

Networks-on-Chip (NoC)

Mark McDermott

Introduction

- **Paradigm shift in SOC design**

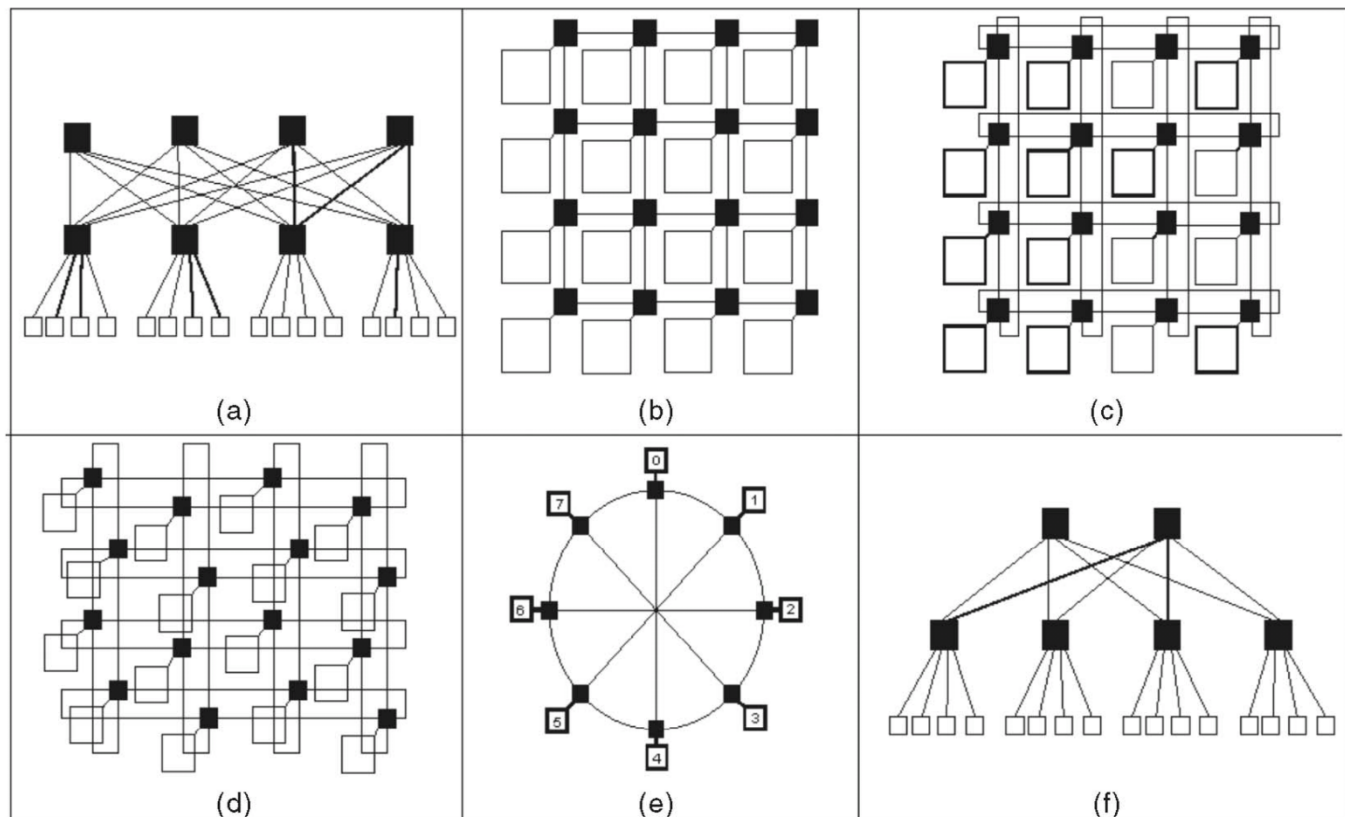
- **Communication challenges**
 - Synchronous communication infeasible
 - Errors due to integrity issues
 - RLC effects
 - Cross-coupling effects
 - Delay Insensitive may be a better approach.

- **Network-on-Chip (NoC)**
 - Packet switching based communication.
 - Extremely high bandwidth by pipelined signal transmission.
 - Synchronous and asynchronous (delay insensitive) communication between routers.
 - Support for error control schemes.

Topologies

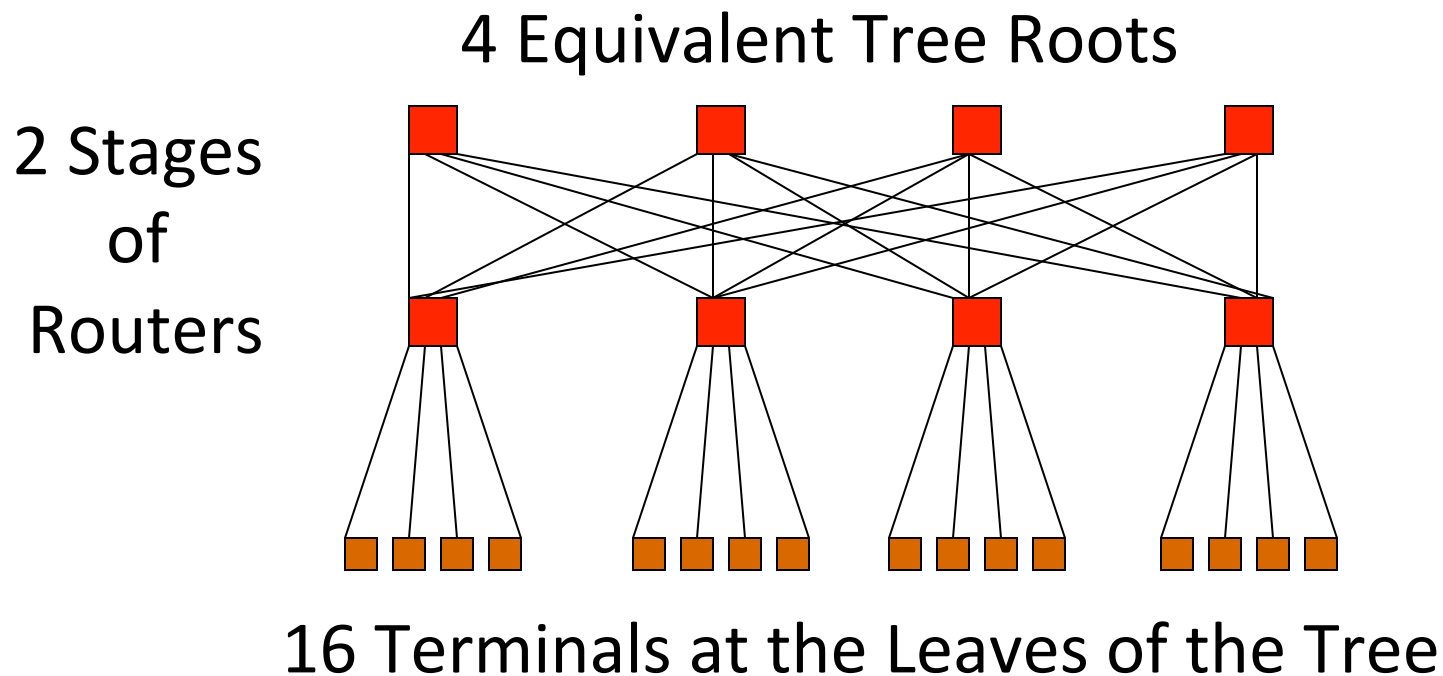
- **Heritage of networks with new constraints**
 - Need to accommodate interconnects in a 2D layout
 - Cannot route long wires (clock frequency bound)

- a) SPIN,
- b) CLICHE' & Mesh
- c) Torus
- d) Folded torus
- e) Octagon
- f) BFT



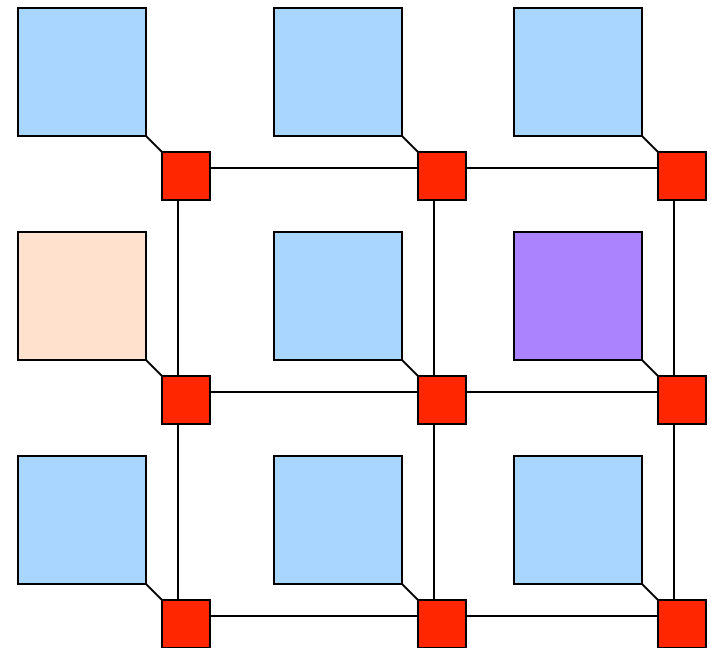
Architectures: SPIN

- **SPIN: Scalable, Programmable, Integrated Network**
 - Every level has same number switches
 - Network grows as $(N \log N)/8$
 - Trades area overhead and decreased power efficiency for higher throughput
 - Illustrative of performance vs. power consumption



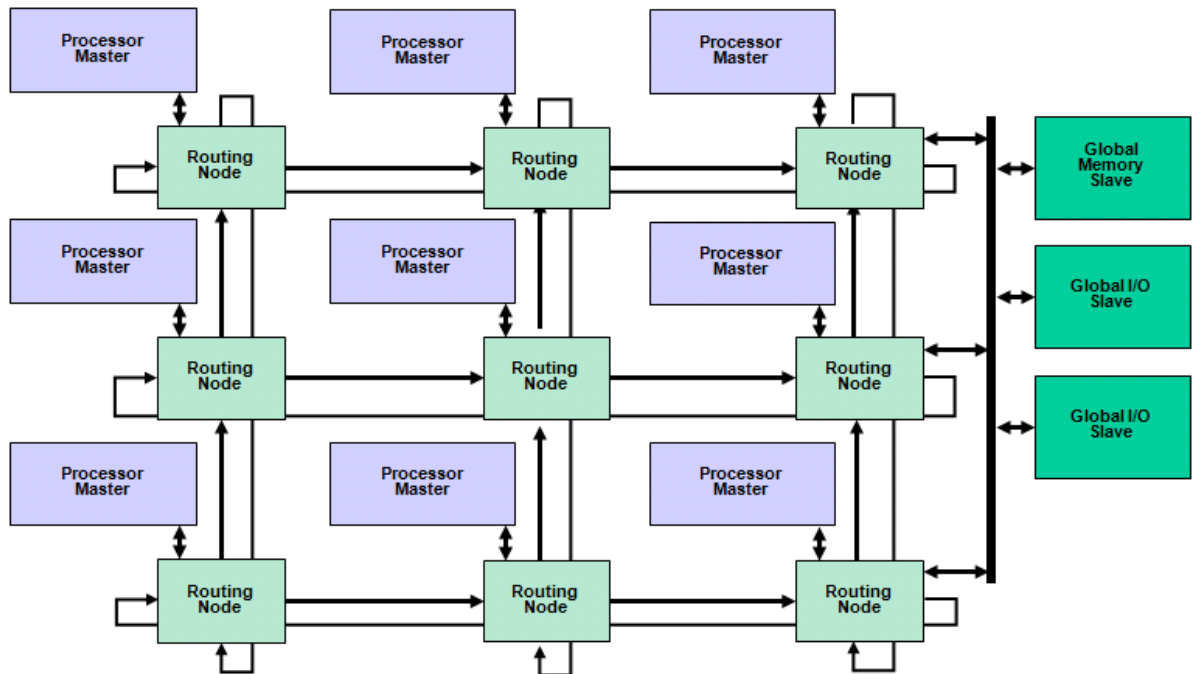
Architectures: CLICHE

- **CLICHÉ: Chip-Level Integration of Communicating Heterogeneous Elements**
 - Two-dimensional mesh network layout for NoC design
 - All switches are connected to the four closest other switches and target resource block, except those switches on the edge of the layout
 - Connections are two unidirectional links



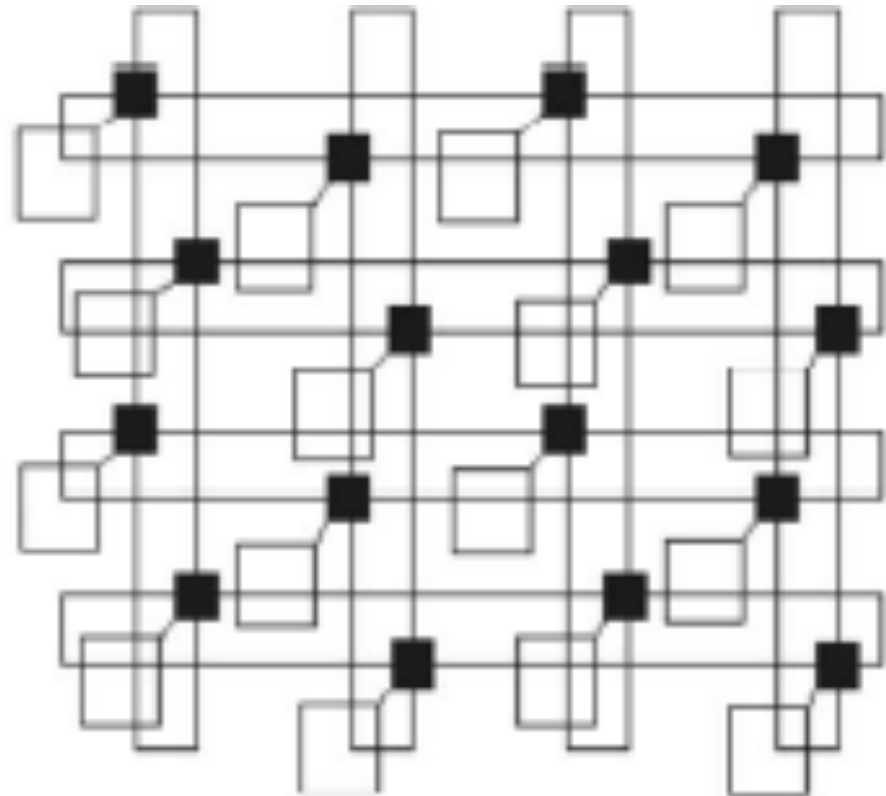
Architectures: Torus

- **Similar to mesh based architectures**
 - Wires are wrapped around from the top component to the bottom and rightmost to leftmost
 - Smaller hop count
 - Higher bandwidth
 - Decreased Contention
 - Increased chip space usage



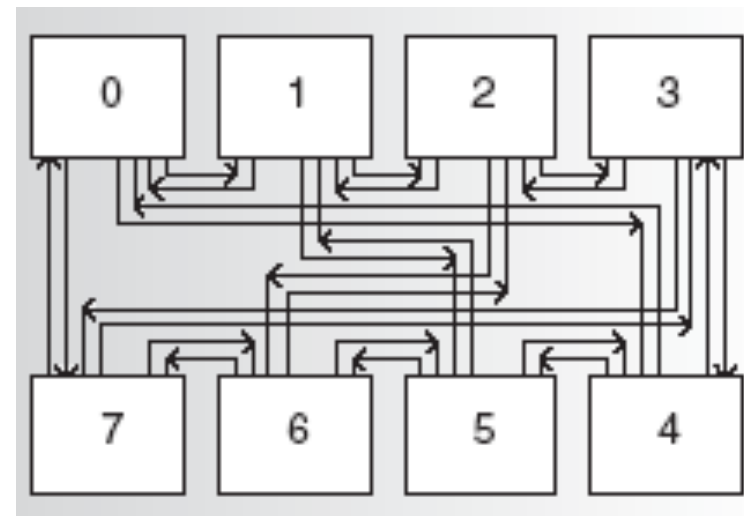
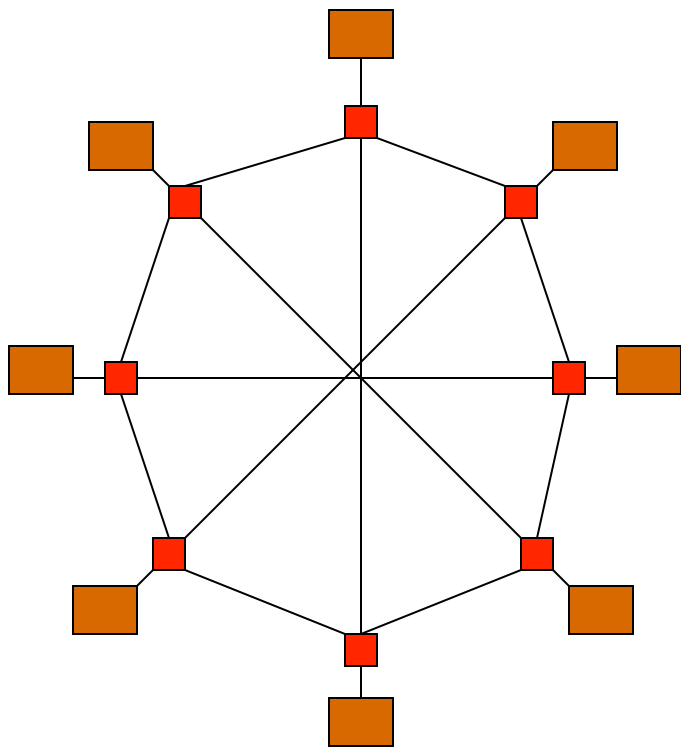
Architectures: Folded Torus

- **Similar to Torus**
 - Torus, the long end-around connections can yield excessive delays
 - Avoided by folding the torus



Architectures: Octagon

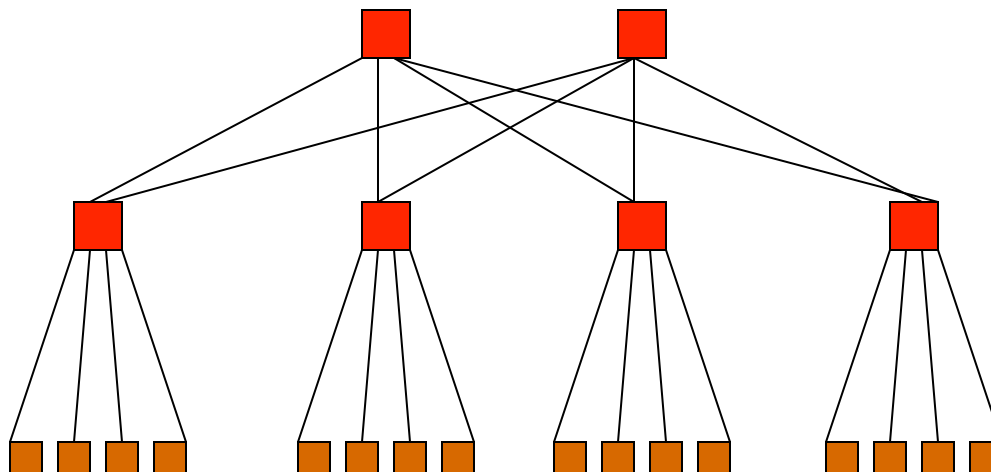
- **Standard model: 8 components, 12 interconnects**
 - Design complexity increases linearly with number of nodes
 - Largest packet travel distance is two hops
 - High throughput
 - Shortest path routing easy to implement



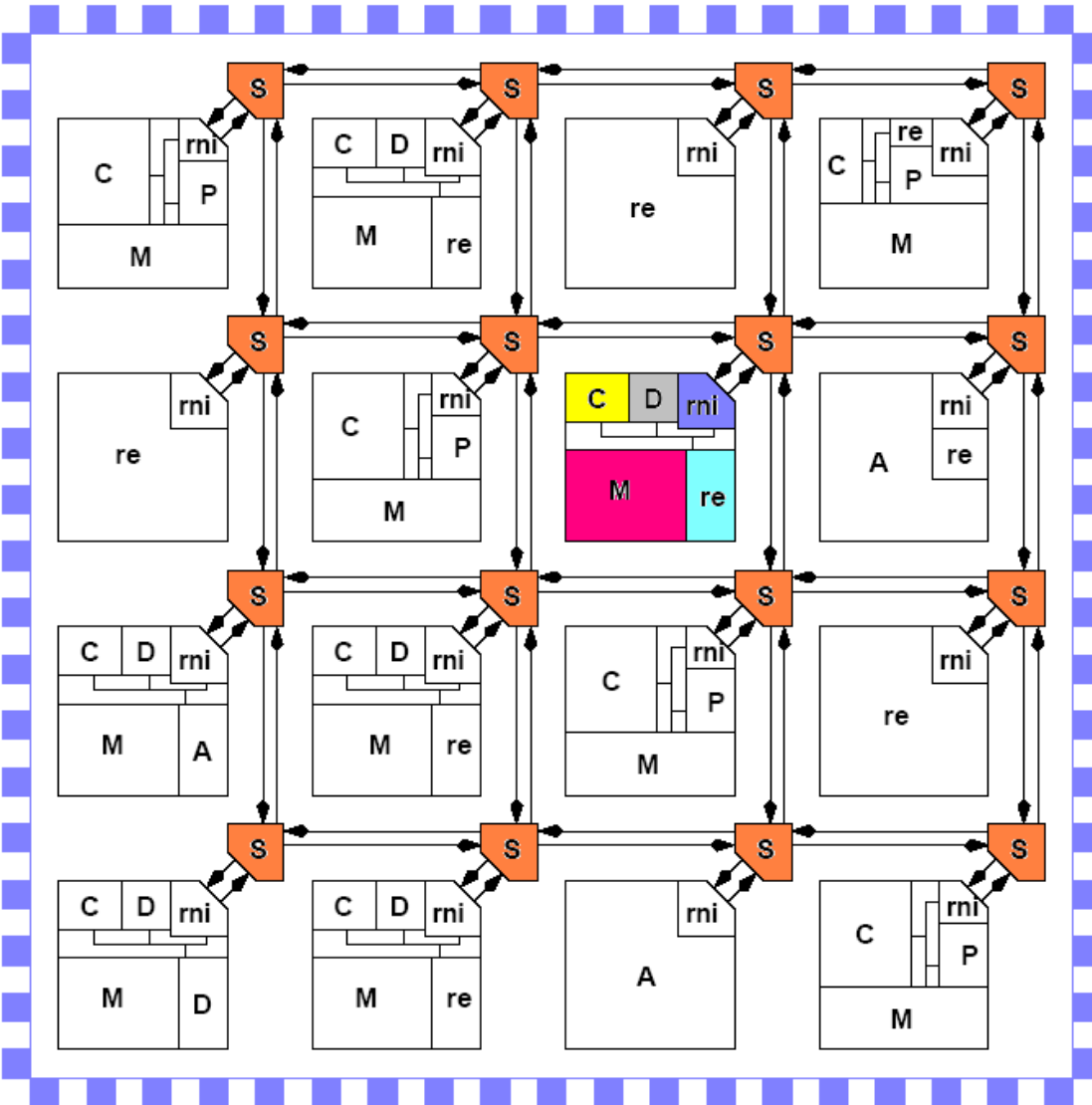
Architectures: BFT

■ BFT: Butterfly Fat Tree

- Each node in tree model has coordinates (level, position) where level is depth and position is from left to right
- Leaves are component blocks
- Interior nodes are switches
- Four child ports per switch and two parent ports
- $\log N$ levels, i th level has $n/(2^{i+1})$ switches, $n = \text{leaves (blocks)}$
- Use traffic aggregation to reduce congestion



Mesh based router architecture



S = switch/router

rni = resource network interface

C = cache

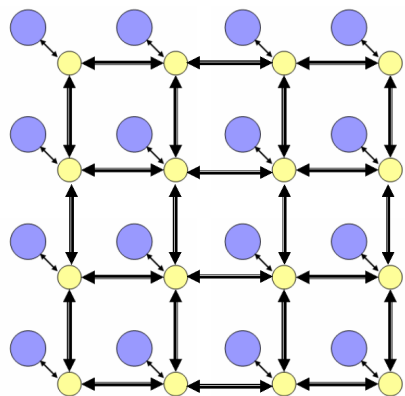
P = processor

M = memory

D = DSP

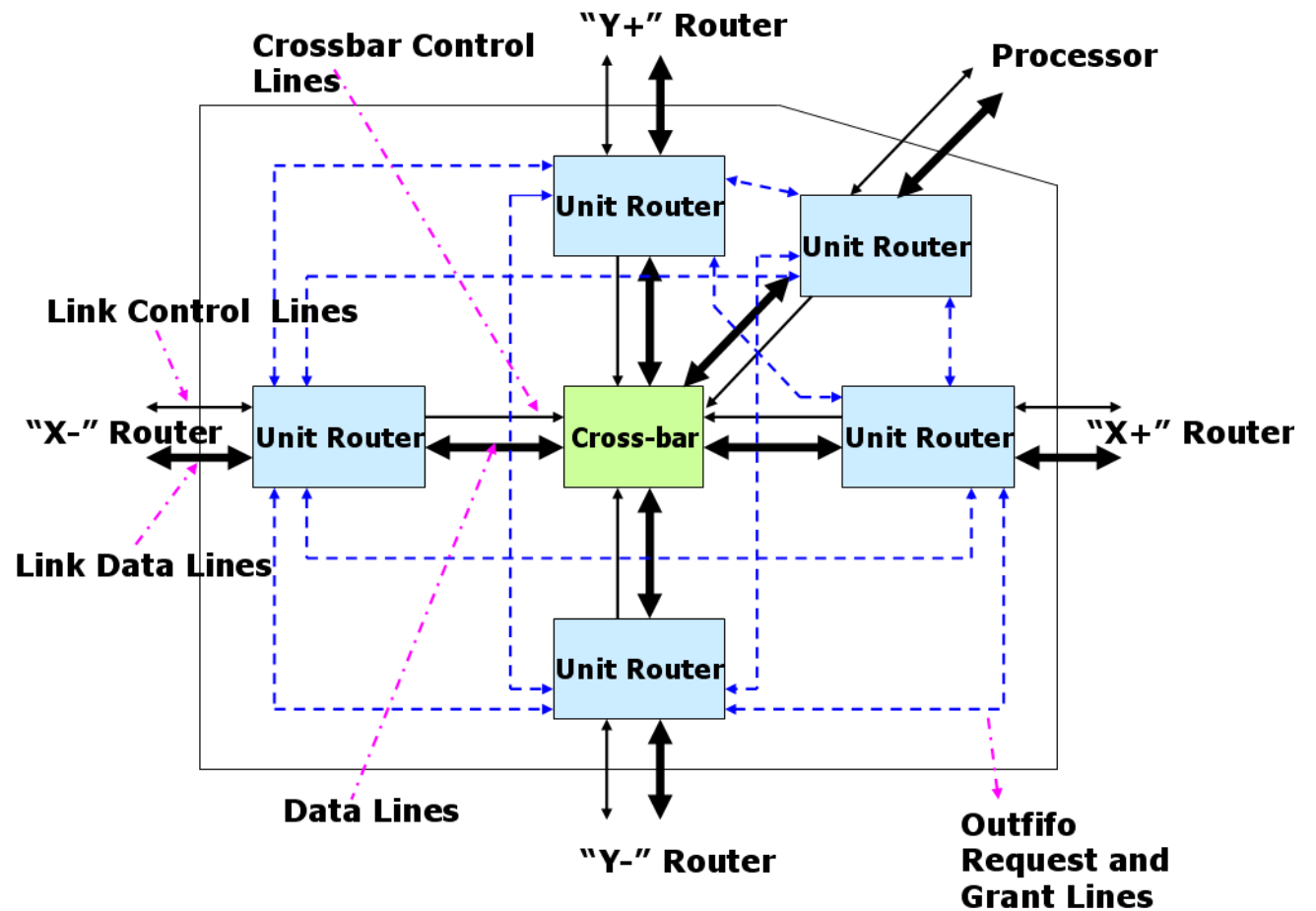
re = reconfigurable logic

Mesh based router architecture

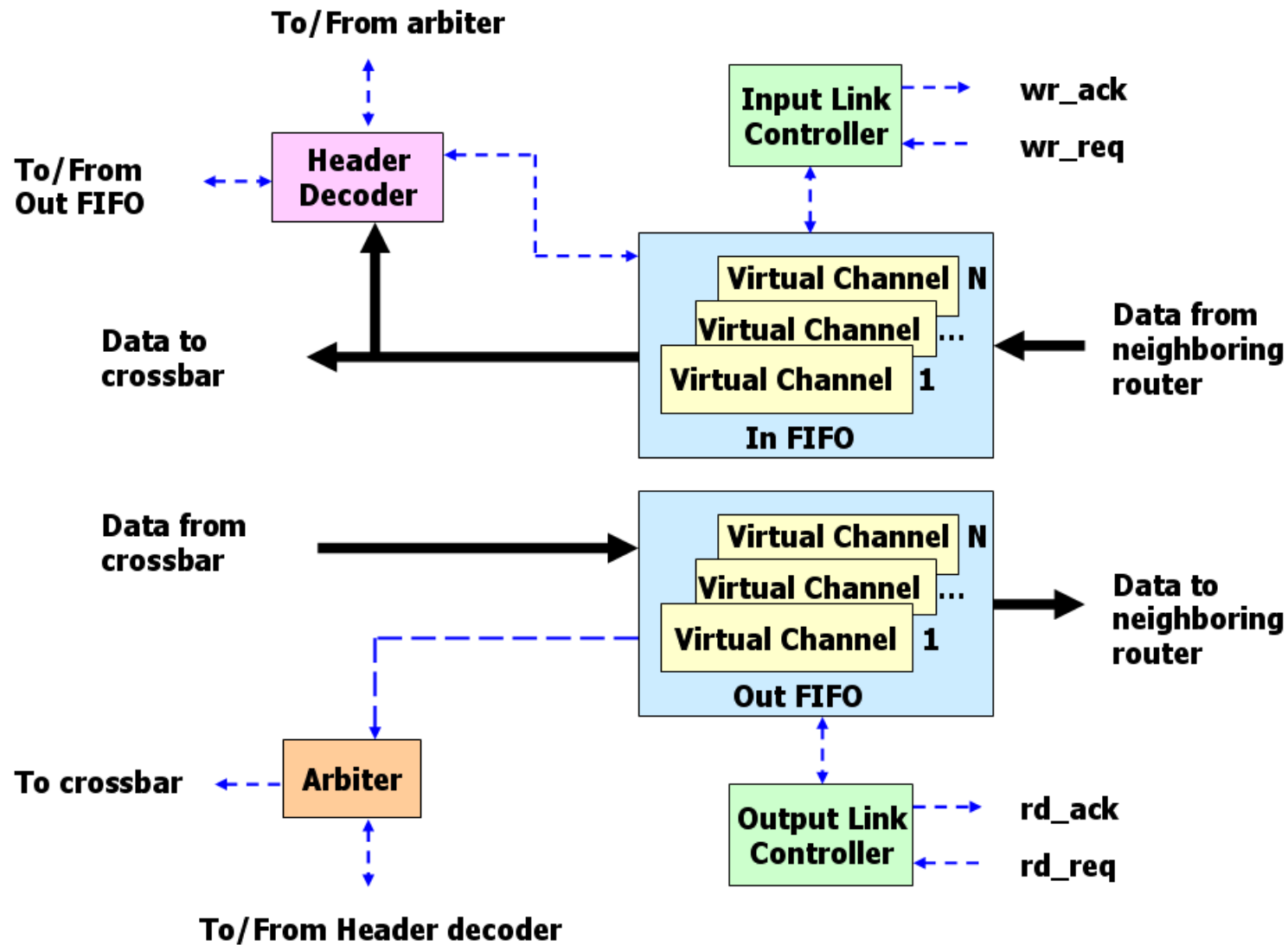


● Router

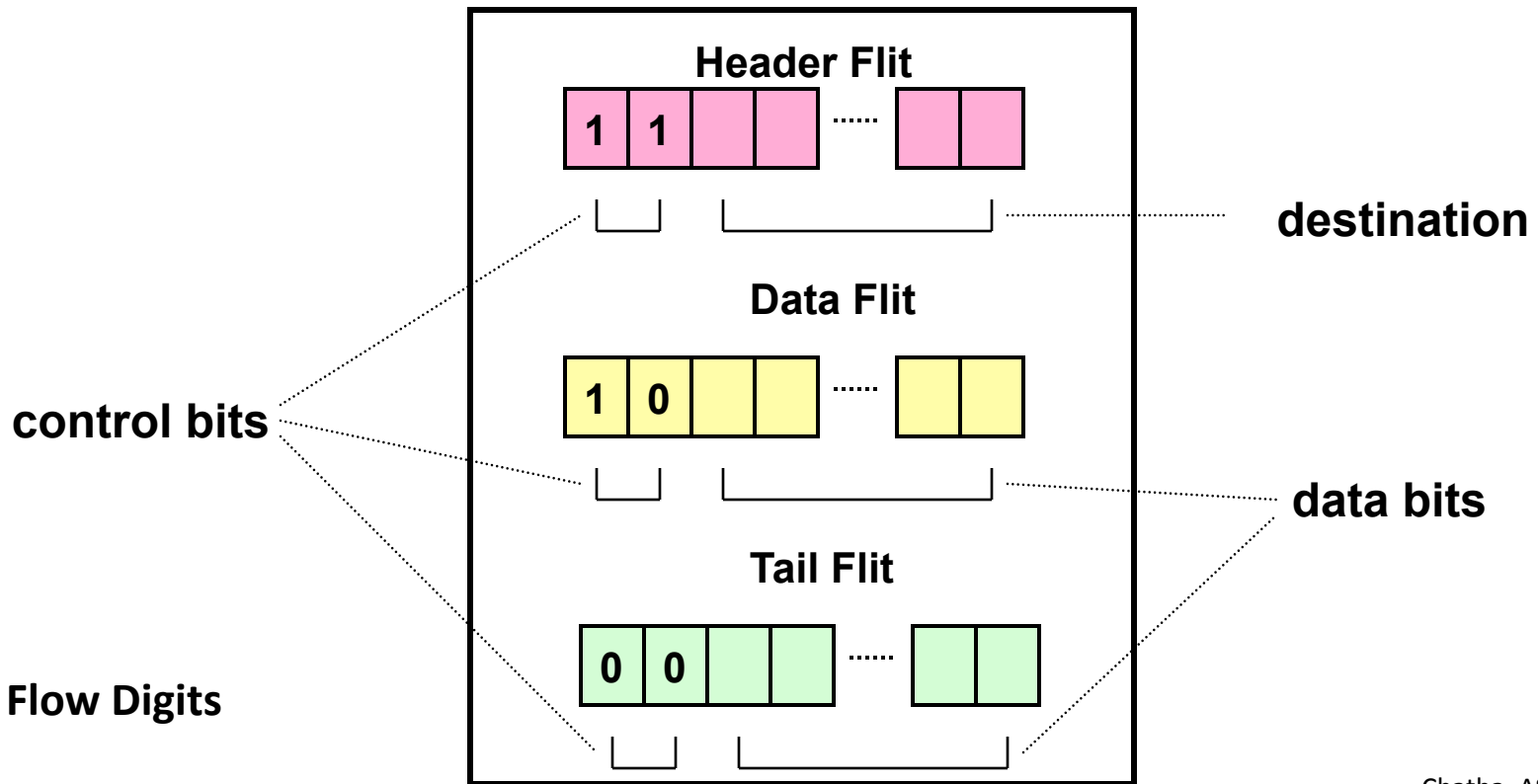
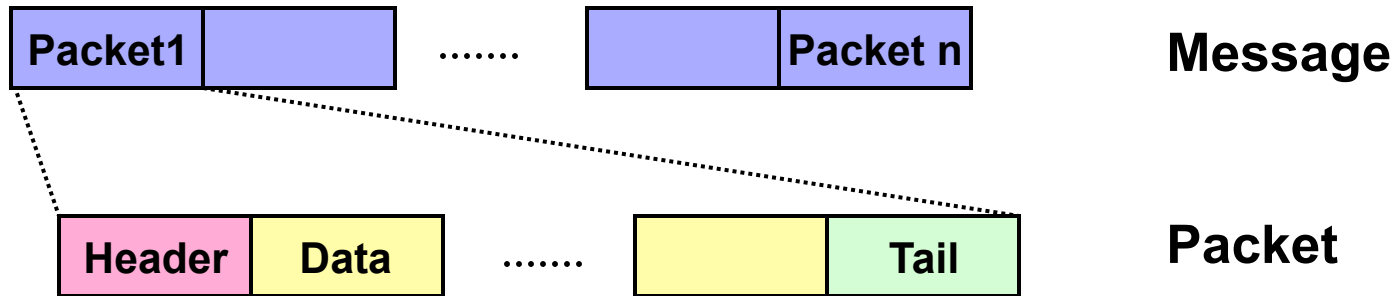
● Processor



Unit Router Architecture



Packet Format



FLIT: Flow Digits

Routing Performance metrics

- **Injection Rate**
 - Number of packets that are injected from the source into the network per unit time
- **Average Network Latency**
 - Average delay experienced by packets as they traverse from source to destination.
- **Average Setup Latency**
 - Average time required to reserve the virtual channels for the GT traffic from source to destination.
- **Acceptance rate**
 - Number of packets reaching each of the destination nodes per clock cycle at a particular injection rate.
- **Average power consumption**
 - Sum of average dynamic and leakage power consumed by the network per clock cycle.

Quality-of-Service levels

- **Guaranteed Throughput (GT)**
 - Throughput and latency guarantees
 - Supports bursty traffic
 - 3 stages – setup, transmit, tear-down
 - setup – virtual channels reserved from source to destination
 - transmit – packets transferred with maximum throughput
 - tear-down – path is set free

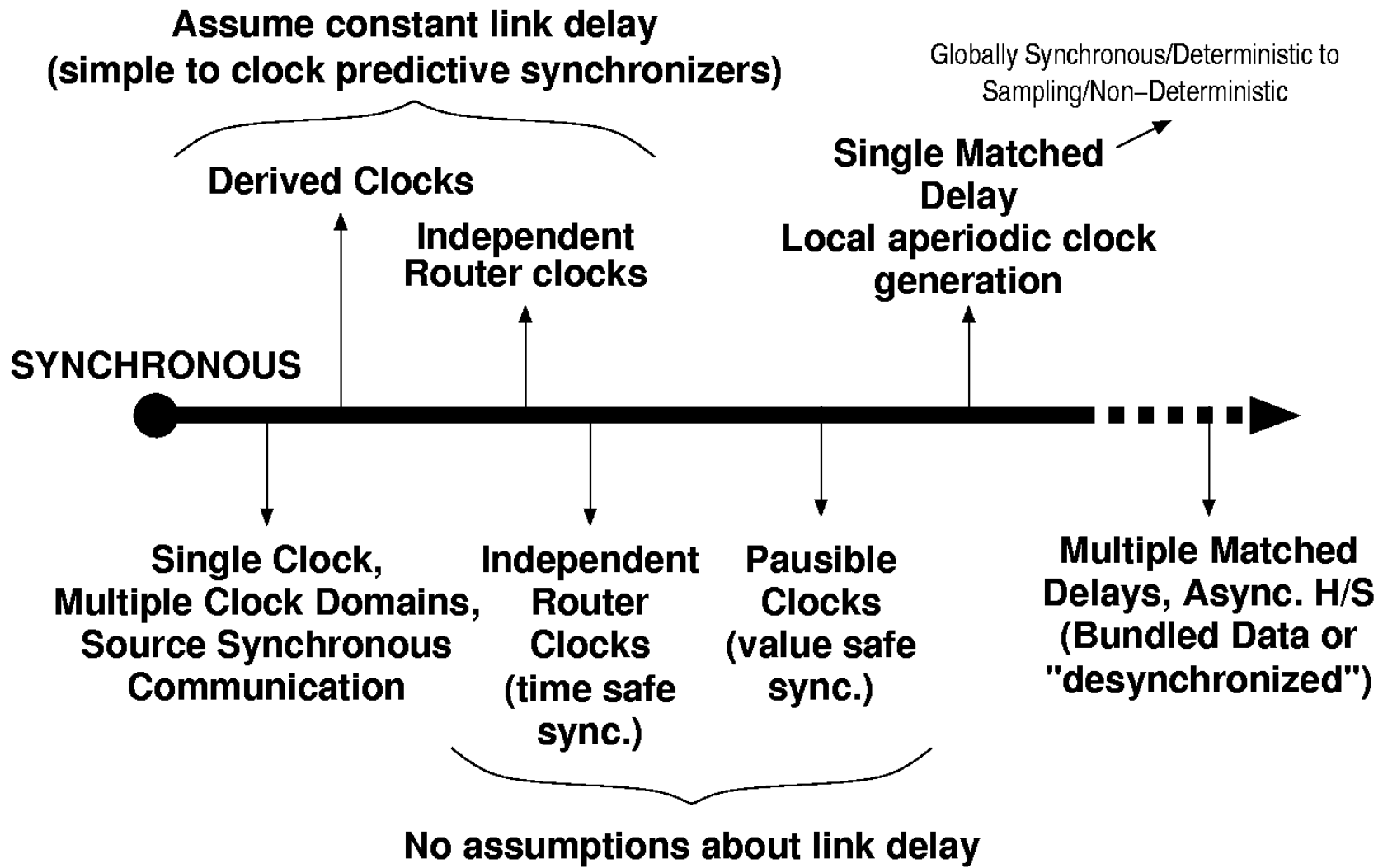
- **Best Effort (BE)**
 - no time guarantees

Quality-of-Service architecture

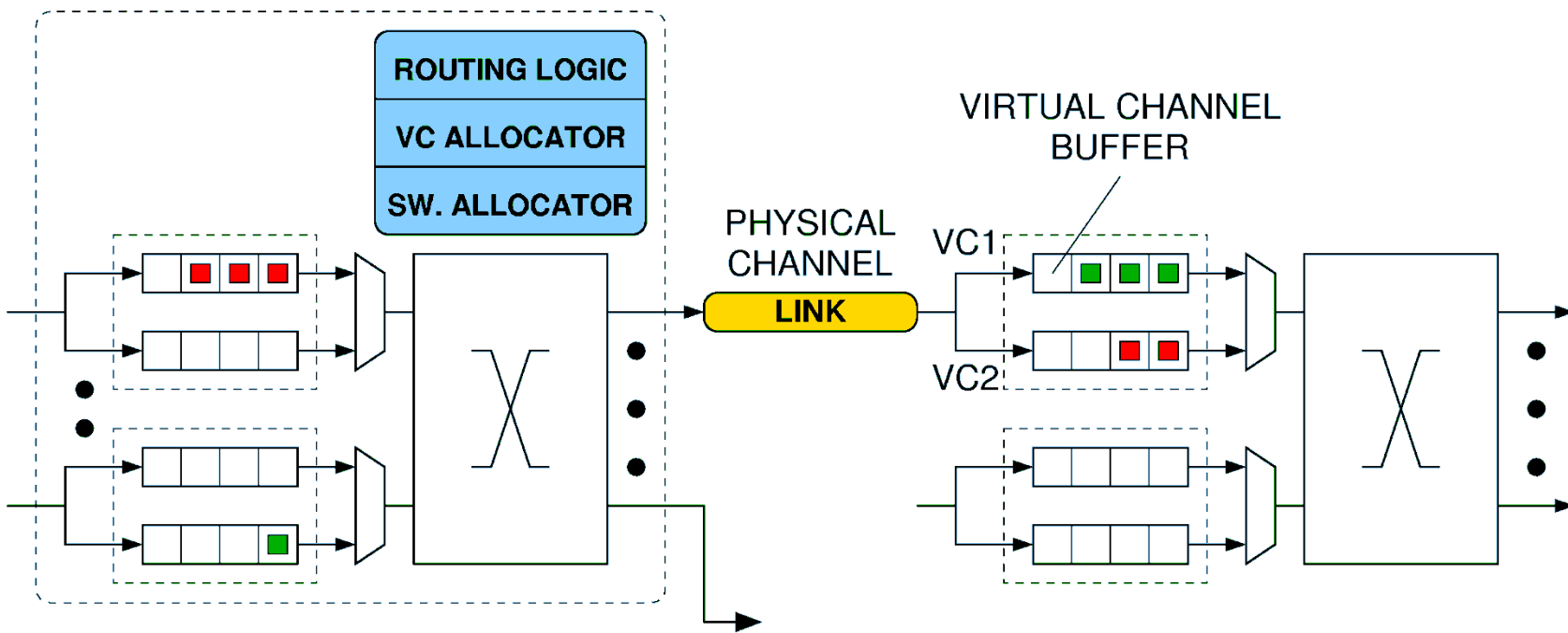
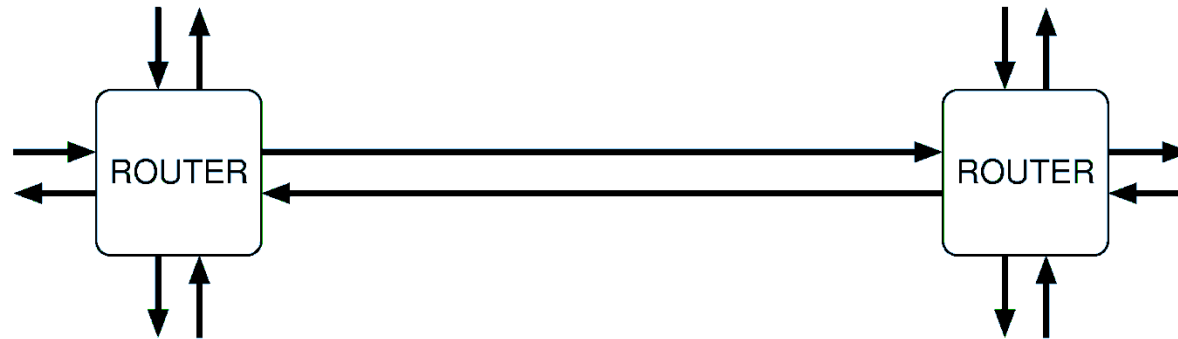
- **Virtual channels divided into two sets**
 - Guaranteed throughput and best effort
 - GT traffic can take over BE virtual channels
- **Higher priority given to GT traffic**
 - Header decoder
 - Arbiter
 - Output link controller
- **Round-robin priority mechanism within each class**

Synchronous Networks

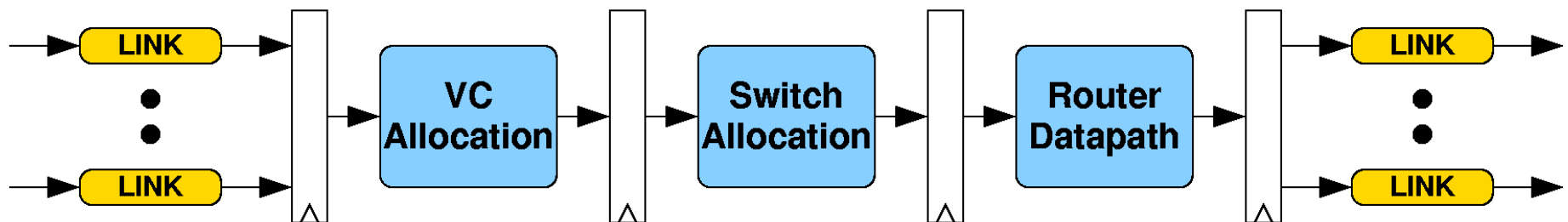
Clocking alternatives for Synchronous Routers



Generic On-Chip Router

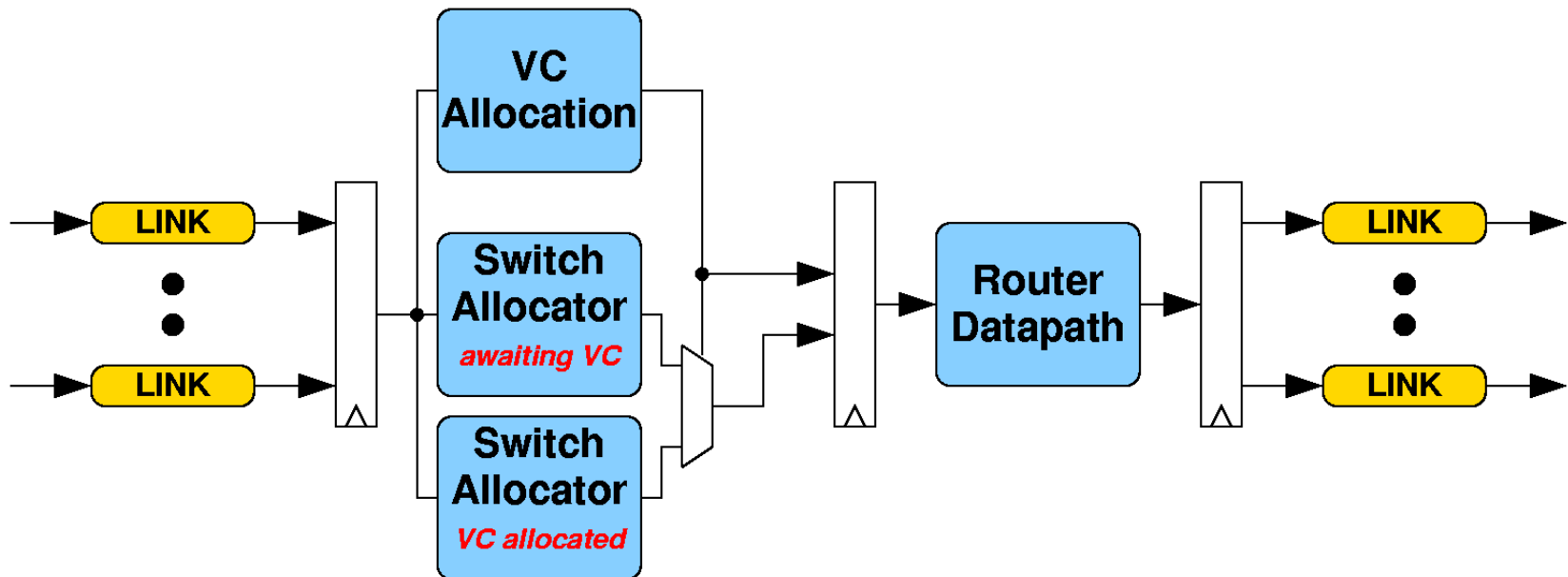


Synchronous Router Pipeline



- **Router Pipeline may be many stages**
 - Increases communication latency
 - Can make packet buffers less effective
 - Incurs pipelining overheads

Speculative Router Architecture

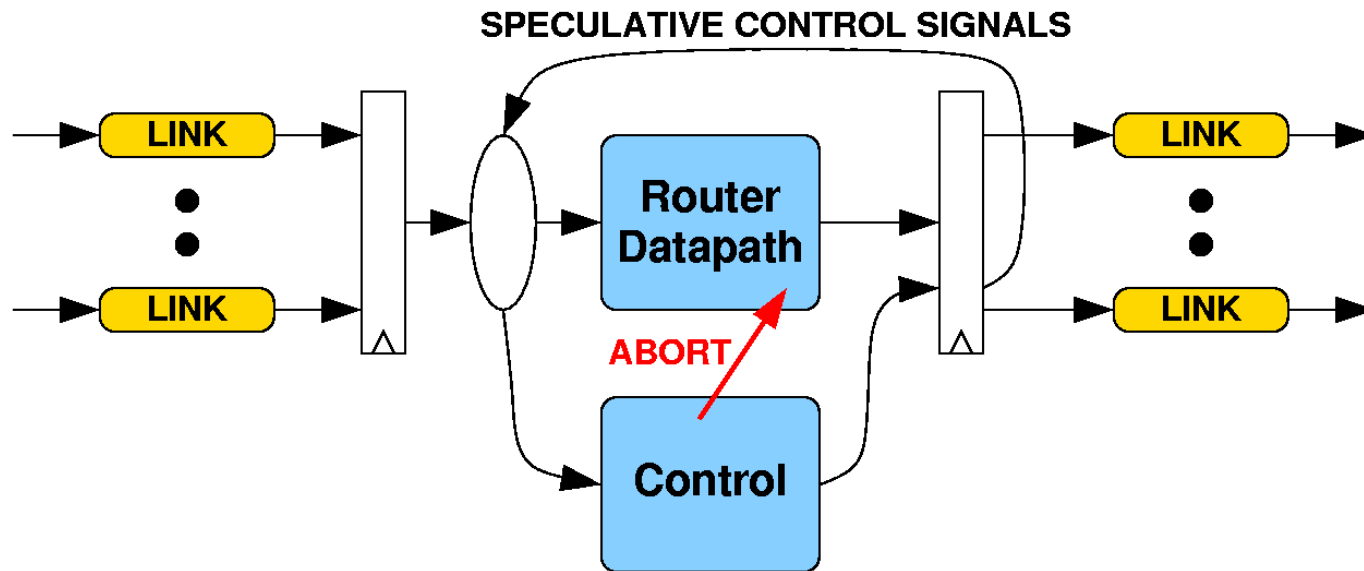
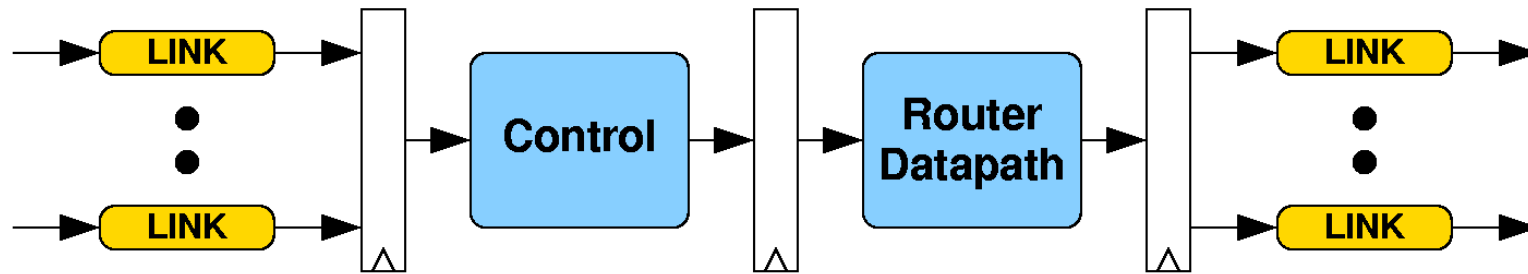


- **VC and switch allocation may be performed concurrently:**
 - Speculate that waiting packets will be successful in acquiring a VC
 - Prioritize non-speculative requests over speculative ones

Li-Shiuan Peh and William J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers", In Proceedings HPCA'01, 2001.

Mullins, Cambridge

Single Cycle Speculative Router



R. D. Mullins, A. West and S. W. Moore, “*Low-Latency Virtual-Channel Routers for On-Chip Networks*”, In Proceedings ISCA’04.

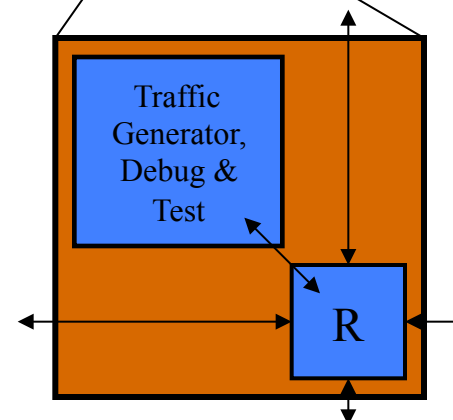
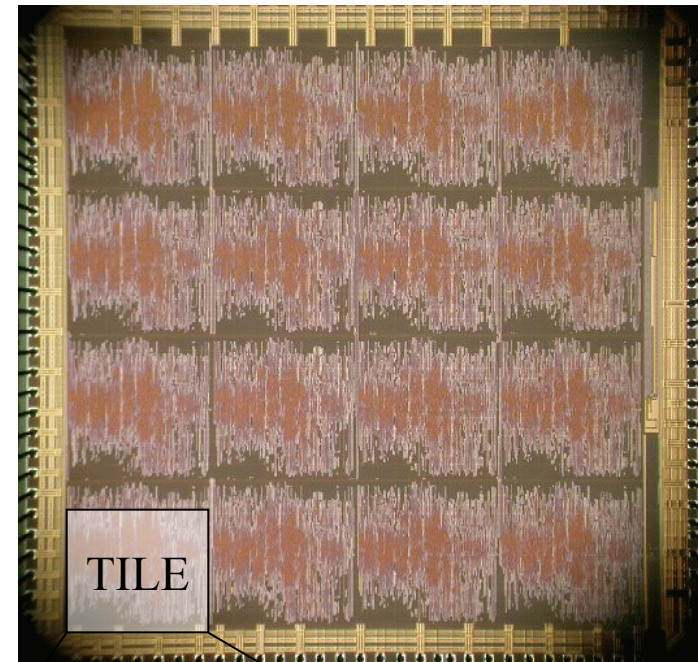
Mullins, Cambridge

Single Cycle Speculative Router

- **Single cycle router made possible by use of speculation**
- **Clock period is almost unchanged (compared to pipelined design)**
 - **Approx. 30 FO4 (simple standard-cell design)**
- **Presence of clock simplifies design**
 - **Arbitration**
 - **Fast combinational matrix arbiters**
 - **Can easily be extended to handle priority traffic etc.**
 - **Speculation**
 - **Aided by the clear notion of a clock “cycle”**
 - **Simple abort logic (abort detection and actual abort)**

Single Cycle Speculative Router

- **4x4 mesh network, 25mm²**
- **Single Cycle Routers (router + link = 1 clock)**
 - Low common case latency
- **4 virtual-channels/input**
- **80-bit links**
 - 64-bit data + 16-bit control
- **250MHz (worst-case PVT)**
16Gb/s/channel, 0.18um.



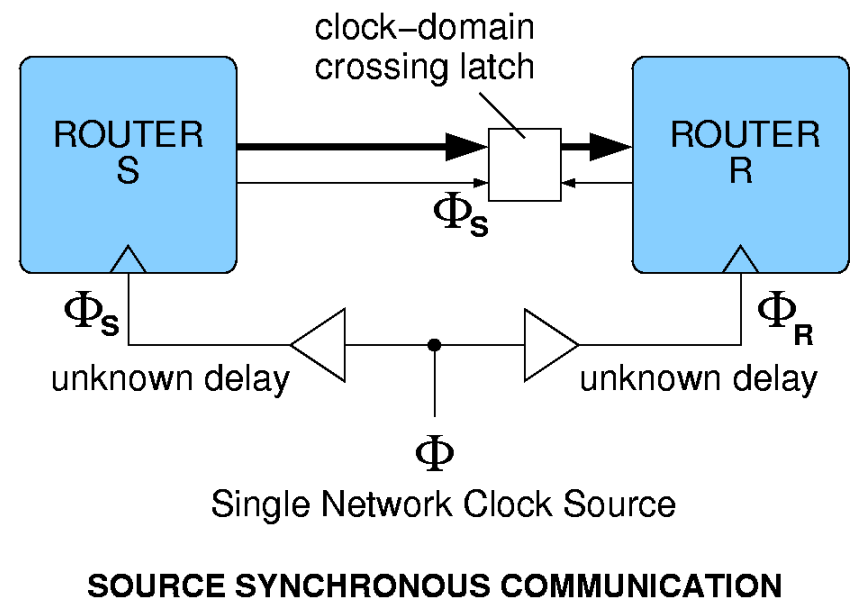
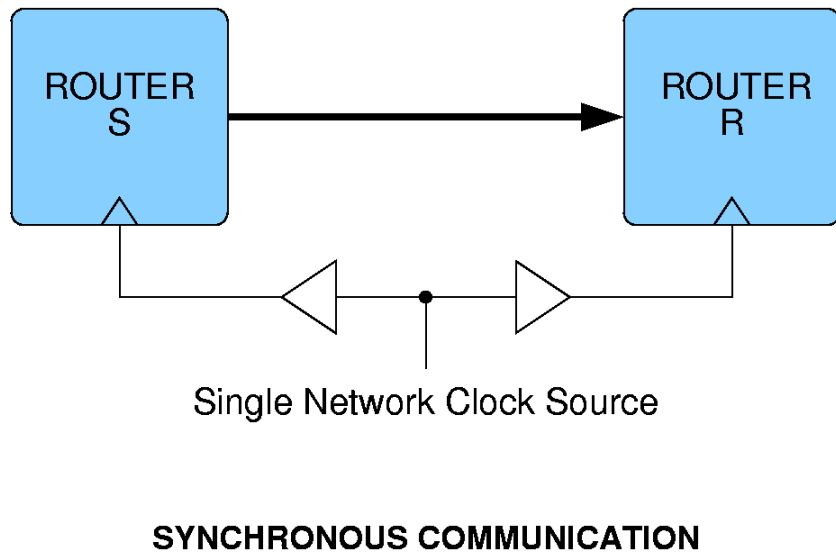
R. D. Mullins, A. West and S. W. Moore, “*The design and implementation of a low-latency on-chip network*”, In Proceedings ASP-DAC’06

Beyond a Single Global Clock

Limitations of Fully-Synchronous Networks

- **Difficult to distribute clock**
 - Network spread over die & may have irregular layout
 - Minimising skew costs complexity and power
- **Alternatives/extensions to PLL and H-tree:**
 - Clock deskewing techniques
 - Distributed Clock Generator (DCG).
 - Distributed PLLs
 - Standing-wave oscillators and rotary clock schemes
 - Resonant global clocks, optical clock distribution etc.
- **Single Network Clock Frequency**
 - Communicating synchronous IP blocks may operate at different and potentially adaptive clock frequencies
 - What is most appropriate network clock frequency?
 - We don't want to have to generate and distribute a very high frequency clock in order to emulate an asynchronous network

Frequency Distribution

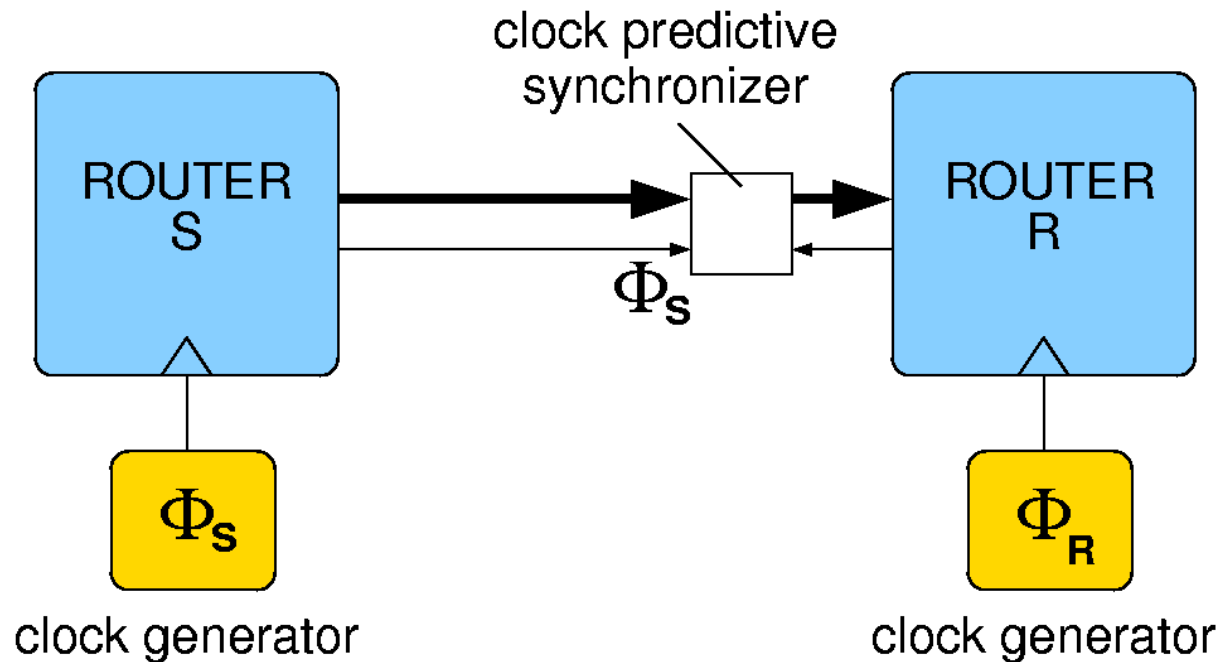


- Clock skew may force the system to be partitioned into multiple clock domains
- Can exploit the fact that only the phase of each router's clock differs, simple error-free clock-domain crossing possible (single clock source)

Router clocks derived from a single source

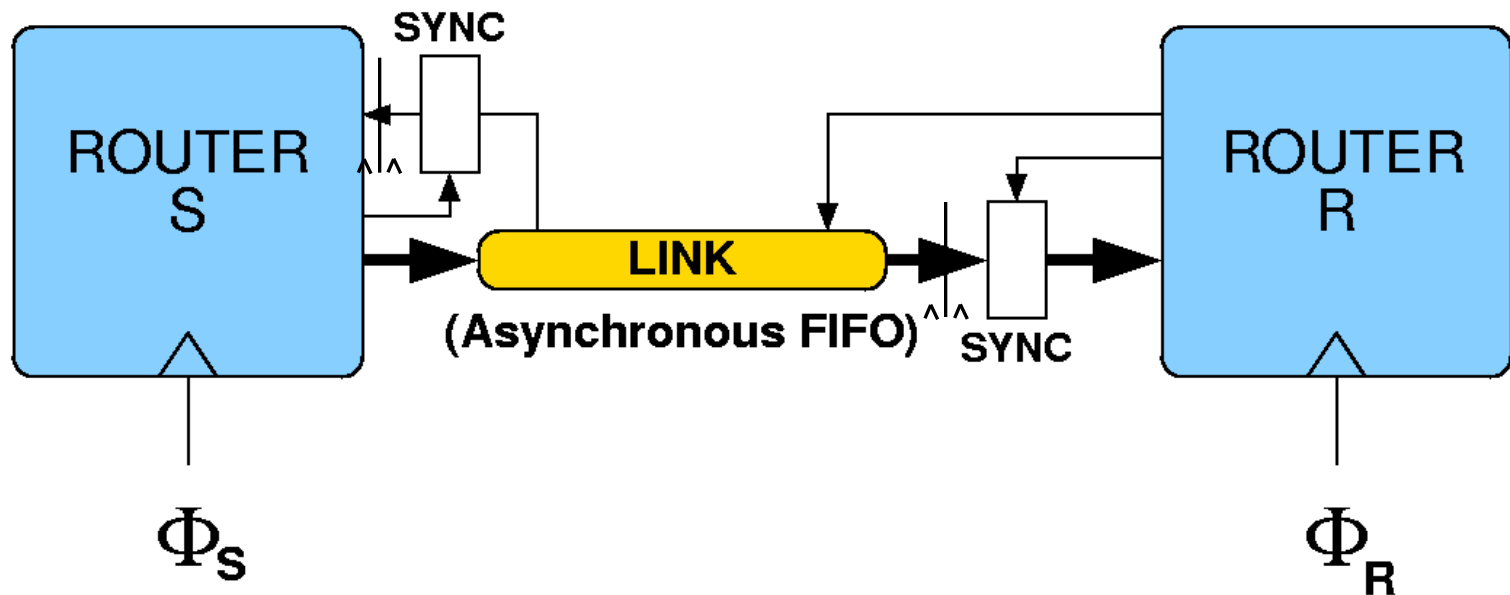
- **Each router's clock may be generated from the global network clock, either by:**
 - Clock division or
 - Clock multiplication
- **Clock domain crossing techniques can exploit known clock frequency relationships**

Locally Generated Clocks (periodic & free-running)



- **Can exploit knowledge about clocks (when crossing clock domains) even if all we know is that they are periodic, examples:**
 - predictive synchronizers [Dally][Frank/Ginosar]
 - asynchronous FIFOs [Chakraborty/Greenstreet]

Synchronous Routers with Asynchronous Links



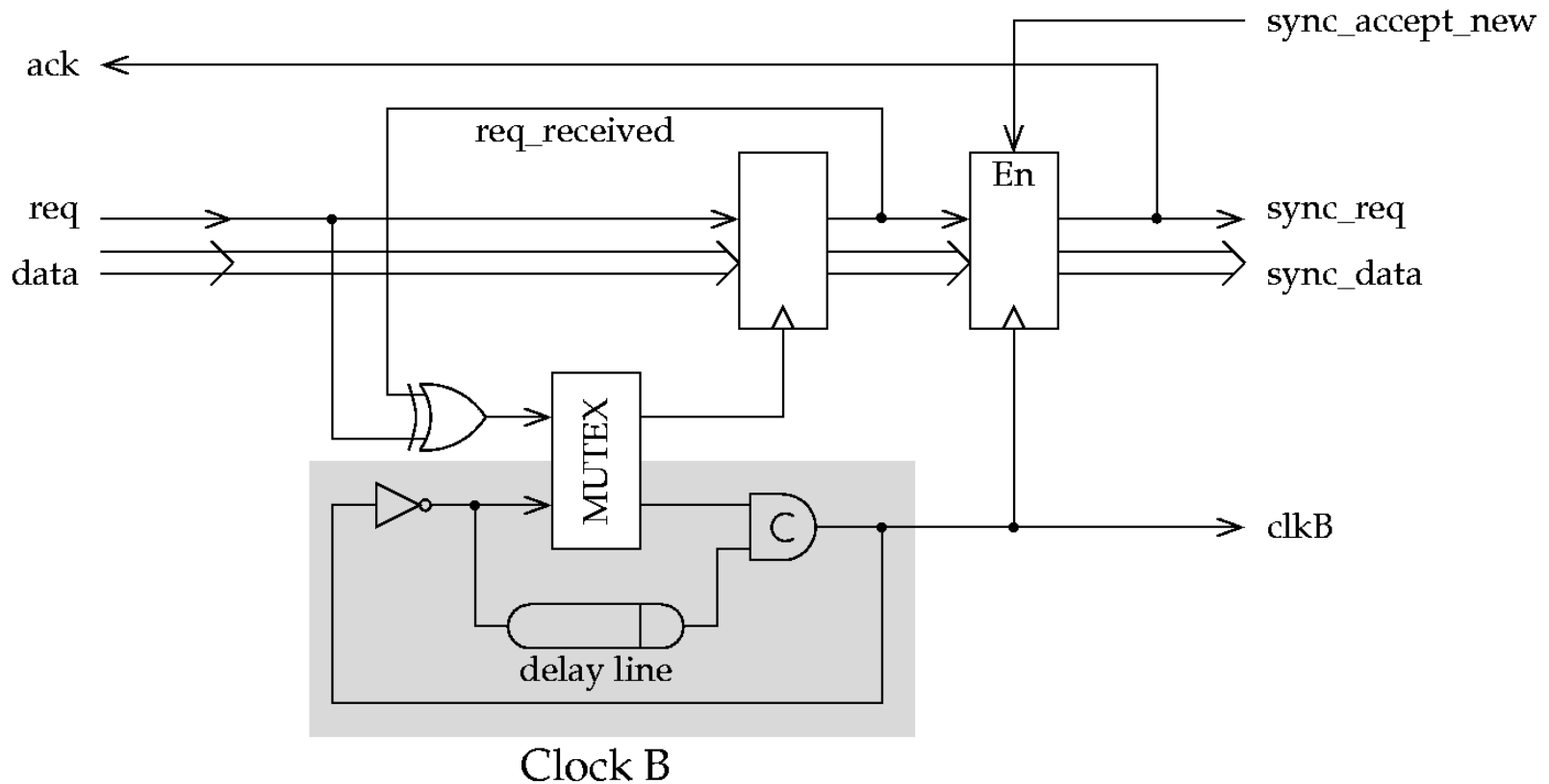
■ Synchronization:

- Time Safe: *e.g.* Traditional 2 FF synchronizers
- Value Safe: Clock Pausing/Data-driven clocks

Locally Clocked Routers/Asynchronous Interconnect (GALS style network)

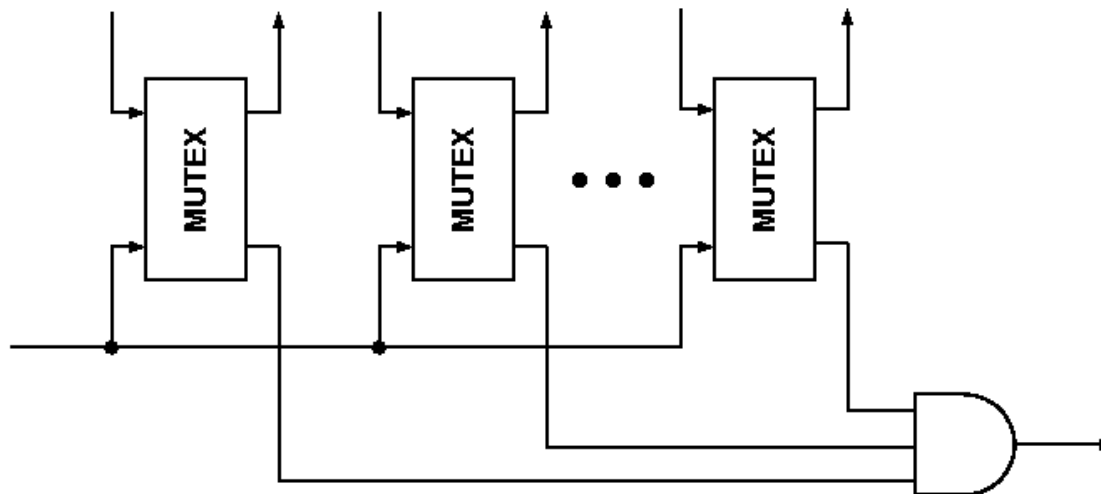
- **Can support asynchronous interconnects**
 - No longer exploiting periodic nature of router clocks
 - Correct operation is independent of the delay of the link
- **GALS interfaces with pause-able clocks**
 - If necessary clock is stretched, data is always transferred reliably (value safe)
 - Need to construct local delay line

GALS – Clock Pausing



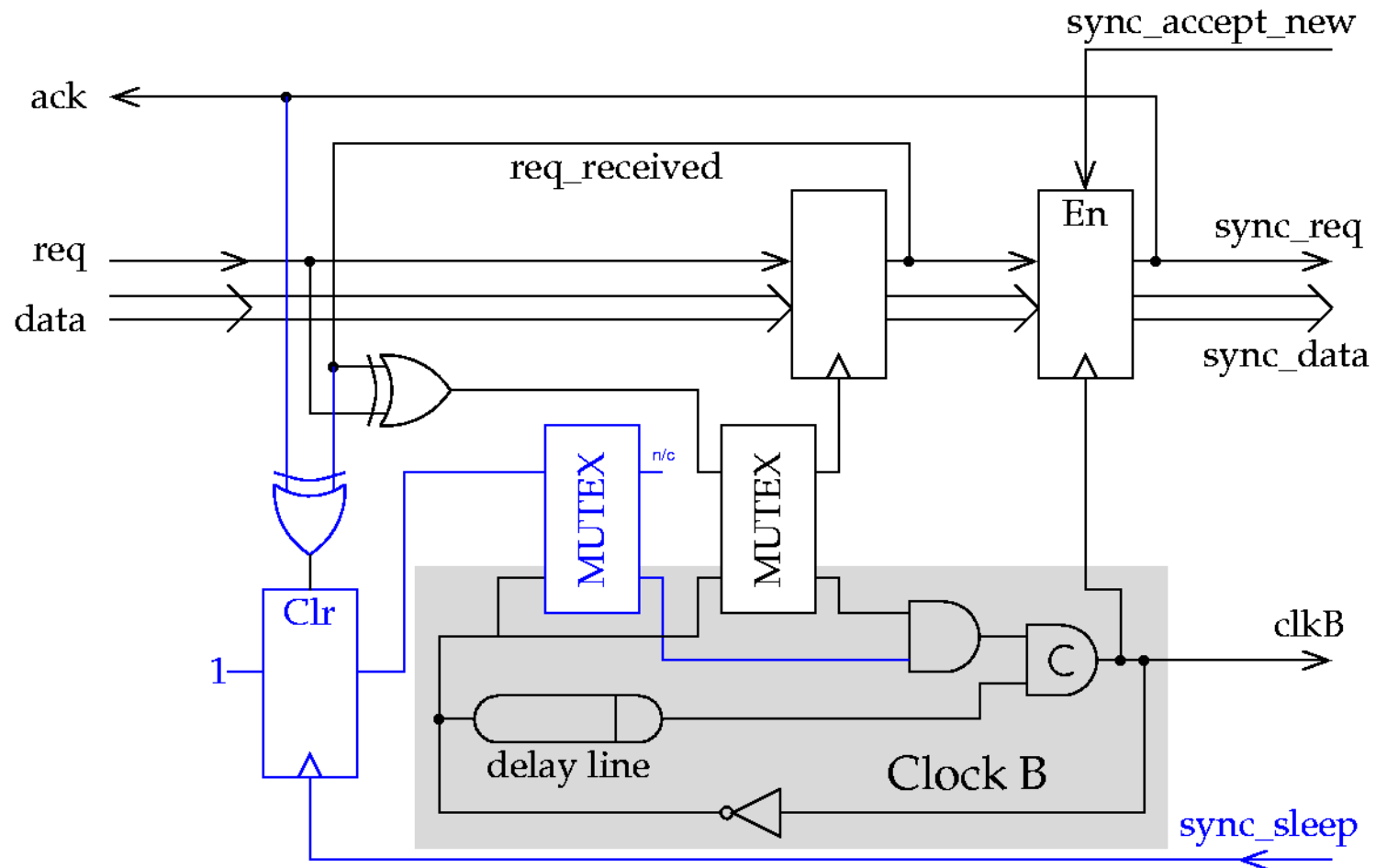
- **Simple GALS interface (receiver)**
- **Note: Req/Ack uses 2-phase handshaking protocol**

GALS – Multiple Inputs



- Clock is free running (although it can be paused)
- It is the clock that really determines if asynchronous data is transferred into the synchronous clock domain on a particular cycle
- Impact on performance in on-chip network requiring multiple input data/control ports?

GALS – Stoppable Clock



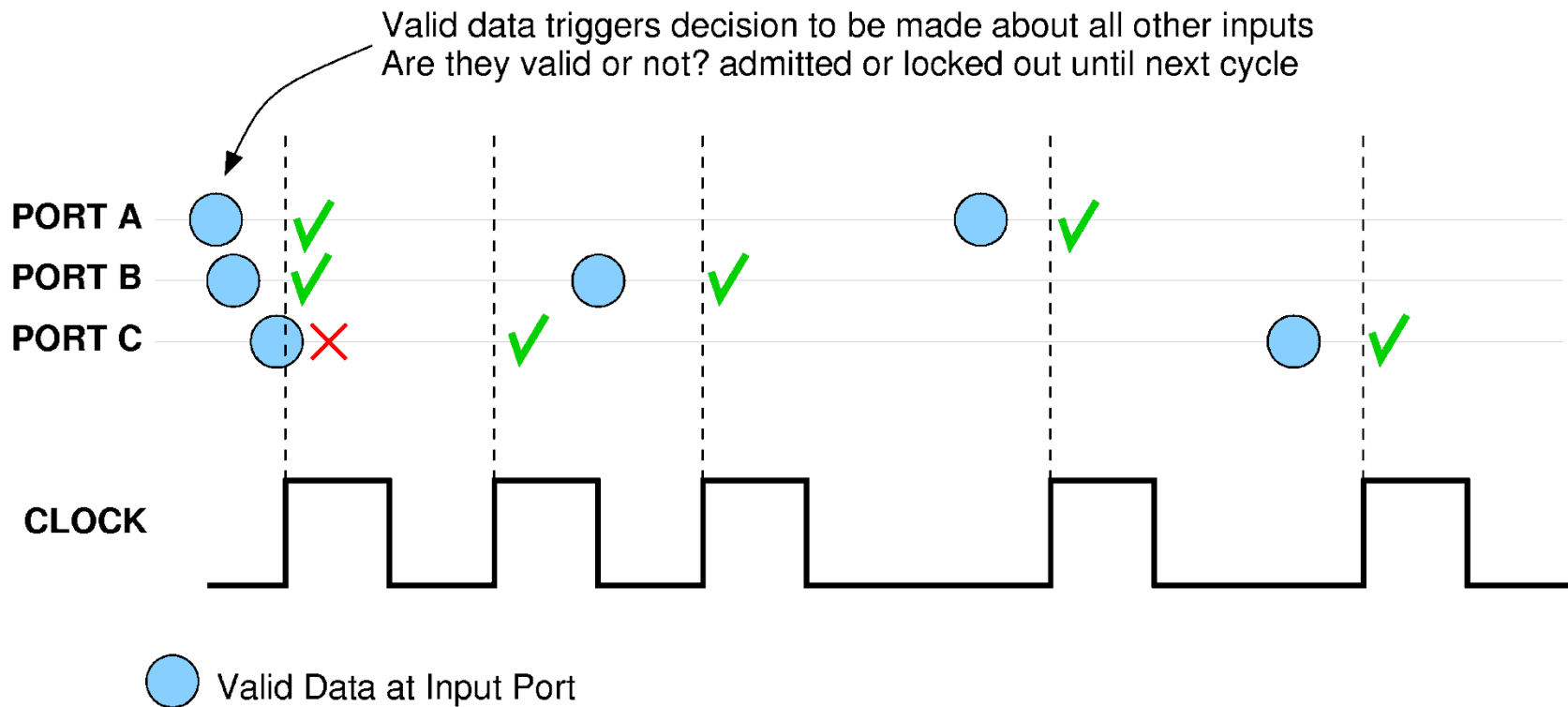
Local aperiodic clock generation

- **Discard free-running clock but retain a single delay assumption for router**
- **Options for clock pulse generation:**
 1. Use stoppable GALS interface and attempt to stop every cycle – overheads?
 2. Wait for data/null-data from all neighbours before generating pulse (global synchrony!)
 3. Data driven clock
 4. Traditional asynchronous bundled-data approach (with a single delay assumption for whole router)
- **Can still exploit synchronous router implementation**

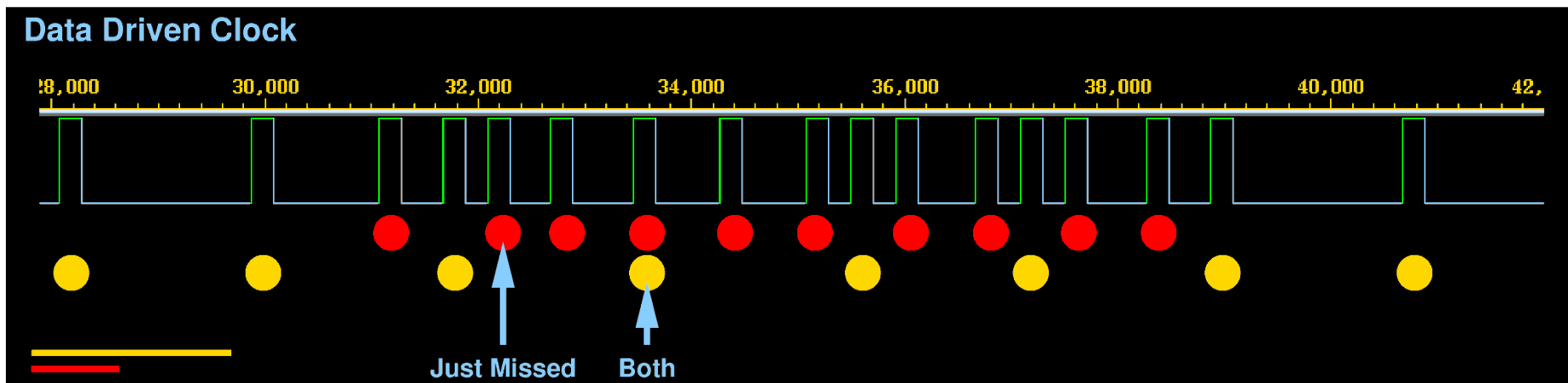
Data-Driven Local Clock

- **Idea:**
 - If data at any input, sample all inputs
 - Determine which inputs are to be admitted on next clock cycle (requires MUTEX)
 - Ensure data that is not admitted is 'locked out' for next clock cycle
 - After all MUTEXes have made a decision (and never faster than the delay line!) generate a clock pulse
- **Similarities to stoppable GALS interface and asynchronous priority arbiters**

Data-Driven Clock Waveform



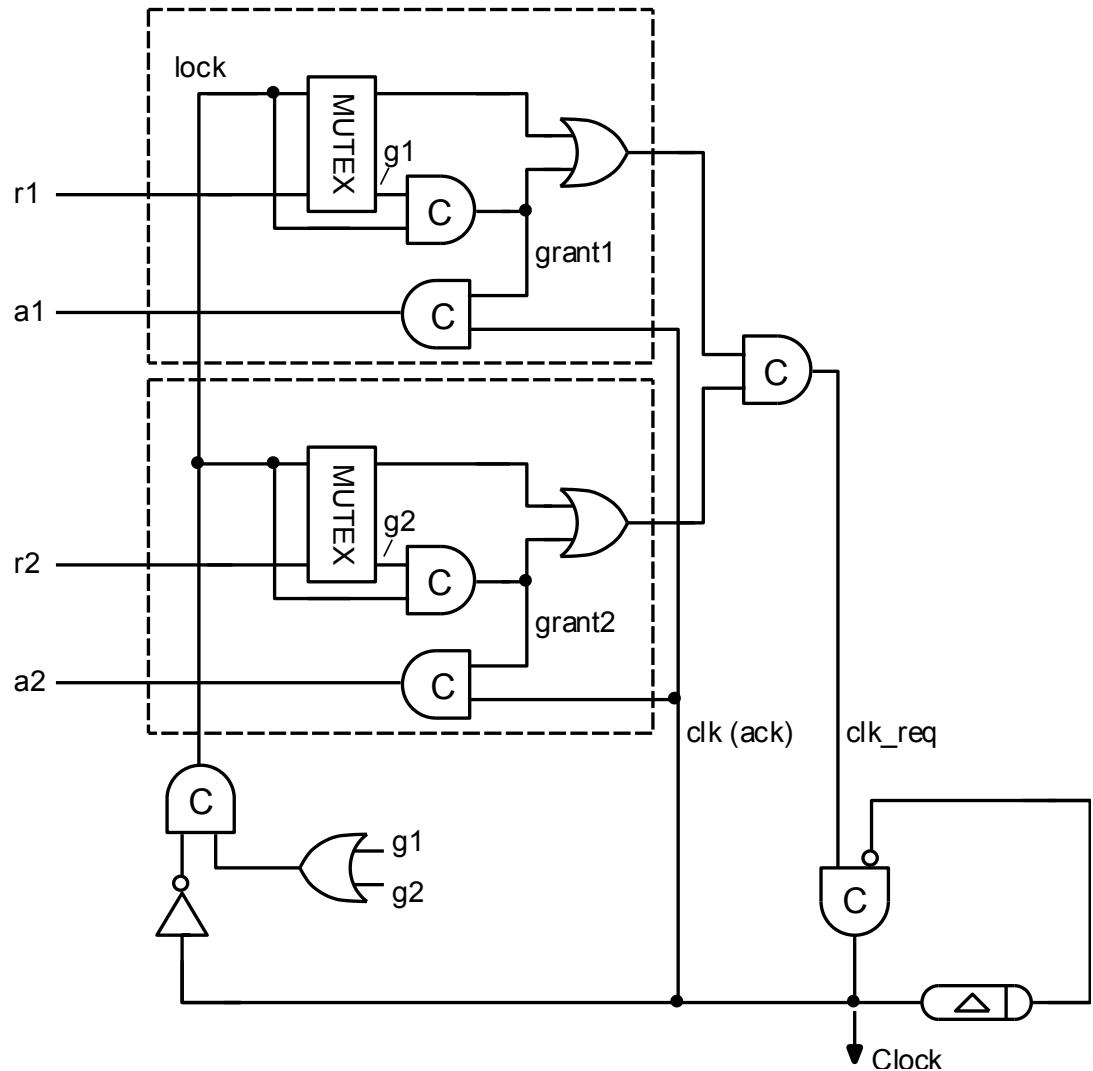
Data-Driven Clock Waveform



- Imagine data from two packets arriving at a single router node at different rates
- An aperiodic clock may be generated to minimize latency and power
- Minimum clock period set by delay line
- Value safe synchronization (no chance data is ever lost)

Data-Driven Local Clock

- May be generalized to n-input ports. Only the control interfaces are shown here (r1,a2 and r2,a2)
- *grantn* is simply used to control the latching of data at each input port (register enable)



Synchronous Routers - Summary

- **Can design high-performance single cycle routers**
- **Design is simplified by presence of global synchrony**
- **Distribution of global clock can be eased by:**
 - New clock generation/distribution techniques
 - Source synchronous communication
- **Network operating frequency**
 - Relax global synchrony further
 - Data-driven clocking determines most appropriate router clock frequency automatically

Asynchronous On-Chip Networks

Why are asynchronous NoCs interesting?

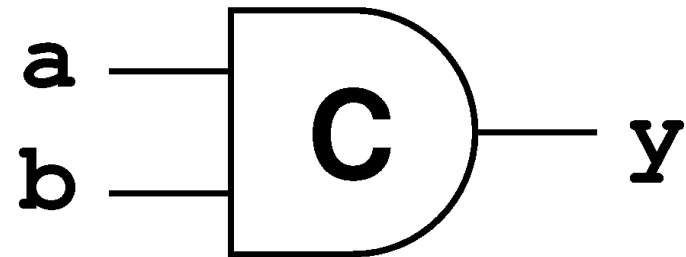
- **Simple/elegant solution when networked IP blocks run at different clock frequencies**
 - Data driven, no superfluous switching activity
 - No synchronization/clock alignment issues at interfaces
- **Ability to exploit data/path-dependent delays**
 - Low-latency common or high-priority paths through router
- **No clock distribution issues**
- **Security and EMI advantages**
 - Clock focuses EM emissions
 - The presence of a clock can also aid fault-induction and side-channel analysis attacks

Why are asynchronous NoCs interesting?

- **Freedom to optimize network links**
 - Not constrained by need to distribute/generate multiple clock frequencies. Can exploit high-frequency narrow links.
 - Dynamic latency/throughput trade-offs (adaptive pipeline depth)
 - Exploit dynamic optimizations on links (e.g. DVS)
- **Reduced design time**
 - Easy to use interfaces, modularity.
 - Robust and simple implementation
- **Some arguments for reduced power**

Asynchronous Circuit Basics

- **Control in asynchronous circuits often relies on simple handshaking protocols (req/ack event cycles)**
- **Delay-insensitive event-driven system - every signal transition is acknowledged**
- **The C-element is a fundamental building block of many asynchronous circuits**
 - **Can be thought of as a AND-gate for events**



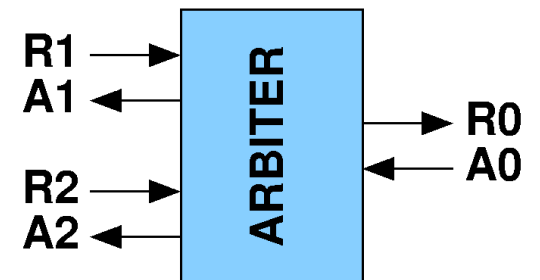
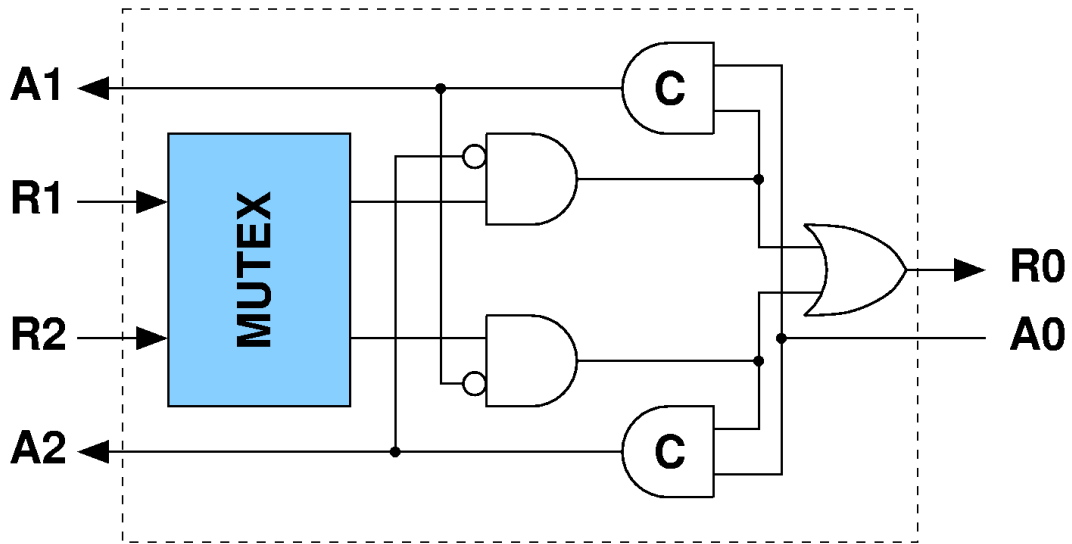
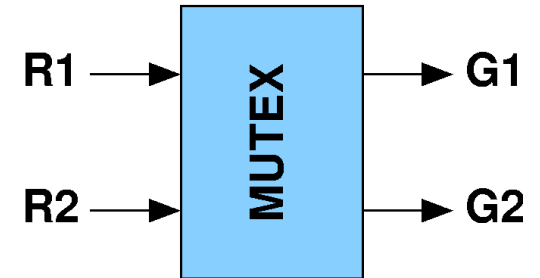
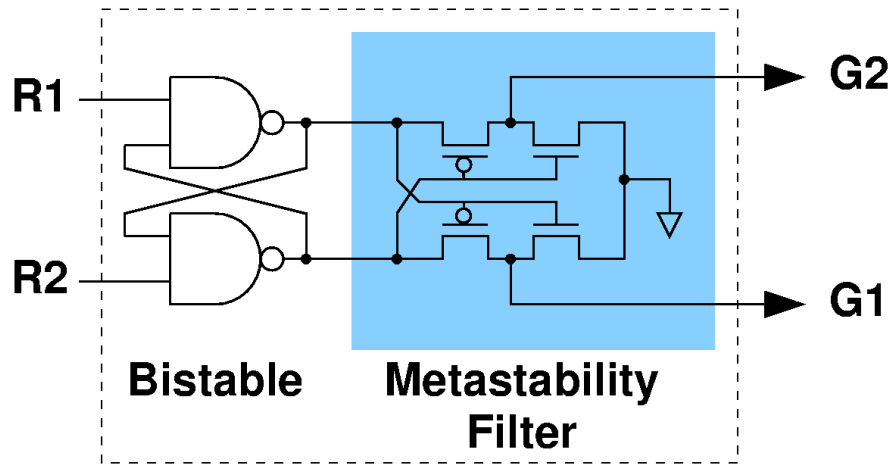
**IF inputs match in state
 THEN copy it for output
 ELSE hold previous state;**

if a=b then y:=a

$y = ab + y(a+b)$

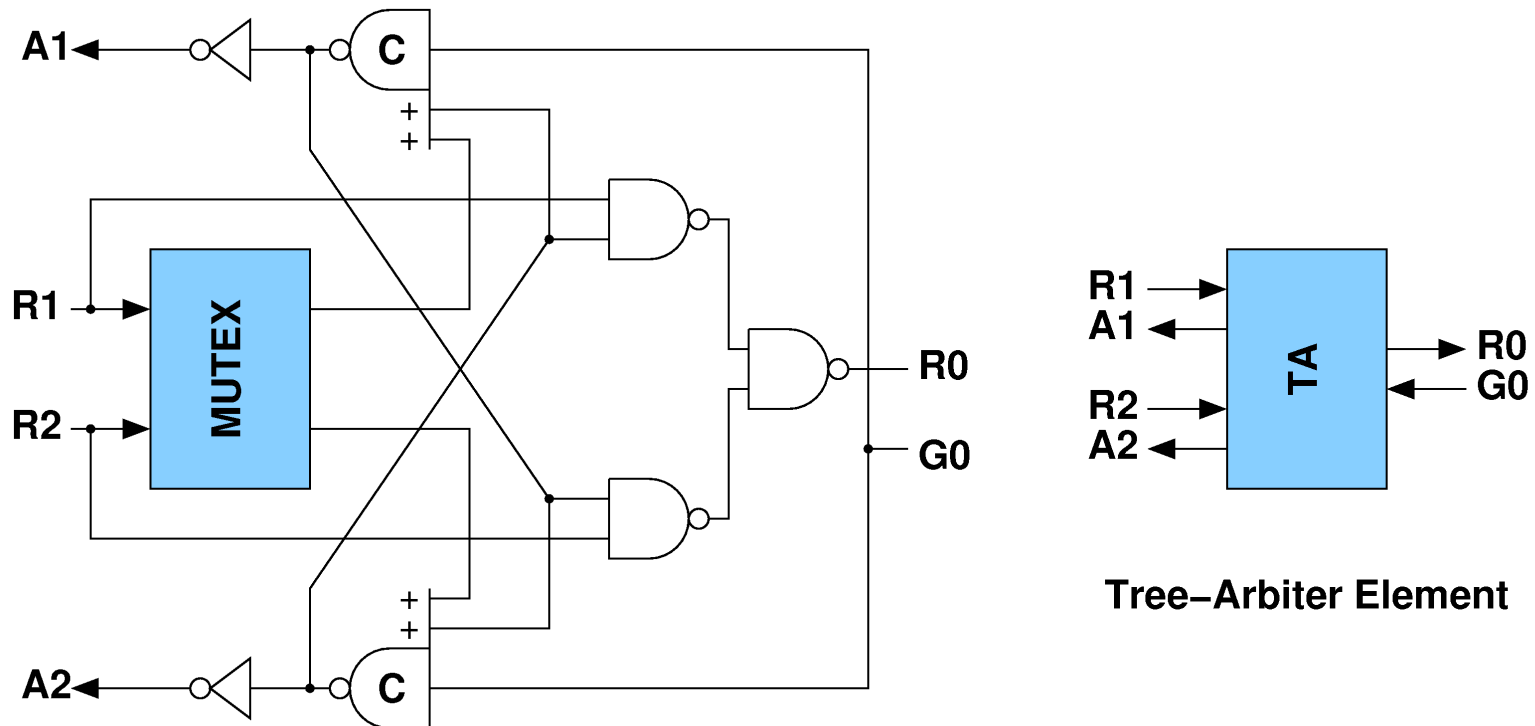
(State Holding Element)

Arbitration



(Arbitrated Call)

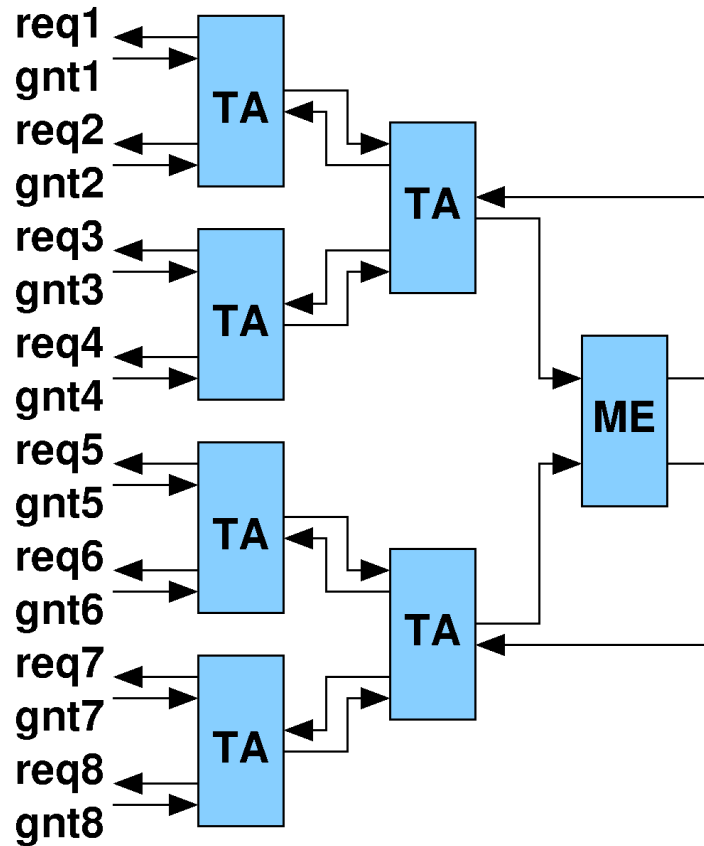
Tree Arbiter Element



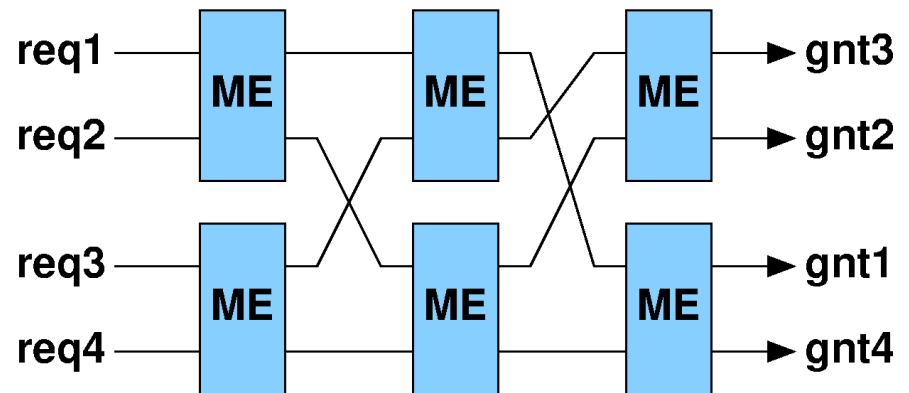
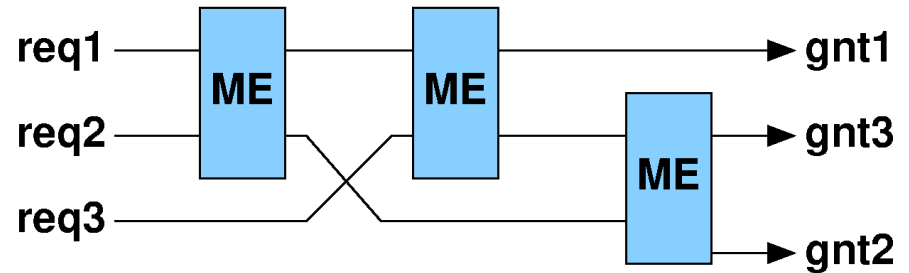
M. B. Josephs and J. T. Yantchev, “*CMOS Design of the Tree Arbiter Element*”, IEEE Trans. On VLSI Systems 4(4), pp.472-476, Dec. 1996

J. Bainbridge, “*Asynchronous System-on-Chip Interconnect*”, Ph.D. Thesis, Dept. of Computer Science, University of Manchester.

Multiway Arbiters



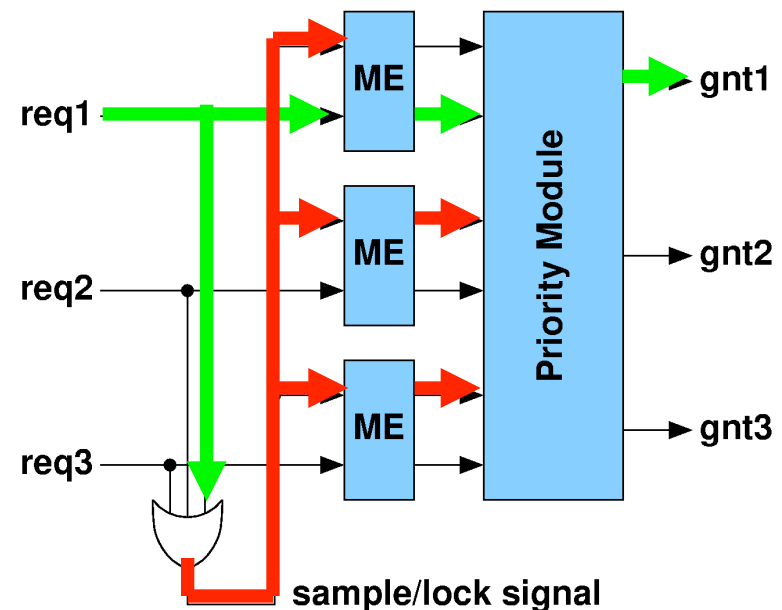
Tree Arbitrer



Cascaded MUTEXes

Static Priority Arbiters

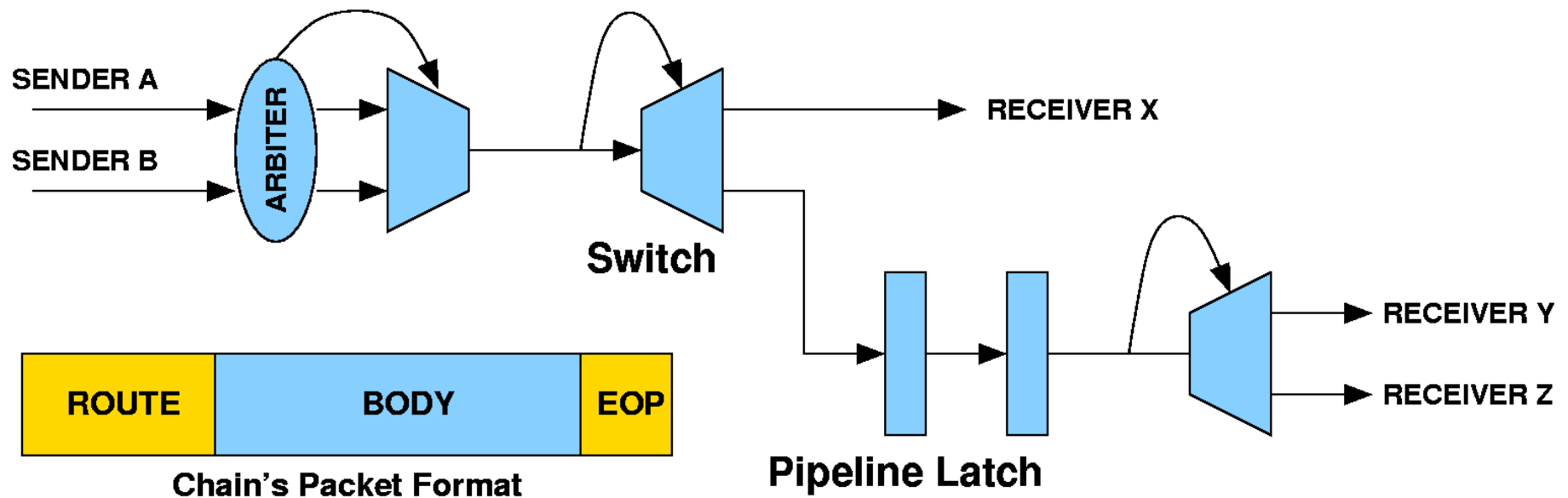
- **“Priority Arbiters”** Bystrov/
Kinniment/Yakovlev (ASYNC’00)
- **First stage samples/locks**
current request vector
- **Static or dynamic priority**
- **Original design updated to**
tackle performance and QoS
**issues Felicijan/Bainbridge/
Furber (ICM’03)**



Basis for sample/lock and prioritise arbiter

Delay-Insensitive Switched Interconnect

- The basic DI latch can be extended to support steering, multiplexing and arbitration



J. Bainbridge and S. Furber, "CHAIN: A Delay-Insensitive Chip Area Interconnect", IEEE Micro, Vol. 22, No. 5, 2002

CHAIN

- **Basic link is 6 wires**
 - 2-bits of data (1-of-4) + end of packet + ack
 - any N-of-M code could be used
 - around 1Gbps (0.18um, 160Mbps per wire)
 - Links may be ganged together
- **Route information tapped off and used to steer remainder of packet**
- **If arbitration is required, arbiter grant is retained for duration of packet (no fragmentation of packets)**

Asynchronous on-chip networks

- **How do we build more complex on-chip routers?**
 - Support for virtual-channels
 - QoS
- **Challenges**
 - Multi-way & prioritised arbitration
 - Control overheads
 - Arbitration and DI circuits can be slow!
 - How can control overheads be hidden?

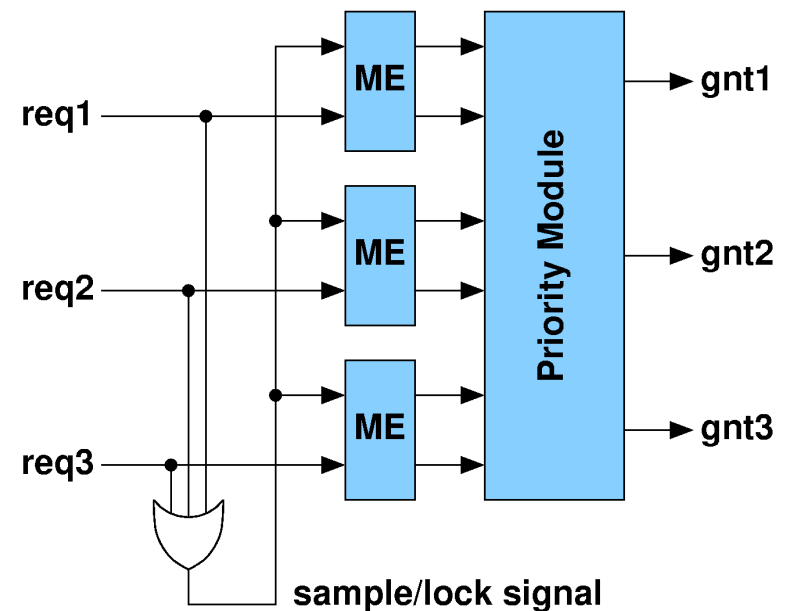
Low-Latency Best-Effort Asynchronous Networks

Improving Network Latency

- **Asynchronous router latency can be high**
 - Fine-grain pipelining can provide good throughput figures but control overheads can extend latency
 - Completion detection, RTZ phase, H/S
 - Fast combinational matrix arbiters have also been replaced by cascaded MUTEXes or complex priority arbiters
 - Overheads even greater in a BE router that must allocate VCs dynamically
- **Approaches to reduce latency?**
 - Speculation
 - Decoupled control and data networks

Low-Latency Asynchronous Routers

- **Exploit speculation?**
 - Use Priority arbiter organization
 - Assume only a single grant will be present after lock is asserted
 - Use MUTEX grant outputs to steer data immediately
- **Issues**
 - Complex abort procedure?
 - Invalid data and DI encoding?
 - Careful not to make common-case slower



Basis for sample/lock and prioritise arbiter

Decoupled Control and Data Networks

- **Control network runs ahead of data network, hiding latency of scheduling logic**
 - In an asynchronous environment, each network will operate at its natural rate
- **Control network latency will be much lower compared to data network**
 - Narrower links and simpler datapath
 - No virtual channels - little arbitration, less switching
 - Less traffic, single control flit per packet only
 - Could also exploit ‘fat’ wires and early requests to send packet
- **Separate control and data networks can also be exploited in synchronous network [Peh/Dally]**

L. Peh and W. J. Dally, “Flit-Reservation Flow Control”, In Proceedings HPCA’00.

Mullins, Cambridge

Decoupled Control and Data Networks

- **Schedule is queued and steers incoming data flits (data flits contain no routing information)**
- **Scheduler could perform VC allocation or both VC and switch allocation in advance**
- **Control network could also control power-gating of data network, waking network/links as needed from sleep mode.**

Decoupled Control and Data Networks

- **Design Decisions**
 - Design can be simplified by keeping input port VC requests in order
 - Has obvious implications for performance
 - Out-of-order VC allocation scheme also possible
 - Performing switch allocation ahead of time could be inefficient
 - Order data actually arrives could be different
- **Decoupled control and data networks may help hide scheduling overheads. More appropriate than speculation for asynchronous NoCs?**

References

- [L. Benini and D. Bertozzi, “Network-on-chip architectures and design methods”, in Proceedings of IEE Computer Digital Technology, March 2005.
- P. P. Pande et al., “Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures, IEEE Transaction on Computers, August 2005.
- W. J. Dally and B. Towles, “Route Packets, Not Wires: On-Chip Interconnection Networks”, Design Automation Conference, June 2001.
- P. Guerrier and A. Greiner, “A Generic Architecture for On-Chip Packet-Switched Interconnections,” Proc. Design and Test in Europe (DATE), pp. 250-256, Mar. 2000.
- S. Kumar et al., “A Network on Chip Architecture and Design Methodology”, ISVLSI, 2002.
- F. Karim et al., “An Interconnect Architecture for Networking Systems on Chip”. IEEE Micro Sep-Oct 2002.
- [P. Pande et al., “Design of a Switch for Network on Chip Applications”, ISCAS 2003.

References

- Chakraborty and M. Greenstreet, “Efficient Self-Timed Interfaces for Crossing Clock Domains”, In Proceedings ASYNC’03**
- L. F. G. Sarmenta, G. A. Pratt and S. A. Ward, “Rational Clocking”, ICCD’95**
- J. Rabaey et al., “A 1-V heterogeneous reconfigurable DSP IC for wireless baseband digital signal processing,” IEEE Journal of Solid State Circuits, Vol. 35, No. 11, Nov. 2000, pp. 1697 - 1704**
- A. Adriahtenaina et al., “SPIN: a Scalable, Packet Switched, On-chip Micro-network,” Proc. Design and Test in Europe (DATE), Mar. 2003.**
- L. Benini and G. De Micheli, “Networks on Chips: A New SoC Paradigm,” Computer, vol. 35, no. 1, Jan. 2002, pp. 70-78.**
- S. Kumar et al., “A network on chip architecture and design methodology,” in Proc. ISVLSI, 2002.**
- K. Goossens et al., “Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip,” IEE Proc.-Comput. Digit. Tech., Vol. 150, No. 5, Sep. 2003, pp. 294-302.**

