A Low-Overhead Asynchronous Interconnection Network for GALS Chip Multiprocessors

Michael N. Horak, *University of Maryland*
Steven M. Nowick, *Columbia University*
Matthew Carlberg, *UC Berkeley*
Uzi Vishkin, *University of Maryland*

In IEEE Transactions on Computer-Aided Design (TCAD), vol. 30:4, pp. 494-507 (April 2011)
[also in ACM/IEEE Int. Symposium on Networks-on-Chip (NOCS-10)]

Challenges for Designing Networks-on-Chip

- Power Consumption
  - Will exceed future power budgets by a factor of **10x** [1]
  - Global clocks: consume large fraction of overall power
- Performance Bottlenecks
  - Large network latencies cause performance degradation
- Increased Designer Resources
  - Many techniques are incompatible with current CAD tools
  - Difficulties integrating heterogeneous modules
    - Chips partitioned into ***multiple timing domains***

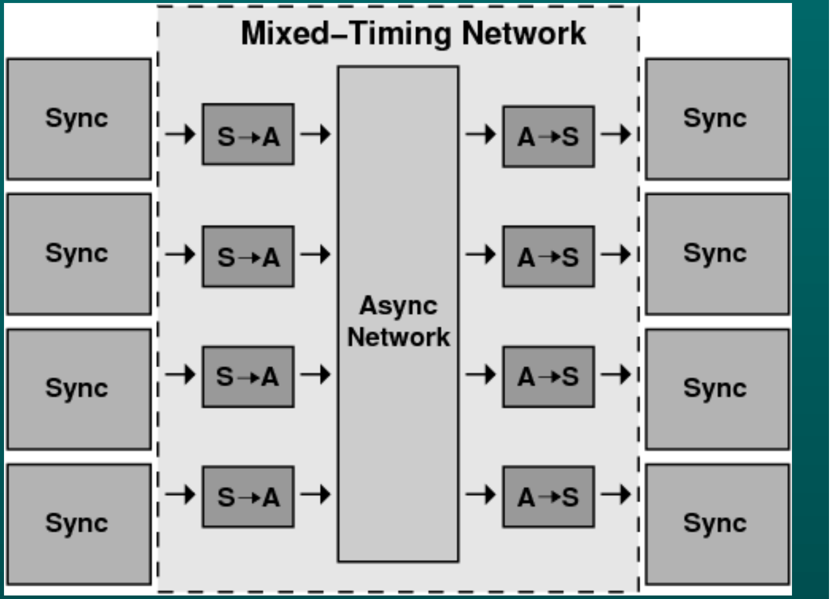
[1] J.D. Owens, W.J. Dally, R. Ho, D.N. Jayasimha, S.W. Keckler, and L.-S. Peh.
Research challenges for on-chip interconnection networks. *IEEE Micro*, 27(5):96-108, 2007.

Potential Advantages of Asynchronous Design

- Lower Power
  - No clock power consumed: without clock gating
  - Idle components inherently consume low power
- Greater Flexibility/Modularity
  - No clock distribution
  - Easier integration between multiple timing domains
  - Supports reusable components
- Lower System Latency
  - End-to-end traffic without clock synchronization
- More Resilient to On-Chip Variations
  - Correct operation depends on localized timing constraints

Mixed-Timing (GALS) System

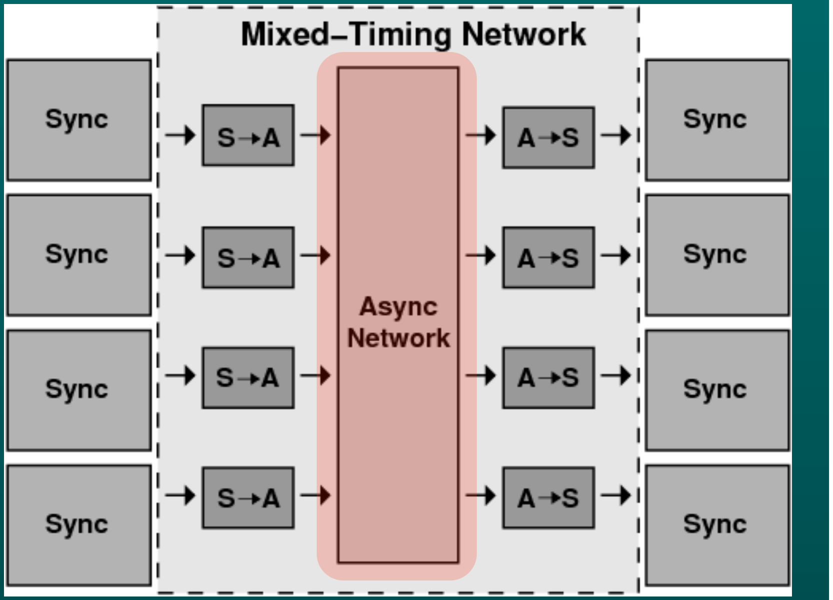
- Globally Asynchronous, Locally Synchronous [2]



[2] D. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems.* PhD thesis, Stanford Univ., 1984.

Mixed-Timing (GALS) System

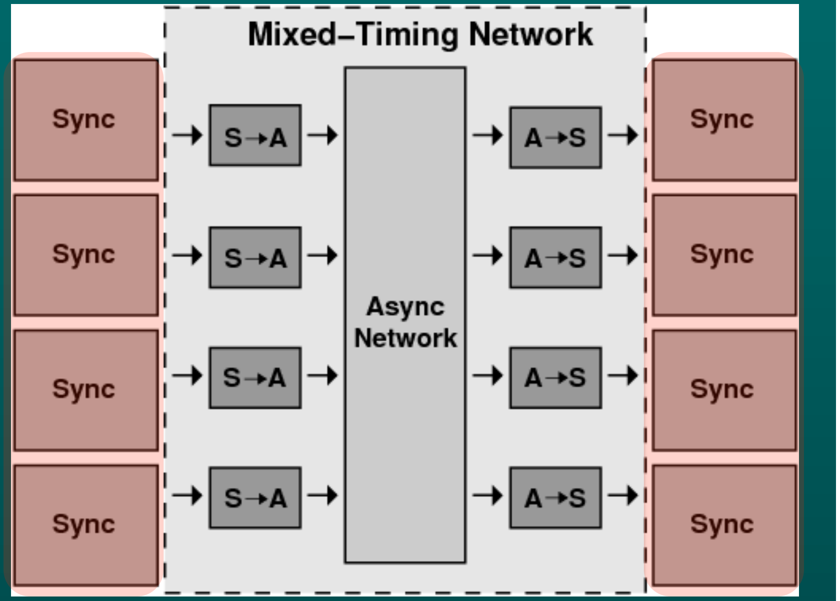
- Globally Asynchronous, Locally Synchronous [2]
- Asynchronous Network
  - – Clockless network fabric



[2] D. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems.* PhD thesis, Stanford Univ., 1984.

Mixed-Timing (GALS) System

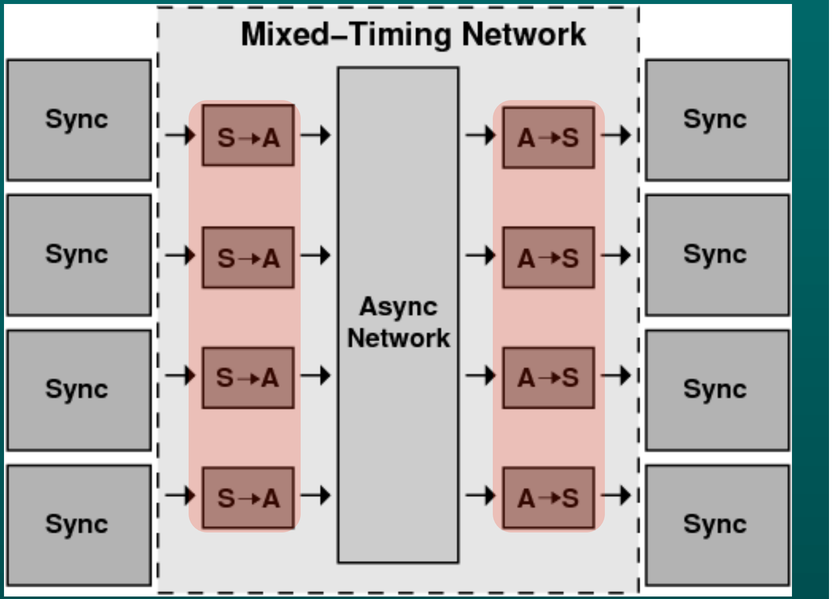
- Globally Asynchronous, Locally Synchronous [2]
- Asynchronous Network
  - Clockless network fabric
- Synchronous Terminals
  - Different unrelated clocks



[2] D. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems.* PhD thesis, Stanford Univ., 1984.

Mixed-Timing (GALS) System

- Globally Asynchronous, Locally Synchronous [2]
- Asynchronous Network
  - Clockless network fabric
- Synchronous Terminals
  - Different unrelated clocks
- Mixed-Timing Interfaces
  - Provide robust communication between Sync and Async domains



[2] D. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems.* PhD thesis, Stanford Univ., 1984.

Advances in GALS Networks-on-Chip

- Commercial Designs
  - Fulcrum Microsystems *(now Intel's Switch & Router Division [SRD])* (A. Lines. IEEE Micro Magazine [2004])
    - FocalPoint chips: high-performance Ethernet routing
  - Silistix, Inc. (J. Bainbridge, S. Furber. IEEE Micro Magazine [2002])
    - CHAIN™ works tool suite: heterogeneous SOCs
- Recent Work
  - Asynchronous Network-on-Chip (ANoC) (Beigne, Clermidy, Vivet et al. Async-05)
    - Wormhole packet-switched NoC with low-latency service
  - MANGO Clockless Network-on-Chip (T. Bjerregaard. DATE-05)
    - Offers quality-of-service (QoS) guarantees
  - RasP On-Chip Network (S. Hollis, S.W. Moore. ICCD-06)
    - Utilizes high-speed pulse-based signaling
  - SpiNNaker Project (Khan, Lester, Plana, Furber et al. IJCNN-08)
    - Massively-parallel neural simulation

GALS NOCs: Typical Current Targets

- Low- to Moderate-Performance Embedded Systems
  - 200-500 MHz
  - High system latency
- "Four-Phase Return-to-Zero" Protocols
  - Two round-trips/link per transaction
- "Delay-Insensitive Data" Encoding (dual-rail, 1-of-4)
  - Lower coding efficiency than single-rail
- Complex-Functionality Router Nodes
  - 5-port routers with layered services (QoS, etc.)
  - High latency/high area
- Custom Circuit Techniques:
  - Pulse-based signaling, low-swing signalling
  - Dynamic logic, specialized cells

Outline

- Introduction
- Target GALS Network Design
- Background: XMT Processor / MoT Network
- Asynchronous Network Primitives
- Experimental Results
- Conclusions

Target GALS Network Design

- Shared-Memory Chip Multiprocessors
  - Medium- to High-Performance

Target GALS Network Design

- Shared-Memory Chip Multiprocessors
- "Heterochronous" Timing [3]
  - Most general GALS timing model
  - Support multiple synchronous domains with unrelated clocking
  - Promotes reuse of Intellectual Property (IP) modules

[3] D. Messerschmitt, "Synchronization in Digital System Design", IEEE Journal on Selected Areas in Communications, October 1990

Target GALS Network Design

- Shared-Memory Chip Multiprocessors
- "Heterochronous" Timing
- Transition Signaling (Two-Phase)
  - Most existing GALS NOCs use "four-phase handshaking"
    - *2 roundtrip link communications* per transaction
  - Benefits of Two-Phase:
    - *1 roundtrip link communication* per transaction
    - improved throughput, power….
  - Challenge of Two-Phase: designing lightweight implementations
    - Most existing 2-phase designs use:
      - complex slow registers: double latch, double-edge-triggered, capture/pass
        - » [Seitz/Su "Mosaic" 93, Brunvand 91, Sutherland 89]
      - custom circuit components

Target GALS Network Design

- Shared-Memory Chip Multiprocessors
- "Heterochronous" Timing
- Transition (Two-Phase) Signaling
- Single-Rail Bundled Data
  - Most existing GALS NOCs use "delay-insensitive" link encodings
    - provide great timing-robustness ==> cost = *poor coding efficiency*
    - examples: dual-rail, 1-of-4
  - "Single-Rail Bundled Data" benefits:
    - *re-use synchronous datapaths:* 1 wire/bit + added "request"
    - *excellent coding efficiency*
  - Challenge: requires matched delay for "request" signal
    - 1-sided timing constraint: "request" must arrive after data stable

Target GALS Network Design

- Shared-Memory Chip Multiprocessors
- "Heterochronous" Timing
- Transition (Two-Phase) Signaling
- Single-Rail Bundled Data
- High Performance
  - Low System-Level Latency
    - minimize end-to-end delay under light to moderate traffic
  - High Sustained Throughput
    - maximize steady-state throughput under heavy traffic

Target GALS Network Design

- Shared-Memory Chip Multiprocessors
- "Heterochronous" Timing
- Transition (Two-Phase) Signaling
- Single-Rail Bundled Data
- High Performance
- Standard Cell Methodology
  - Use existing standard cell libraries
    - only exception: analog arbiter circuit
  - Challenge: timing analysis using existing tools

Target GALS Network Design

- Shared-Memory Chip Multiprocessors
- “Heterochronous” Timing
- Transition (Two-Phase) Signaling
- Single-Rail Bundled Data
- High Performance
- Standard Cell Methodology
- Fine-Grained Network Topology
  - Lightweight network nodes
    - *low-functionality* low-radix router components
    - avoids 5-port router with North/South/East/West/Local ports

Outline

- Introduction
- Target GALS Network Design
- Background: XMT Processor / MoT Network
  - eXplicit Multi-Threading (XMT) Architecture
  - Mesh-of-Trees (MoT) Network Topology
  - Synchronous Router Nodes
- Asynchronous Network Primitives
- Experimental Results
- Conclusions

XMT Parallel Architecture

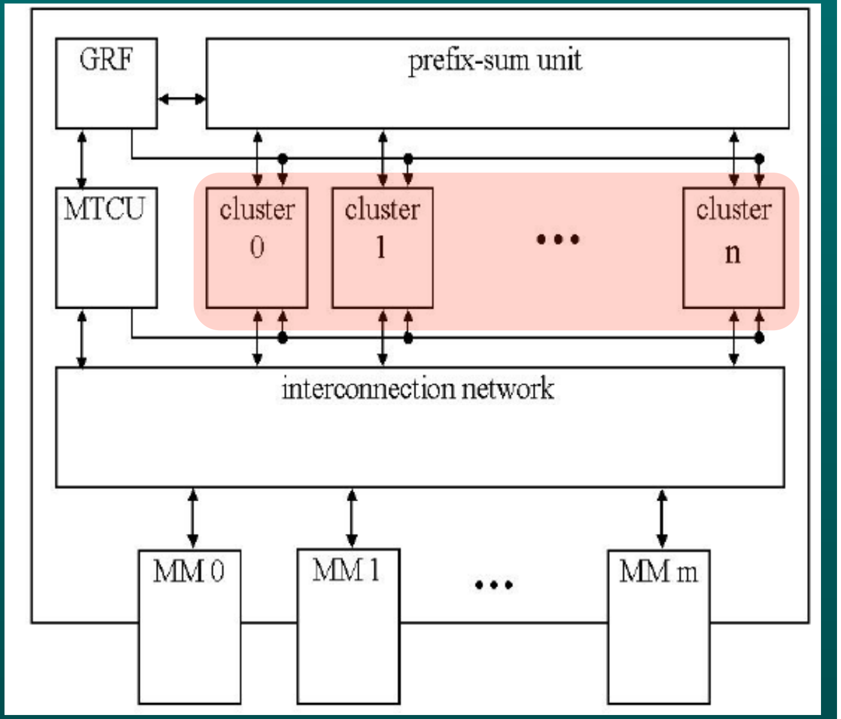
- XMT = "eXplicit Multi-Threading" (1997-present) [4]
  - Led by Prof. Uzi Vishkin at University of Maryland, College Park
- Based on Parallel Random Access Model (PRAM)
  - Largest body of parallel algorithmic theory
- Ease of Programmability
  - XMT-C language + optimizing compiler
  - Single-Program Multiple-Data (SPMD) programming methodology
- Demonstrated to Provide Significant Speedups
  - Performs well on irregular computations (BFS, ray-tracing)
  - 100x speedup for VHDL circuit simulations compared to serial [5]

[4] D. Naishlos, J. Nuzman, C.-W. Tseng and U. Vishkin, *"Towards a first vertical prototyping of an extremely fine-grained parallel programming approach"*, SPAA 2001

[5] P. Gu and U. Vishkin, *"Case study of gate-level logic simulation on an extremely fine-grained chip multiprocessor",* Journal of Embedded Computing, April 2006

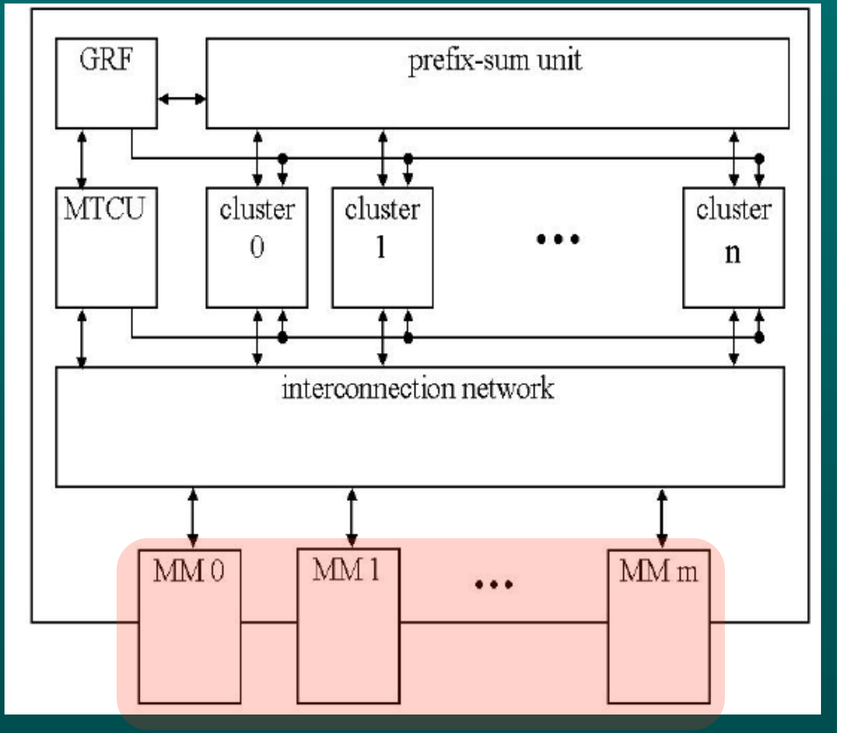
XMT Parallel Architecture

- Processing Clusters
  - Group of simple pipelined cores, e.g. 16 Thread Control Units (TCU)
  - Each TCU executes to completion with little to no synchronization
  - "IOS" = independence-of-order semantics: no WAW/WAR/RAW data hazards between threads



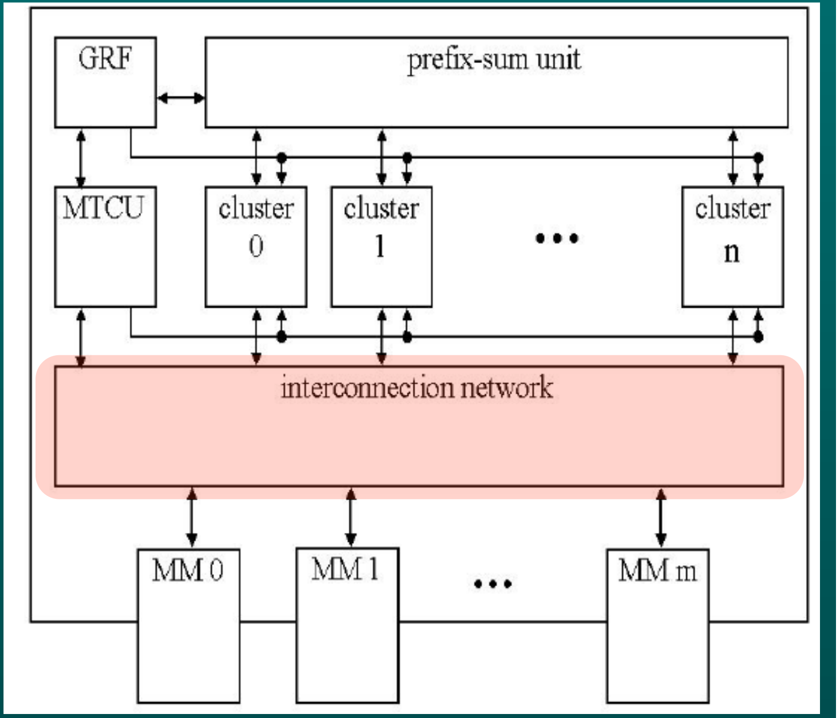
XMT Parallel Architecture

- Processing Clusters
  - Groups of simple pipelined cores, e.g. 16 Thread Control Units (TCU)
  - Each TCU executes to completion with little or no synchronization
- Distributed Caches
  - Shared global L1 data cache
  - No cache coherence problem



XMT Parallel Architecture

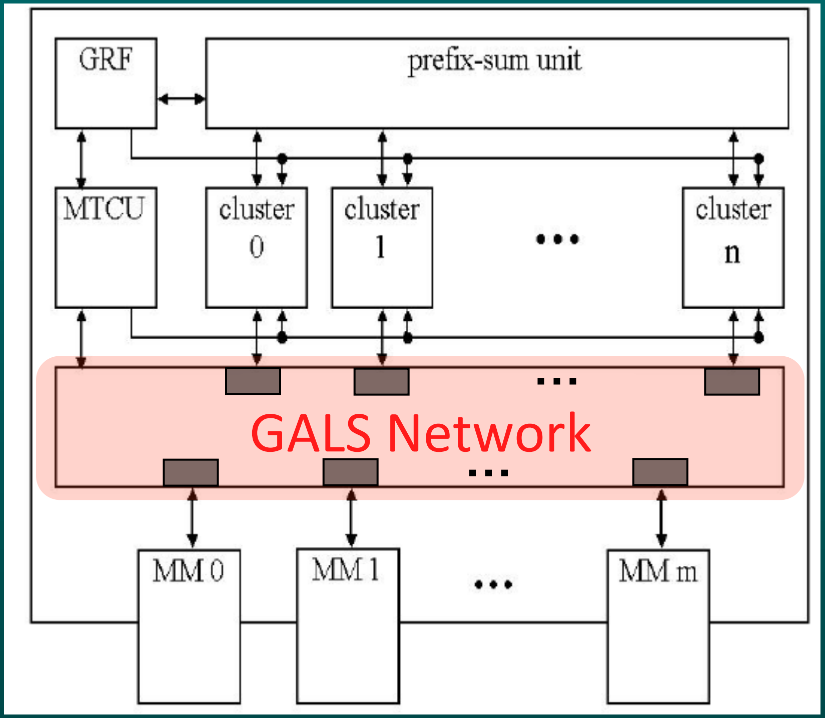
- Processing Clusters
  - Groups of simple pipelined cores, e.g. 16 Thread Control Units (TCU)
  - Each TCU executes to completion with little to no synchronization
- Distributed Caches
  - Shared global L1 data cache
  - No cache coherence problem



- NOC Challenge: high bandwidth/low power requirements
  - Many concurrent memory requests (load/store)
  - **Short packets:** 1-2 flits/dynamically-varying traffic
  - **Low latency:** required for system performance

Proposed XMT Parallel Architecture: with GALS Interconnection Network

-



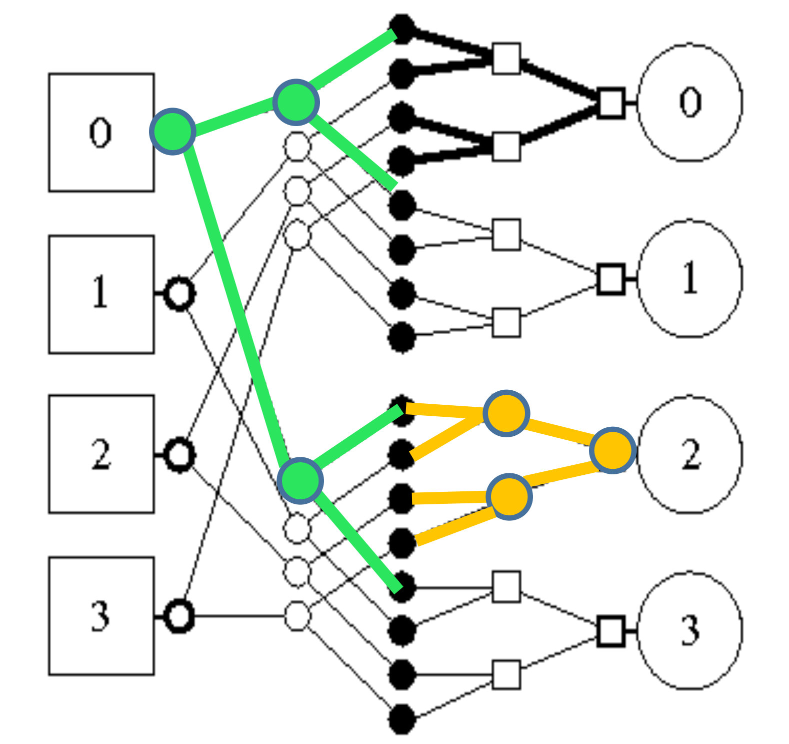
Mesh-of-Trees Network Topology

- Variant of classic MoT
- N fan-out trees
  - Routing only
  - Root at source terminals

- N fan-in trees
  - Arbitration only
  - Root at destination terminals

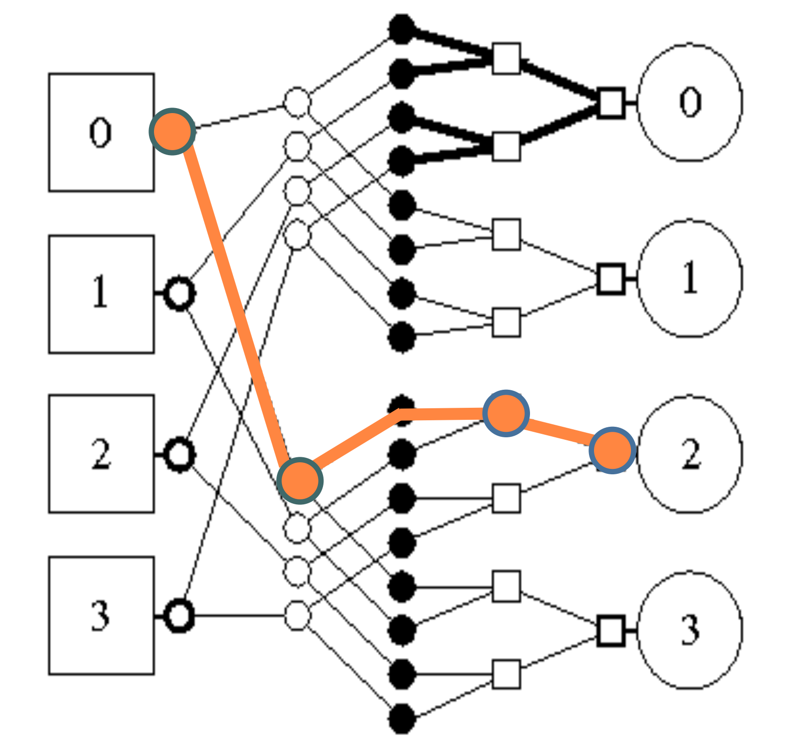
Routing

Arbitration



Mesh-of-Trees Network Topology

- High Throughput
  - Unique routing paths (source/sink)
  - Avoids interference penalties
- Fixed Path Length
  - Logarithmic depth
- Distributed Low-Radix Routing
  - Limited functionality nodes
  - Wormhole deterministic routing

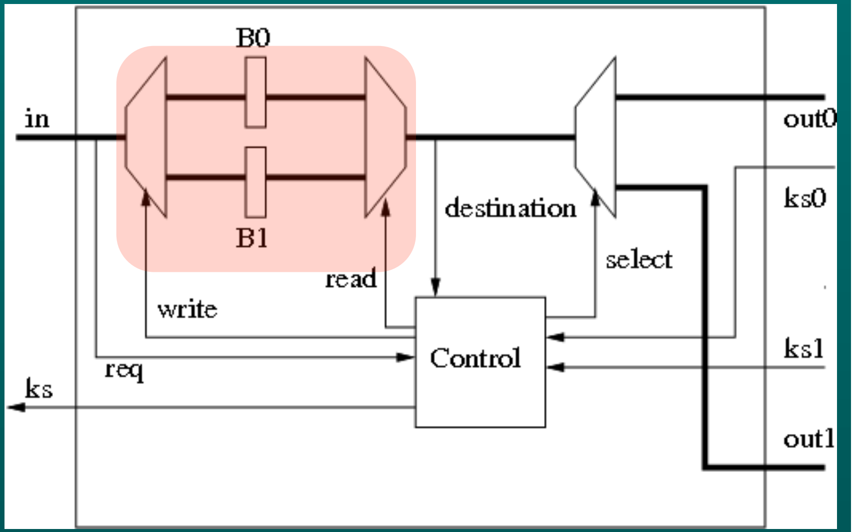


- Shown to Perform Well for CMPs
  - Provides very high sustained throughput [6]
  - High saturation throughput: ~91%

[6] A.O. Balkan, G. Qu, U. Vishkin, "*Mesh-of-Trees and alternative interconnection networks for single-chip parallelism*", IEEE Transactions on Very Large Scale Integration Systems, April 2009

Synchronous Routing Primitive

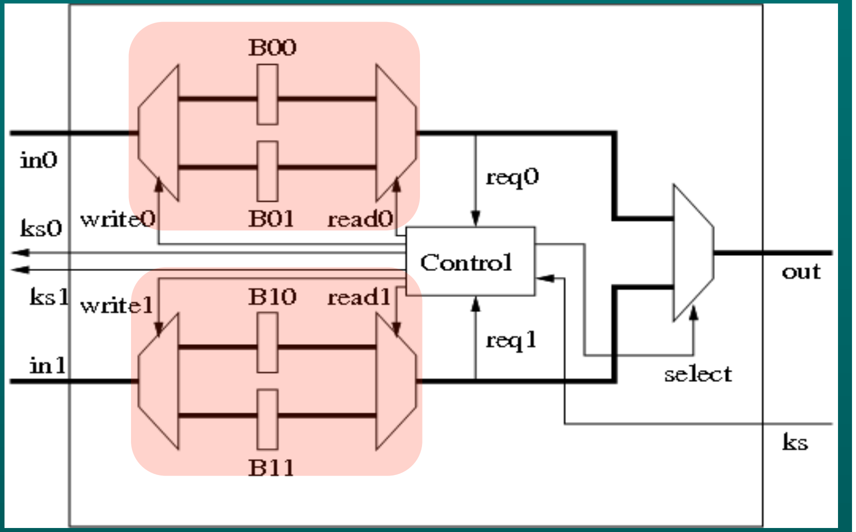
- Fan-Out Component [7]
  - 1 Input, 2 Outputs
  - Synchronous Flow Control
    - Back-pressure mechanism
    - Signal to previous stage when new data can be accepted



- Based on "Latency-Insensitive Design" [Carloni et al., TCAD 01]
  - 2-Register FIFO: B0, B1
  - Allows 1 flit/cycle in steady-state
    - Accept new data and forward stored data concurrently
  - Cost: *1 extra auxiliary register (flipflop-based)*

[7] A.O. Balkan, G. Qu, U. Vishkin. *"A Mesh-of-Trees Interconnection Network for Single-Chip Parallel Processing",* IEEE ASAP Symposium (2006)

Synchronous Arbitration Primitive



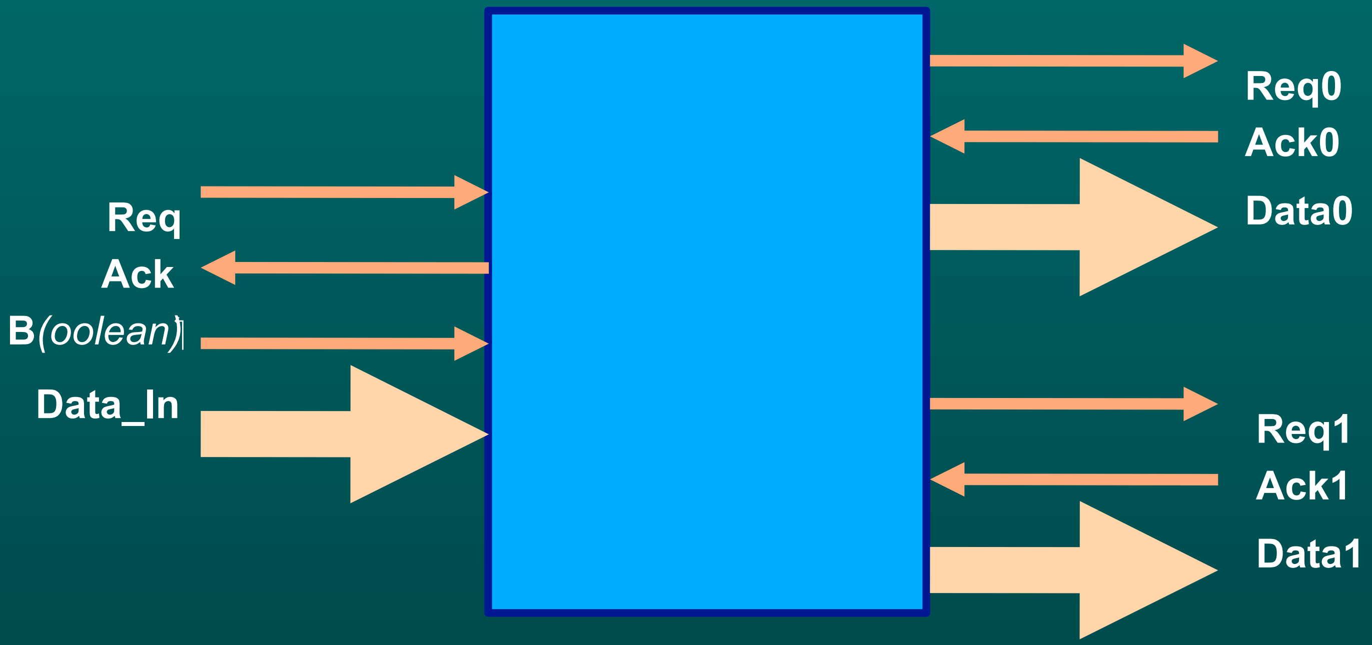
- Fan-In Component [7]
  - 2 Inputs, 1 Output
  - Synchronous Flow Control
    - Back-pressure mechanism
- Based on "Latency-Insensitive Design"
  - 2-Stage FIFOs at each input port
  - When empty, latency = 1 cycle
  - When stalled, latency = 2+ cycles
    - Depends on back-pressure and synchronous arbitration
  - Cost: *total of 4 registers (flip-flop based)*

[7} A.O. Balkan, G. Qu, U. Vishkin. *"A Mesh-of-Trees Interconnection Network for Single-Chip Parallel Processing",* IEEE ASAP Symposium (2006)

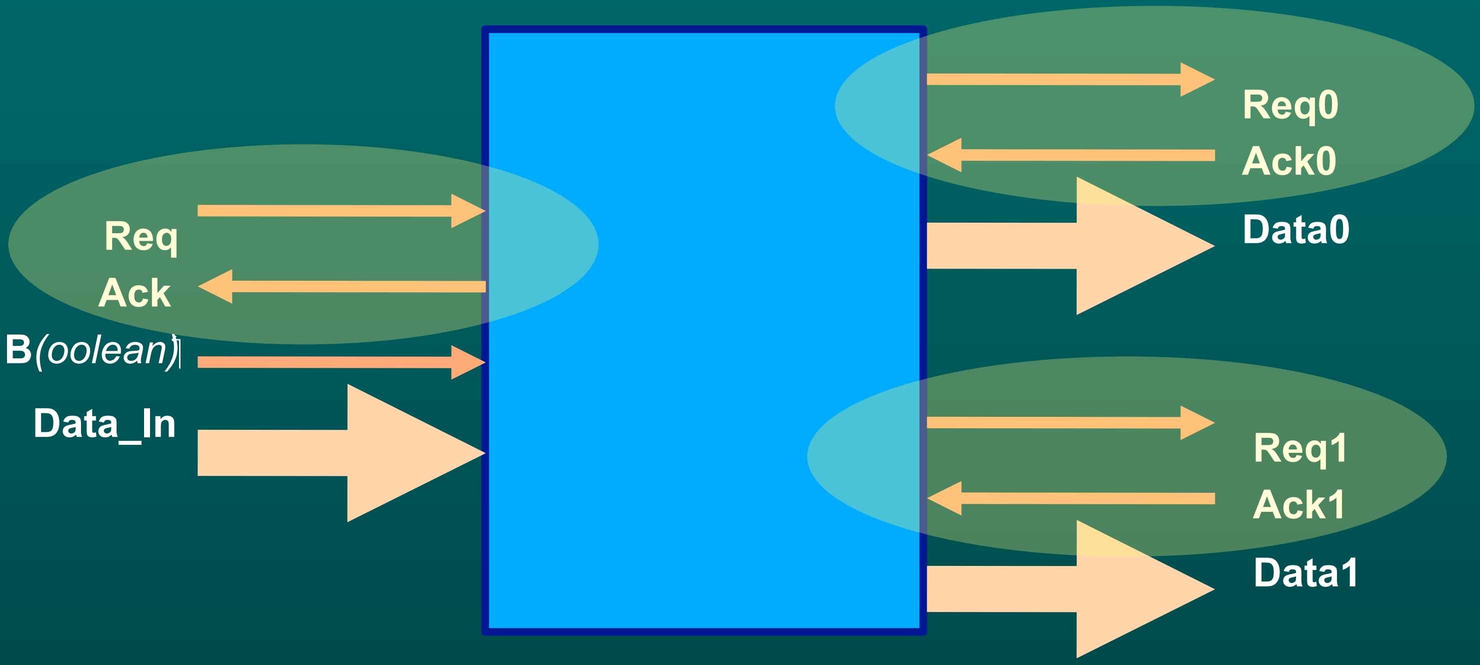
Outline

- Introduction
- Target GALS Network Design
- Background: XMT Processor / MoT Network
- Asynchronous Network Primitives
  - Routing primitive (Fan-out)
  - Arbitration primitive (Fan-in)
  - Mixed-timing interfaces
- Experimental Results
- Conclusions

New Routing Primitive

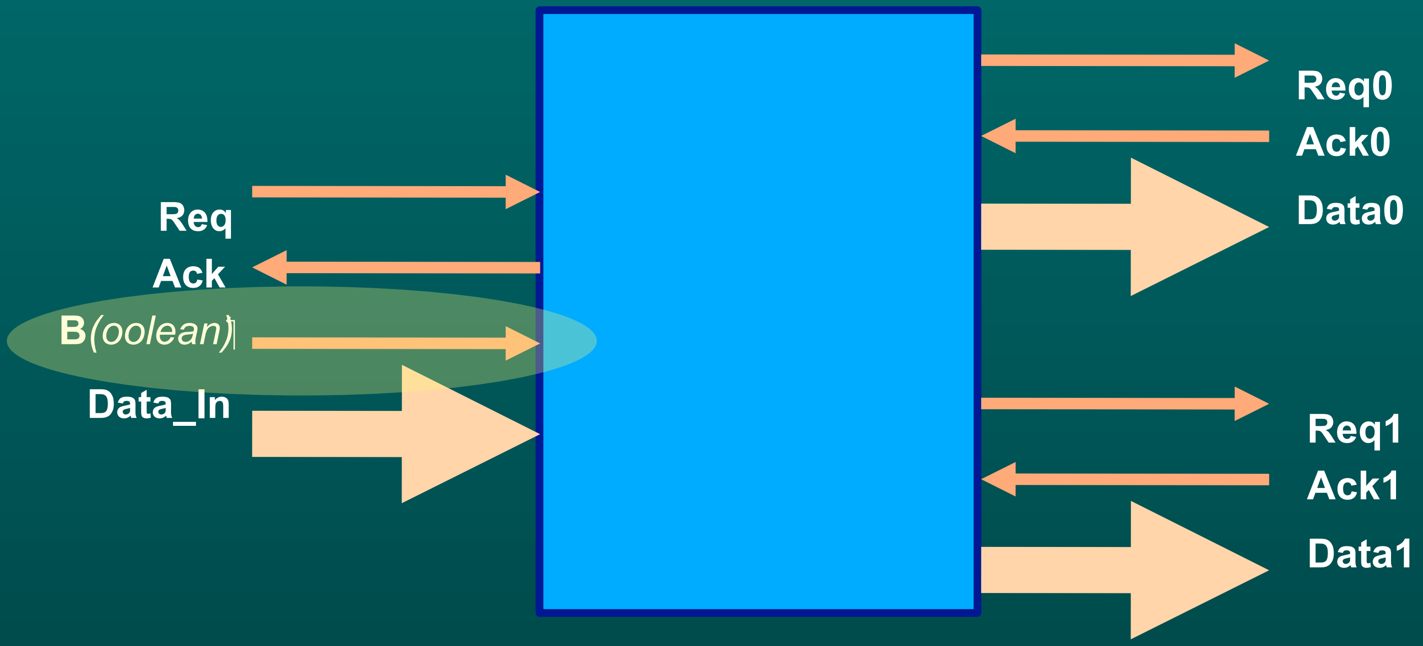


New Routing Primitive



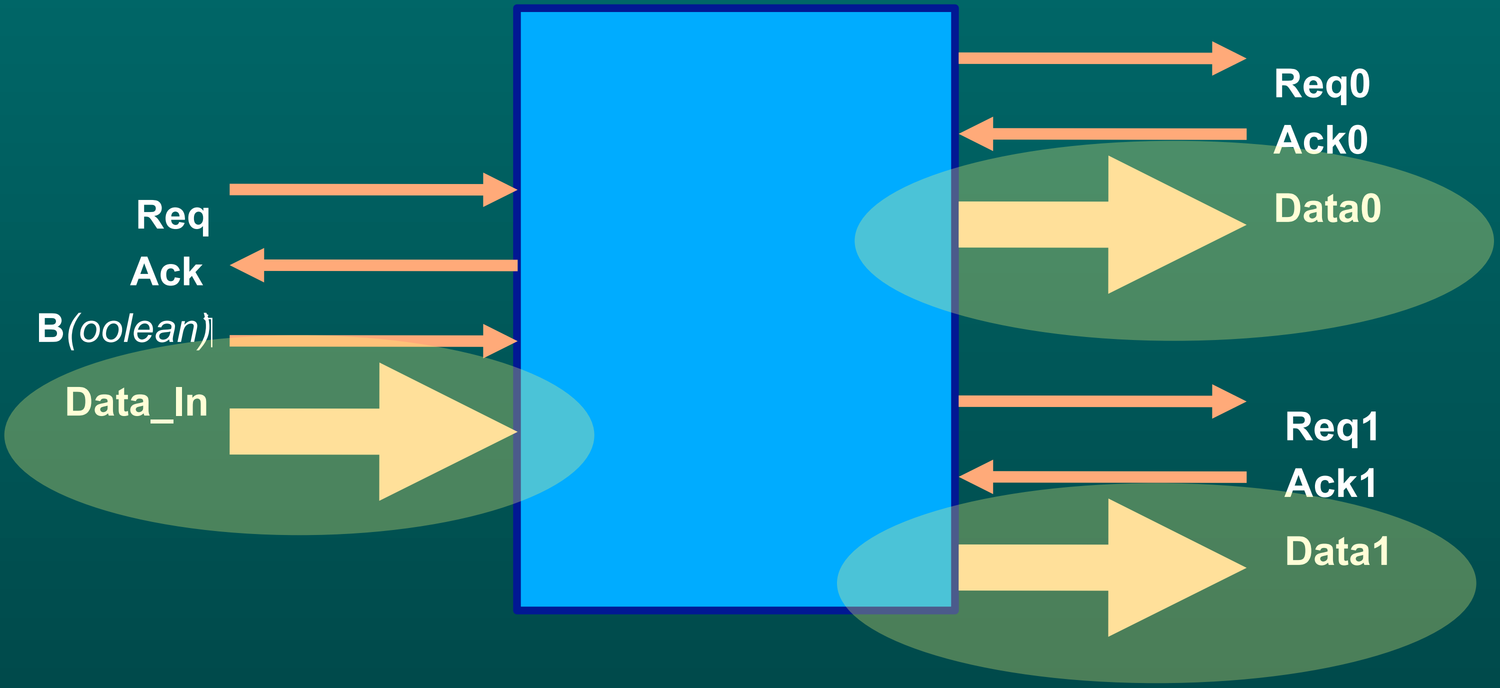
Handshaking Signals (Request / Acknowledge)

New Routing Primitive



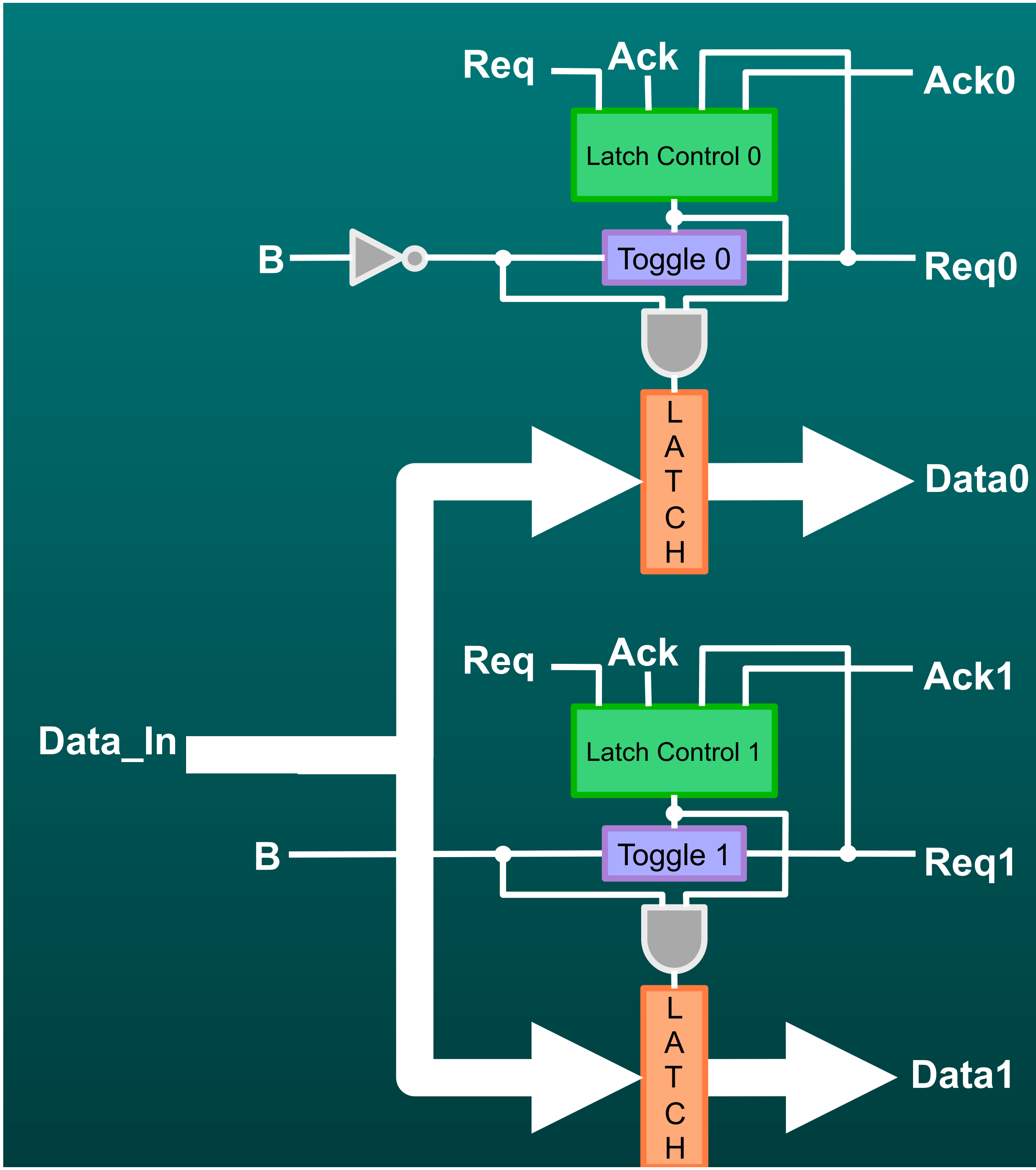
Binary Routing Signal

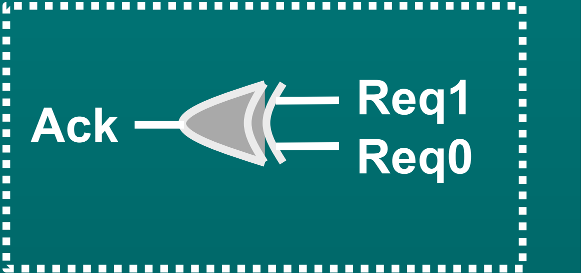
New Routing Primitive

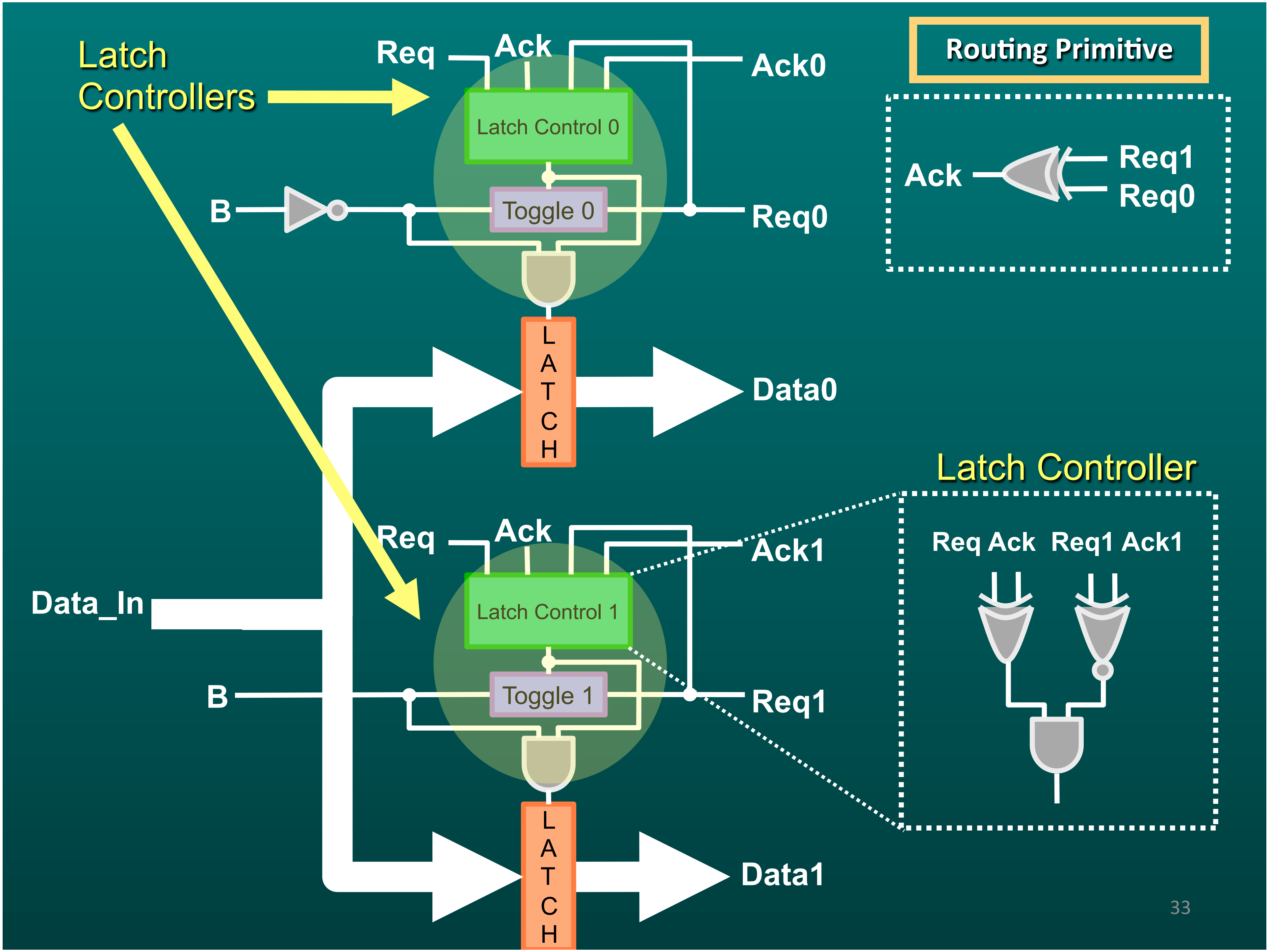


Data Channels

Routing Primitive





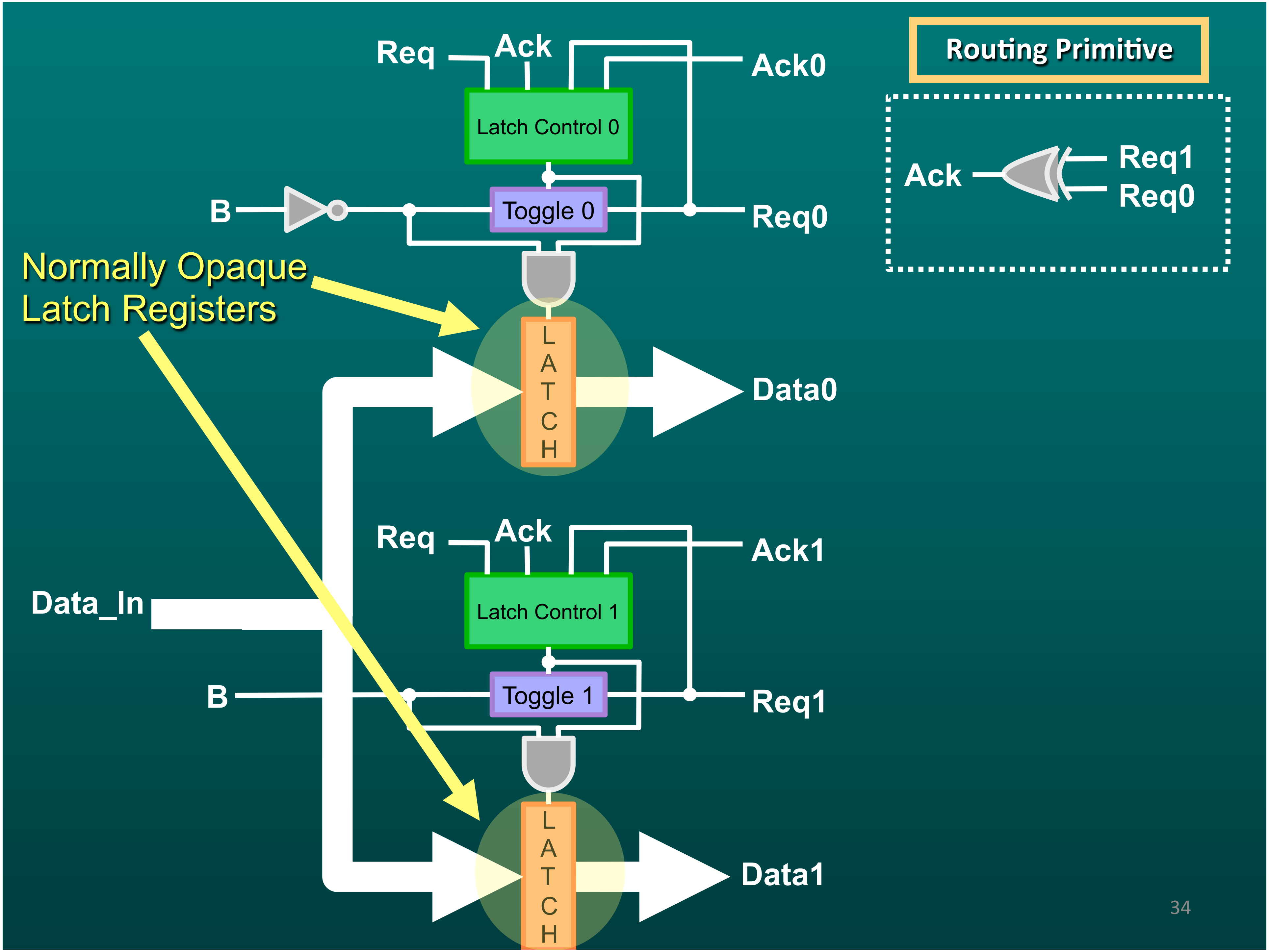


Latch Controllers

Routing Primitive

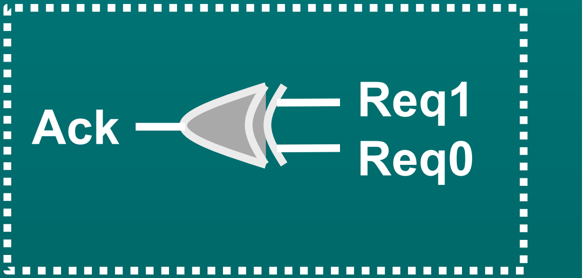
Latch Controller

Routing Primitive

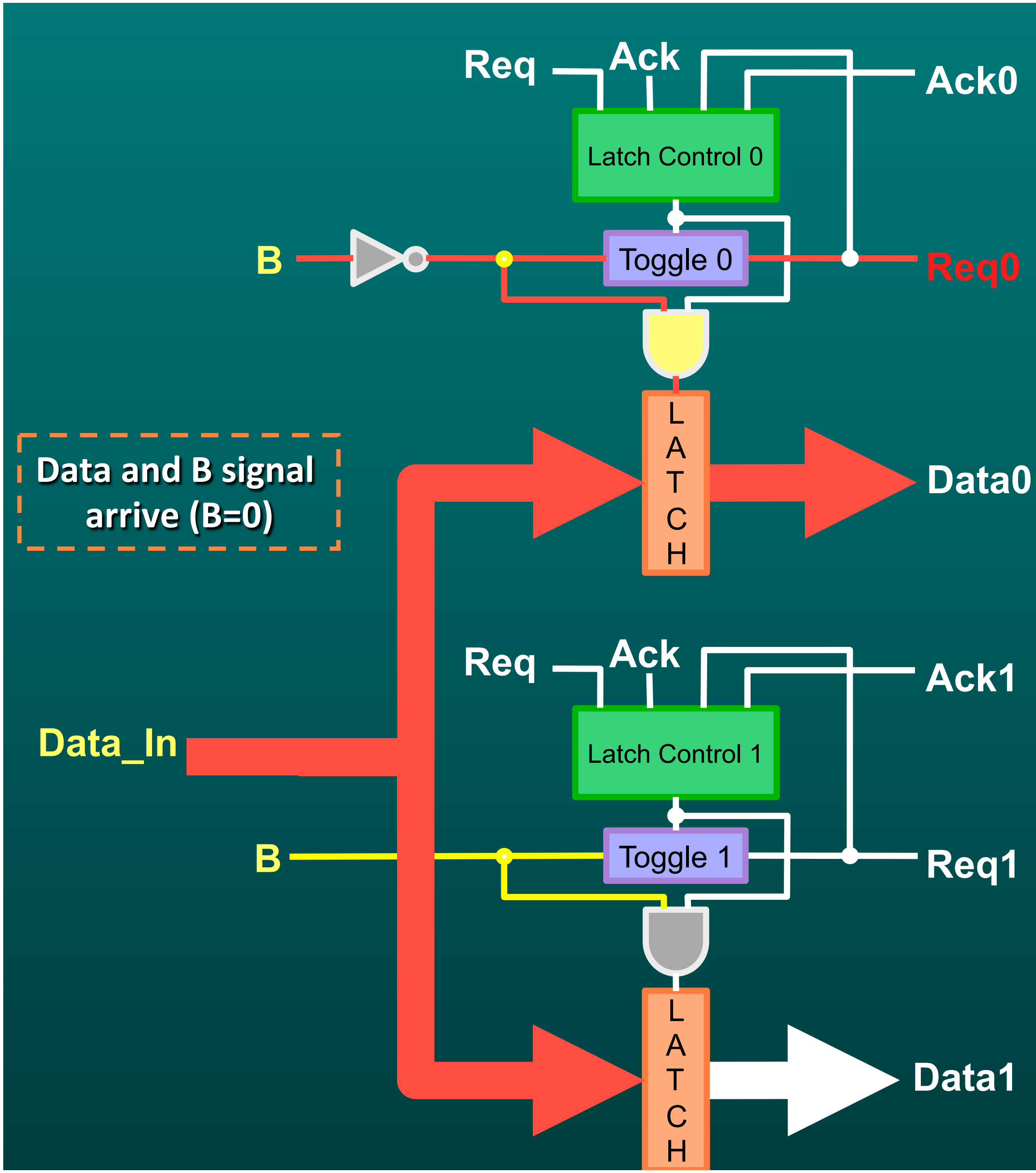


Normally Opaque Latch Registers

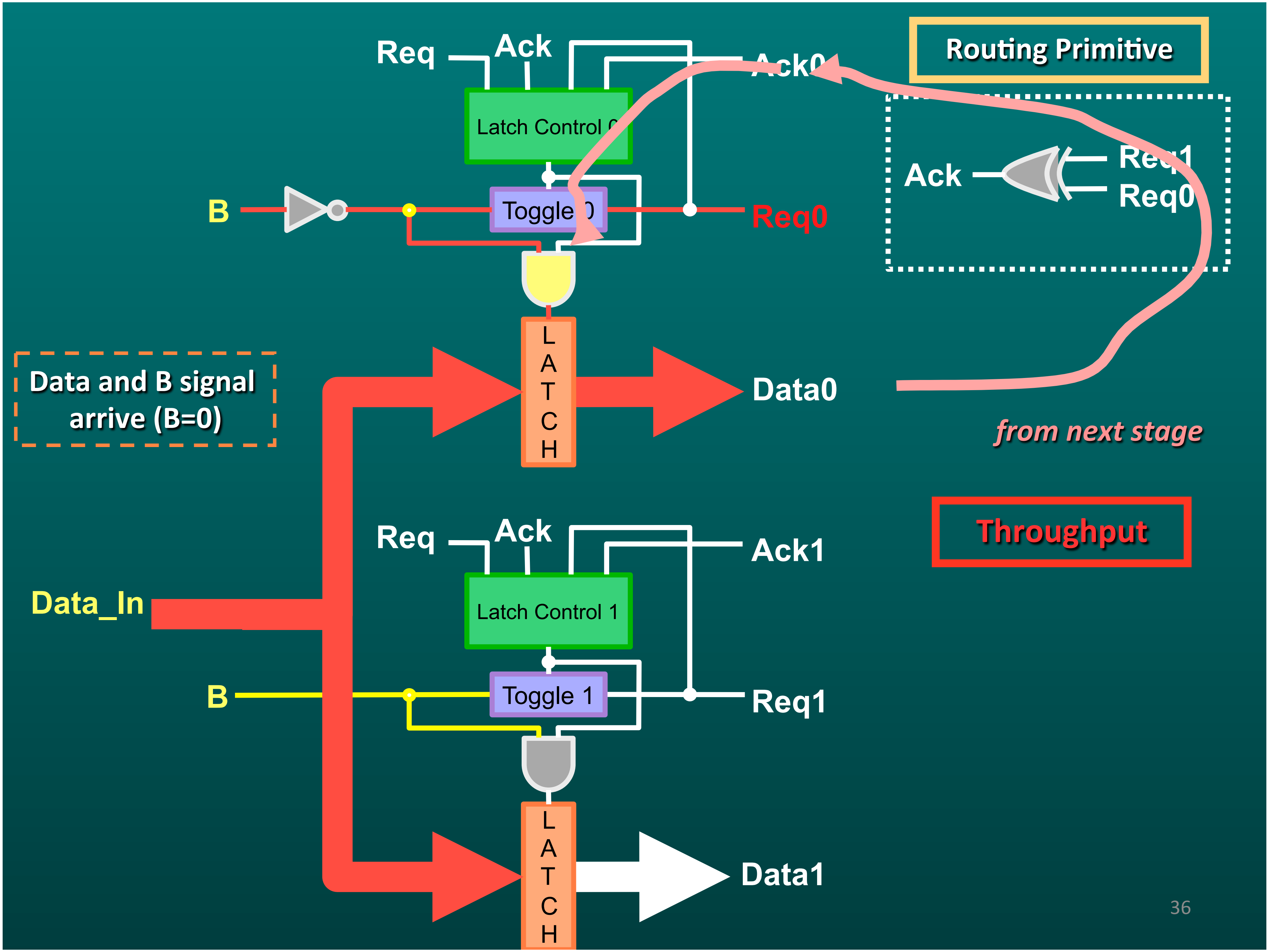
Routing Primitive



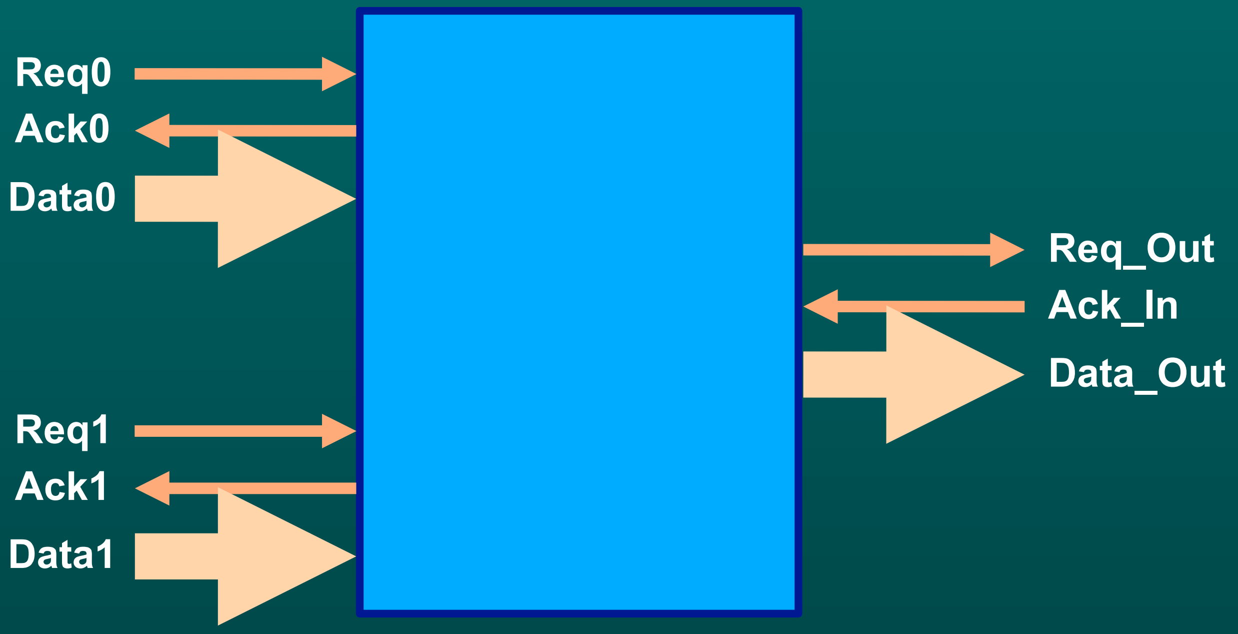
Data and B signal arrive (B=0)



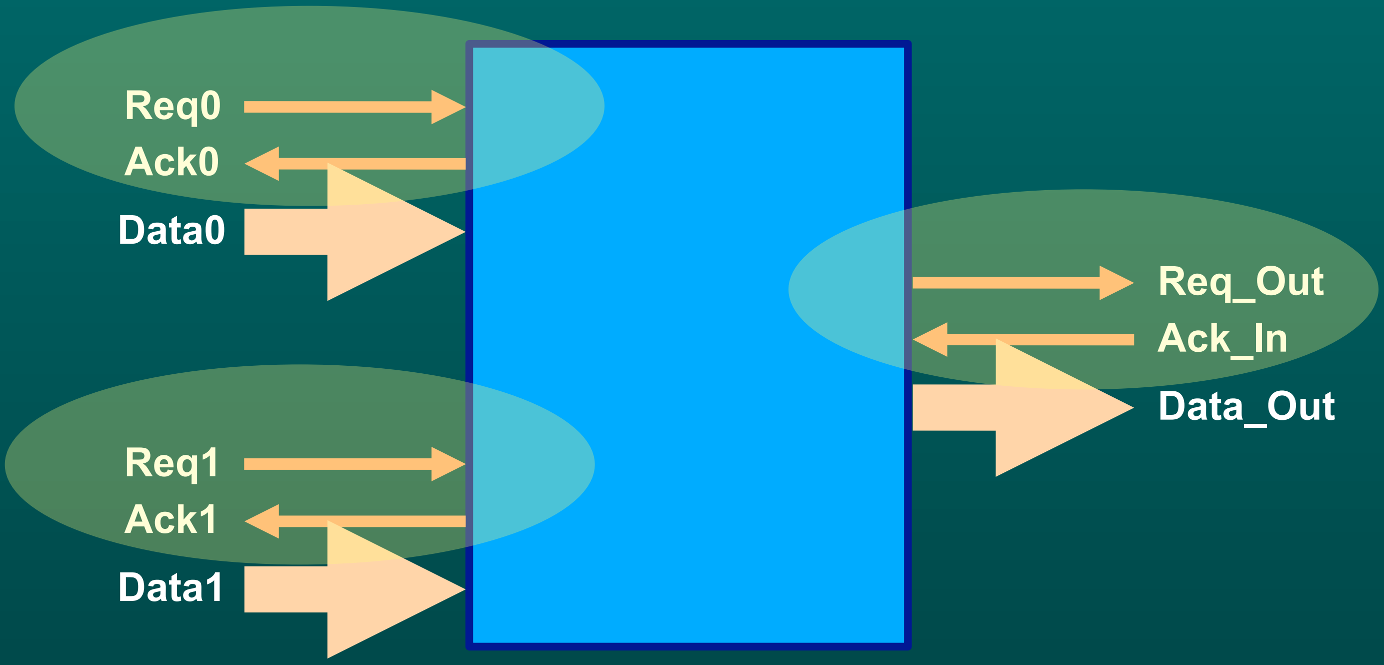
Latency


Req
Ack
Ack0
Latch Control 0
B
Toggle 0
Req0
L
A
T
C
H
Data0
Routing Primitive
Ack
Req1
Req0
from next stage
Data and B signal arrive (B=0)
Req
Ack
Ack1
Latch Control 1
Data_In
B
Toggle 1
Req1
L
A
T
C
H
Data1
Throughput

New Arbitration Primitive

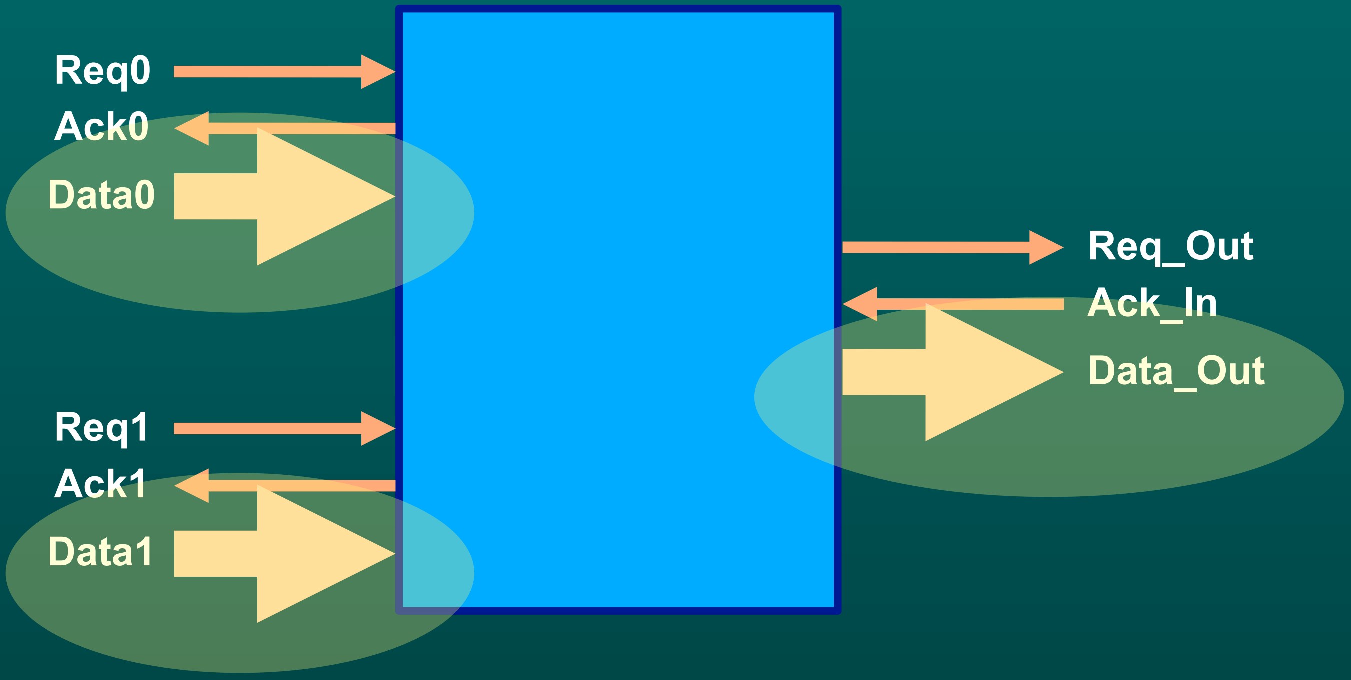


New Arbitration Primitive

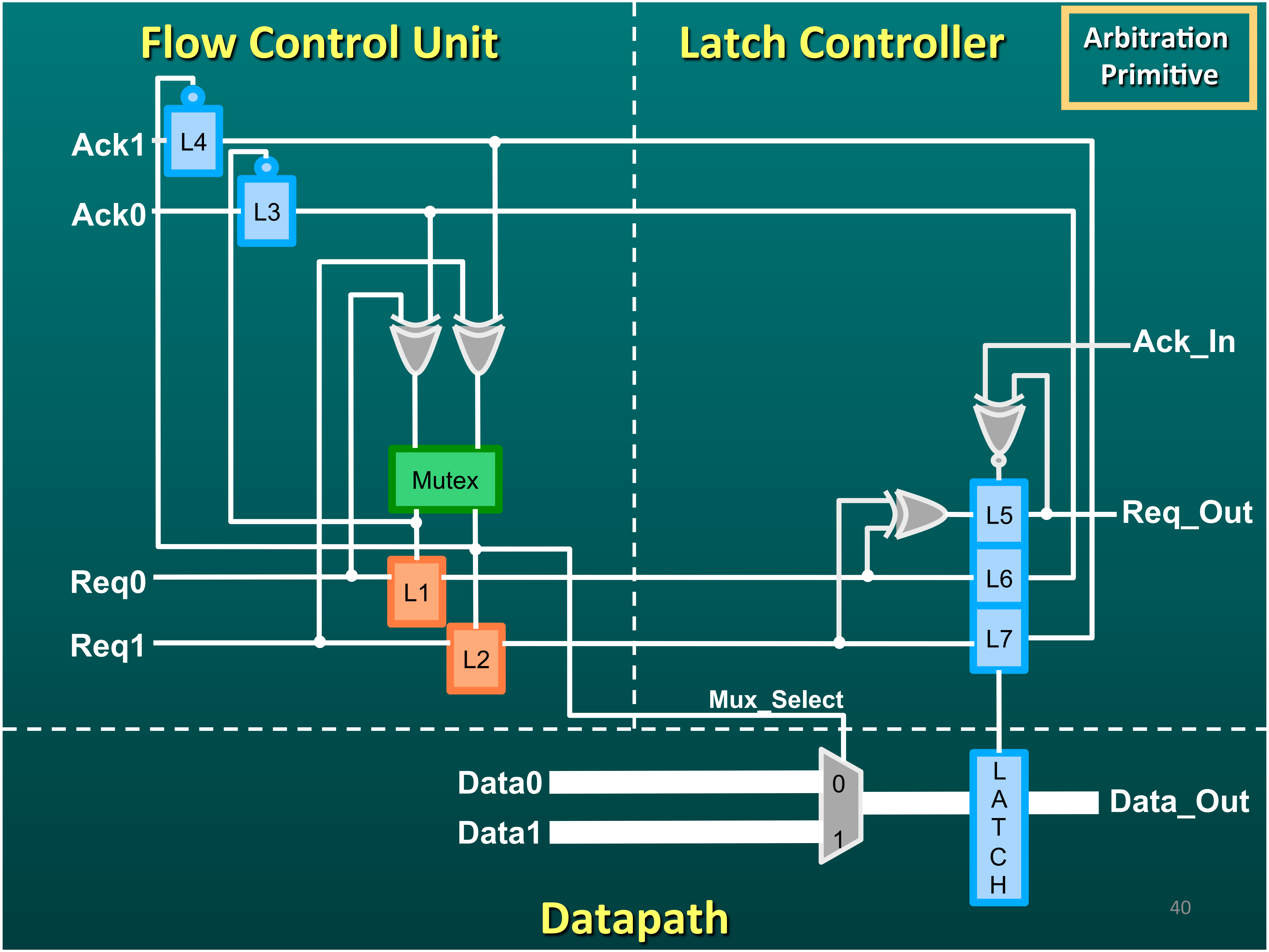


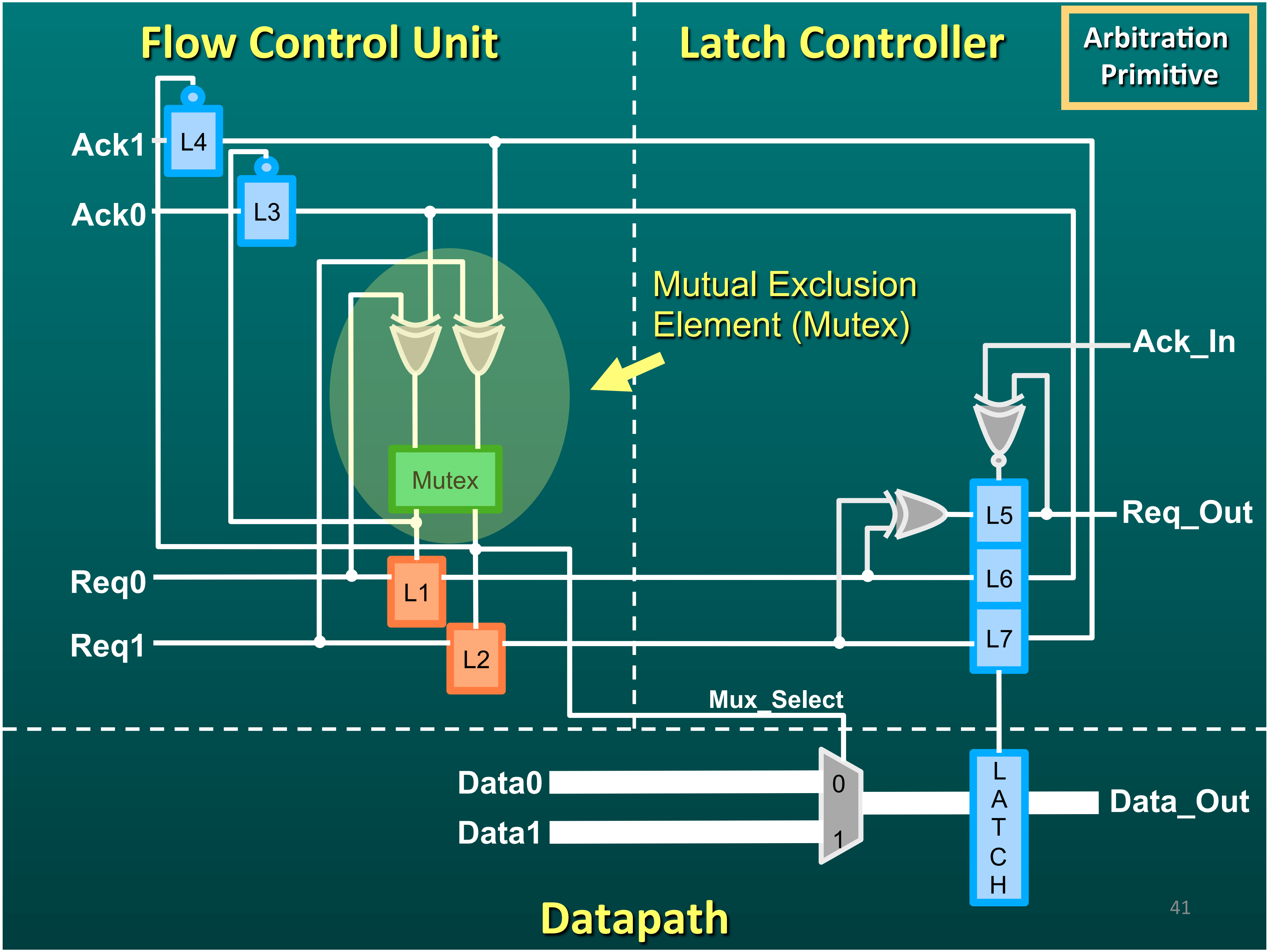
Handshaking Signals (Request / Acknowledge)

New Arbitration Primitive



Data Channels

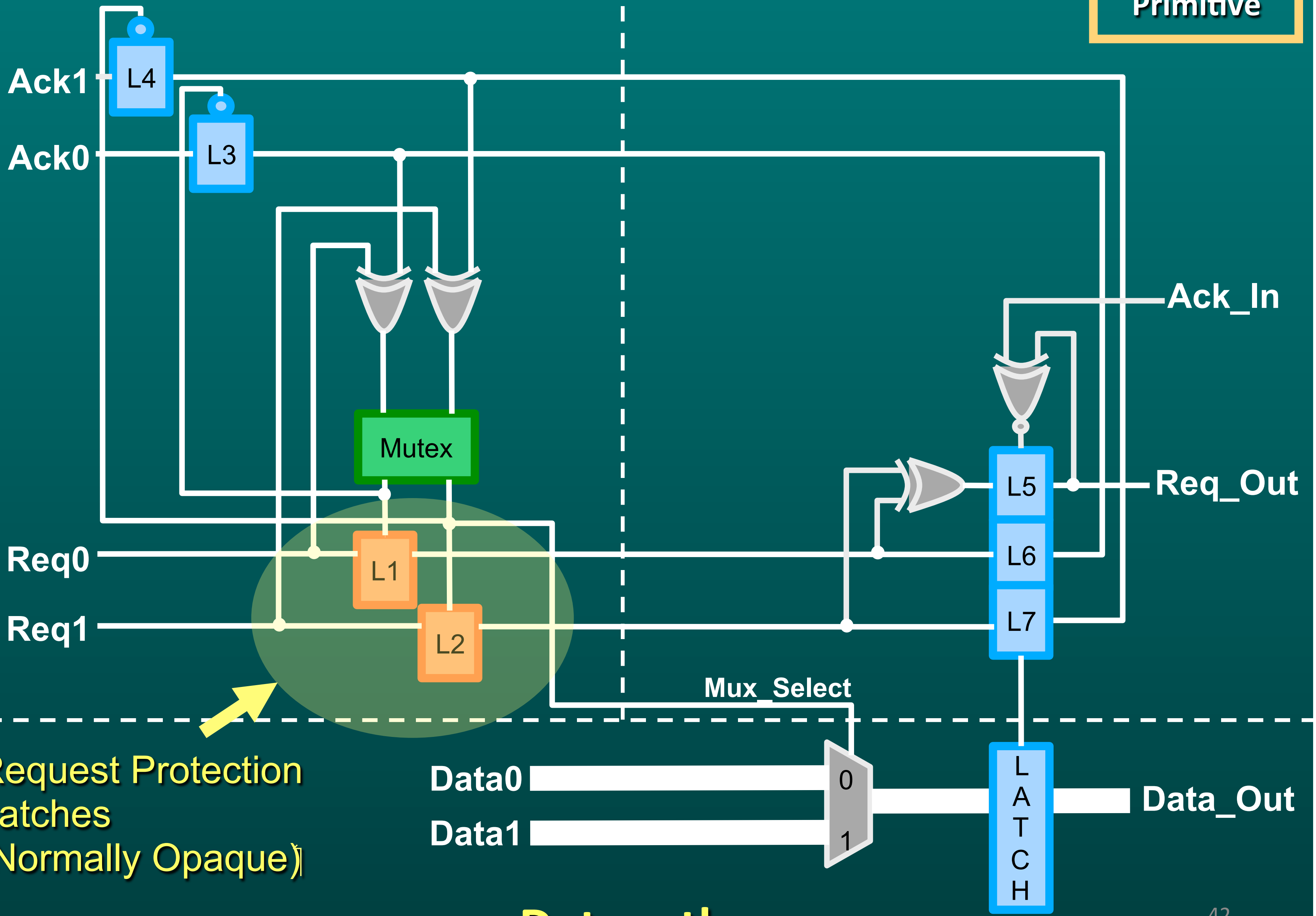

Flow Control Unit
Latch Controller
Arbitration Primitive
Ack1
L4
Ack0
L3
Mutex
Req0
L1
Req1
L2
Ack_In
L5
Req_Out
L6
L7
Mux_Select
Data0
Data1
0
1
LATCH
Data_Out
Datapath


Flow Control Unit
Latch Controller
Arbitration Primitive
Ack1
L4
Ack0
L3
Mutual Exclusion Element (Mutex)
Ack_In
Mutex
L5
Req_Out
L6
L7
Req0
L1
Req1
L2
Mux_Select
Data0
0
Data1
1
LATCH
Data_Out
Datapath

Arbitration Primitive

Flow Control Unit

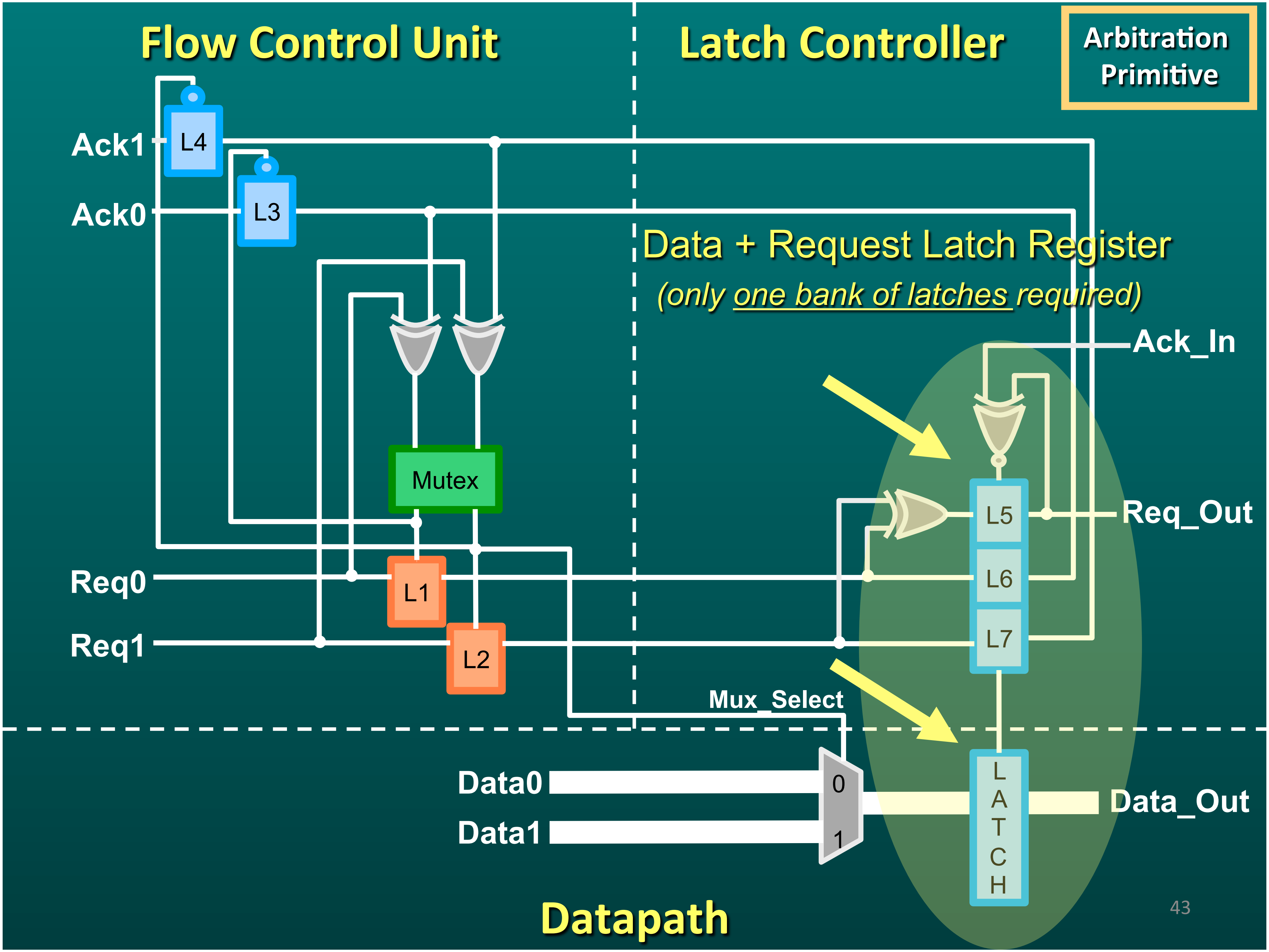
Latch Controller



Request Protection Latches (Normally Opaque)

Datapath

Arbitration Primitive



Flow Control Unit

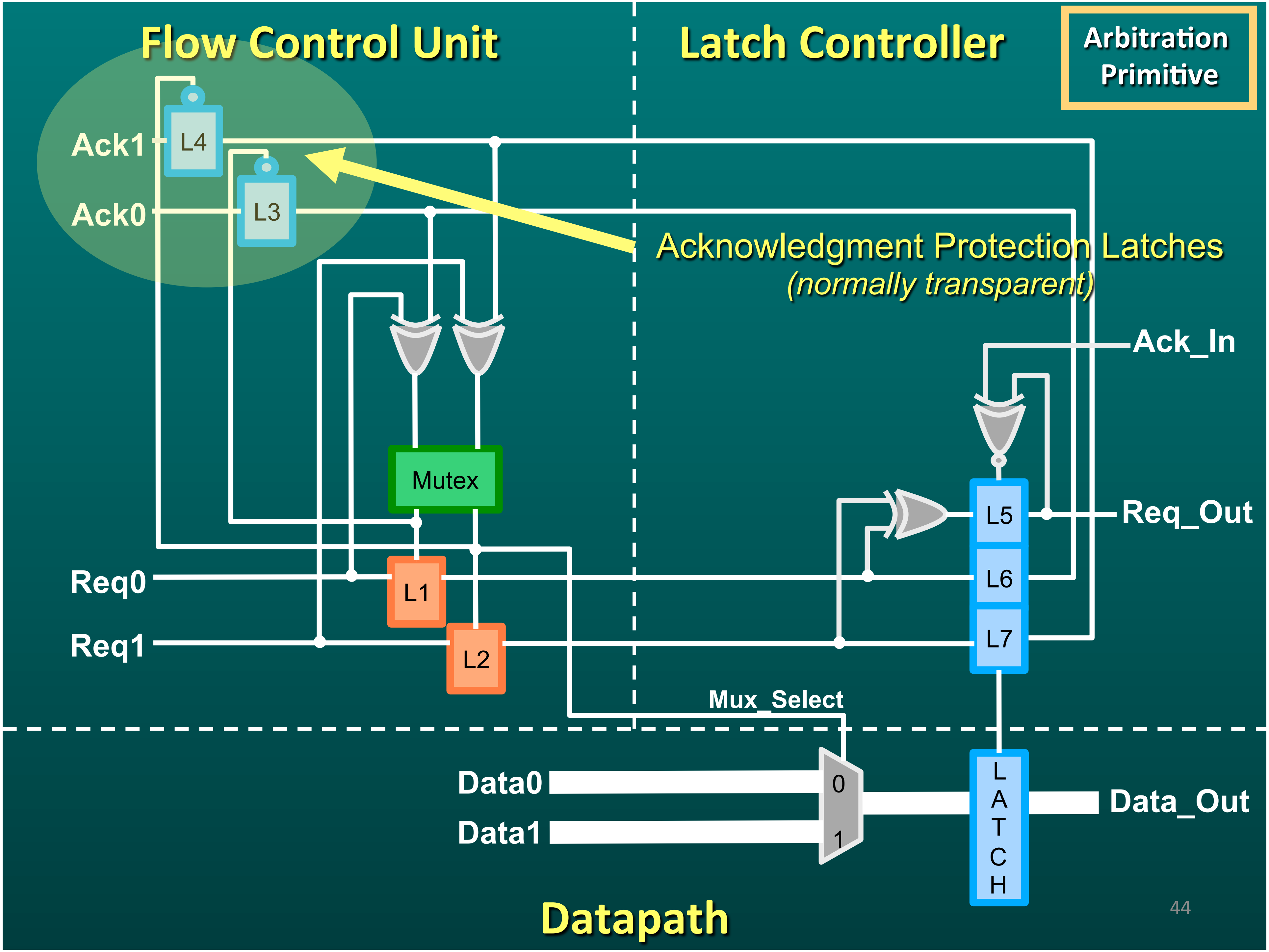
Latch Controller

Data + Request Latch Register

(only one bank of latches required)

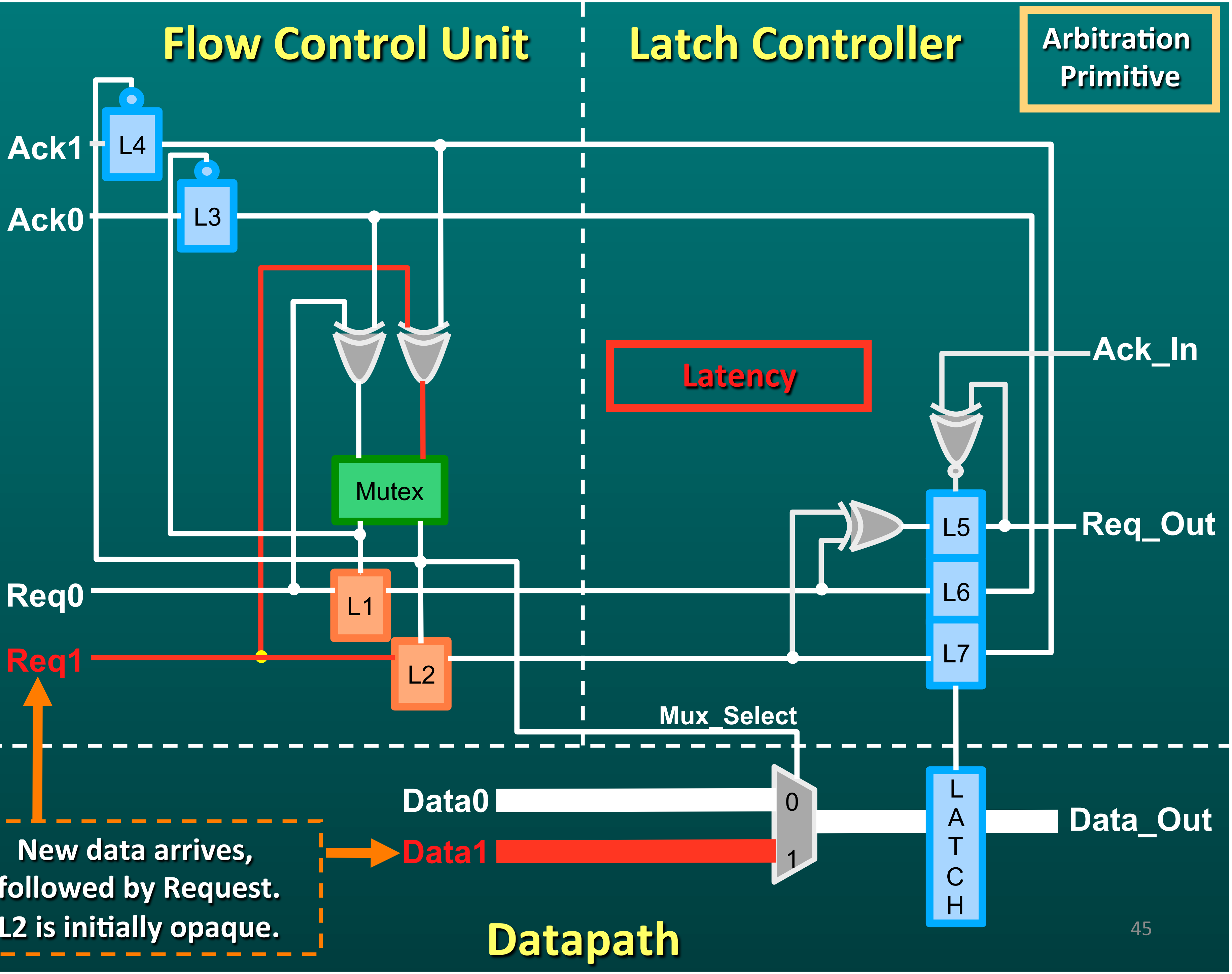
Datapath

Arbitration Primitive



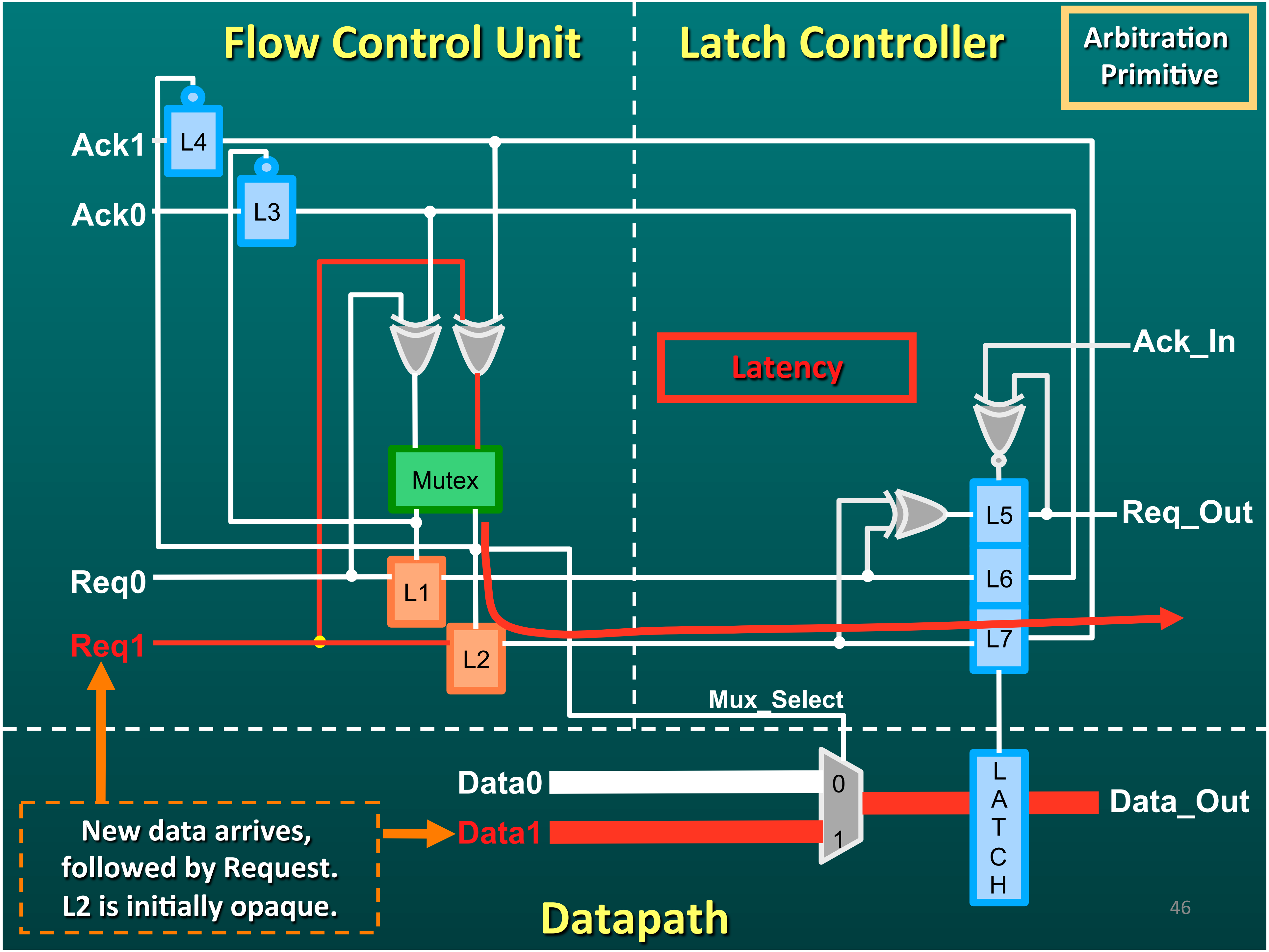
Acknowledgment Protection Latches
(normally transparent)

Arbitration Primitive

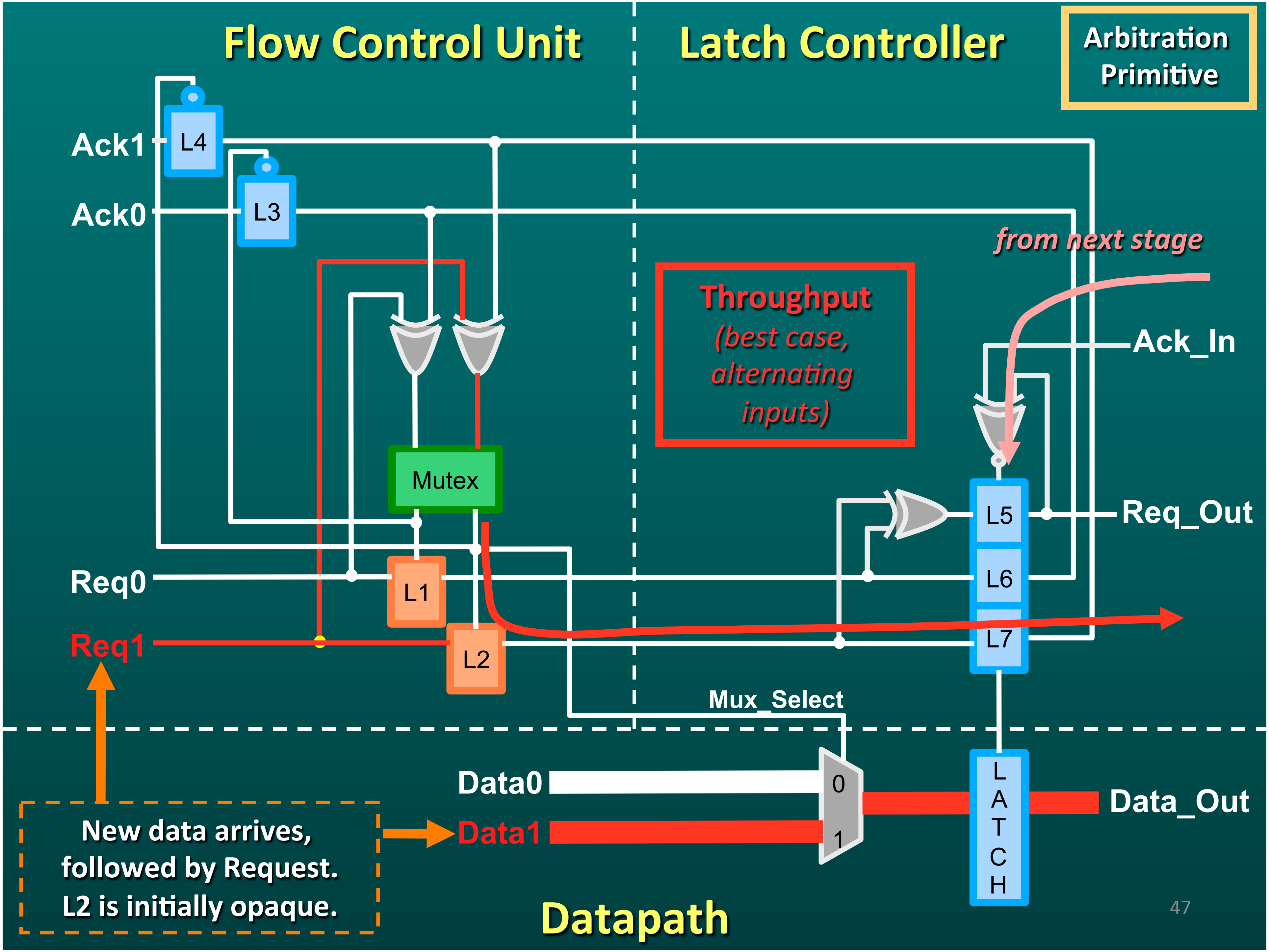


New data arrives, followed by Request. L2 is initially opaque.

Arbitration Primitive



New data arrives, followed by Request. L2 is initially opaque.



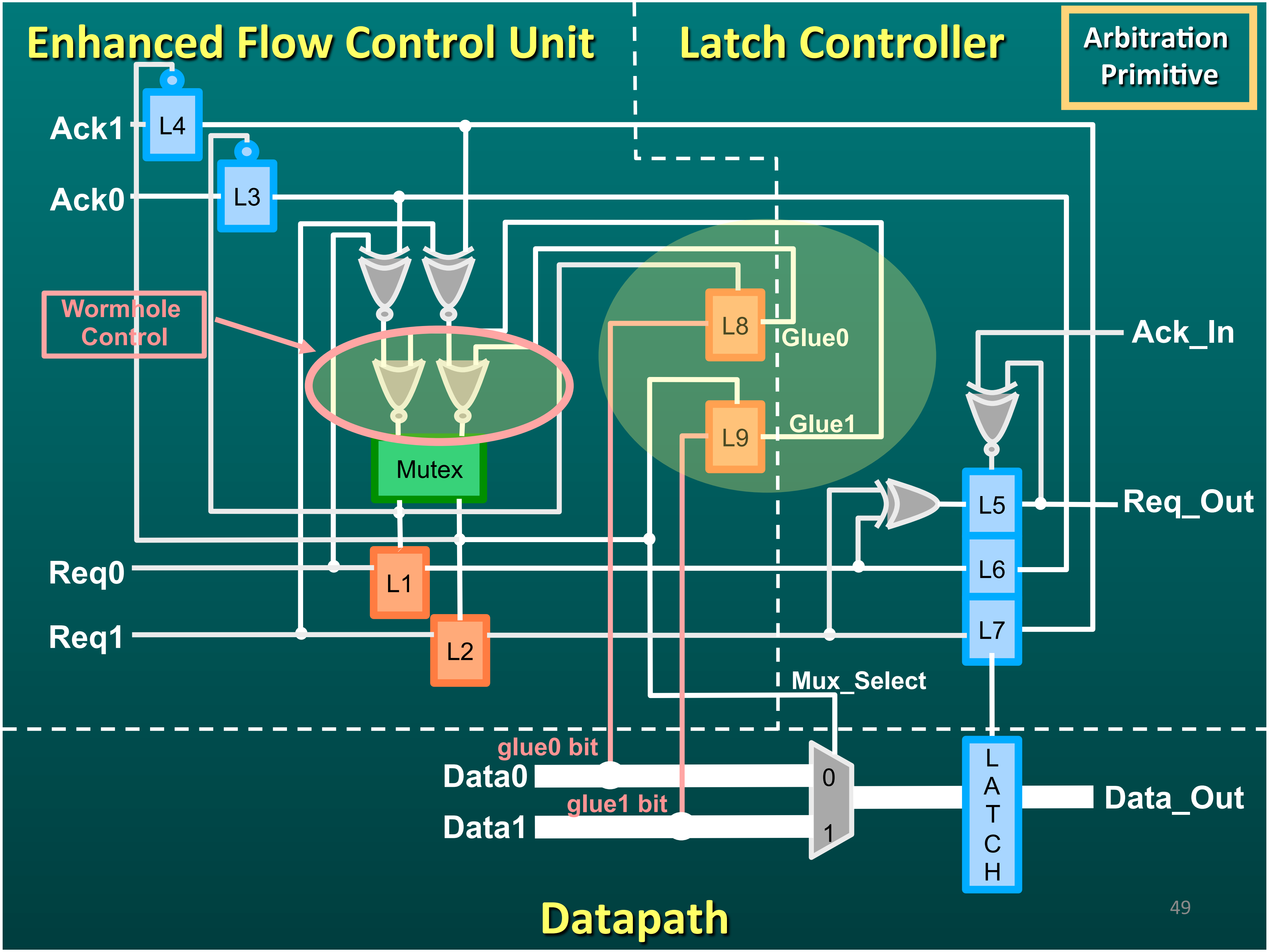
Throughput

(best case, alternating inputs)

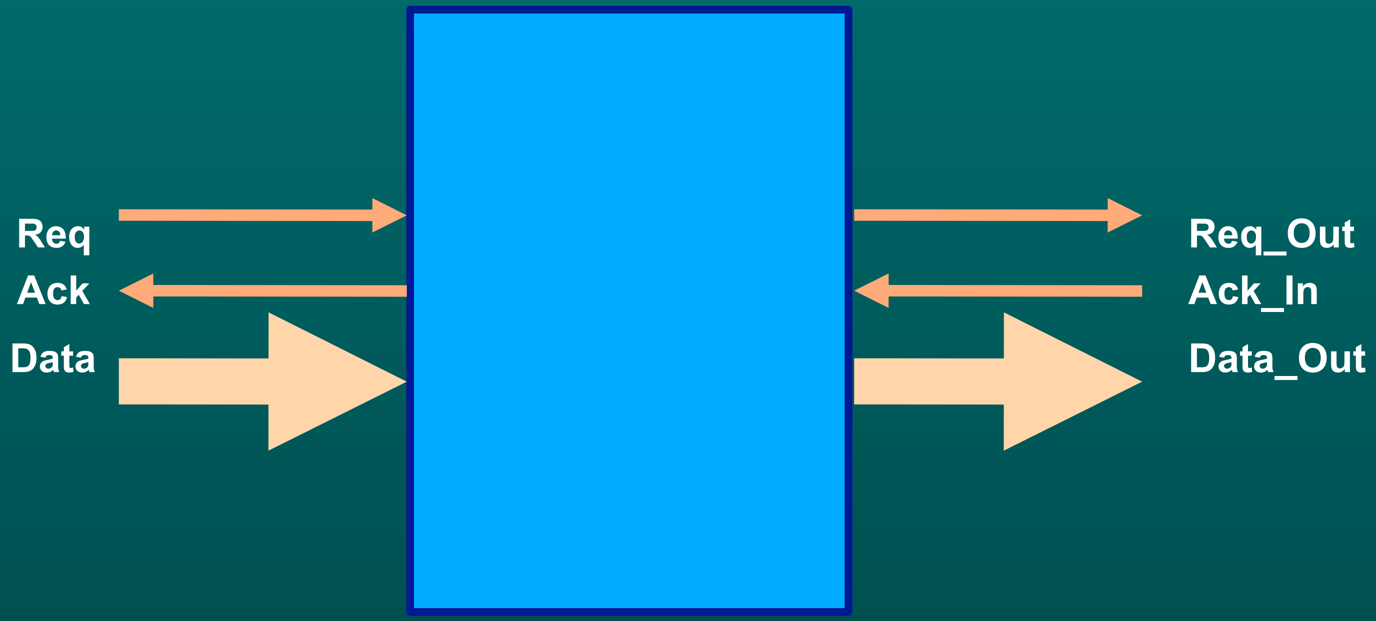
New data arrives,
followed by Request.
L2 is initially opaque.

Wormhole Routing Capability

- Goal: support transmission of multi-flit packets
  - example: XMT *"store* packets" = 2 flits (address + data)
- Solution: add 1 extra "glue bit" to each flit
  - Glue bit = 1 → not last flit in packet
  - Enhanced arbitration primitive: bias mutex decision
    - "winner-take-all" strategy [Dally/Towles]
    - header flit takes over mutex: glue = 1
    - last flit releases mutex: glue = 0


Enhanced Flow Control Unit
Latch Controller
Arbitration Primitive
Ack1
L4
Ack0
L3
Wormhole Control
Mutex
L8
Glue0
L9
Glue1
Ack_In
L5
Req_Out
L6
L7
Req0
L1
Req1
L2
Mux_Select
glue0 bit
Data0
0
glue1 bit
Data1
1
LATCH
Data_Out
Datapath

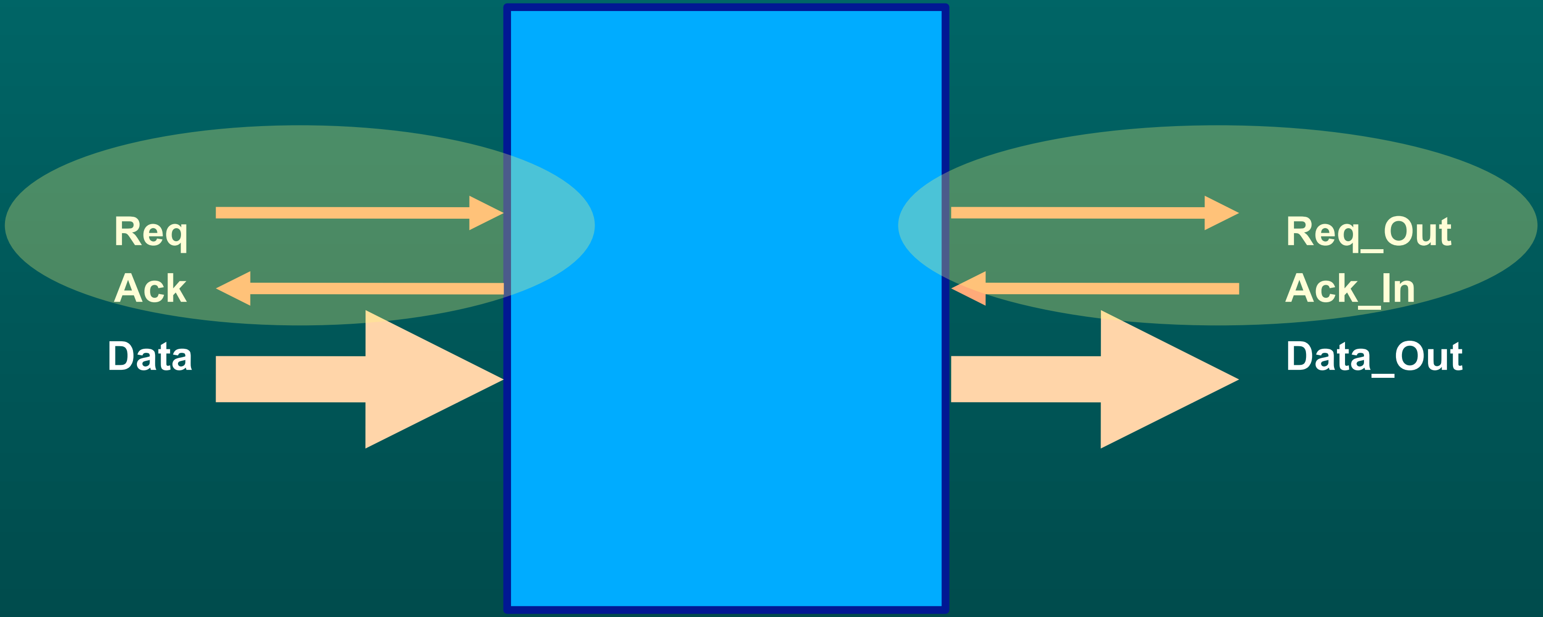
Linear Pipeline Primitive



- Can be inserted for buffering: to improve system-level throughput
- Basis for design of new fan-in/fan-out primitives

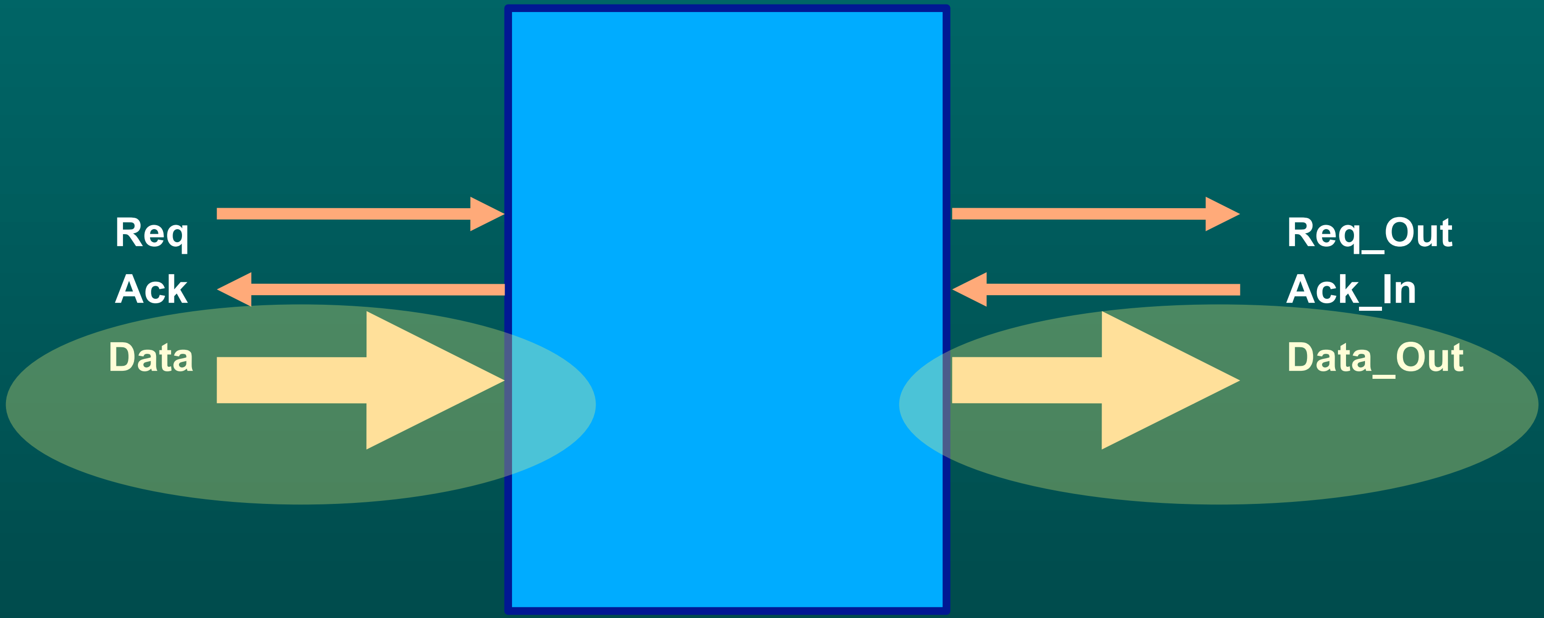
[8] M. Singh and S.M. Nowick. *"MOUSETRAP: High-Speed Transition-Signaling Asynchronous Pipelines,"* IEEE Transactions on VLSI Systems, vol. 15:11, pp. 1256-1269 (Nov. 2007)

Linear Pipeline Primitive



Handshaking Signals (Request and Acknowledgment)

Linear Pipeline Primitive



Data Channels

Mixed-Timing Interfaces

- Use Existing Synchronizing FIFOs [9]
  (with small modifications)
  - Supports arbitrary “heterochronous” timing domains
  - No modification to existing components
- Modular Design
  - Reusable *Put* and *Get* components (either Async or Sync)
  - Each FIFO is array of identical cells
- Supports Low-Power Operation
  - Circular FIFO: data does not move

[9] T. Chelcea and S. Nowick, *“Robust Interfaces for Mixed-Timing Systems”,*
IEEE Transactions on Very Large Scale Integration Systems, August 2004

Outline

- Introduction
- Target GALS Network Design
- Background: XMT Processor / MoT Network
- Asynchronous Network Primitives
- Experimental Results
- Conclusions

Evaluation Methodology

- Direct Comparison with Synchronous MoT Network
  - Identical Technology: IBM 90nm CMOS process
  - Identical Functionality: Same routing and arbitration primitives
  - Identical Topology: 8-terminal networks with same floorplan
- Evaluate at Multiple Levels of Integration
  - Isolated Asynchronous Primitives *(post-layout)*
  - 8-Terminal Asynchronous Network *(pre-layout with wire estimates,*
    -- interconnection of laid-out router primitives)
  - 8-Terminal GALS Network
  - XMT Architecture Co-Simulation on Parallel Kernels

Tool Flow

- Implemented in IBM 90nm technology
  - Placed and routed with Cadence SOC Encounter
  - Simulated as gate-level Verilog with extracted delays
- Standard Cell Methodology
  - ARM 90nm Standard Cells (IBM CMOS9SF)
- Exception: Mutual Exclusion Element
  - Designed using transistor models from IBM 90nm PDK
  - Simulated in Cadence Spectre
  - Measured delays to calibrate Verilog behavioral model

Routing Primitive Comparison: Area and Power

	Area *(μm^2)*	Energy/ Packet *(pJ)*	Leakage Power *(μW)*	Idle Power *(μW)*
Asynchronous	358.4	0.37	0.56	0.6
Synchronous	988.6	2.06	1.82	225.6

- Area:
  - 64% less area: result of lightweight data storage
    - *2 flip-flop registers + extra MUX/DEMUX (sync)* vs. *2 latch registers (async)*
    - *MUX/DEMUX overhead (sync)*
- Energy/Packet (1 flit):
  - 82% less energy per packet
  - Steady-state measurement on random traffic

Routing Primitive Comparison: Latency and Throughput

Component Type	Latency	Maximum Throughput *(GFPS)*		
	(ps)	Single	Random	Alternating
Asynchronous	546	1.07	1.34	1.70
Synchronous	516	1.93	1.93	**1.93**

- Synchronous: Using Max Clock Rate (1.93 GHz)
- Latency:
  - 546 ps (async) vs. 516 ps (sync)
- Max Throughput (Giga-flits/sec):
  - Single-ported traffic: **55%** of sync max. *(no concurrency)*
  - Random traffic: **70%** of sync max.
  - Alternating traffic: **88%** of sync Max. *(most concurrency)*

⇨ ... expect significant future improvements by inserting small # of FIFO stages

Arbitration Primitive Comparison: Area and Power

Component Type	Area *(μm^2)*	Energy/ Packet *(pJ)*	Leakage Power *(μW)*	Idle Power *(μW)*
Asynchronous	349.3	0.33	0.50	0.5
Synchronous	2240.3	3.53	4.13	388.6

- Area:
  - 84% less area
  - Due to low-overhead data storage
    - *4 flip-flop registers (sync)* vs. *1 latch register (async)*
- Energy/Packet (1 flit):
  - 91% less energy per packet
  - Measured steady-state packets arriving at both input ports

Arbitration Primitive Comparison: Latency and Throughput

Component Type	Latency	Max. Throughput *(GFPS)*	
	(ps)	Single	Both Ports
Asynchronous	489	1.08	2.04
Synchronous	474	2.09	2.09

- Synchronous: Using Max Clock Rate (2.09 GHz)
- Latency:
  - 489 ps *(async)* vs. 474 ps *(sync)*
- Max. Throughput (Giga-flits/sec):
  - Single Port only: **51%** of synchronous max.
  - Traffic at Both Ports: **98%** of synchronous max.

⇨ ... expect significant future improvements by inserting small # of FIFO stages

8-Terminal Network Evaluation

- Head-on-Head Comparison with Sync Network
- Projected Network Layout
  - Pre-layout async network
  - Uses post-layout primitives, treated as hard IP macros, with assigned wire delays
  - Extrapolate wire delays based on ASIC floorplan of Sync MoT
- Experimental Setup
  - Evaluate performance under uniformly random input traffic
  - 32-bit flits

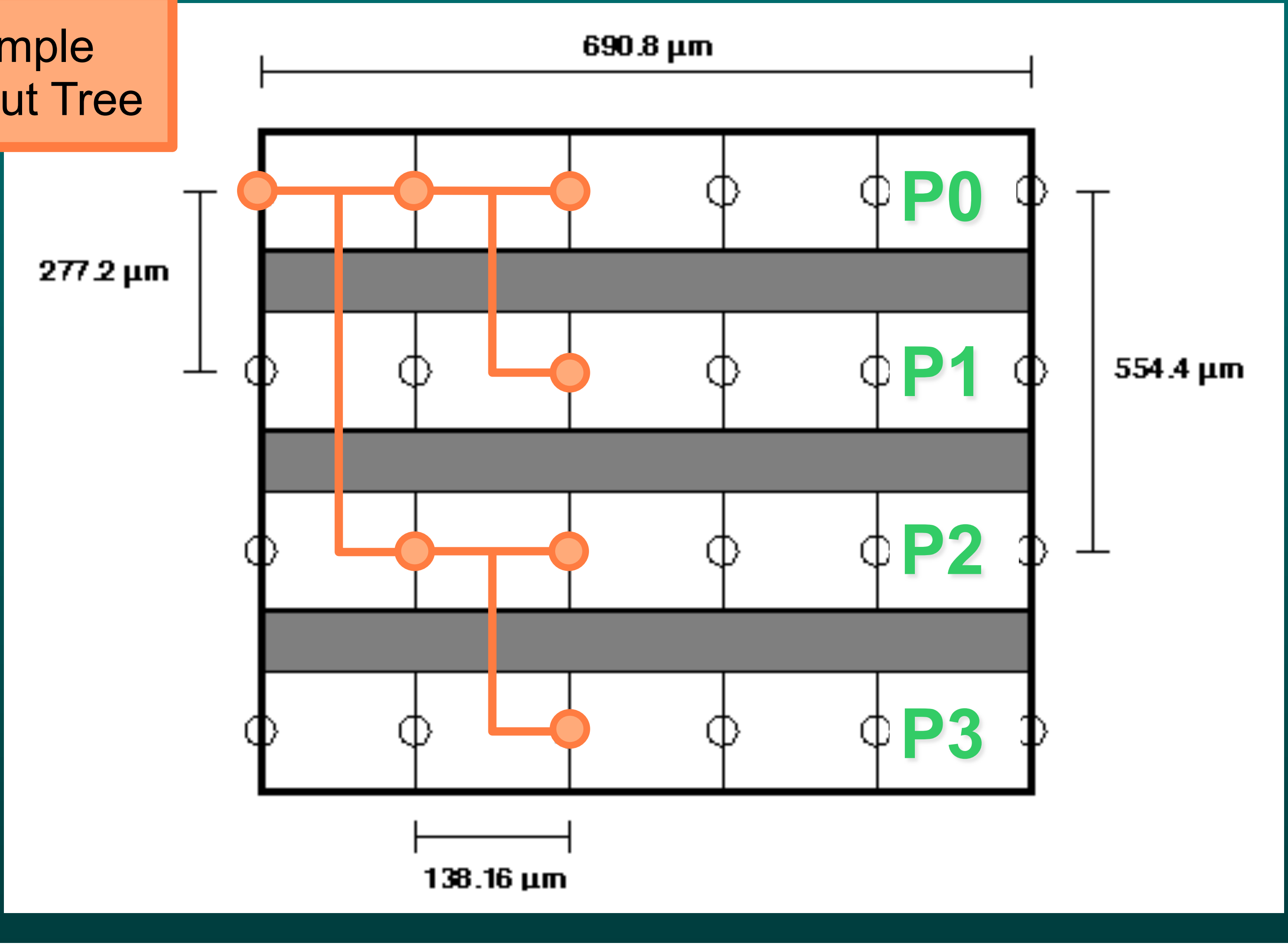
Projected 8-Terminal Network Layout

- Based on Floorplan of Synchronous MoT Test ASIC
  - Designed/fabricated at UMD in March 2007 [10]
- Network divided into 4 partitions (P0,P1,P2,P3)
  - Fan-In Trees exist entirely within one partition
  - Fan-Out Trees distributed among partitions
- Asynchronous Projection Methodology
  - Treat asynchronous primitives are hard IP macros
    - all routing, arbitration primitives have same timing
  - Evenly distribute groups of primitives
  - Assign inter-primitive wire delays based on position
    - delays on wires assigned based on technology specifications

[10] A.O. Balkan, M.N. Horak, G. Qu, U. Vishkin. “*Layout-accurate design and implementation of a high-throughput interconnection network for single-chip parallel processing*”, Hot Interconnects, August 2007

Projected 8-Terminal Network Layout

Example Fan-Out Tree



Current CAD Tool Flows: Sync vs. Async

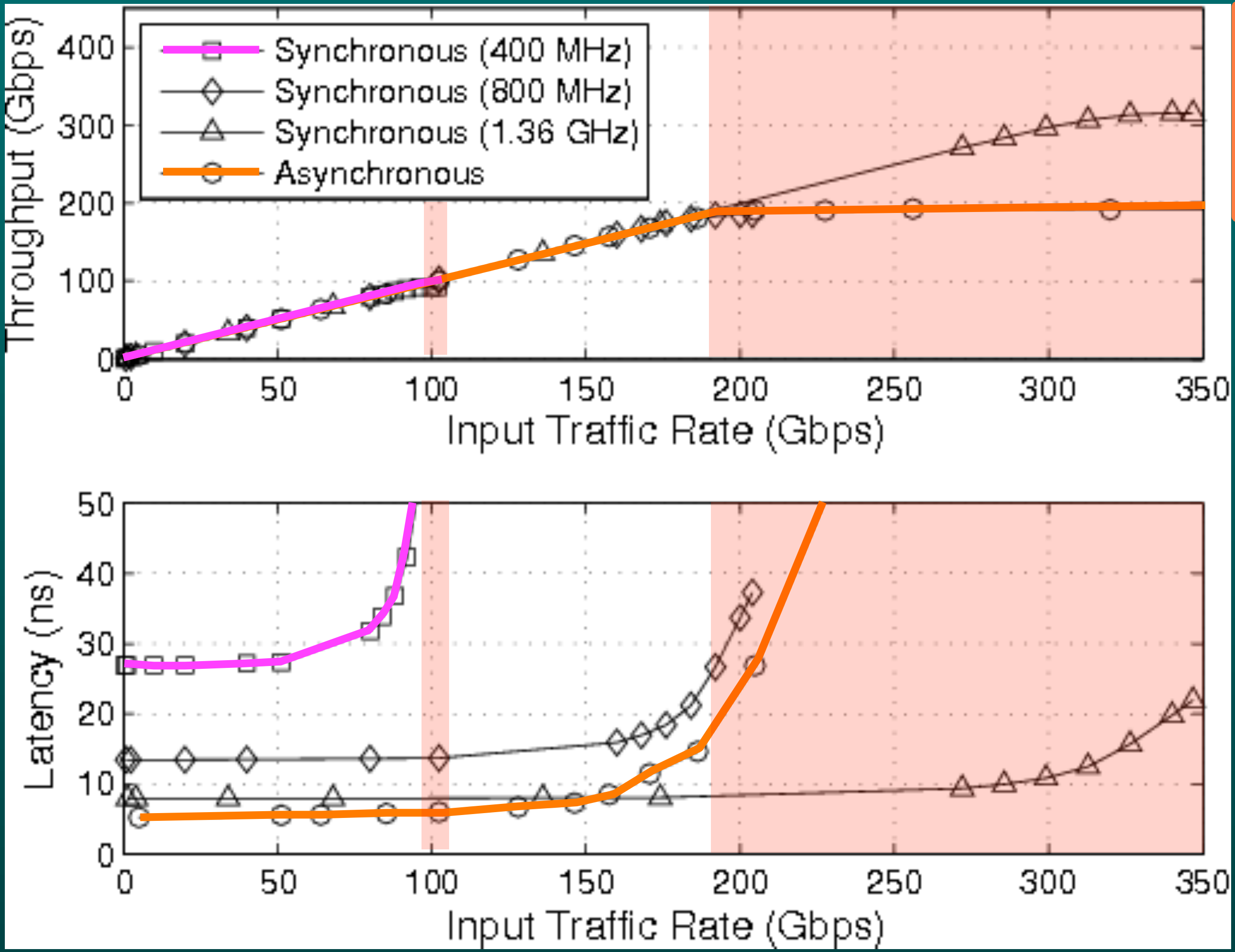
- Synchronous Synthesis:
  - Automatic place/route optimizations
  - Includes cell resizing / repeater insertion
- Asynchronous Synthesis:
  - Limited optimization: hard macros + regular manual placement
  - No cell resizing / repeater insertion

⇨ … much potential for future performance improvement

- Currently Do Not Define Necessary Timing Constraints
  - No automatic path-length matching
  - Necessary to enforce bundling constraint

Async Network Performance Comparison: 400 MHz Sync vs. Async

Comparable throughput for entire range of Sync



Sync Max. Input Rate: 102.4 Gbps

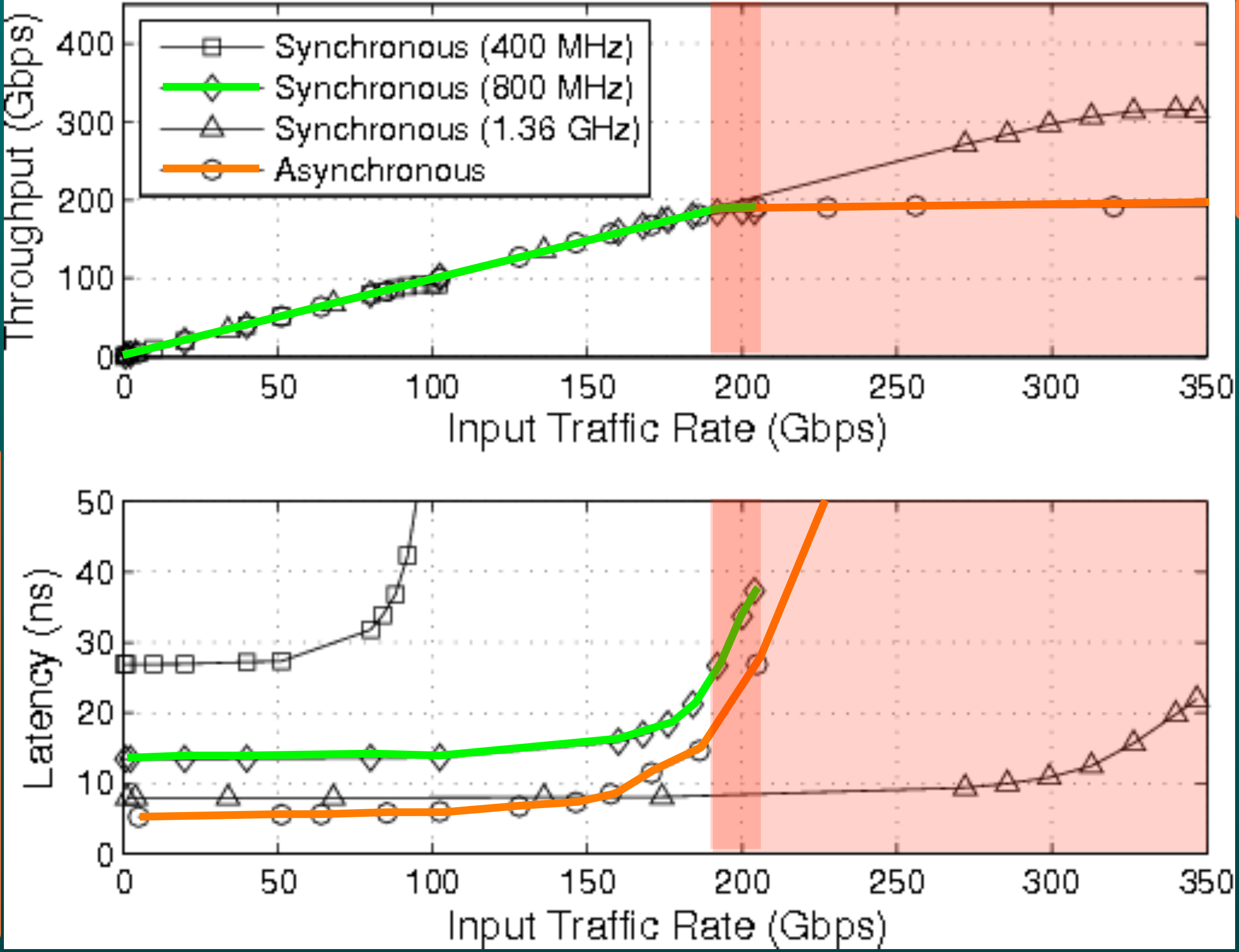
Sync has <u>at least 4.3x higher latency</u> for all Sync input rates

Note: sync max. input rate limited by clock frequency

Async Network Performance Comparison: 800 MHz Sync vs. Async

Comparable throughput for entire range of Sync

Sync has <u>>1.7x higher latency</u> for input rates up to 73% of Sync max. (150 Gbps)

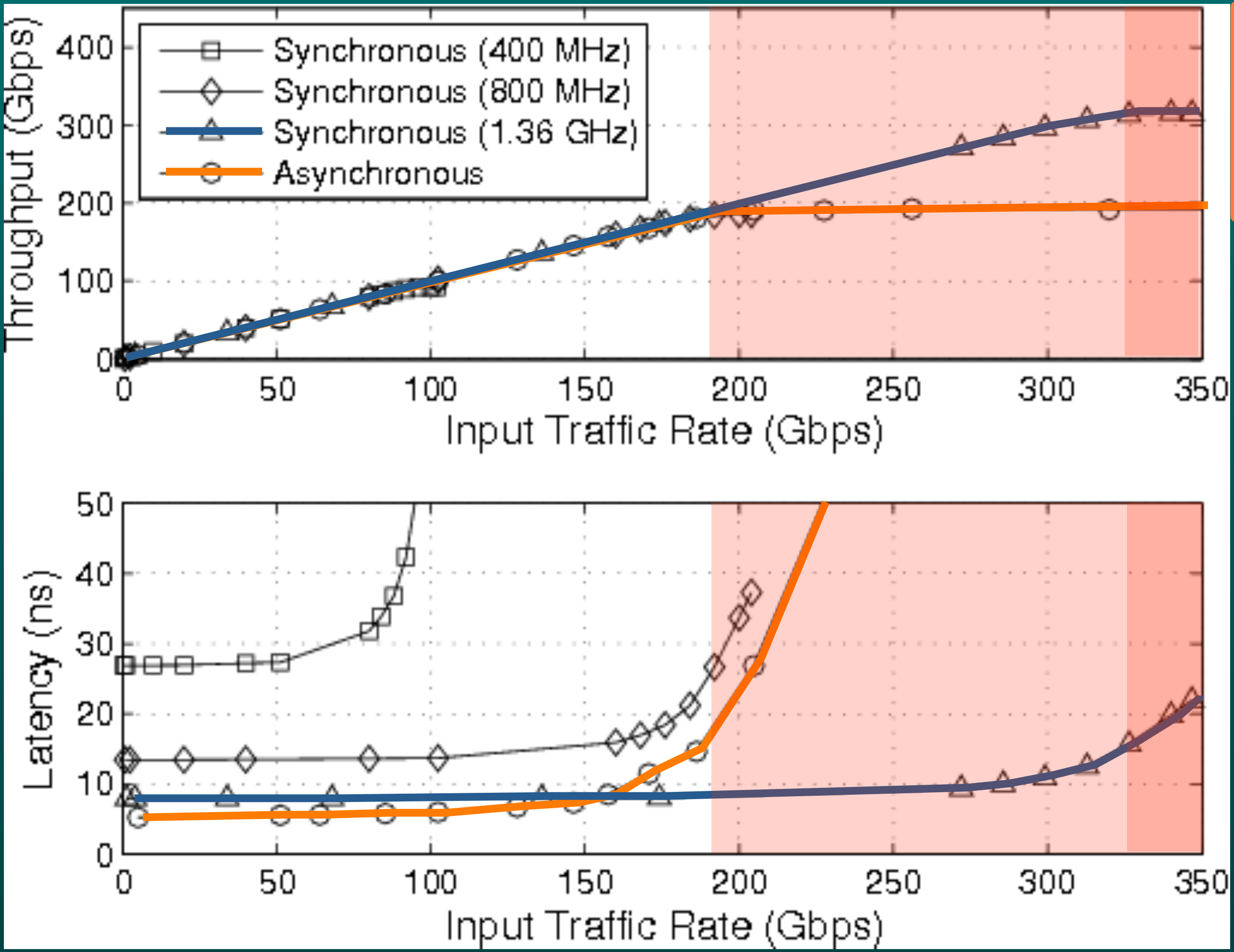


Sync Max. Input Rate: 204.8 Gbps

Note: sync max. input rate limited by clock frequency

Async Network Performance Comparison: 1.36 GHz Sync vs. Async

Comparable throughput for rates up to 55% of Sync max. (190 Gbps)



Sync Max. Input Rate: 348.2 Gbps

Lower latency for input rates up to 43% of Sync max. (150 Gbps)

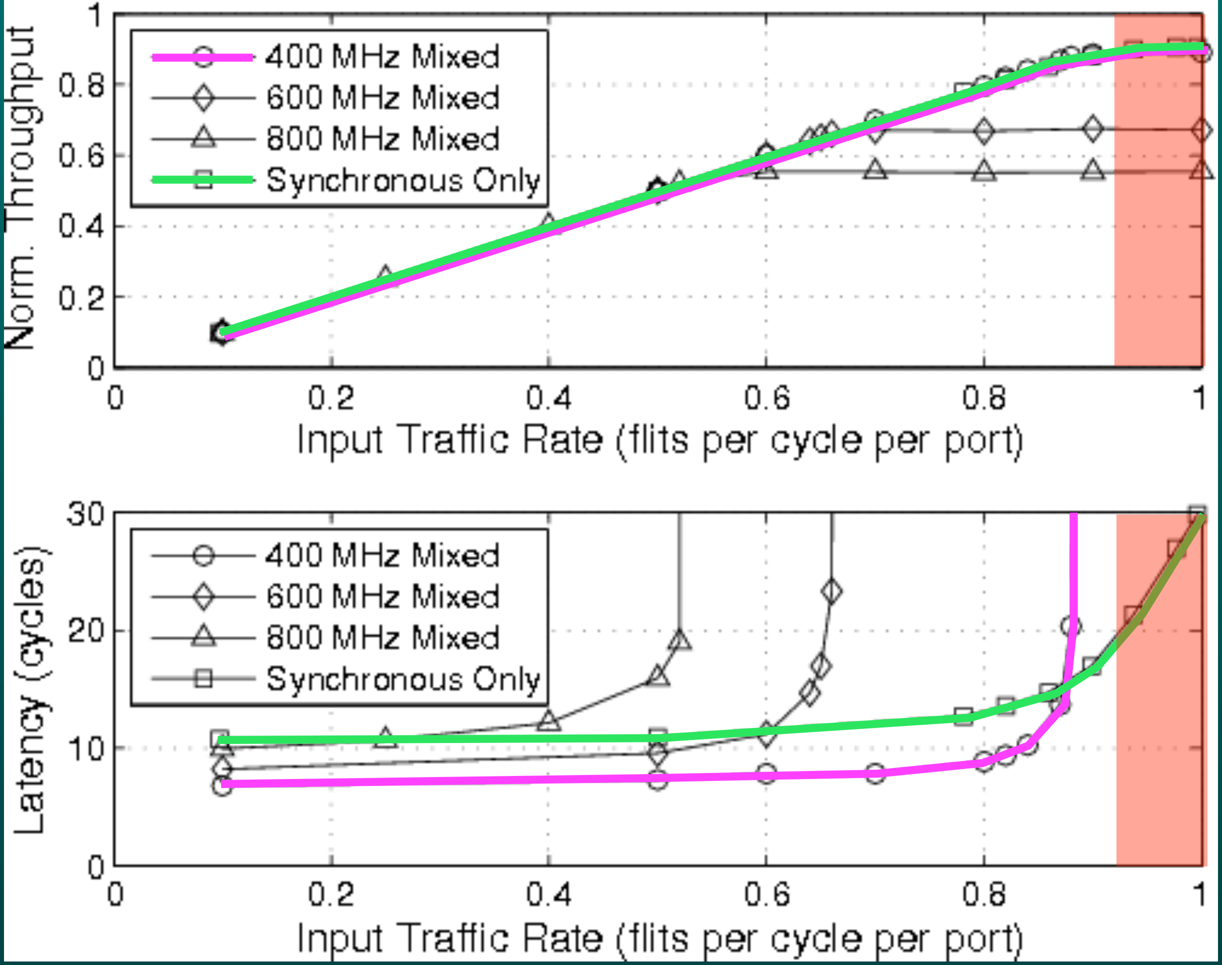
Note: sync max. input rate limited by clock frequency

GALS Network Performance Comparison

- Experimental Setup
  - Create terminals to generate traffic and record measurements
  - Terminals generate uniformly random input traffic
- Results Normalized to Clock Rate
  - Throughput units *(normalized)*: flits per cycle per port
  - Latency units *(normalized)*: # clock cycles
  - Sync network results: always same relative to clock cycles
  - Async network results: vary with clock rate

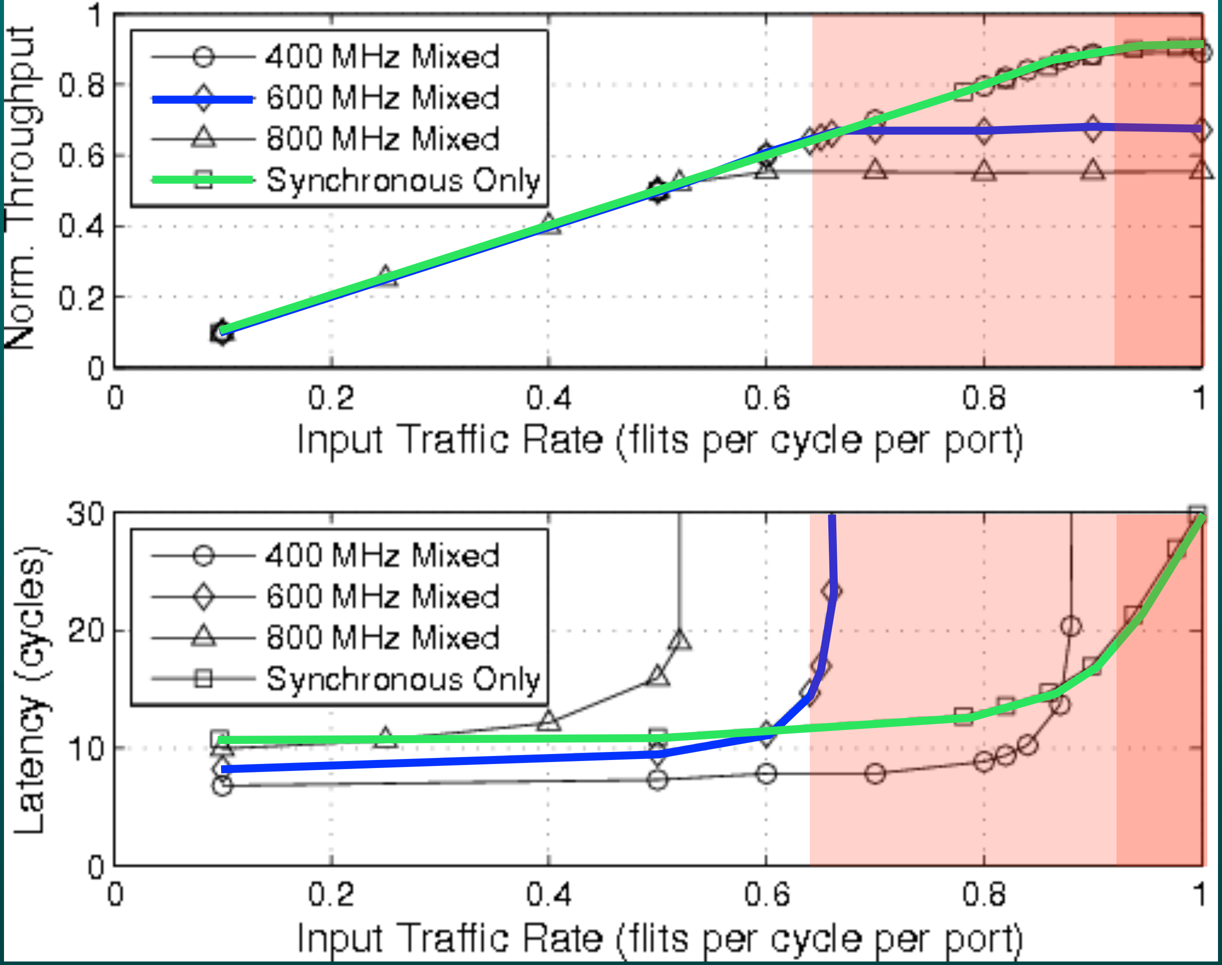
GALS Network Performance Comparison: 400 MHz GALS vs. Sync

<u>Comparable throughput</u> for all traffic rates



Sync has <u>52% higher latency</u> up to 80% input traffic

GALS Network Performance Comparison: 600 MHz GALS vs. Sync

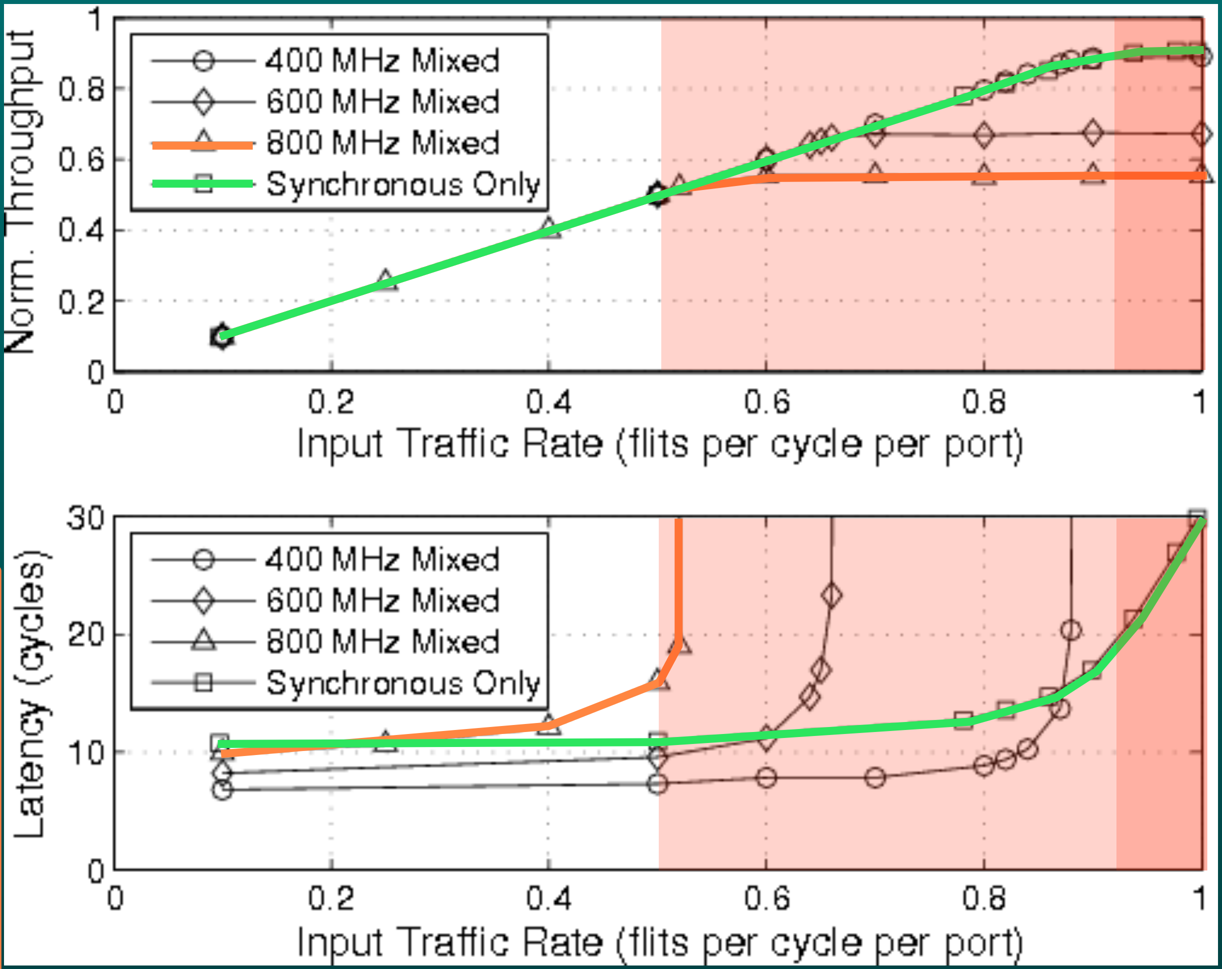


<u>Comparable throughput</u> up to 65% input traffic

<u>Lower latency</u> up to 60% input traffic

GALS Network Performance Comparison: 800 MHz GALS vs. Sync

Comparable throughput up to 52% input traffic



Lower latency up to 29% input traffic, comparable latency up to 40% input traffic

XMT Parallel Kernel Simulations

- Goal: Integrate with Synchronous XMT Parallel Architecture
  - XMT Verilog RTL description with GALS network

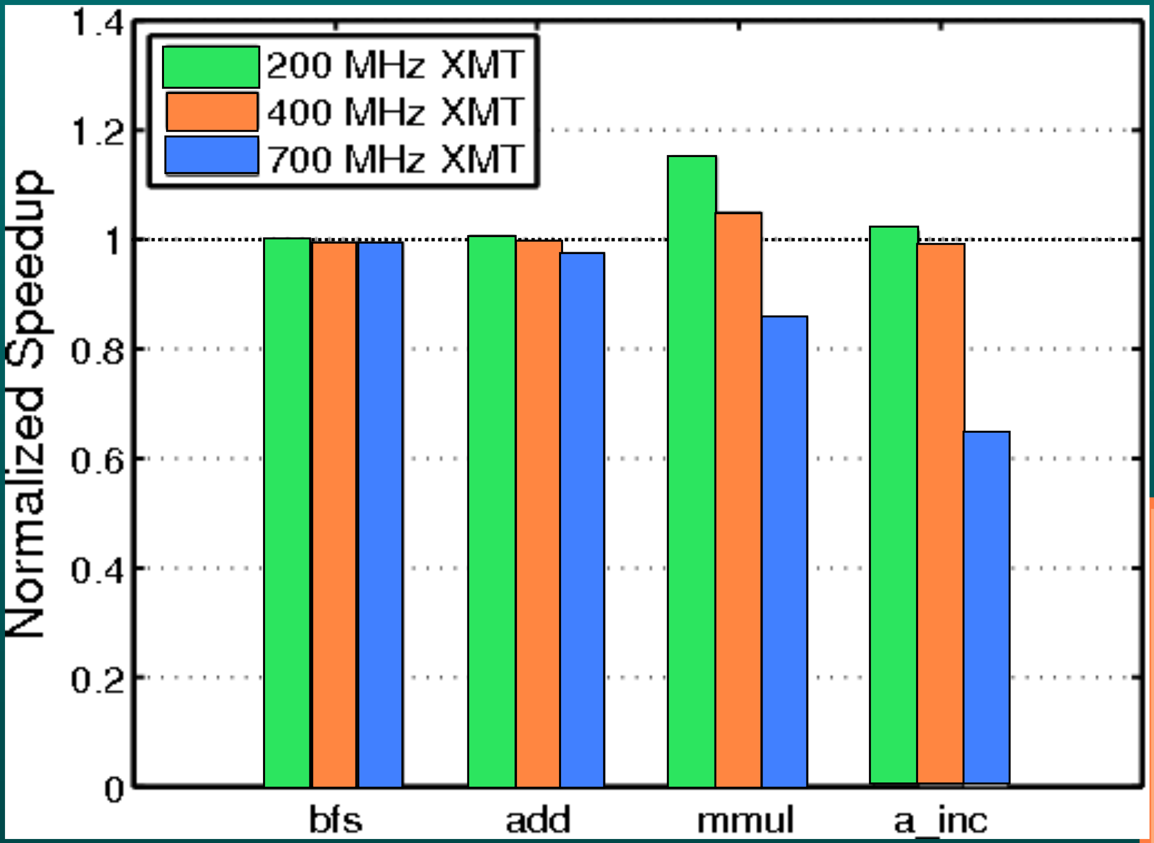
- XMT Parallel Kernels
  - Array Summation (add)
    - Compute sum of 3 million elements in array
  - Matrix Multiplication (mmul)
    - Compute product of two 64 x 64 matrices
  - Breadth-First Search (bfs)
    - Run XMT BFS algorithm with 100,000 vertices and 1 million edges
  - Array Increment (a_inc)
    - Increment all 32k elements of an array

XMT Parallel Kernel Simulations

- XMT Processor Configuration
  - 8 Processing Clusters (16 TCUs each) = 128 TCU's total
  - 8 Distributed L1 D-Cache Modules (64KB total)
- Simulate GALS XMT at Different Clock Frequencies
  - 200, 400, 700 MHz
- Compare Speedups Relative to Synchronous XMT
  - Values greater than 1.0 indicate better performance

GALS XMT Performance Comparison

GALS XMT has similar performance for 200, 400 MHz



Only moderate degradation at 700 MHz (a_inc: 37% decrease)

(Graph arranged in order of increasing network utilization)

Conclusions

- New GALS Network for Chip Multiprocessors
  - Low-overhead network for “heterochronous” Interfaces
- Design of Two New Asynchronous Router Cells
  - Routing and arbitration circuits
- Overview of Results
  - Router Primitives
    - 64-84% less area, 82-91% less energy/packet
    - Latency & throughput (for balanced traffic) = ~2 Gflits/sec
  - System-Level Performance
    - Async network comparison with 800 MHz sync network:
      - Comparable throughput across all input traffic
      - 1.7x lower latency up to 73% max input traffic
    - GALS network comparison with 800 MHz sync network:
      - Comparable throughput up to 52% max input traffic
      - Lower latency up to 29% max input traffic

Future Directions

- Architectual Optimization
  - Insert linear pipeline stages on long wires to improve throughput
- Circuit Optimization
  - Improve designs of routing/arbitration primitives
  - Mixed-timing FIFO optimizations
- Asynchronous Topology Optimization
  - Area improvements using hybrid MoT-Butterfly [Balkan et al., DAC-08]
- Integrate with Synchronous Physical CAD Tool Flow
  - Goal = leverage existing commercial techniques
    - Timing constraint specification and synthesis of unclocked timing paths
    - Build on automated async flow of [Quinton/Greenstreet/Wilton TVLSI '08]
    - Optimized placement, routing, gate resizing and repeater insertion
- Target Alternative Parallel Architectures/Memory Systems

COLUMBIA UNIVERSITY

NOCS 2010

UNIVERSITY OF MARYLAND 18 56

BACKUP SLIDES

Types of Mixed-Timing (GALS) Systems

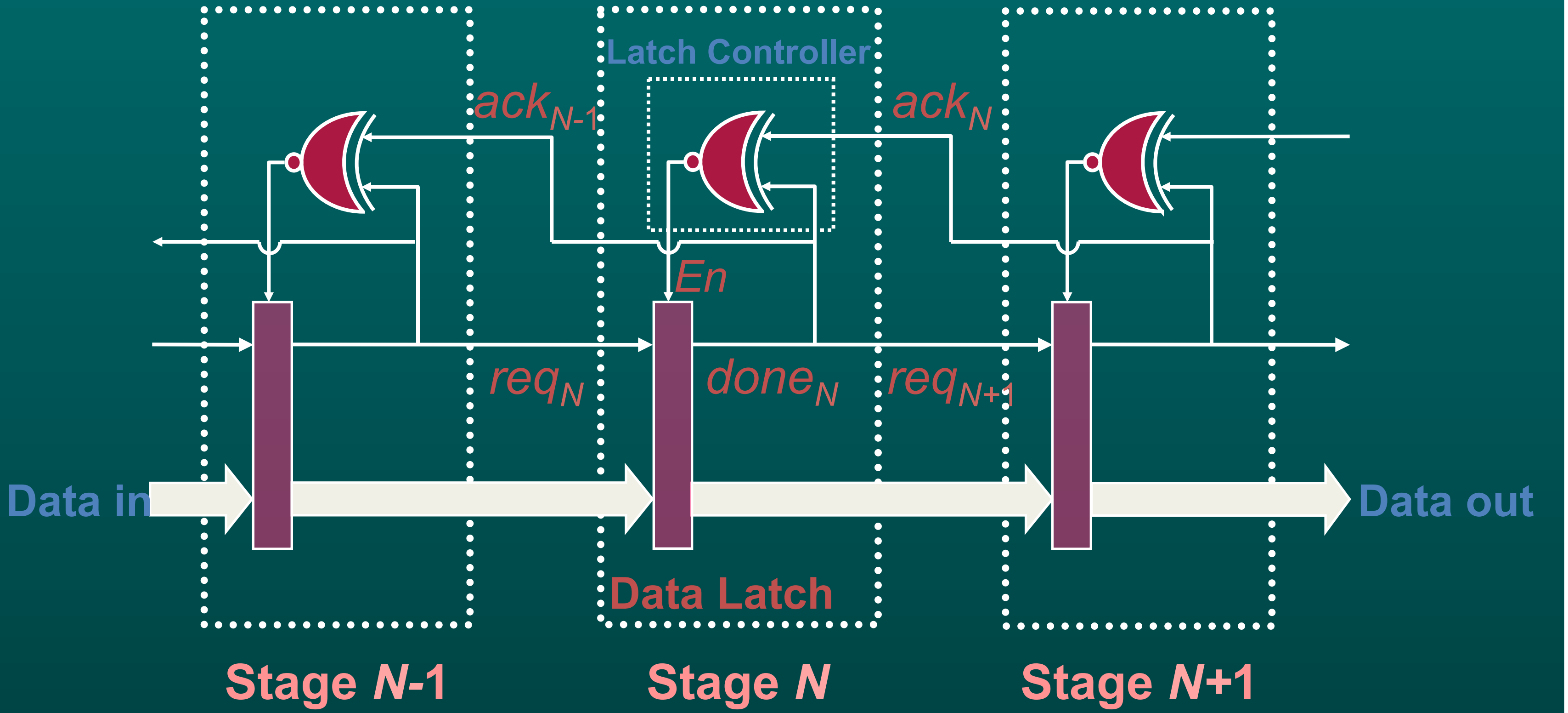
- Pseudochronous
  - Same Frequency, Constant Phase Difference
- Mesochronous
  - Same Frequency, Undefined Phase Difference
- Plesiochronous
  - Nearly exact Frequency and Phase Difference
- Heterochronous
  - Undefined Frequency and Phase Difference

MOUSETRAP Asynchronous Pipelines

- Fast Communication
  - Transition signaling (2-phase) handshaking
- Synchronous-Style Channel Encoding
  - Single-rail bundled data protocol
- Low Latency
  - 1 Transparent D Latch delay for empty stage
- Minimal-Overhead Latch Controller
  - 1 XNOR Gate

MOUSETRAP: A Basic FIFO (no computation)

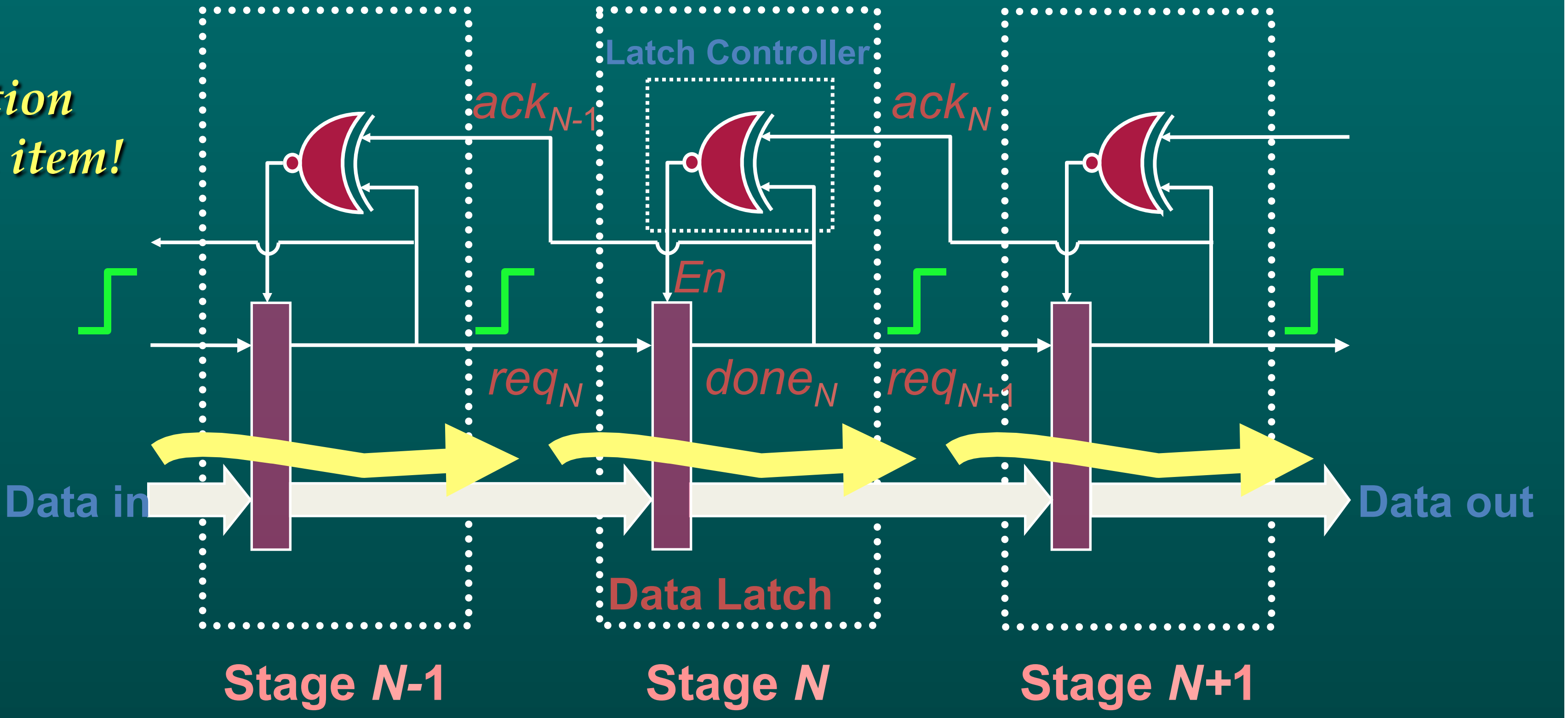
Stages communicate using ***transition-signaling:***



MOUSETRAP: A Basic FIFO (no computation)

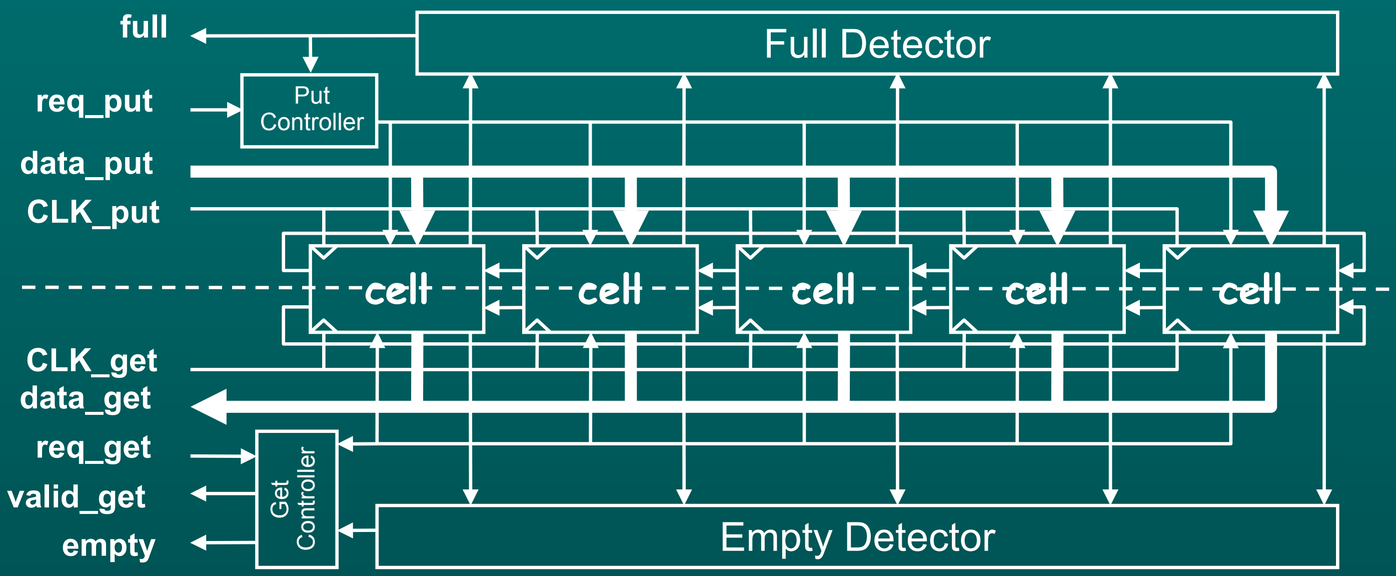
Stages communicate using *transition-signaling:*

1 transition per data item!



One Data Item

Basic Mixed-Clock FIFO (Sync-Sync)



- Sync-Sync FIFO: uses Synchronous *Put* and *Get* Modules
  - Sync-Sync is one of 4 mixed-timing FIFOs
- Mixed Async + Sync FIFO's: modular changes
  - Sync-Async: uses Synchronous *Put* (top) and Asynchronous *Get*
  - Async-Sync: uses Synchronous *Get* (bottom) and Asynchronous *Put*