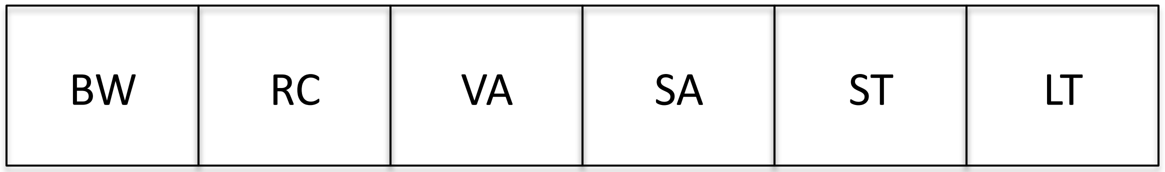
Virtual Channel Router



Router Components

- Input buffers, route computation logic, virtual channel allocator, switch allocator, crossbar switch
- Most OCN routers are input buffered
  - Use single-ported memories
- Buffer store flits for duration in router
  - Contrast with processor pipeline that latches between stages

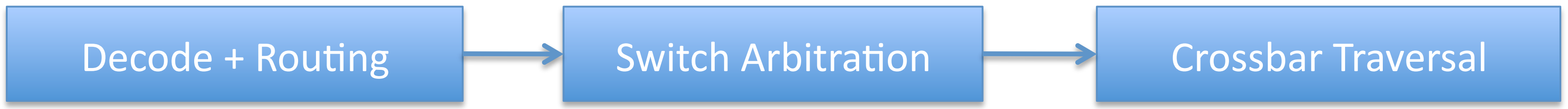
Baseline Router Pipeline



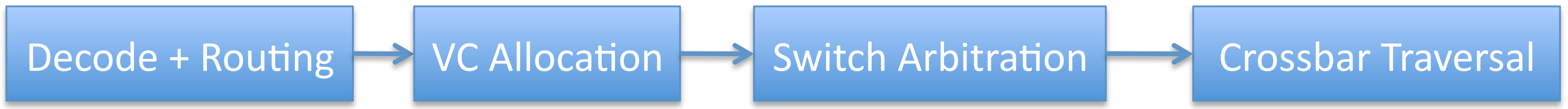
- Logical stages
  - Fit into physical stages depending on frequency

- Canonical 5-stage pipeline
  - BW: Buffer Write
  - RC: Routing computation
  - VA: Virtual Channel Allocation
  - SA: Switch Allocation
  - ST: Switch Traversal
  - LT: Link Traversal

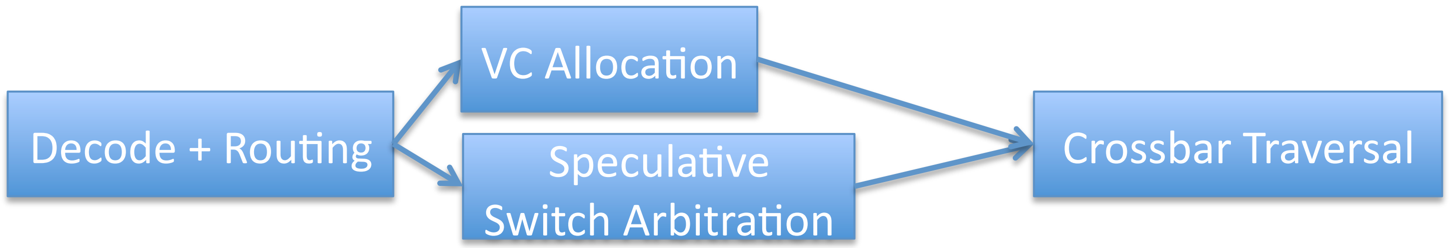
Atomic Modules and Dependencies in Router



Wormhole Router



Virtual Channel Router



Speculative Virtual Channel Router

- Dependence between output of one module and input of another
  - Determine critical path through router
  - Cannot bid for switch port until routing performed

Atomic Modules

- Some components of router cannot be easily pipelined
- Example: pipeline VC allocation
  - Grants might not be correctly reflected before next allocation
- Separable allocator: many wires connecting input/output stages requiring latches if pipelined

Baseline Router Pipeline (2)

	1	2	3	4	5	6	7	8	9
Head	BW	RC	VA	SA	ST	LT			
Body 1		BW	✗	✗	SA	ST	LT		
Body 2			BW	✗	✗	SA	ST	LT	
Tail				BW	✗	✗	SA	ST	LT

- Routing computation performed once per packet
- Virtual channel allocated once per packet
- Body and tail flits inherit this info from head flit

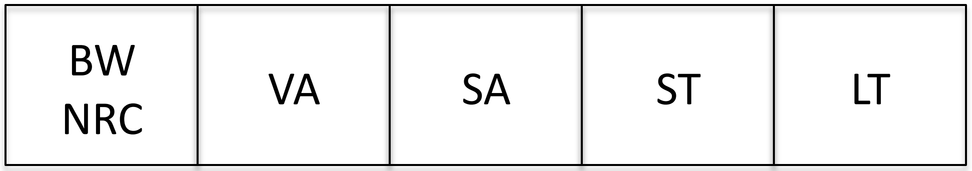
Router Pipeline Performance

- Baseline (no load) delay

$$= \left(5\,cycles + link\,delay\right) \times hops + t_{serialization}$$

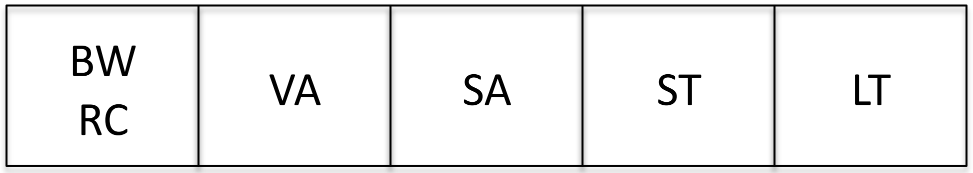
- Ideally, only pay link delay

- Techniques to reduce pipeline stages



Pipeline Optimizations: Lookahead Routing

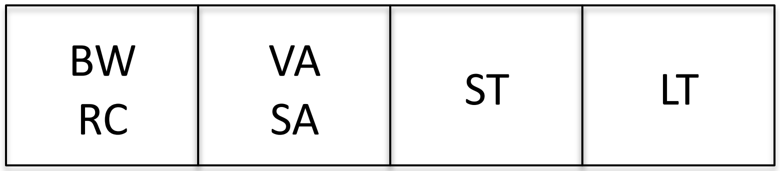
- At current router perform routing computation for next router
  - Overlap with BW



  - Precomputing route allows flits to compete for VCs immediately after BW
  - RC decodes route header
  - Routing computation needed at next hop
    - Can be computed in parallel with VA

Pipeline Optimizations: Speculation

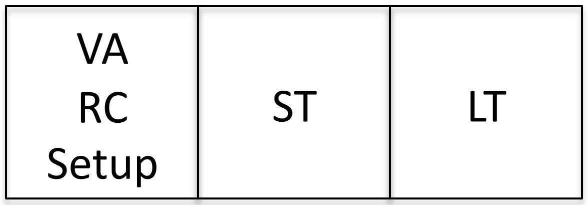
- Assume that Virtual Channel Allocation stage will be successful
  - Valid under low to moderate loads
- Entire VA and SA in parallel



- If VA unsuccessful (no virtual channel returned)
  - Must repeat VA/SA in next cycle
- Prioritize non-speculative requests

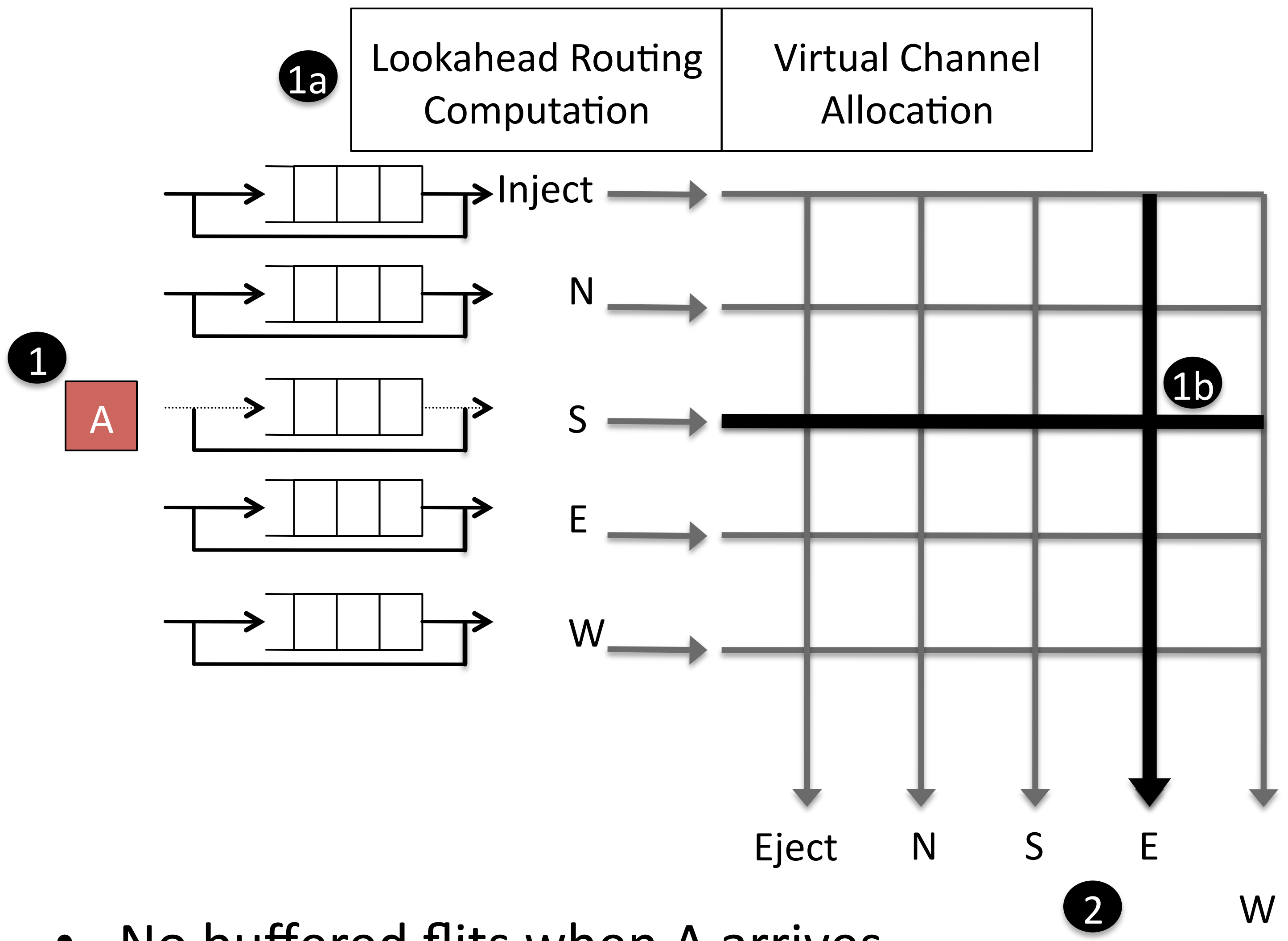
Pipeline Optimizations: Bypassing

- When no flits in input buffer
  - Speculatively enter ST
  - On port conflict, speculation aborted



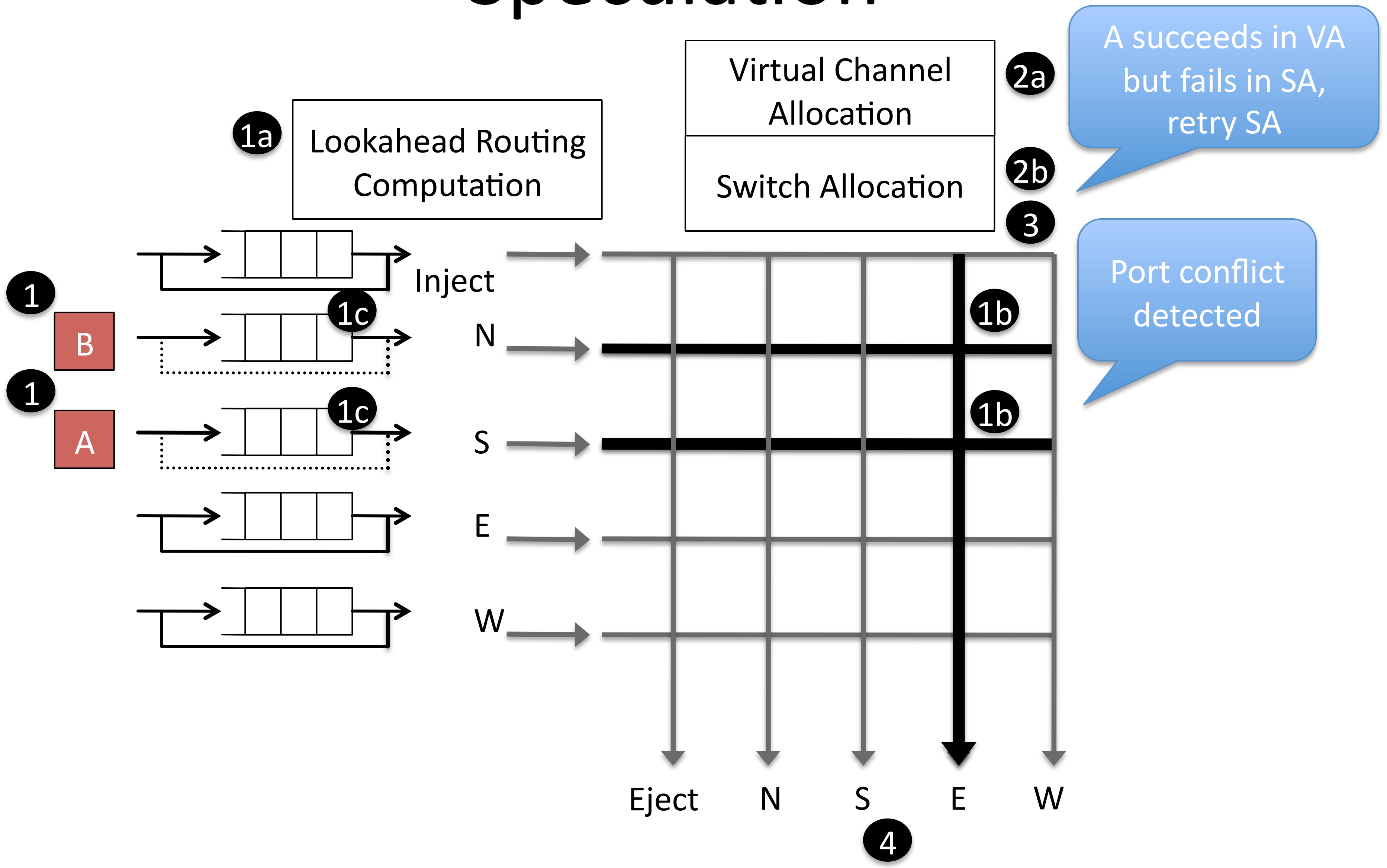
  - In the first stage, a free VC is allocated, next routing is performed and the crossbar is setup

Pipeline Bypassing

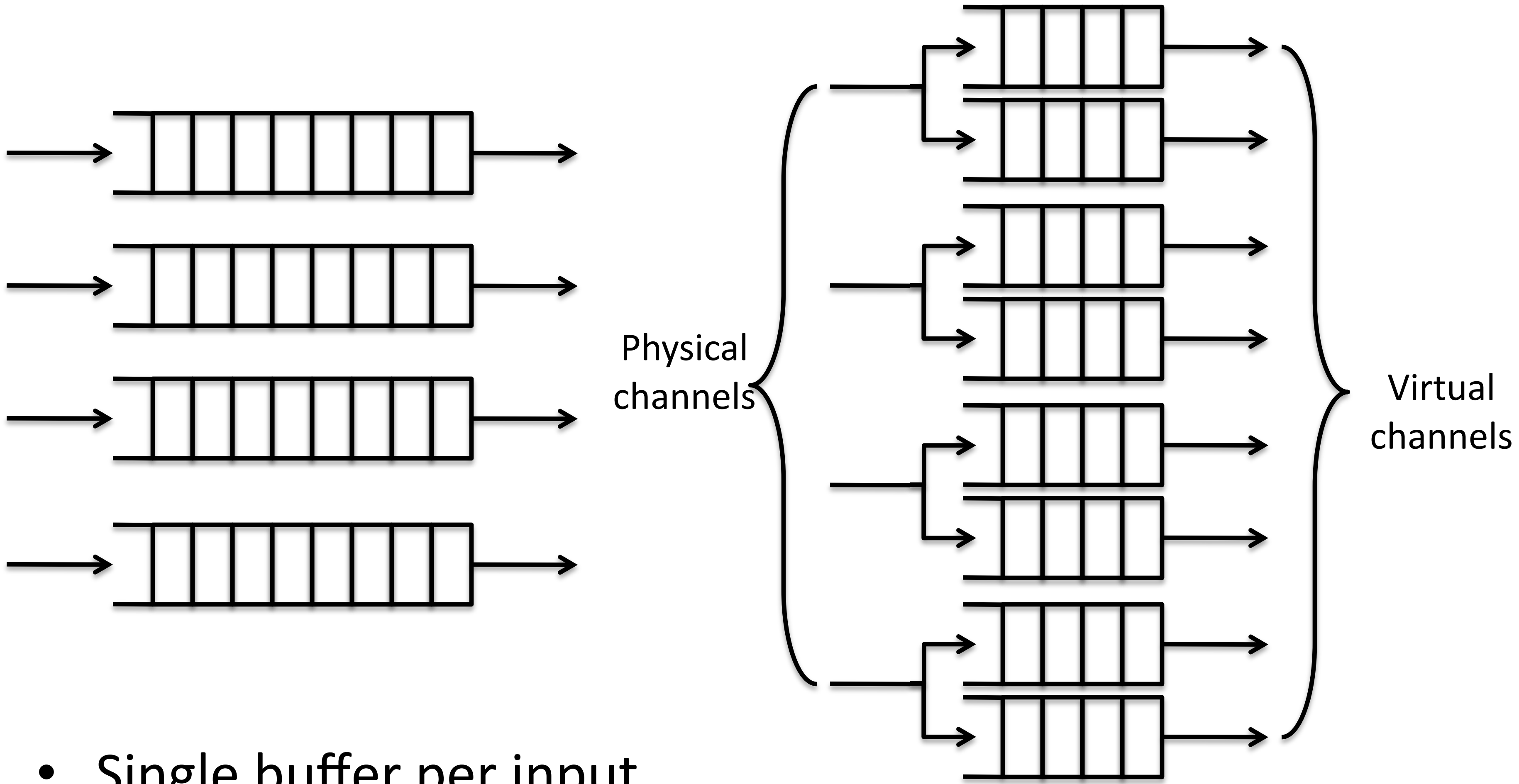


- No buffered flits when A arrives

Speculation

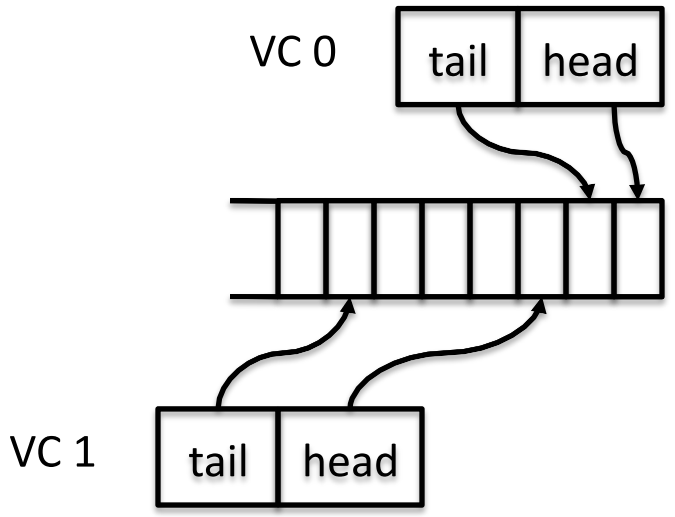

Lookahead Routing Computation
1a
Virtual Channel Allocation
2a
Switch Allocation
2b
3
A succeeds in VA but fails in SA, retry SA
Port conflict detected
1
B
1
A
1c
1c
Inject
N
S
E
W
1b
1b
Eject
N
S
E
W
4

Buffer Organization



- Single buffer per input
- Multiple fixed length queues per physical channel

Buffer Organization



- Multiple variable length queues
  - Multiple VCs share a large buffer
  - Each VC must have minimum 1 flit buffer
    - Prevent deadlock
  - More complex circuitry

Buffer Organization

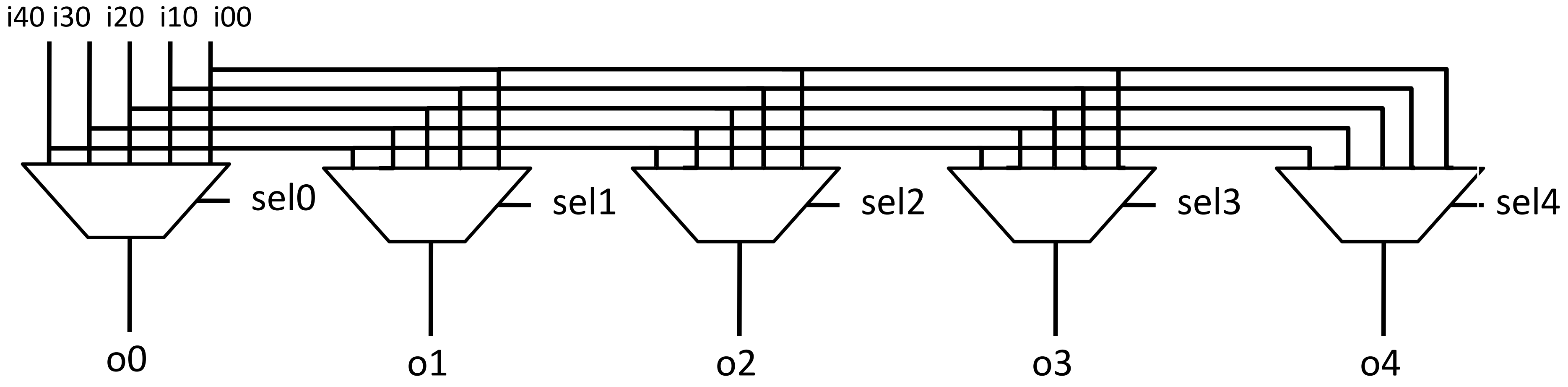
- Many shallow VCs?
- Few deep VCs?

- More VCs ease HOL blocking
  - More complex VC allocator

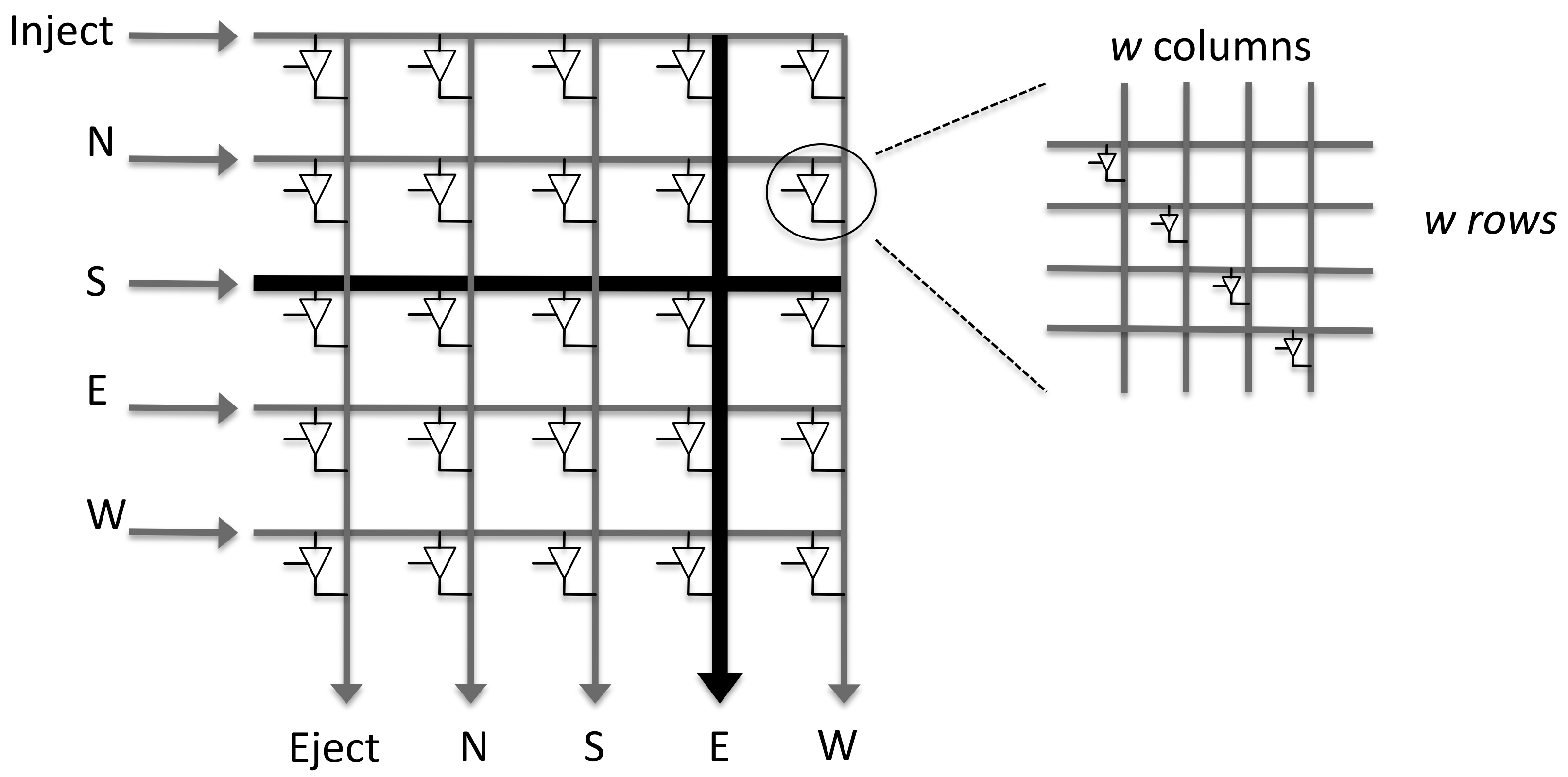
- Light traffic
  - Many shallow VCs – underutilized
- Heavy traffic
  - Few deep VCs – less efficient, packets blocked due to lack of VCs

Switch Organization

- Heart of datapath
  - Switches bits from input to output
- High frequency crossbar designs challenging
- Crossbar composed for many multiplexers
  - Common in low-frequency router designs

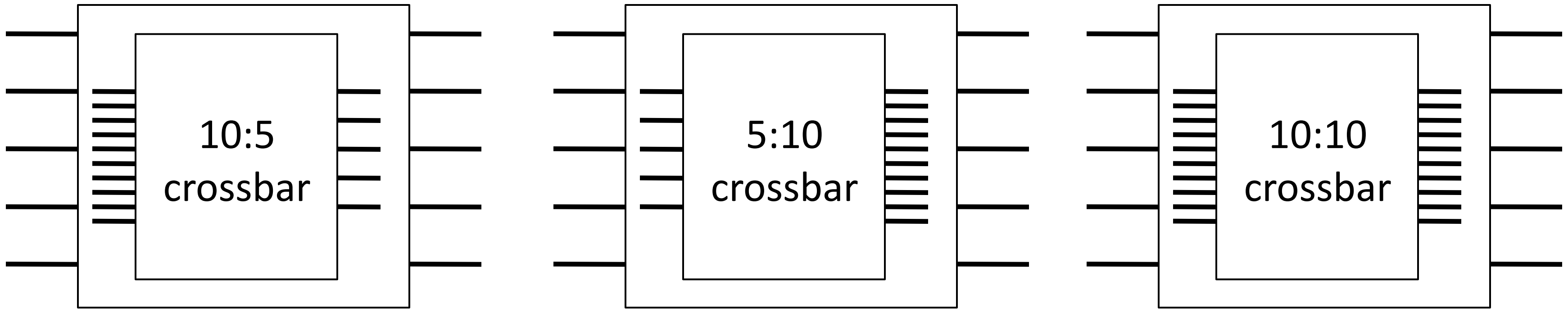


Switch Organization: Crosspoint



- Area and power scale at $O((pw)^2)$
  - p: number of ports (function of topology)
  - w: port width in bits (determines phit/flit size and impacts packet energy and delay)

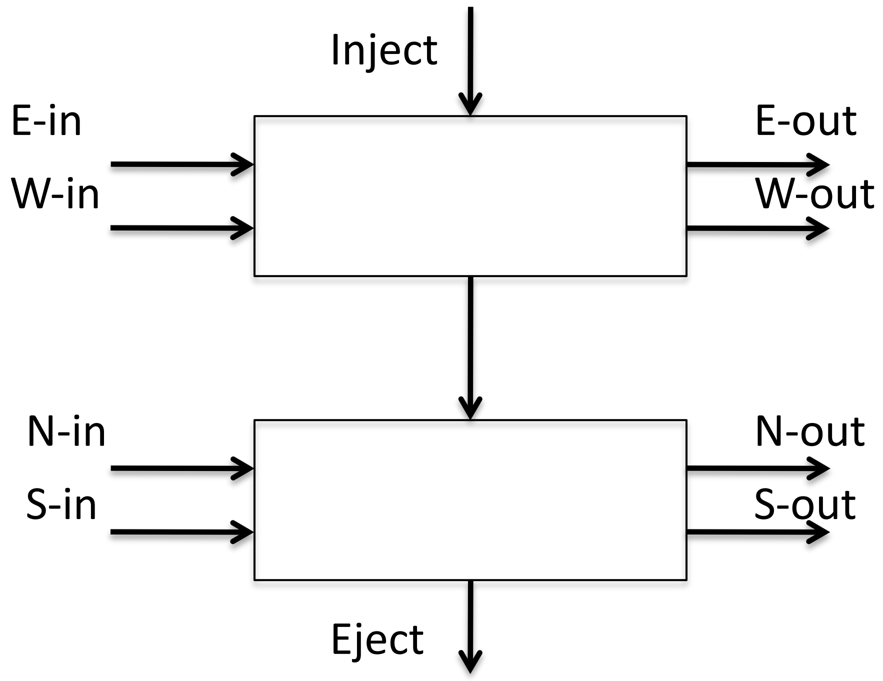
Crossbar speedup



- Increase internal switch bandwidth
- Simplifies allocation or gives better performance with a simple allocator
  - More inputs to select from → higher probability each output port will be matched (used) each cycle
- Output speedup requires output buffers
  - Multiplex onto physical link

Crossbar Dimension Slicing

- Crossbar area and power grow with *$O((pw)^2)$*



- Replace 1 5x5 crossbar with 2 3x3 crossbars
- Suited to DOR
  - Traffic mostly stays within 1 dimension

Arbiters and Allocators

- Allocator matches N requests to M resources
- Arbiter matches N requests to 1 resource
- Resources are VCs (for virtual channel routers) and crossbar switch ports.

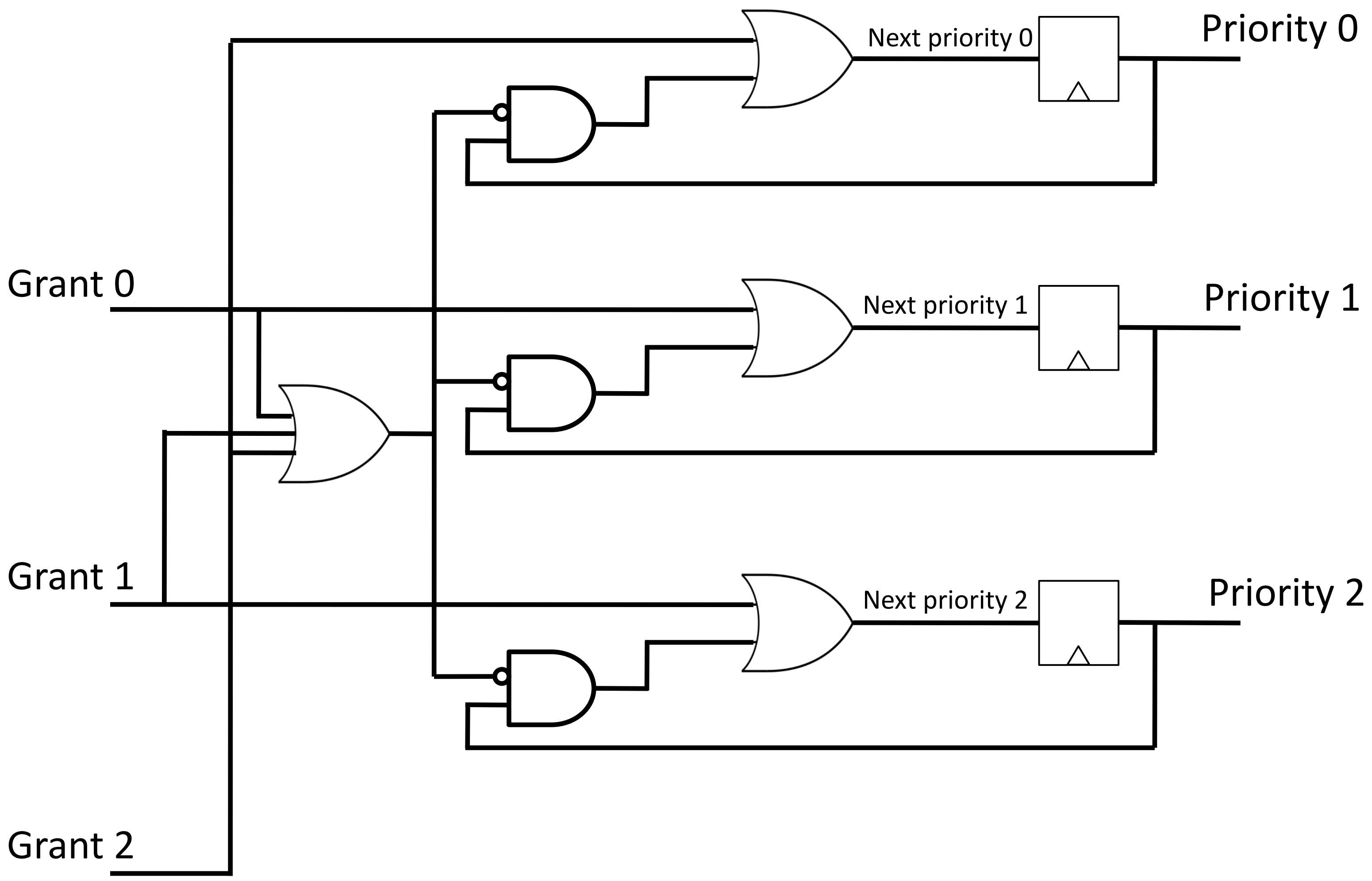
Arbiters and Allocators (2)

- Virtual-channel allocator (VA)
  - Resolves contention for output virtual channels
  - Grants them to input virtual channels
- Switch allocator (SA) that grants crossbar switch ports to input virtual channels
- Allocator/arbiter that delivers high matching probability translates to higher network throughput.
  - Must also be fast and/or able to be pipelined

Round Robin Arbiter

- Last request serviced given lowest priority
- Generate the next priority vector from current grant vector
- Exhibits fairness

Round Robin (2)

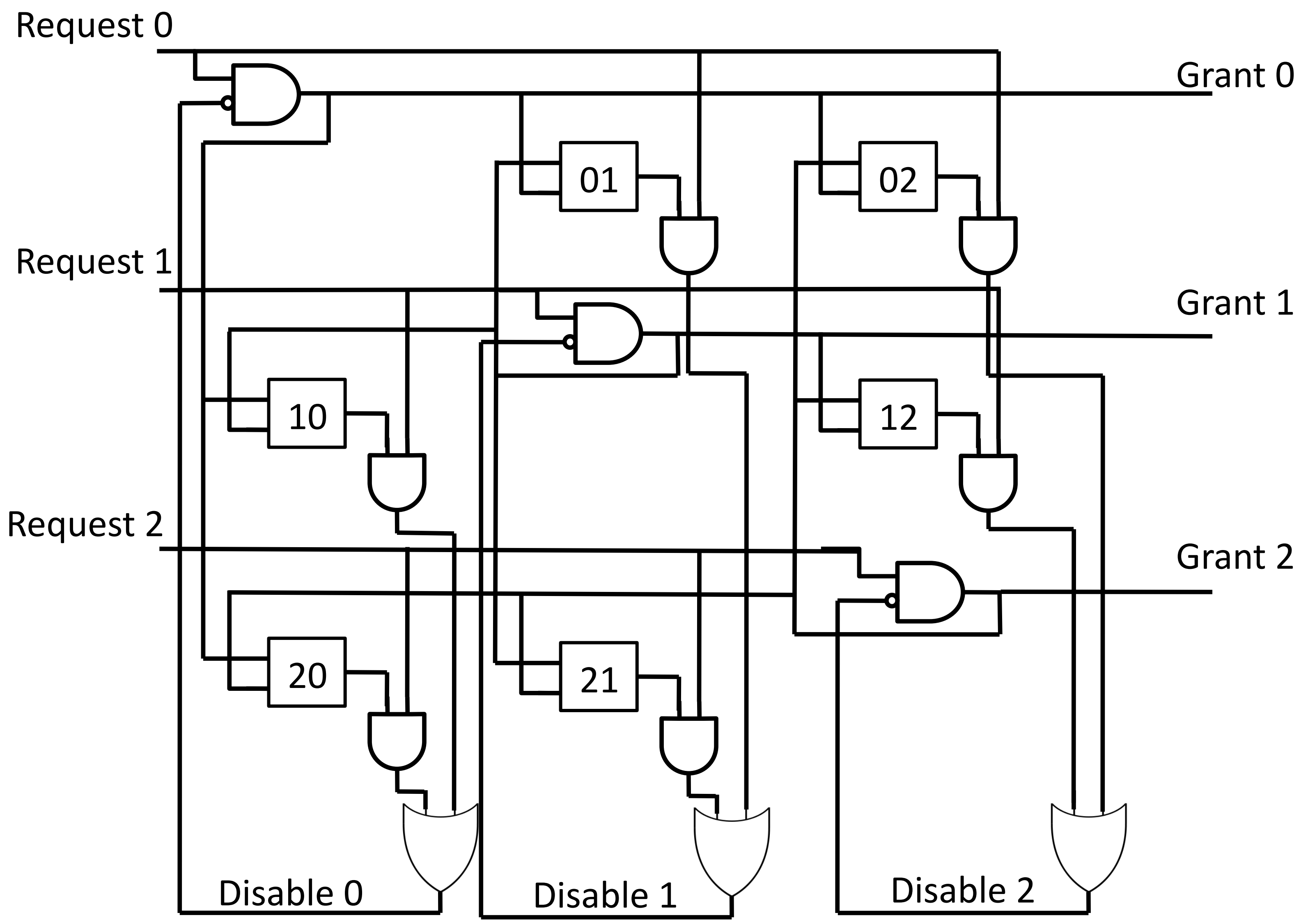


- G_i granted, next cycle P_{i+1} high

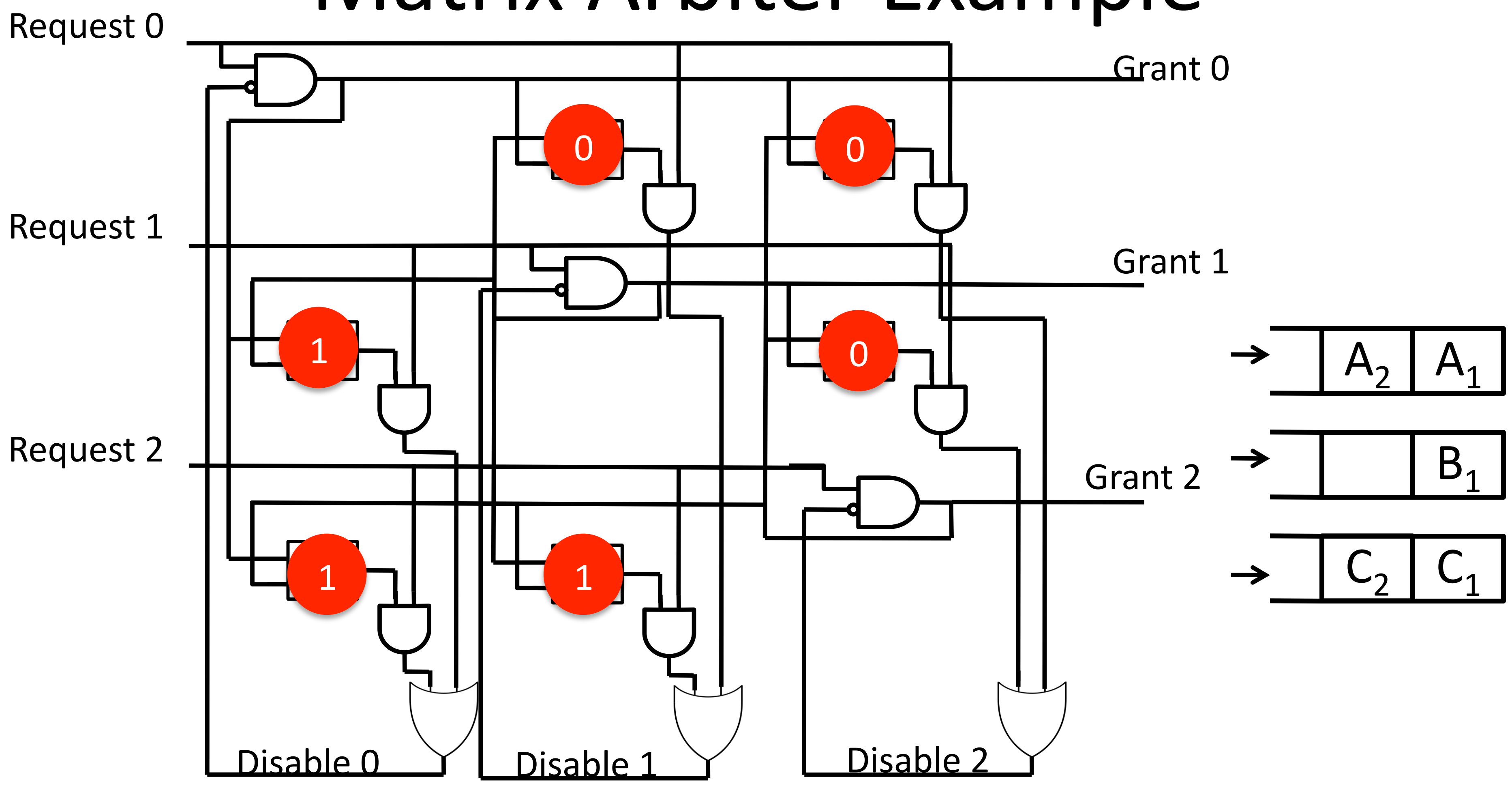
Matrix Arbiter

- Least recently served priority scheme
- Triangular array of state bits w_{ij} for $i < j$
  - Bit w_{ij} indicates request i takes priority over j
  - Each time request k granted, clears all bits in row k and sets all bits in column k
- Good for small number of inputs
- Fast, inexpensive and provides strong fairness

Matrix Arbiter (2)

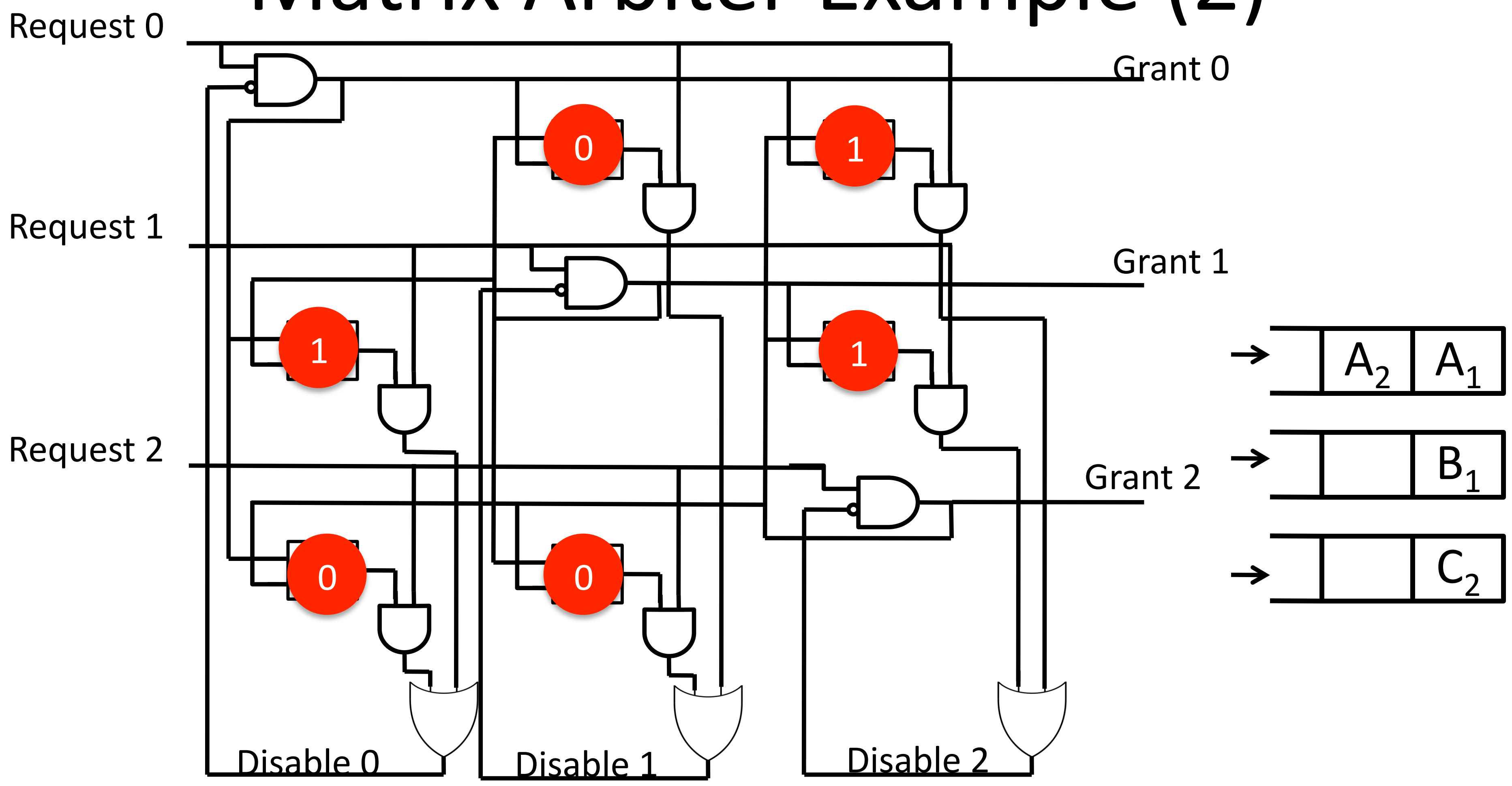


Matrix Arbiter Example



Bit [1,0] = 1, Bit [2,0] = 1 → 1 and 2 have priority over 0
Bit [2,1] = 1 → 2 has priority over 1
C_1 (Req 2) granted

Matrix Arbiter Example (2)

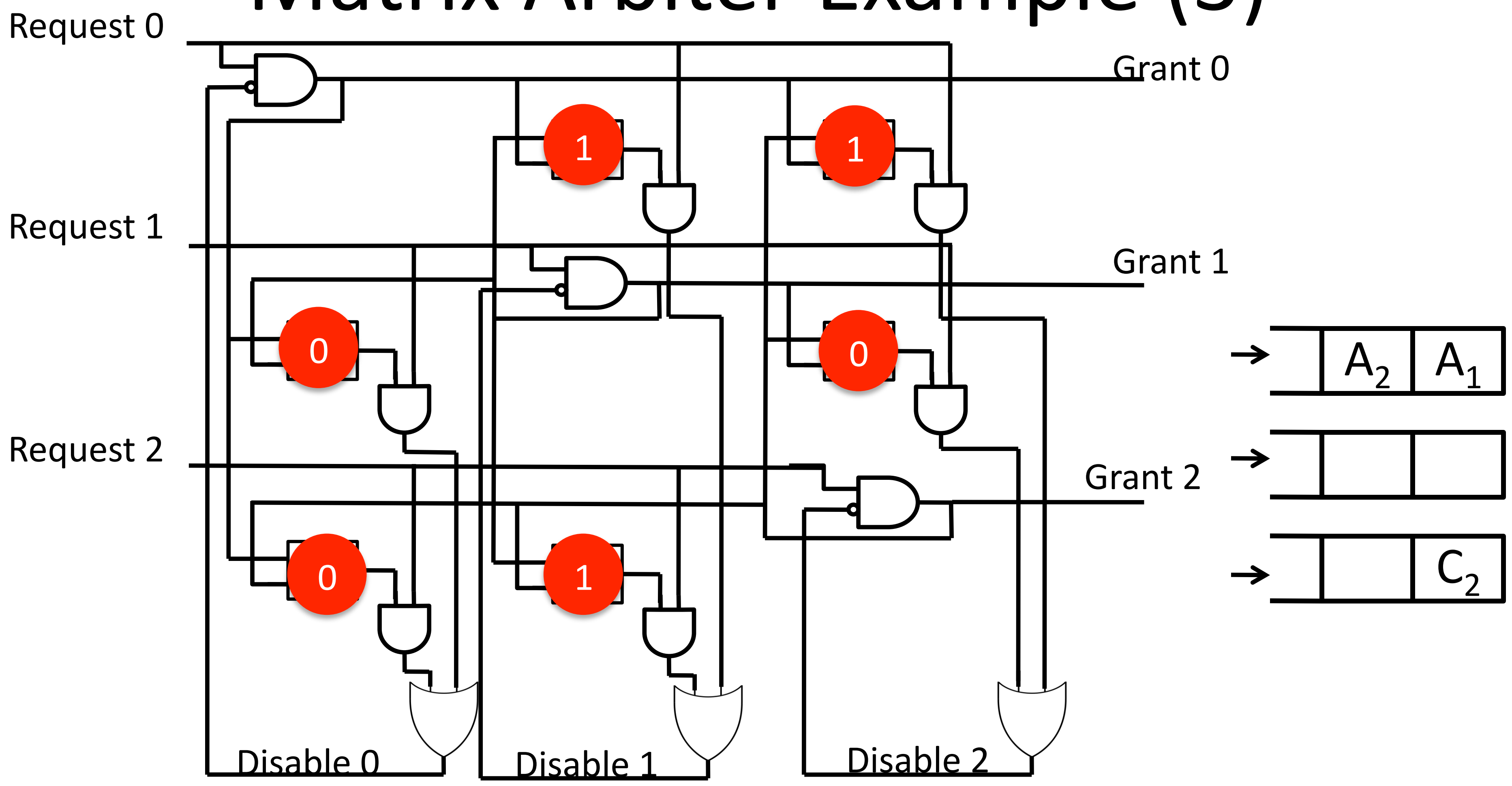


Set column 2, clear row 2

Bit [1,0] = 1, Bit [1,2] = 1 → Req 1 has priority over 0 and 2

Grant B_1 (Req 1)

Matrix Arbiter Example (3)

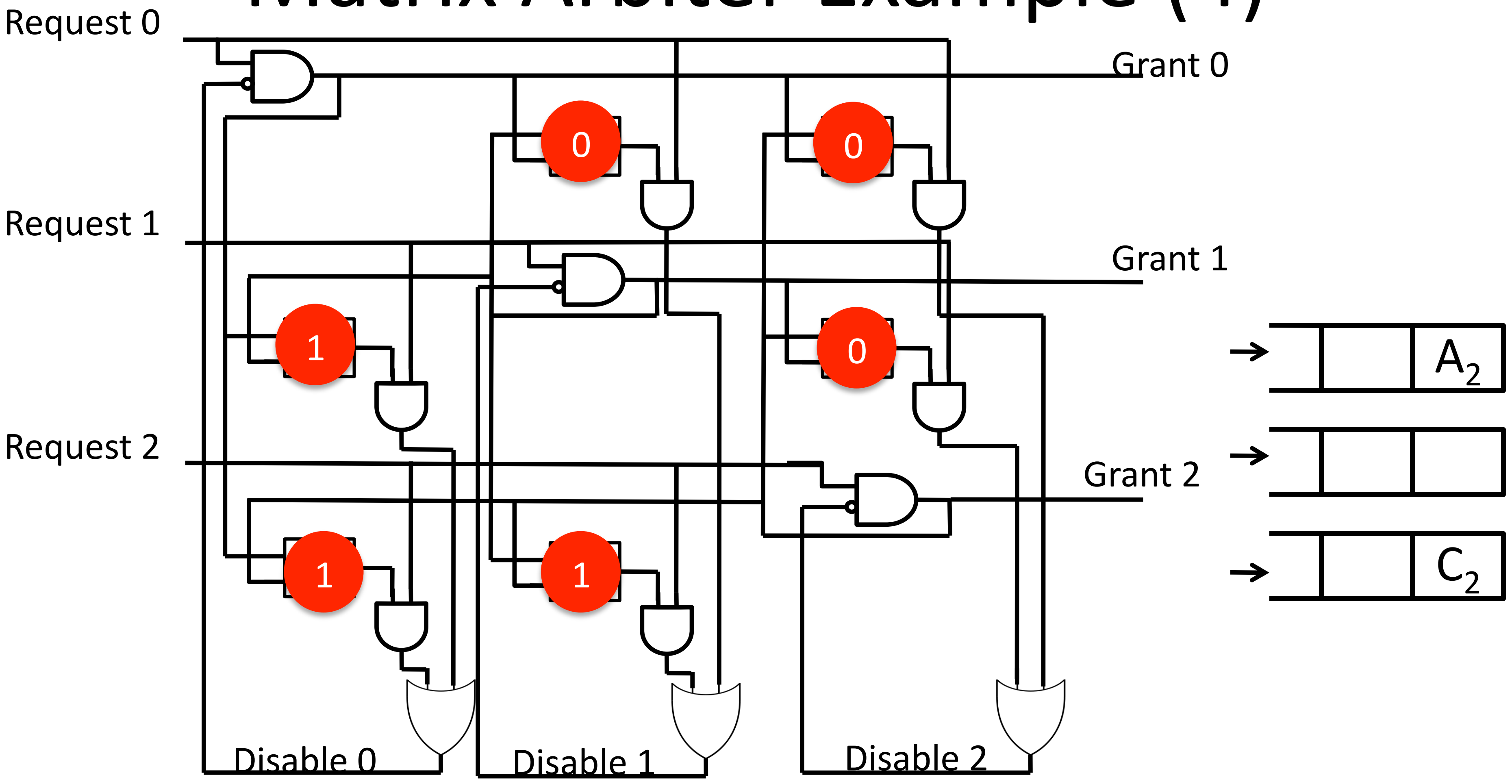


Set column 1, clear row 1

Bit [0,1] = 1, Bit [0,2] = 1 → Req 0 has priority over 1 and 2

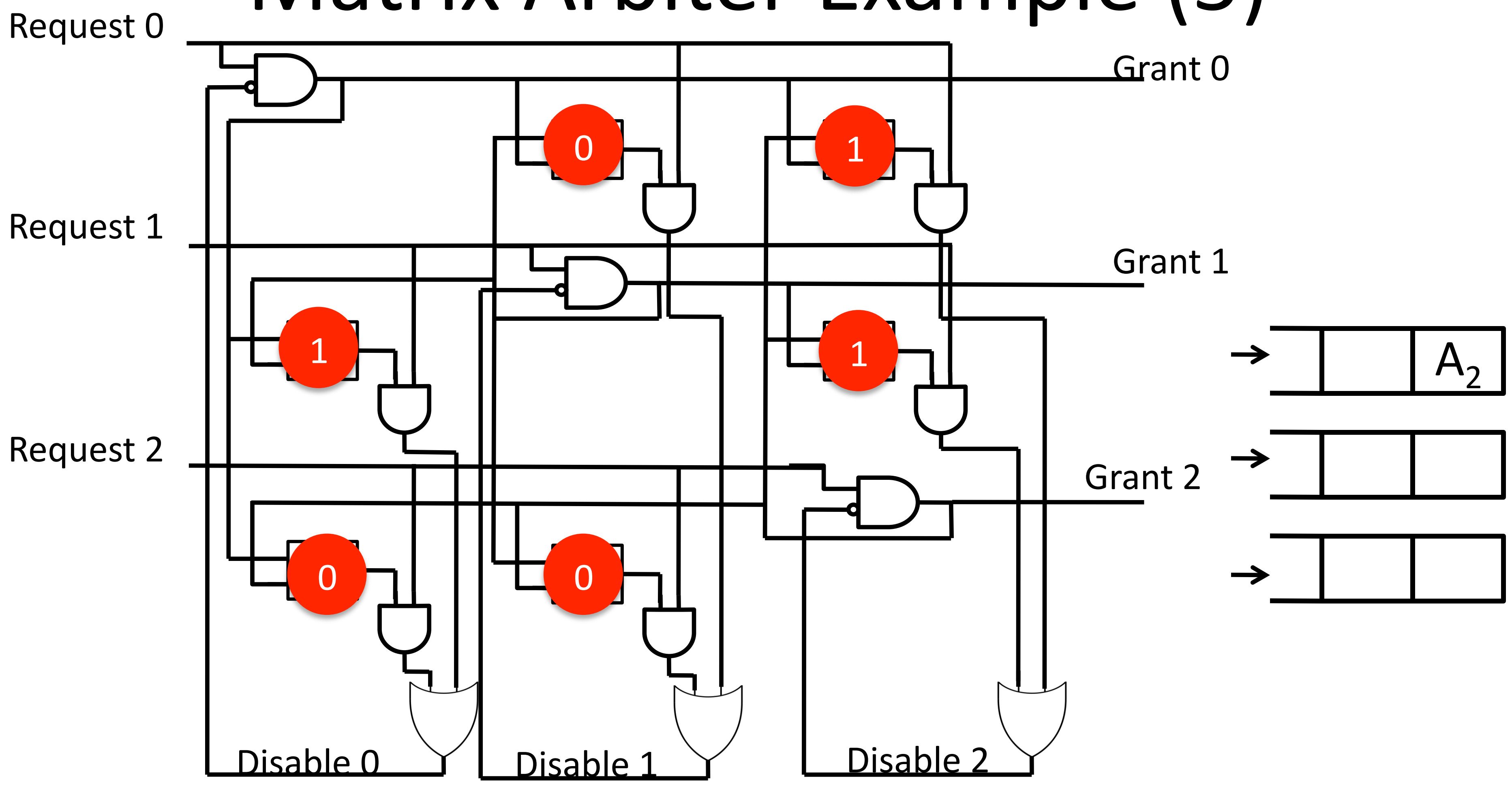
Grant A_1 (Req 0)

Matrix Arbiter Example (4)



Set column 0, clear row 0
Bit [2,0] = 1, Bit [2,1] = 1 → Req 2 has priority over 0 and 1
Grant C_2 (Req 2)

Matrix Arbiter Example (5)



Set column 2, clear row 2
Grant Request A_2

Wavefront Allocator

- Arbitrates among requests for inputs and outputs simultaneously
- Row and column tokens granted to diagonal group of cells
- If a cell is requesting a resource, it will consume row and column tokens
  - Request is granted
- Cells that cannot use tokens pass row tokens to right and column tokens down

avefront Allocator Example

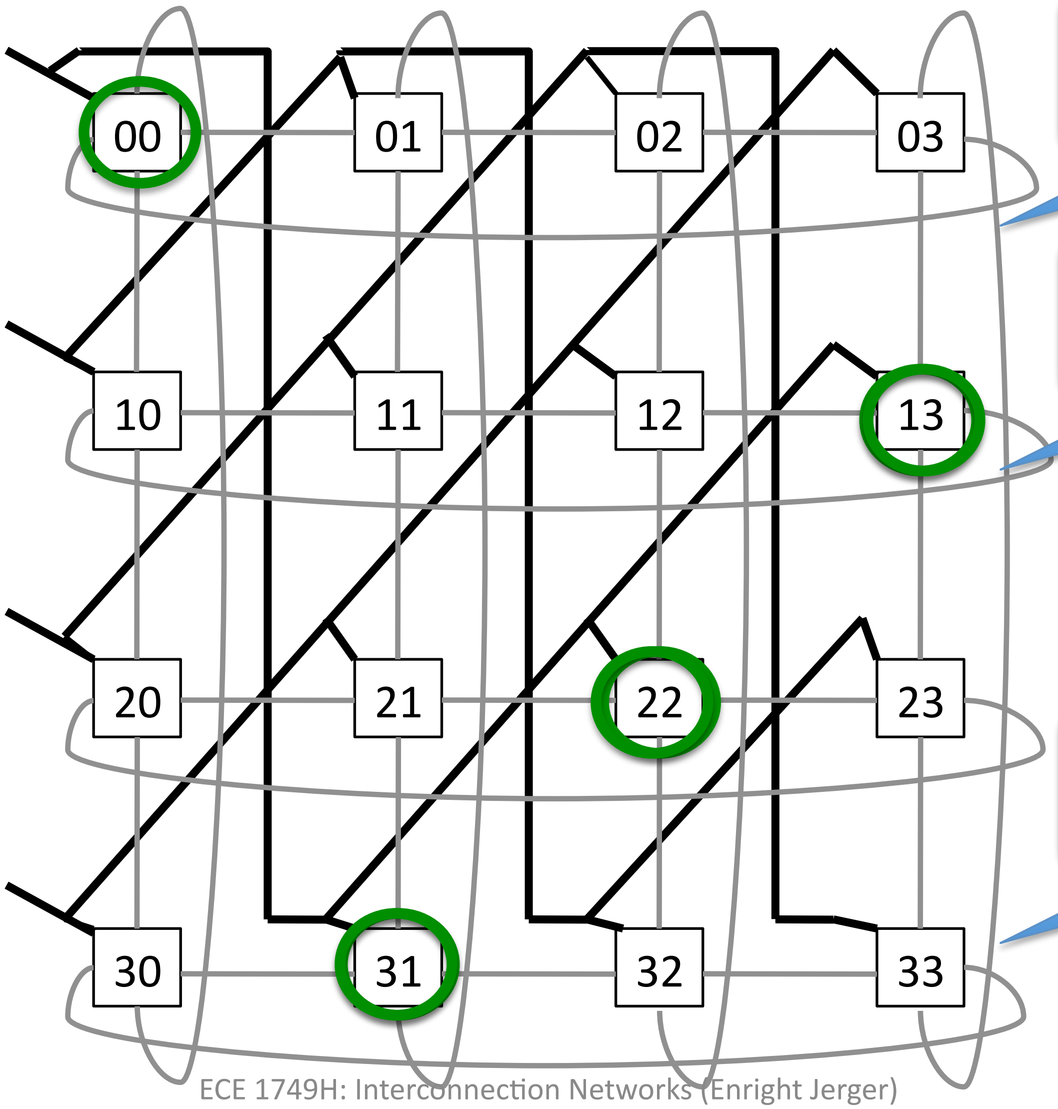
Tokens inserted at P0

A requesting Resources 0, 1 ,2

B requesting Resources 0, 1

C requesting Resource 0

D requesting Resources 0, 2

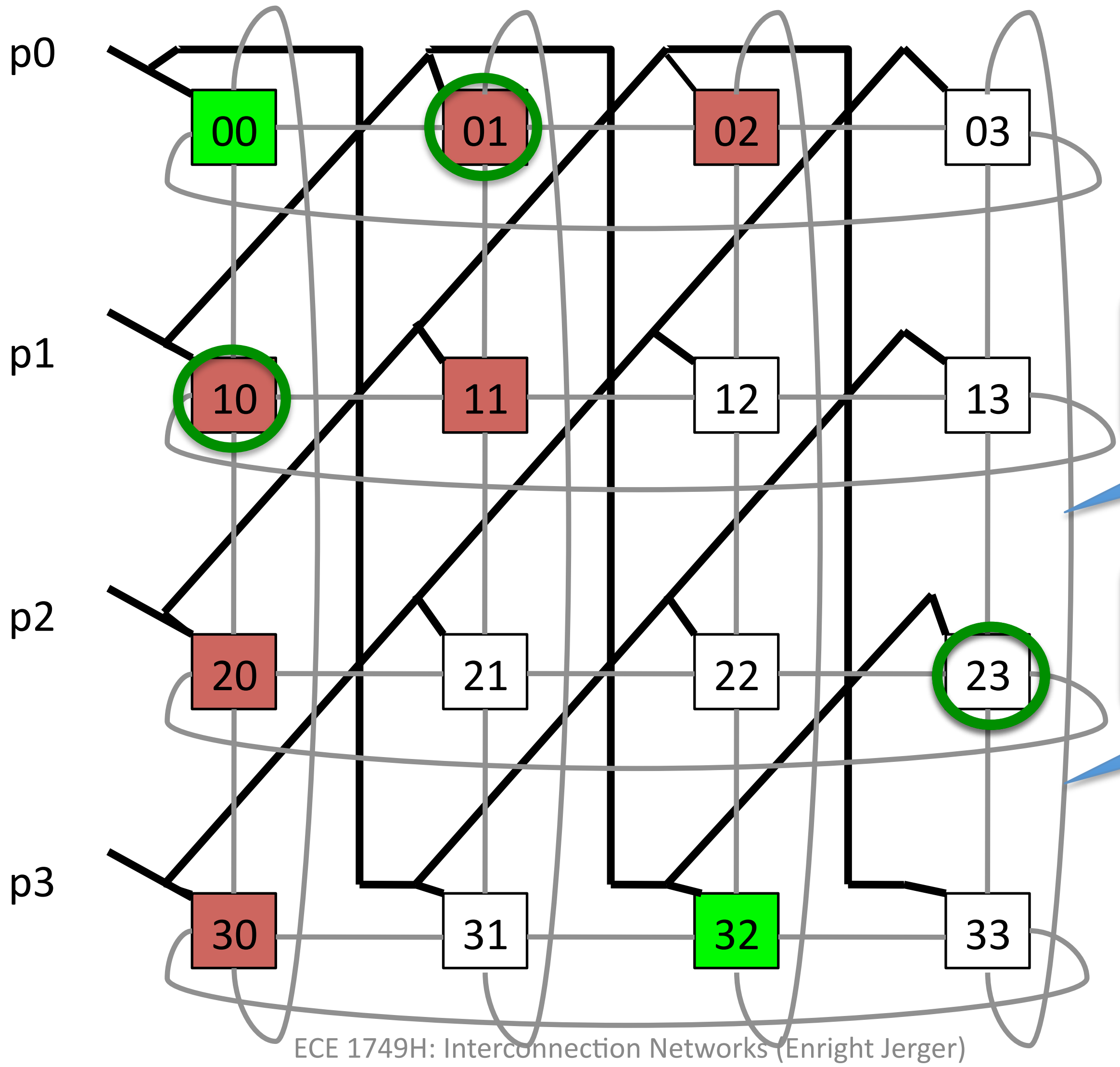


Entry [0,0] receives grant, consumes token

Remaining tokens pass down and right

[3,2] receives 2 tokens and is granted

Wavefront Allocator Example



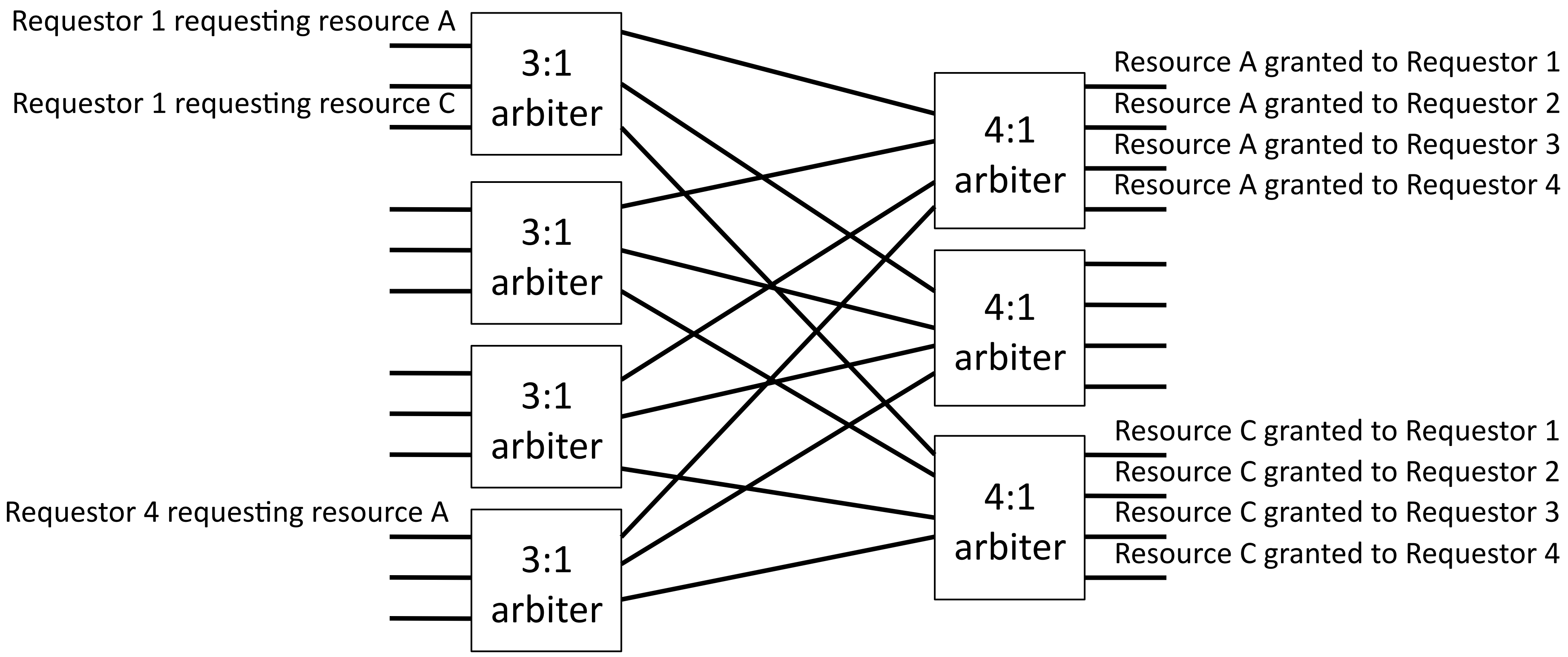
[1,1] receives 2 tokens and granted

All wavefronts propagated

Separable Allocator

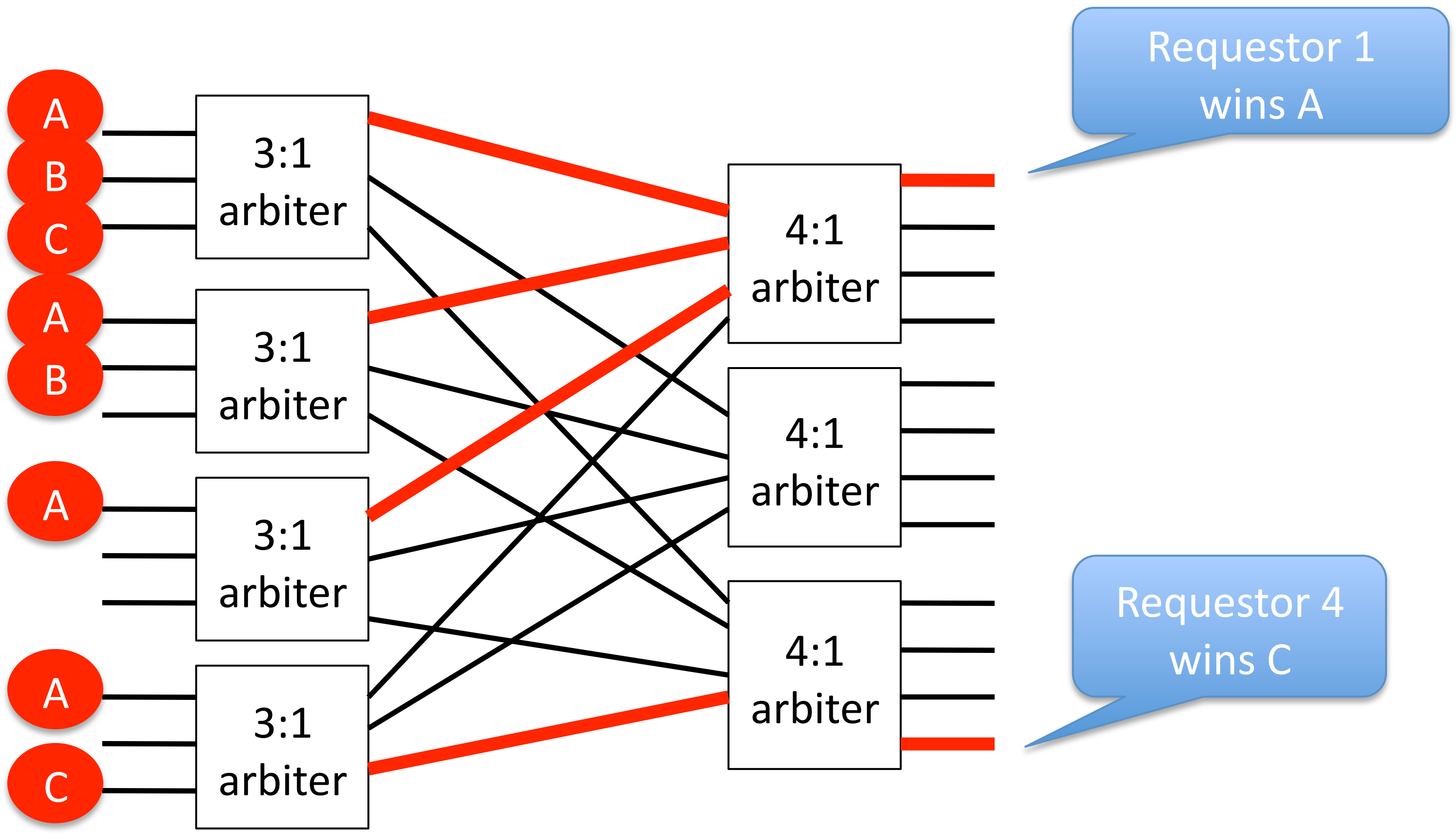
- Need for pipelineable allocators
- Allocator composed of arbiters
  - Arbiter chooses one out of N requests to a single resource
- Separable switch allocator
  - First stage: select single request at each input port
  - Second stage: selects single request for each output port

Separable Allocator


Requestor 1 requesting resource A
Requestor 1 requesting resource C
Requestor 4 requesting resource A
3:1 arbiter
3:1 arbiter
3:1 arbiter
3:1 arbiter
4:1 arbiter
4:1 arbiter
4:1 arbiter
Resource A granted to Requestor 1
Resource A granted to Requestor 2
Resource A granted to Requestor 3
Resource A granted to Requestor 4
Resource C granted to Requestor 1
Resource C granted to Requestor 2
Resource C granted to Requestor 3
Resource C granted to Requestor 4

- A 3:4 allocator
- First stage: 3:1 – ensures only one grant for each input
- Second stage: 4:1 – only one grant asserted for each output

Separable Allocator Example

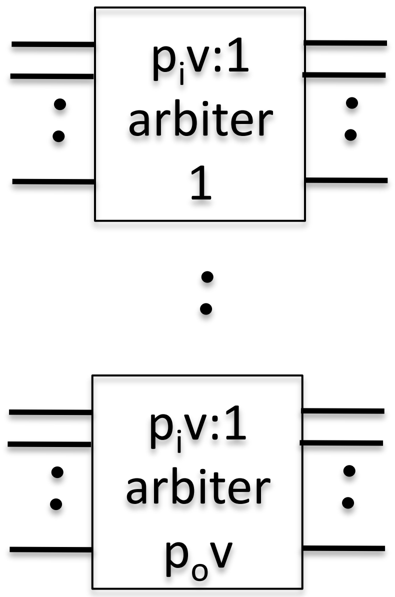


- 4 requestors, 3 resources
- Arbitrate locally among requests
  - Local winners passed to second stage

Virtual Channel Allocator Organization

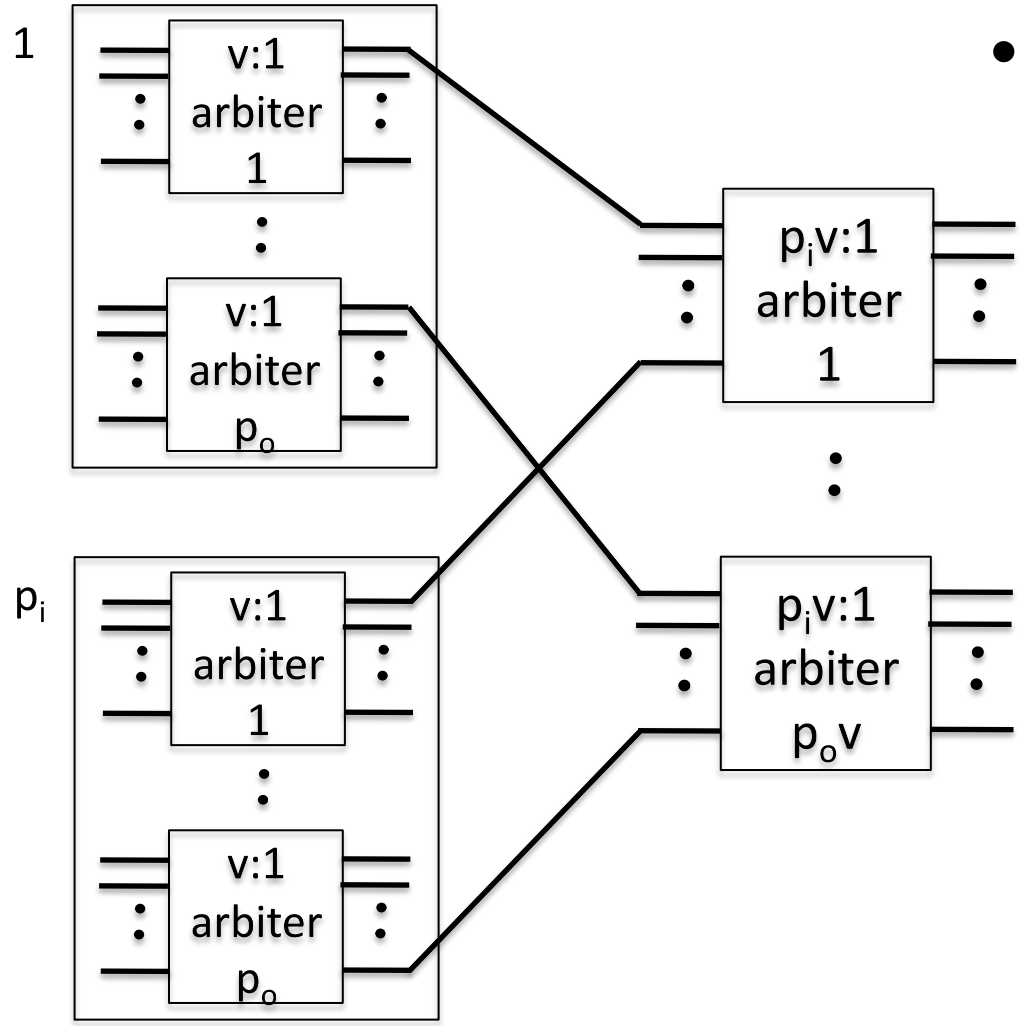
- Depends on routing function
  - If routing function returns single VC
    - VCA need to arbitrate between input VCs contending for same output VC
  - Returns multiple candidate VCs (for same physical channel)
    - Needs to arbitrate among ***v*** first stage requests before forwarding winning request to second stage)

Virtual Channel Allocators



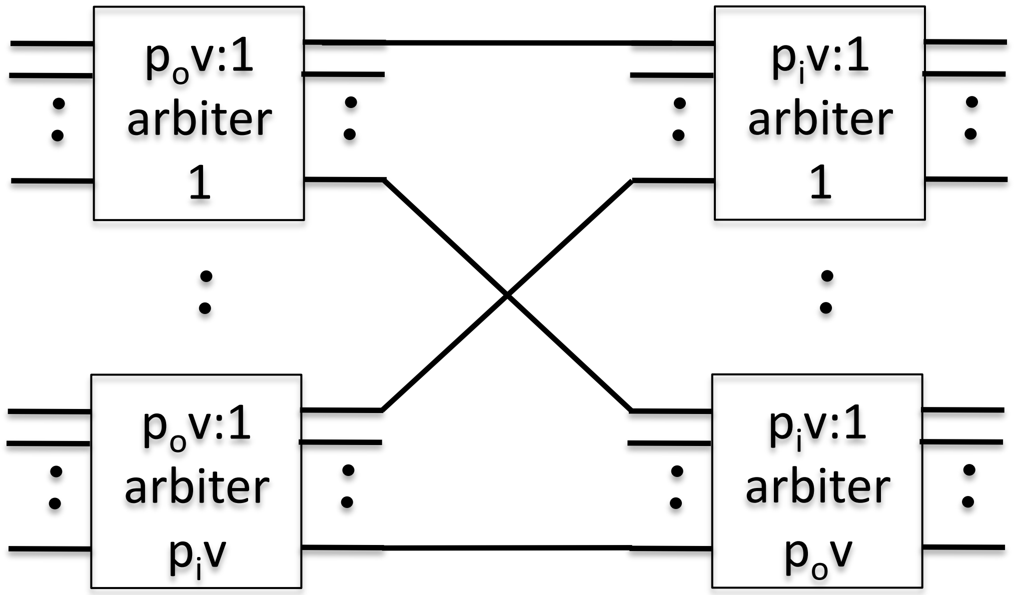
- If routing function returns single virtual channel
  - Need p_iv:1 arbiter for each output virtual channel (p_ov)
- Arbitrate among input VCs competing for same output VC

Virtual Channel Allocators



- Routing function returns VCs on a single physical channel
  - First stage of v:1 arbiters for each input VC
  - Second stage p_iv:1 arbiters for each output VC

Virtual Channel Allocators



- Routing function returns candidate VCs on any physical channel
  - First stage: $p_o v$:1 arbiter to handle max $p_o v$ output VCs desired by each input VC
  - Second stage: $p_i v$:1 for each output VC

Adaptive Routing & Allocator Design

- Deterministic routing
  - Single output port
  - Switch allocator bids for output port

- Adaptive routing
  - Returns multiple candidate output ports
    - Switch allocator can bid for all ports
    - Granted port must match VC granted
  - Return single output port
    - Reroute if packet fails VC allocation

Separable Switch Allocator

- First stage:
  - P_i v:1 arbiters
    - For each P_i input, select among v input virtual channels
- Second stage:
  - P_o p_i:1 arbiters
    - Winners of v:1 arbiters select output port request of winning VC
    - Forward output port request to p_i:1 arbiters

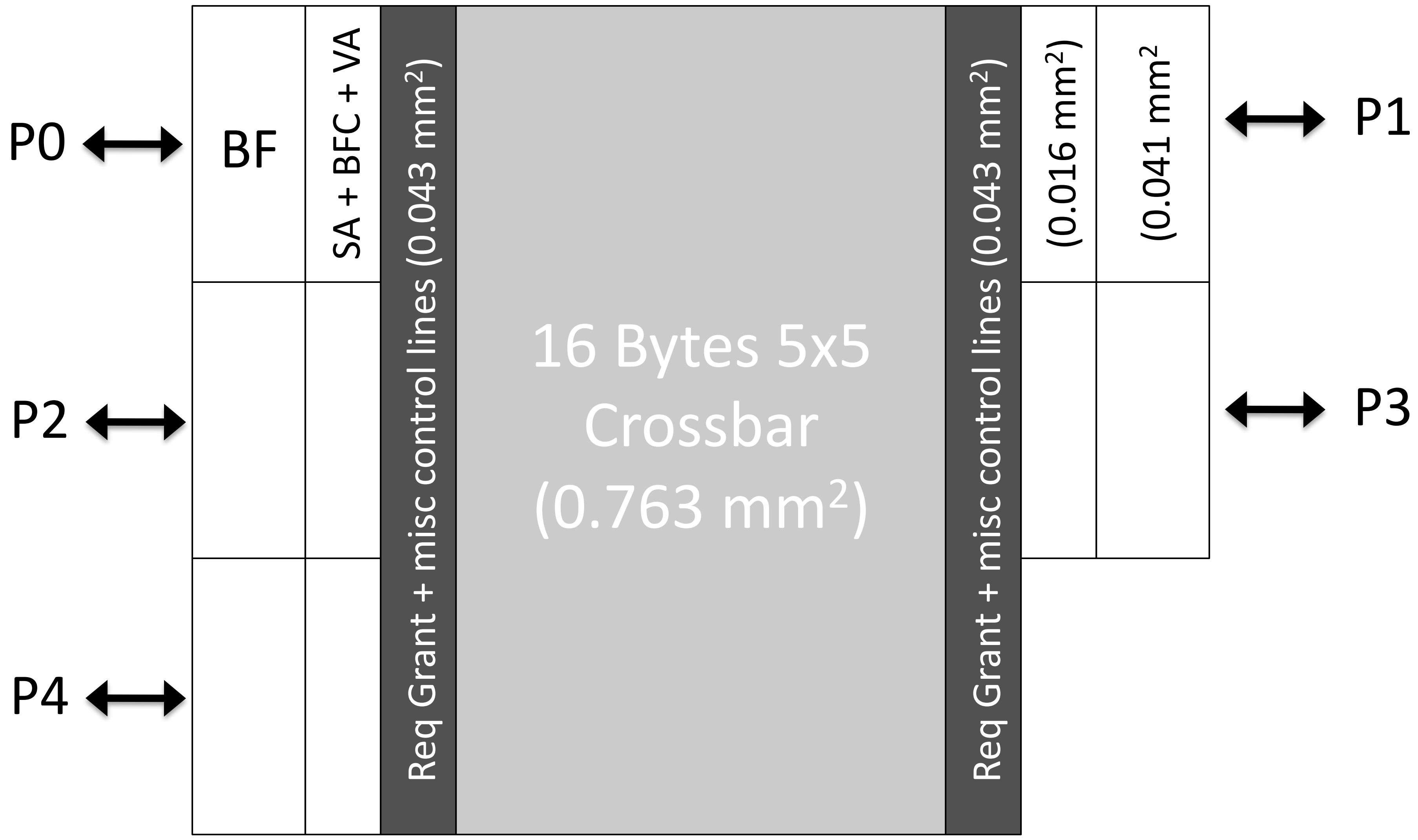
Speculative VC Router

- Non-speculative switch requests must have higher priority than speculative ones
  - Two parallel switch allocators
    - 1 for speculative
    - 1 for non-speculative
    - From output, choose non-speculative over speculative
  - Possible for flit to succeed in speculative switch allocation but fail in virtual channel allocation
    - Done in parallel
    - Speculation incorrect
      - Switch reservation is wasted
  - Body and Tail flits: non-speculative switch requests
    - Do not perform VC allocation → inherit VC from head flit

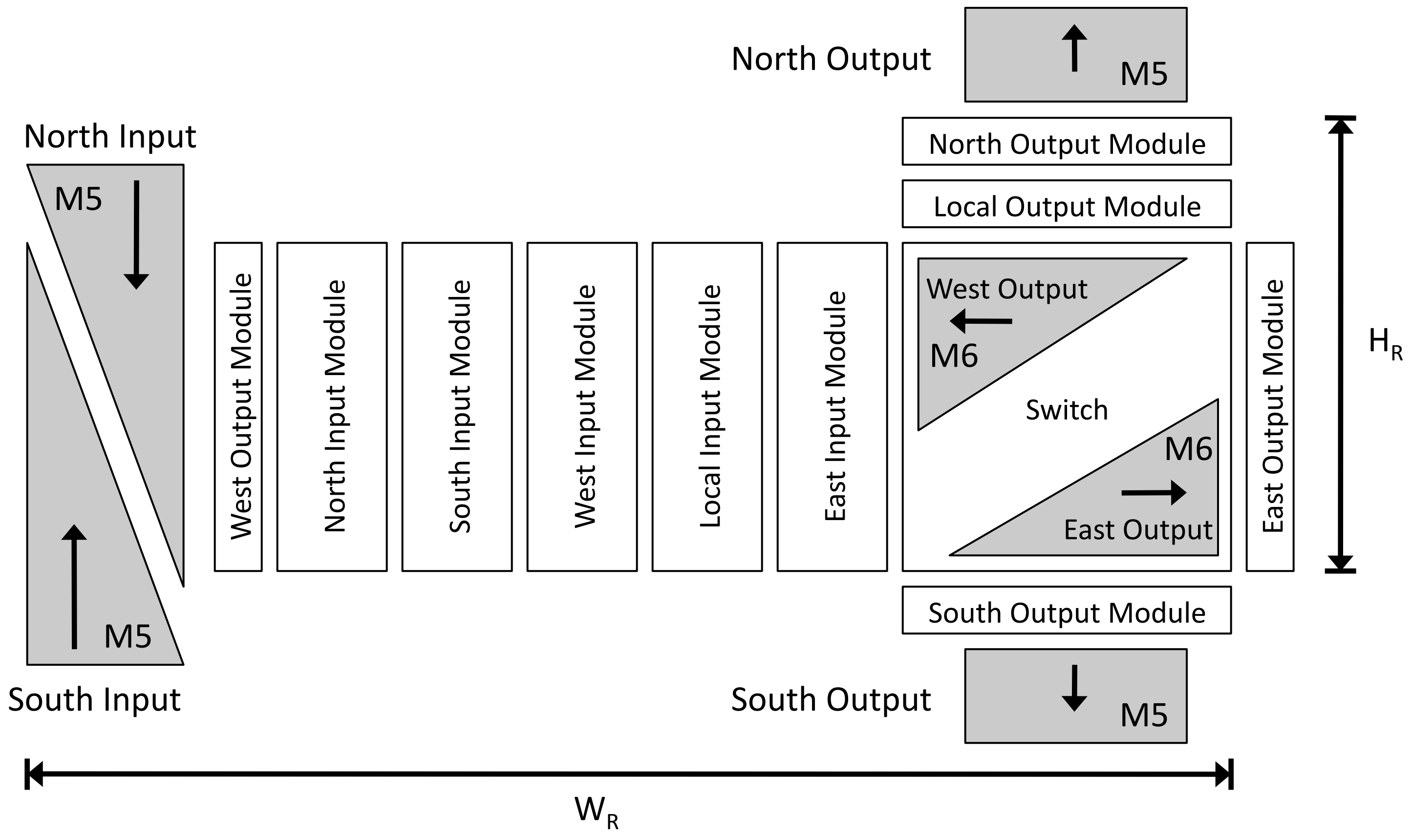
Router Floorplanning

- Determining placement of ports, allocators, switch
- Critical path delay
  - Determined by allocators and switch traversal

Router Floorplanning


P0
P1
P2
P3
P4
BF
SA + BFC + VA
Req Grant + misc control lines (0.043 mm²)
16 Bytes 5x5 Crossbar (0.763 mm²)
Req Grant + misc control lines (0.043 mm²)
(0.016 mm²)
(0.041 mm²

Router Floorplanning

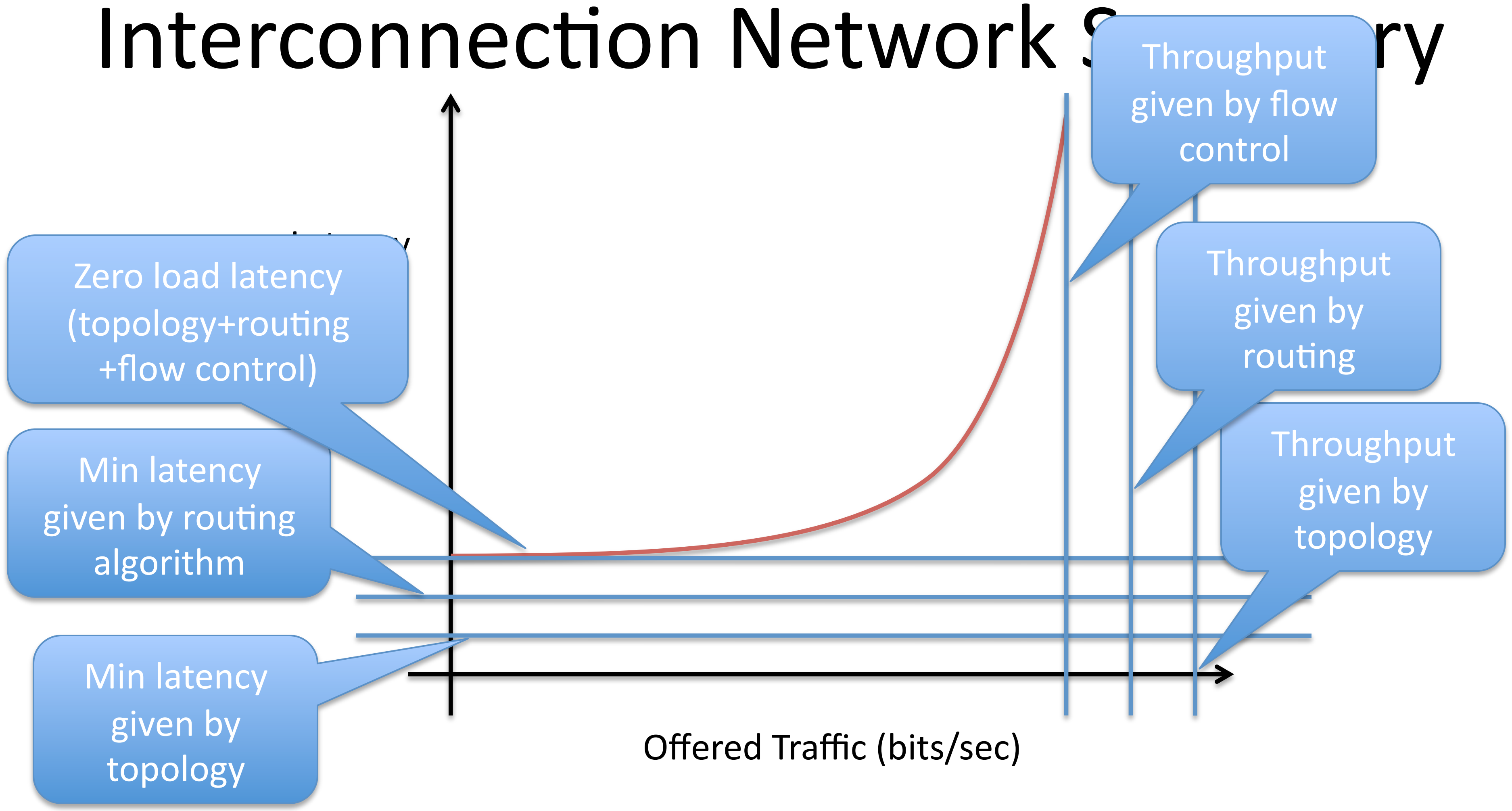


- Placing all input ports on left side
  - Frees up M5 and M6 for crossbar wiring

Microarchitecture Summary

- Ties together topological, routing and flow control design decisions
- Pipelined for fast cycle times
- Area and power constraints important in NoC design space

Interconnection Network S ry



- Latency vs. Offered Traffic

Towards the Ideal Interconnect

- Ideal latency
  - Solely due to wire delay between source and destination

$$T_{ideal} = \frac{D}{v} + \frac{L}{b}$$

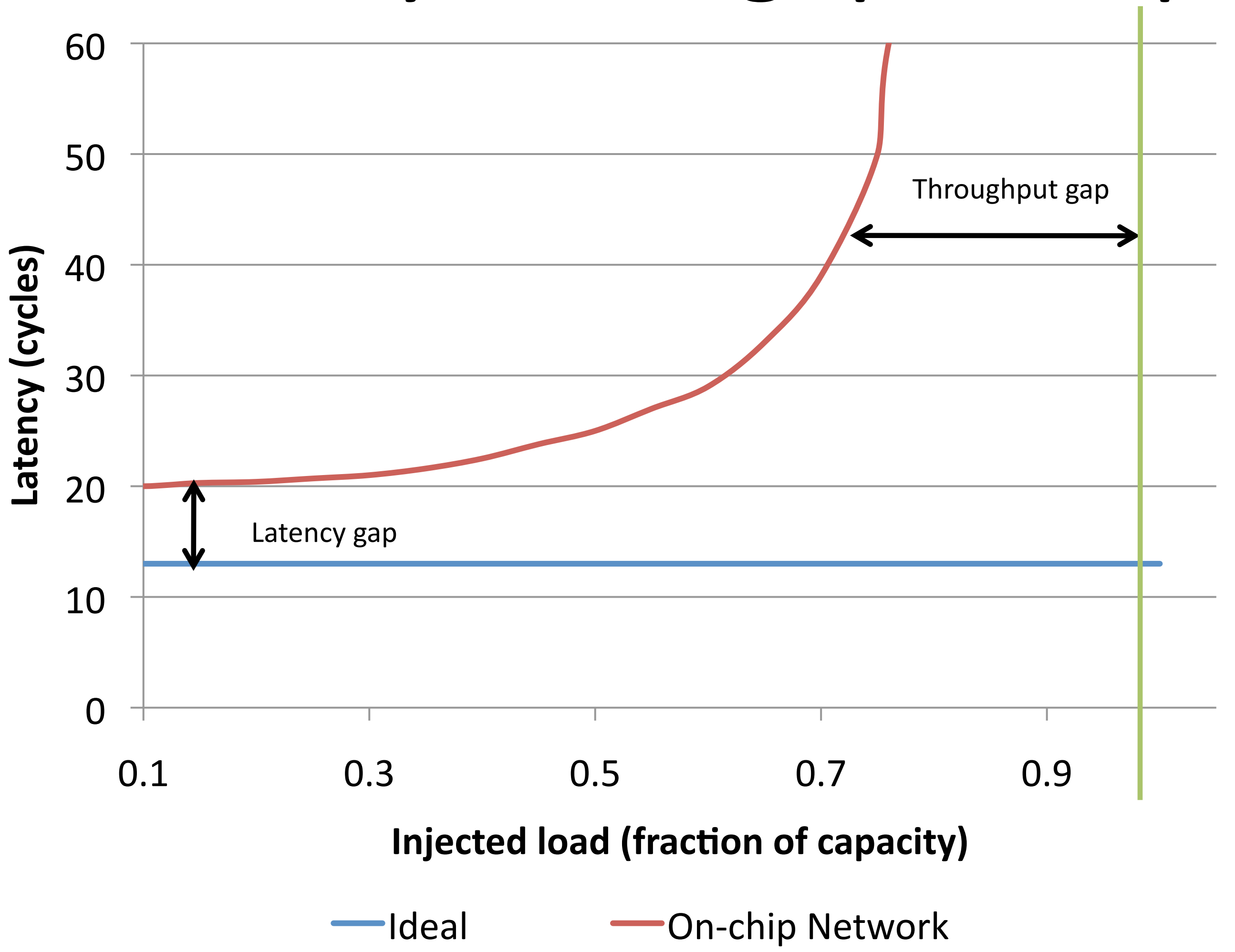
  - D = Manhatten distance
  - L = packet size
  - b = channel bandwidth
  - v = propagation velocity

State of the Art

- Dedicated wiring impractial
  - Long wires segmented with insertion of routers

$$T_{actual} = \frac{D}{v} + \frac{L}{b} + H \cdot T_{router} + T_c$$

Latency Throughput Gap



- Aggressive speculation and bypassing
- 8 VCs/port

Towards the Ideal Interconnect

- Ideal Energy
  - Only energy of interconnect wires

$$E_{ideal} = \frac{L}{b} \cdot D \cdot P_{wire}$$

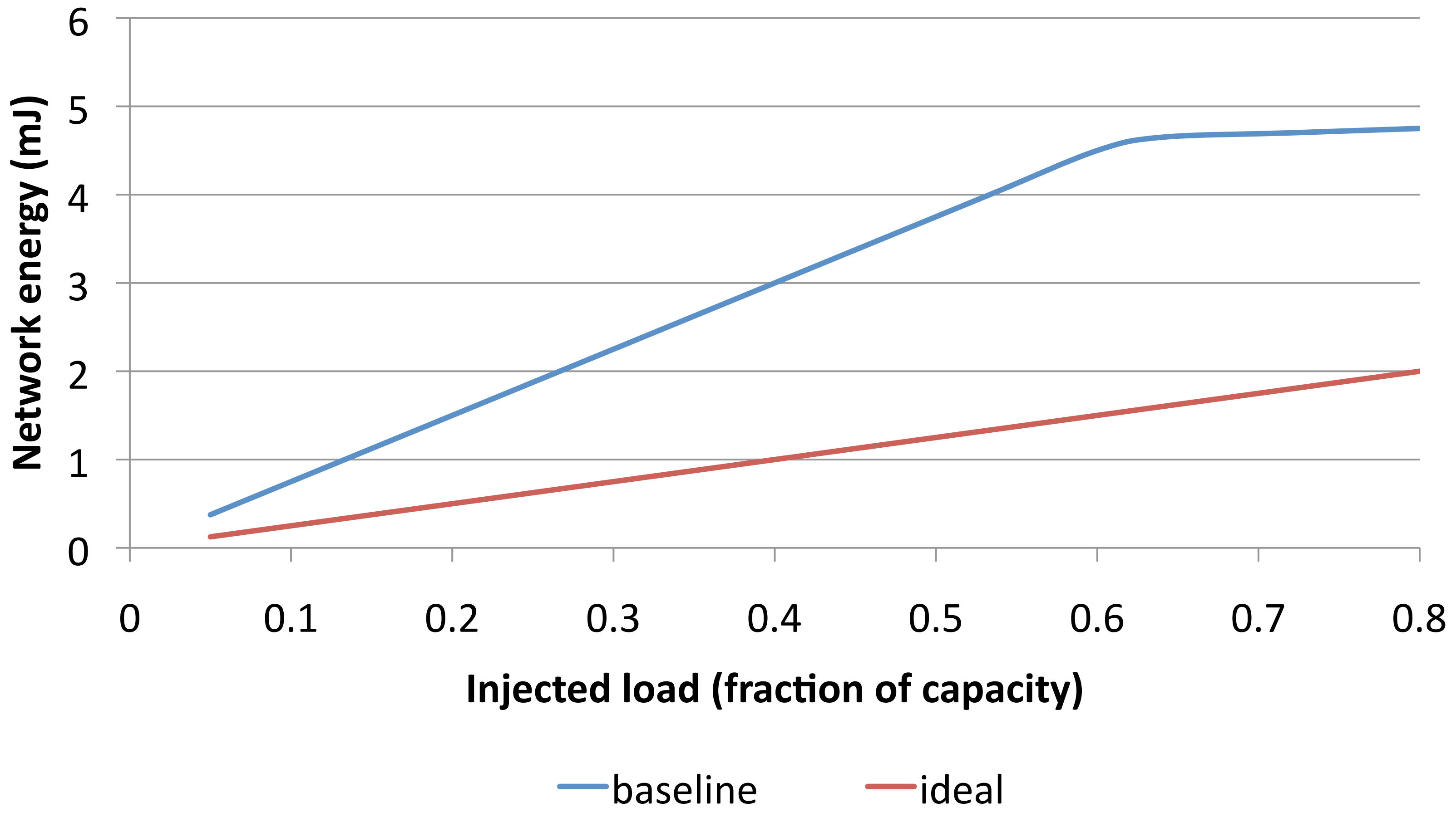
  - D = Distance
  - P_{wire} = transmission power per unit length

State of the Art

- No longer just wires
  - P_{router} = buffer read/write power, arbitration power, crossbar traversal

$$E_{actual} = \frac{L}{b} \cdot \left(D \cdot P_{wire} + H \cdot P_{router} \right)$$

Power Gap



Network energy (mJ)
0
1
2
3
4
5
6
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Injected load (fraction of capacity)
baseline
ideal

Key Research Challenges

- Low power on-chip networks
  - Power consumed largely dependent on bandwidth it has to support
  - Bandwidth requirement depends on several factors
- Beyond conventional interconnects
  - Power efficient link designs
  - 3D stacking
  - Optics
- Resilient on-chip networks
  - Manufacturing defects and variability
  - Soft errors and wearout

Next Week

- Paper 1: Flattened Butterfly
  - Presenter: Robert Hesse
- Paper 2: Design and Evaluation of a Hierarchical On-Chip Interconnect
  - Presenter: Jason Luu
- Paper 3: Design Trade-offs for Tiled CMP On-Chip Networks
- Paper 4: Cost-Efficient Dragonfly

- Two critiques due at the start of class