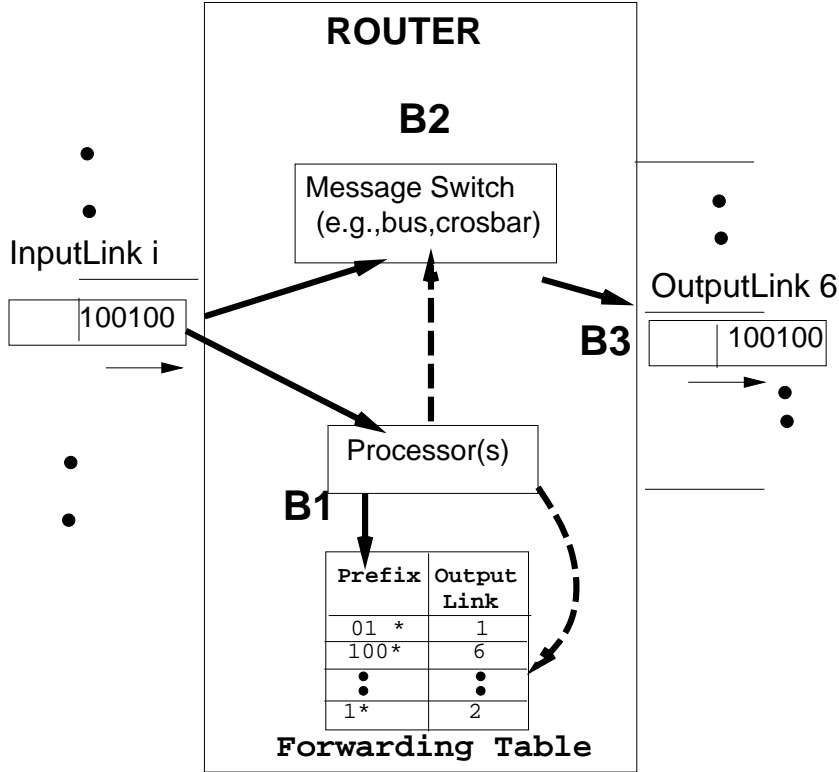


Switching

George Varghese

January 13, 2005

Three Central Bottlenecks in a Router



- Lookups (Chapters 10,11,12), Switching (this lecture), and Scheduling (Chapter 14)

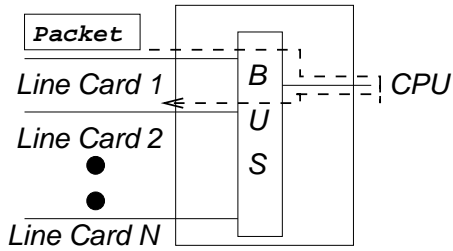
Outline

- Shared Memory Switches
- Bus versus Crossbar
- Simple Input Queued Switches (Gigaswitch)
- Head of Line (HOL) Blocking
- Output Queued Switches (Knockout) solution to HOL
- Solving HOL via Input Queued Switches, Virtual Queues and Randomization (PIM)
- Solving HOL via Input Queued Switches, Virtual Queues and without Randomization (iSLIP)

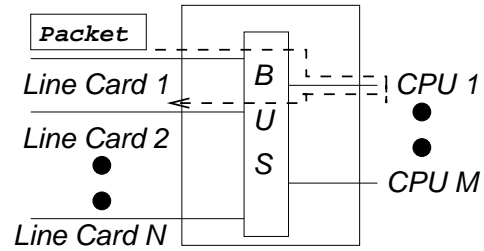
Shared Memory Switches

- Packets read into memory from input links; read out of memory to appropriate output links.
- Main problem: memory bandwidth. With 8 input and 8 need memory 16 times faster than link speeds.
- Solve, up to a point, using wide memory access width. Bits come in serially, accumulated into an input shift register, load cell into cell-wide. Reverse process on output.
- Datapath uses central memory of 4K cells on-chip, augmented with flow control and off-chip packet buffers. Does not scale.
- But are memory and power optimal because data is only moved *once*. New work in this old idea using interleaved DRAMs

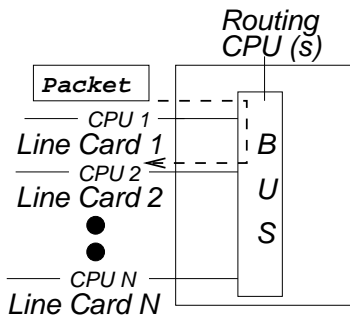
Router Evolution from Bus to Crossbar



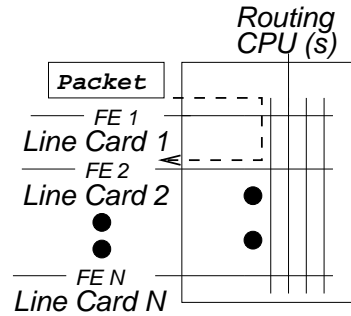
A) PALEOZOIC: BUS, SHARED CPU



B) PALEOLITHIC: BUS, SHARED CPU's

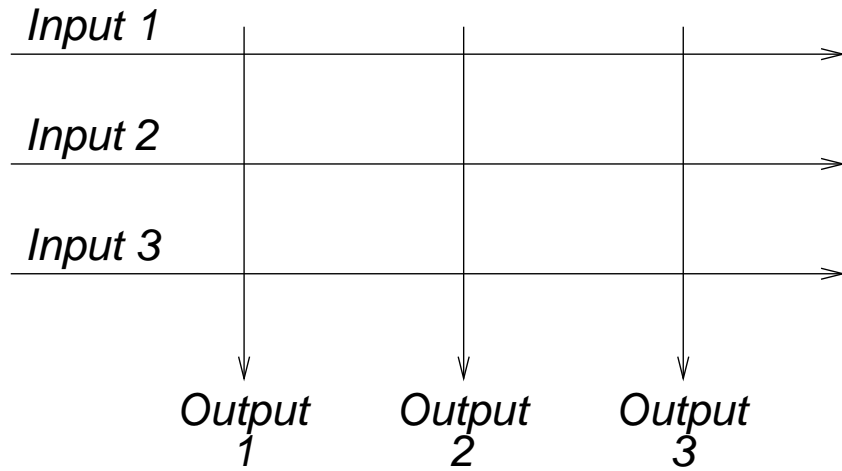


C) NEOLITHIC: BUS, PER LINE CARD CPU's



D) MODERN: CROSSBAR, PER LINE CARD FORWARDING ENGINES

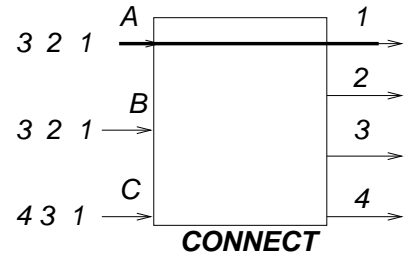
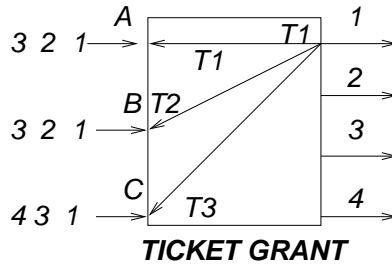
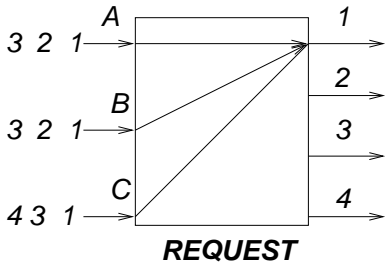
Basic Crossbar



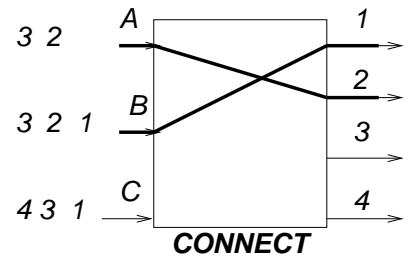
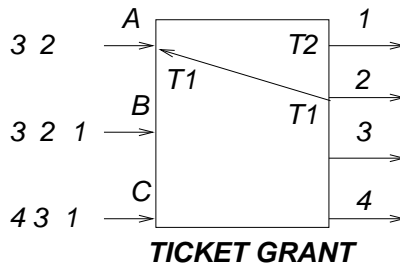
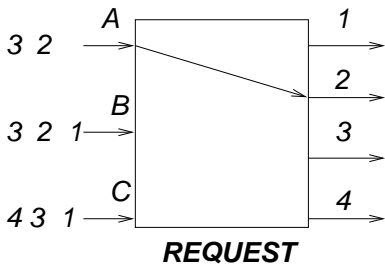
- Potential N -fold parallelism over single bus throughput.
- But how to schedule, and prevent two inputs on same output bus?

Take-a-ticket Scheduling in Gigaswitch

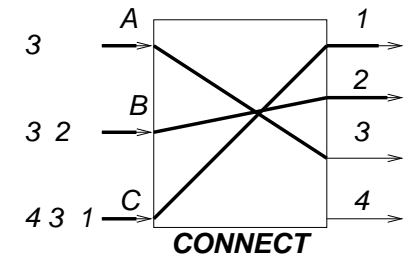
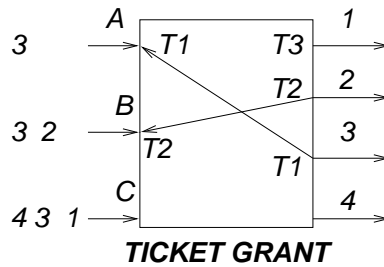
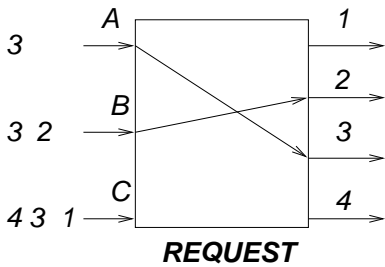
Round 1



Round 2

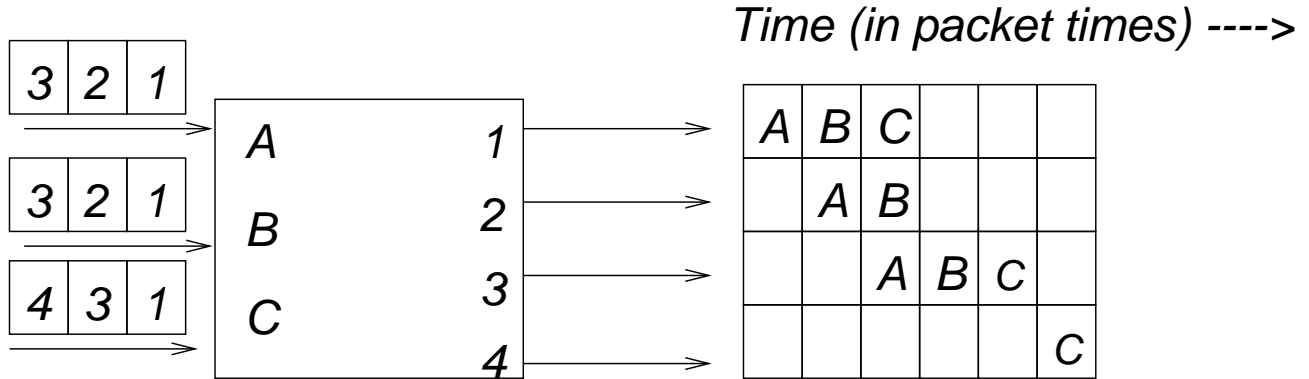


Round 3



- Each output maintains a queue for all inputs waiting to send.
- Queue stored at inputs using a simple ticket number mechanism used.

Head-of-Line Blocking



- Each queue held hostage by the progress of head
- Such *head of line* (HOL) blocking reduces parallelism opportunities shown as blanks.

HOL Throughput Loss Analysis

- Assume each input packet destined to each of N outputs with probability $1/N$.
- With equal size packets and one initial trial, probability that an output O is idle is probability that *none* of the inputs choose O .
- Each input does not choose O with probability $1 - 1/N$. Hence probability O is idle is $(1 - 1/N)^N$, converges to $1/e$.
- Thus probability that output is busy is $1 - 1/e$ which is 0.63. More accurate analysis by Karol et al shows 58% throughput

Output Queuing

- Draconian measure: get rid of input queuing and queue on output. Creates as many problems as it solves.
- Can run fabric N times faster than input but this is very expensive.
- One practical solution via Knockout switch

Knockout Idea

- Assume receiving N cells to same destination in any cell time is rare, and expected number is $k \ll N$.
- Optimize expected case (**P11**) by using fabric speedup of k . Realize (**P5**) by k parallel busses.
- Also needs output queue that accepts cells k times faster than output link speed.
- A distributor (that does run k times as fast) sprays arriving cells into k memories in round-robin order, and departing cells are read out in the same order.

Fair discarding in Knockout

Great attention paid to fair dropping via a hardware tournament for case when $N > k$ cells get sent to the same output.

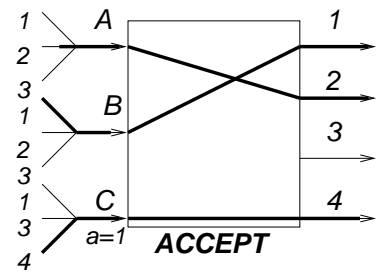
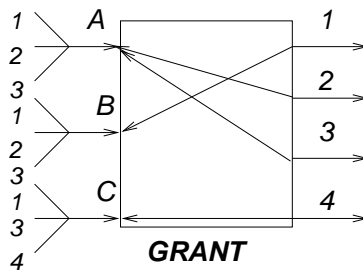
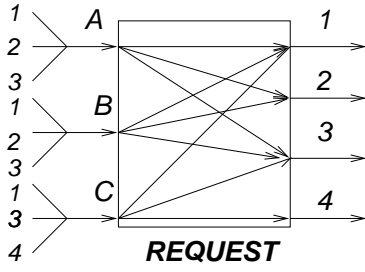
- *Two Contenders, one winner:* Choose one cell fairly from two choices. 2 by 2 hardware *concentrator* randomly picks a winner and a loser.
- *Many contenders, one winner:* Suppose $k = 1$ and $N > 2$. Create a *knockout tree* of 2 by 2 concentrators as in a tennis match to create a winner.
- *Many contenders, more than one winner:* Naive idea would be to create $k > 1$ knockout trees. Complexity: for fairness, losers of earlier knockout trees have to be considered for later trees.
- Naive way: start Position j tree strictly after logic for Position $j - 1$ tree to collect all eligible losers. Done faster via pipelining. Lovely picture in Peterson and Davy!

Parallel Iterative Matching

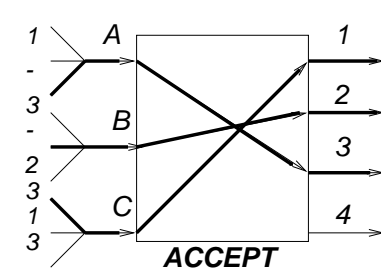
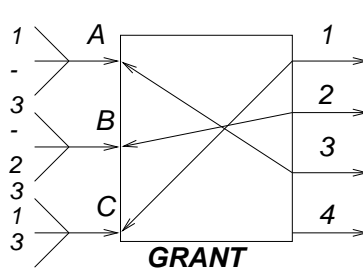
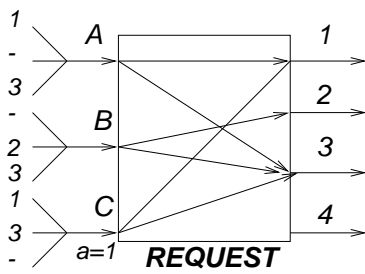
- Reconsider input queuing, but retrofit it to avoid Head-of-Line blocking by allowing an input to schedule not just head but *all* cells that can make progress. Feasible via 2 observations.
- *Observation 1:* Need only work on scheduling to N possible output queues, as cells in same input to same output cannot be scheduled concurrently. Decompose single input queue into separate input queue per output at each input called *virtual output queues (VOQs)*.
- *Observation 2:* Communicating scheduling needs of any input port takes only a bitmap of N bits which is feasible for $N < 64$.

PIM in Action

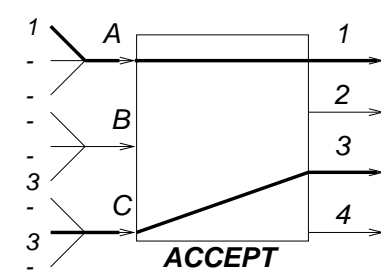
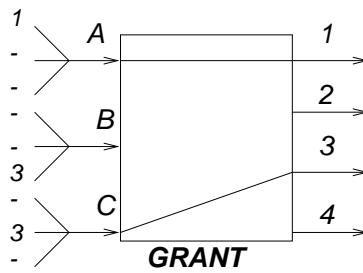
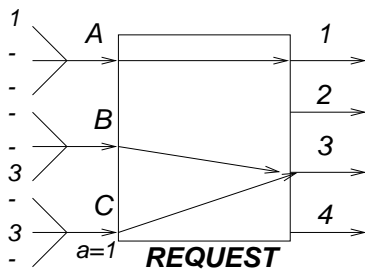
Round 1



Round 2



Round 3



Avoiding randomization with iSLIP

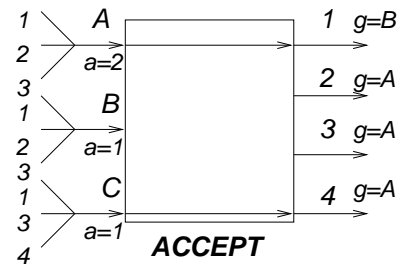
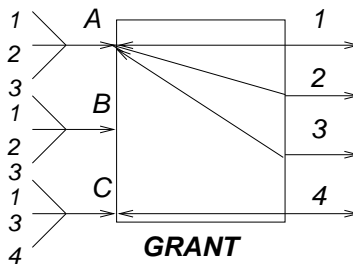
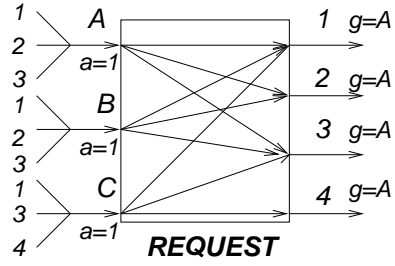
- PIM has two potential problems. First, it uses randomization, which can be hard to implement right and requires a logarithmic number of iterations to attain maximal matches.
- iSLIP essentially “derandomizes” PIM and gets close to maximal matches after just one or two iterations.
- Base idea: while in PIM, an output port chooses an winning input randomly for fairnes (like Ethernet), iSLIP does so using a round robin pointer (like Token Rings).

iSLIP Mechanisms

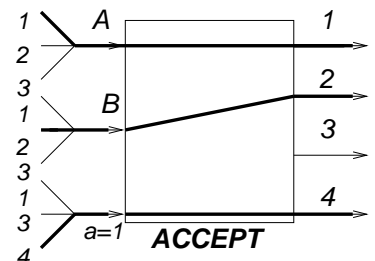
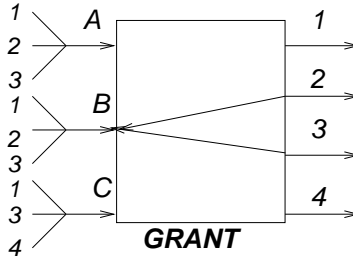
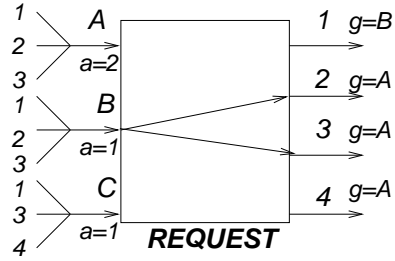
- Each output (respectively input) maintains a pointer g (respectively a) initially set to the first input (respectively output) port.
- When an output has to choose between multiple input requests, it chooses the lowest input number $\geq g$. Similarly, when an input port has to choose between multiple output port requests, it chooses the lowest output port number that is $\geq a$.
- If an output port is matched to an input port X , then the output port pointer is incremented to the first port number greater than X in circular order.
- “rotating priority” allows fairness at the cost of $2N$ extra $\log_2 N$ pointers in addition to the N^2 scheduling state needed.

iSLIP in action

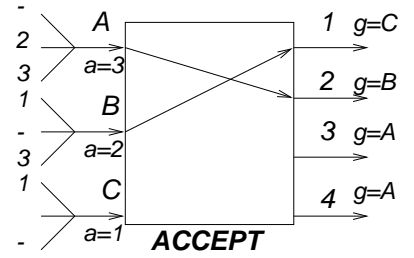
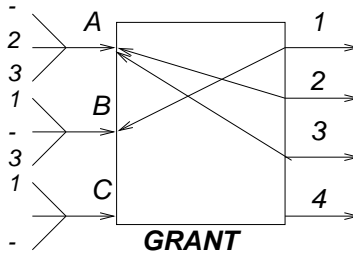
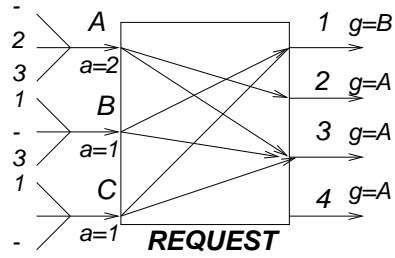
Round 1, Iteration 1



Round 1, Iteration 2

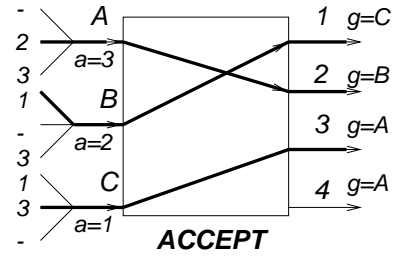
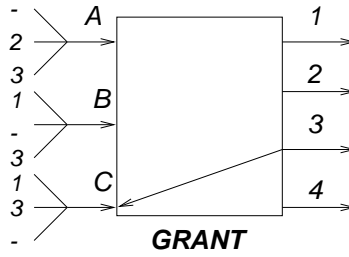
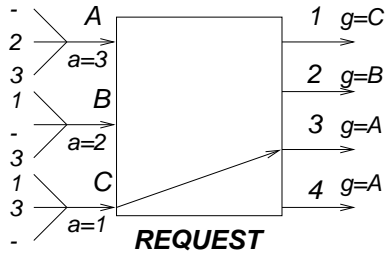


Round 2, Iteration 1

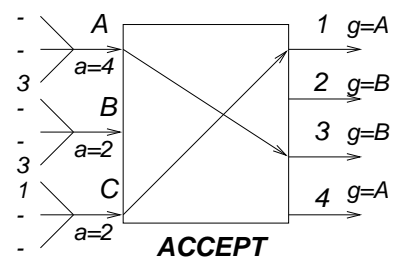
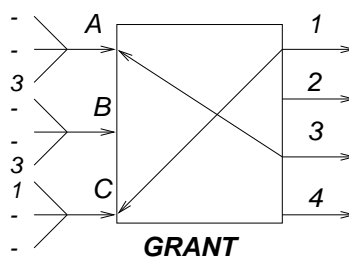
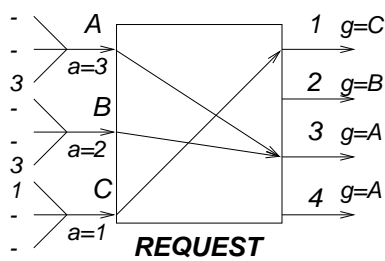


iSLIP in action continued

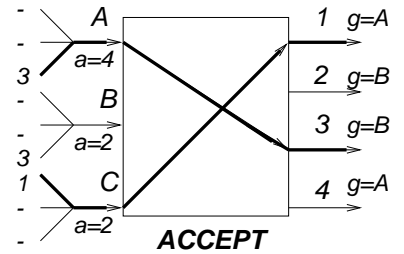
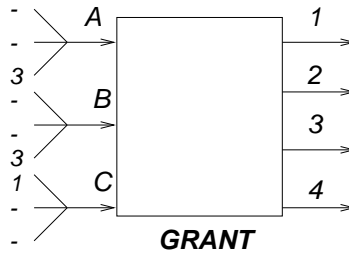
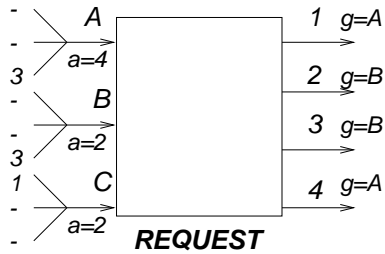
Round 2, Iteration 2



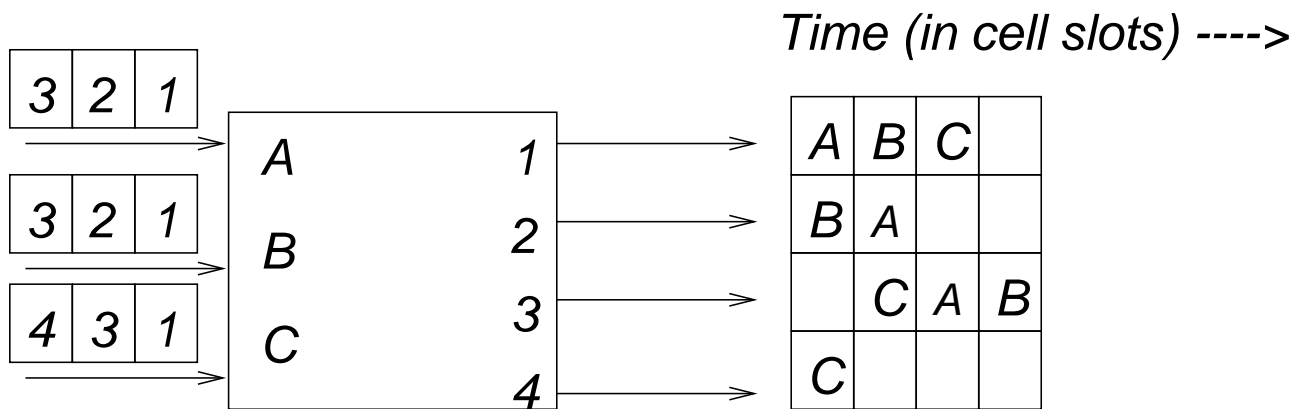
Round 3, Iteration 1



Round 3, Iteration 2



So why does iSLIP help in throughput?



- Compare with corresponding picture for Take-a-ticket. 4 cell times to finish backlog compared to 6.

iSLIP subtleties

- *Startup Cost*: Poor parallelism at start but as soon as pointers desynchronize, iSLIP does very well 1 iteration. Cisco GSR.
- **Grant pointers incremented only after accept**: Not doing so can cause traffic from I to O to be persistently starved, and also causes pointers to synchronize and reduce bandwidth.
- **All pointers incremented only after the first iteration accept is granted**: Once again, this rule prevents starvation but the scenario is more subtle.

iSLIP Implementation Notes

- Hardware implementation via a Programmable Priority Encoder (see Chapter 2). Easy to put all PPEs in a single centralized chip and route control wires from ports. To deal with latency of centralization, can work with a pipeline of m cells from each VOQ.
- For multiple iterations, appears that *Grant* of iteration $k + 1$ must started after accept of iteration k . Can partially pipeline by simple observation: if input I receives *any* grant in iteration k , then I must accept exactly one and so be unavailable in iteration $k + 1$.
- Thus Grant of iteration $k + 1$ can start immediately after the Grant phase of iteration k , overlapping with the Accept phase of iteration k . Reduces the overall completion time by nearly a factor of two.

Scaling to Larger Switches

- Tiny Tera and GSR only do small switches
- With forces like DWDM would like to have larger switches. Juniper T-series and Cisco CRS-1. See text for some strategies via interconnection networks such as Benes and Delta networks.