A Fast, Energy Efficient Z-Comparator

Justin Hensley
Montek Singh
Anselmo Lastra
University of North Carolina, Chapel Hill

Key Features

- **Asynchronous Logic**
- **Reduced energy consumption**
  - 25% lower energy consumption
  - Bit-comparisons computed as needed
- **Increased performance**
  - 1.67 times faster for the average case
  - Comparison available as soon as computed
- **Compute on demand paradigm for graphics**

Overview

- Asynchronous logic tutorial
- Dynamic logic review
- Comparator architecture
- Comparator operation
- Simulation results
- Conclusions and future work

Asynchronous Logic
Advantages

- **Higher performance**
  - "average-case" versus "worst-case" performance
  - Avoids overhead of clock distribution
- **Lower Power**
  - No power wasted by switching clock
  - Inactive components consume negligible power
- **Better electromagnetic compatibility**
  - Smooth radiation spectra (no clock spikes)
  - Less interference with receivers
- **Greater flexibility/modularity**

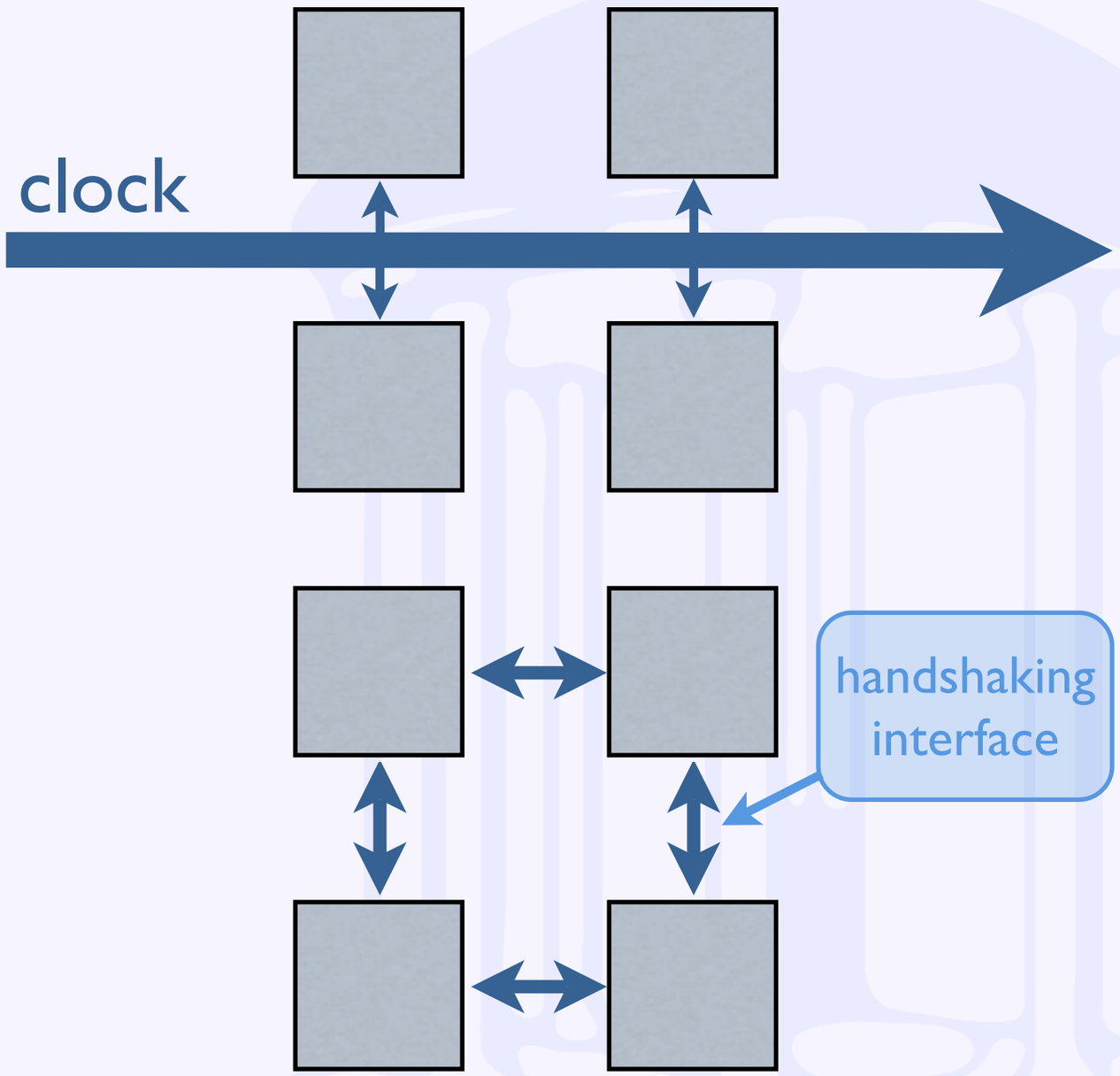
Asynchronous Logic
Challenges

- **Hazards**
  - Glitches can cause serious problems
  - Communication must be hazard free
- **Testability/debugging**
  - No clock - can not "single-step" design
- **Lack of commercial tools**
  - Mostly homegrown tools
  - Use synchronous tools
    - Fitting a square peg in a round hole

Asynchronous Logic
Clocking Methodology

- **Synchronous**
  - *Global* clock
  - Centralized control

- **Asynchronous**
  - Distributed clock
  - Distributed control



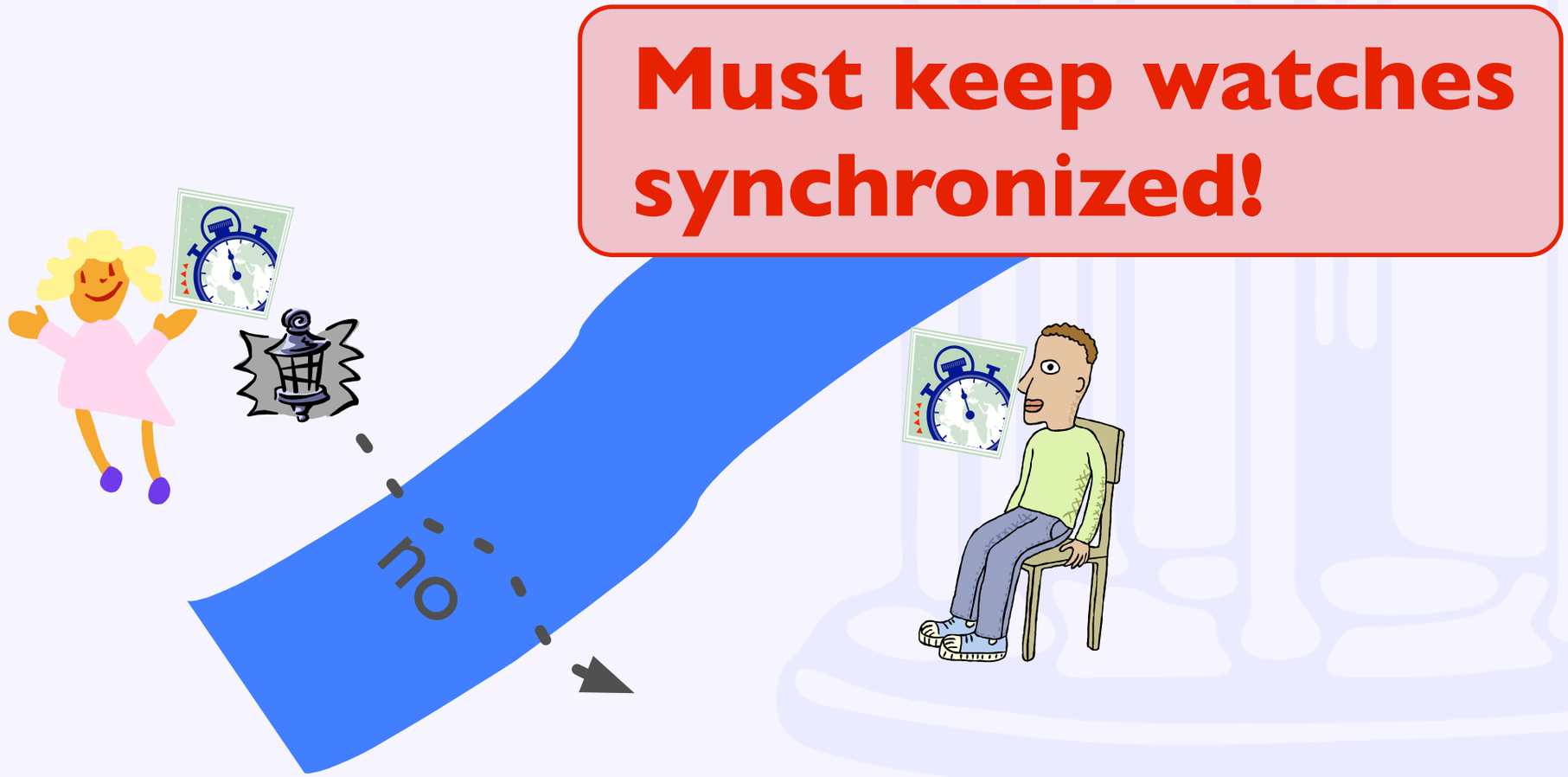
Asynchronous Logic
Communication (1/4)

- Alice and Bob live on opposite sides of a river.

How can Alice send a yes or no message to Bob at midnight? (yelling doesn't count)

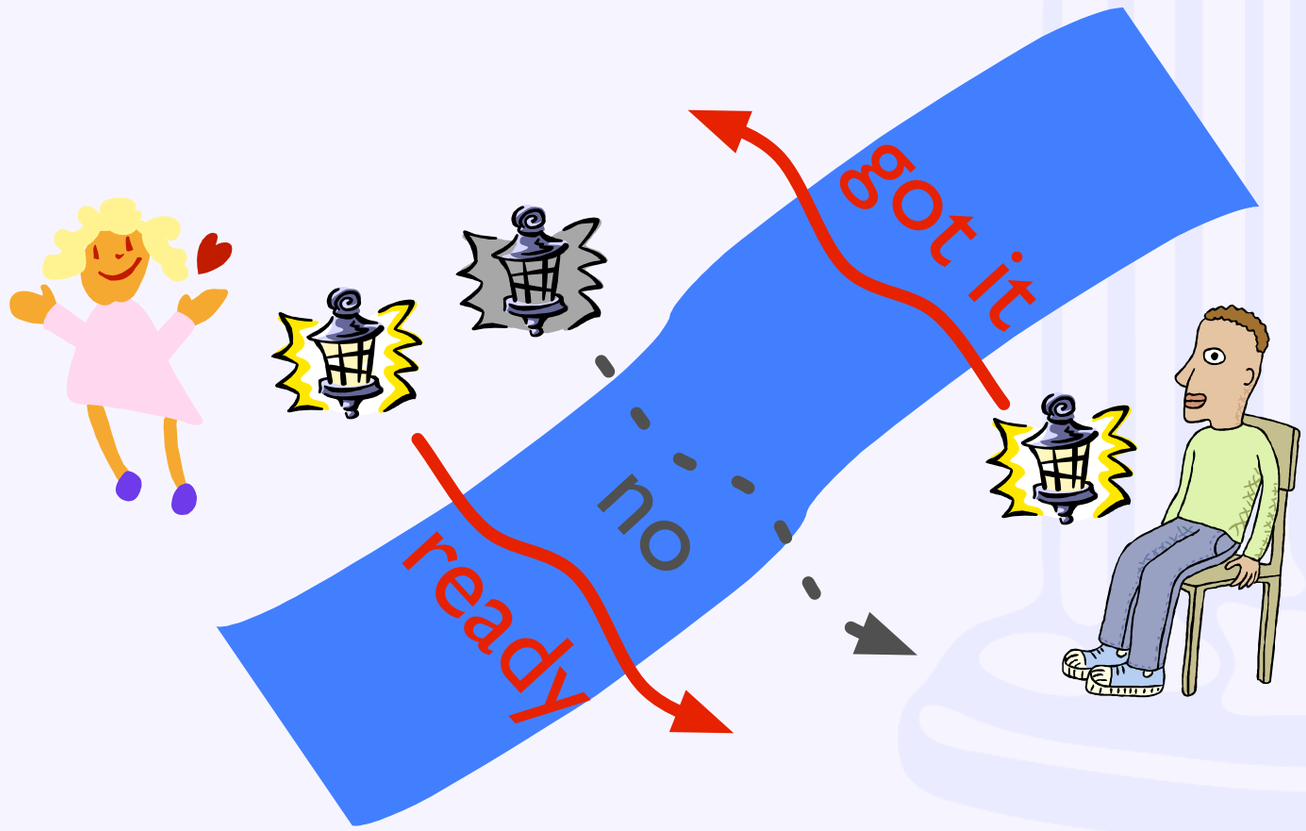
Asynchronous Logic
Communication (2/4)

- Alice and Bob have synchronized watches
- Alice shines a light at midnight if answer is yes → synchronous logic



Asynchronous Logic Communication (3/4)

- **Alice has two lamps**
  - One lamp is used for the actual message
  - Other lamp used to indicate message ready
- **Bob has one lamp**
  - Used to indicate message received



Asynchronous Logic
Bundled Data

- **Single-rail "Bundled Datapath"**
  - Simplest approach

Features:

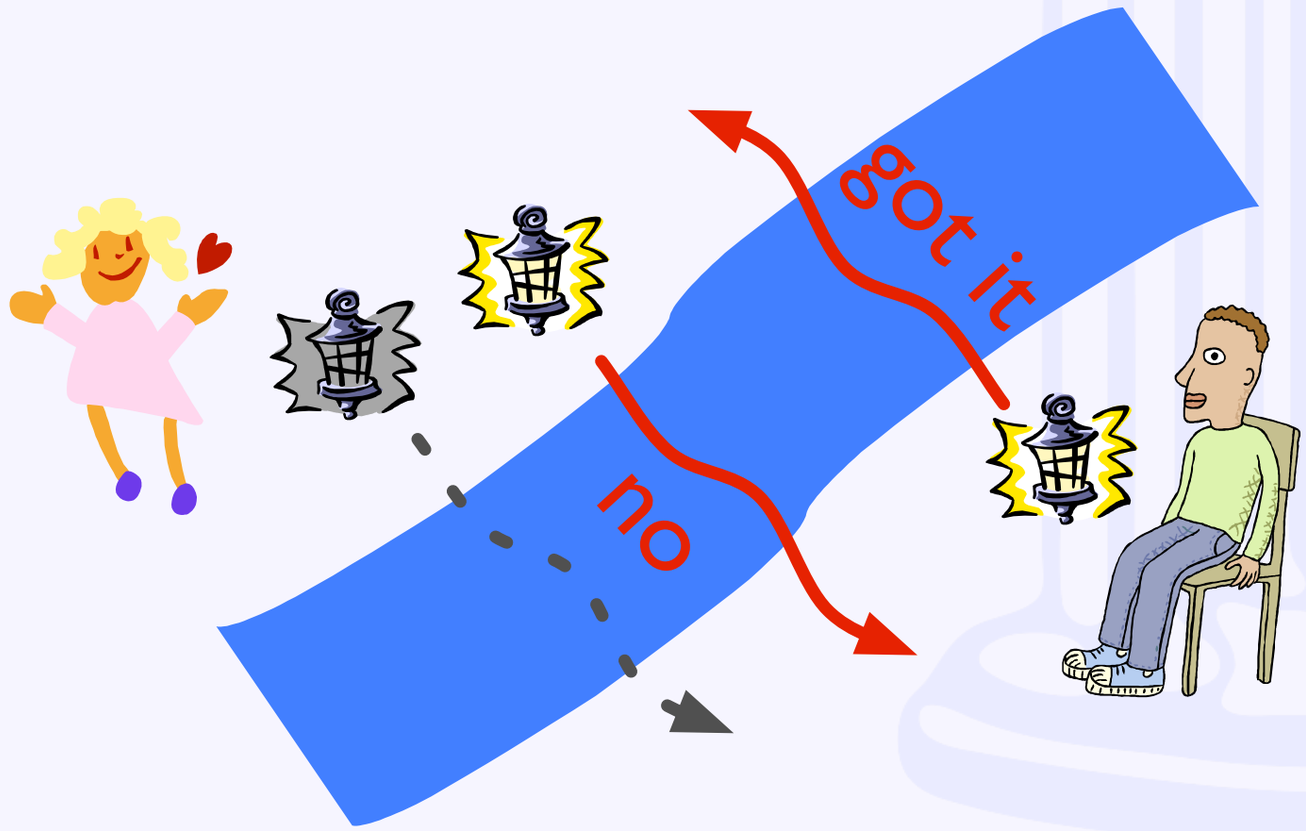
- **Datapath:** 1 wire per bit plus done signal (n+1)
- **Completion:** matched delay produces a ***done*** signal

+Practical, allows the reuse of synchronous components

-Matched delay introduces timing assumption

Asynchronous Logic
Communication (4/4)

- **Alice has two lamps**
  - One lamp to signifies yes
  - One lamp to signifies no
- **Bob has one lamp**
  - Used to indicate message received



Asynchronous Logic
Dual-Rail

- **Dual-rail Datapath**
  - Works well with dynamic logic

Features:

- **Datapath:** 2 wires per bit
- **Completion: via detectors**

code	meaning
0:0	"reset"
0:1	logic 0
1:0	logic 1
1:1	unused

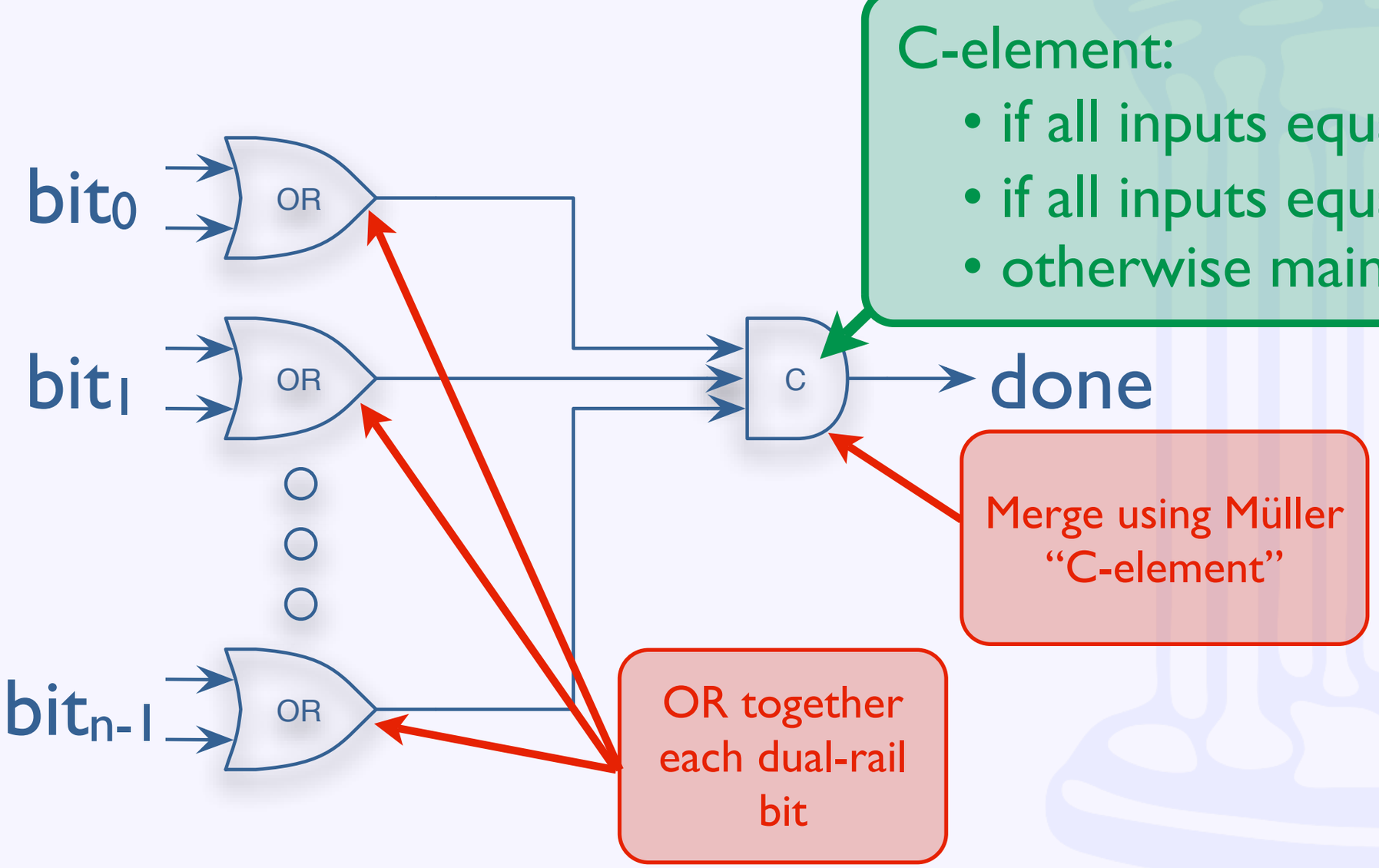
+Robust operation

-Completion detection adds additional gate delays to critical path and increases size

Asynchronous Logic
Completion Sensing

- Combines dual-rail signals
- Indicates when all bits are valid

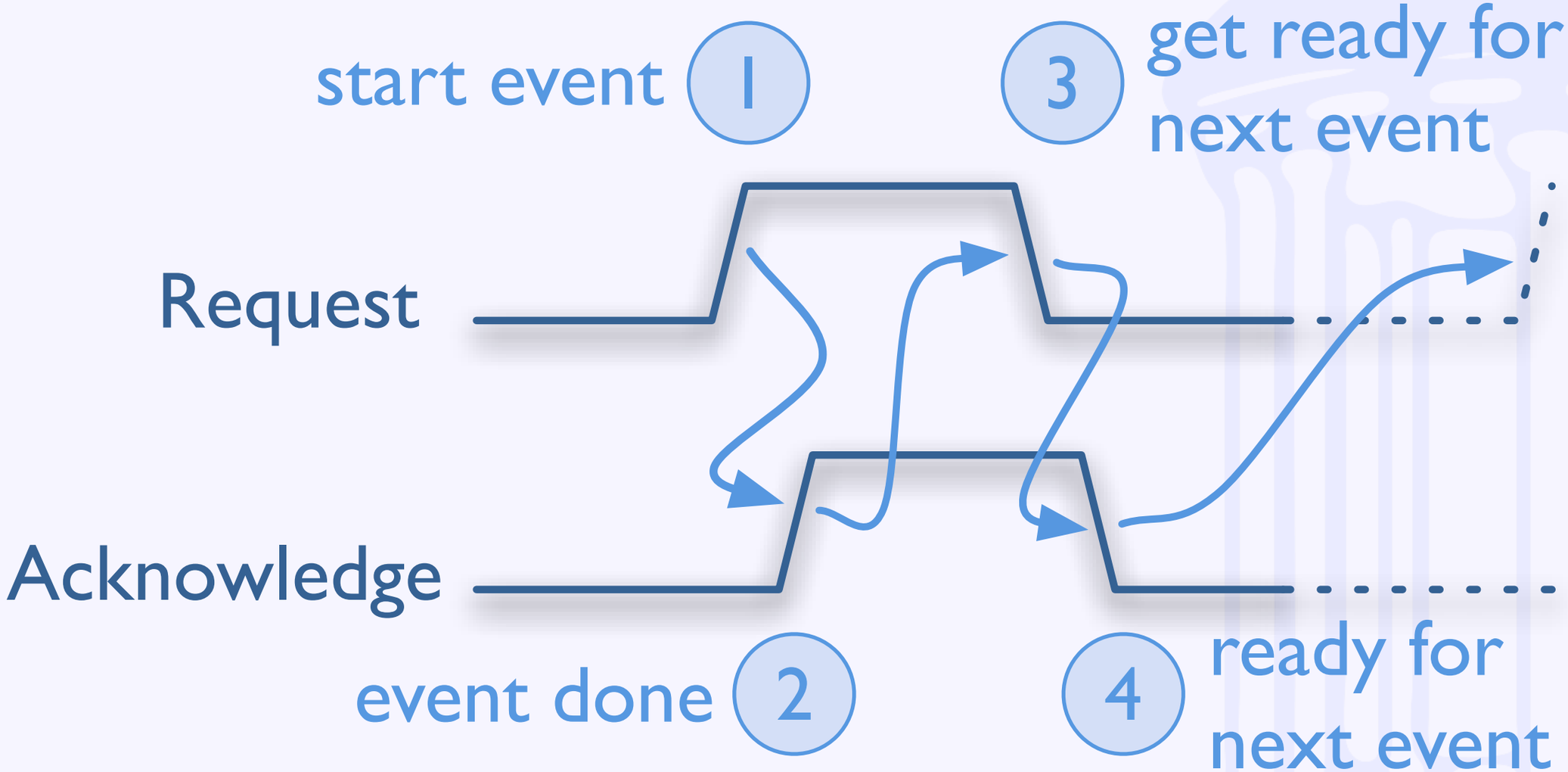
C-element:
- if all inputs equal 1, output → 1
- if all inputs equal 0, output → 0
- otherwise maintain value



Asynchronous Logic
Handshaking Protocols

- **4-phase handshaking**
  - level-sensitive - simpler implementation
  - extra work to return to signals to zero
- **2-phase handshaking**
  - Event-based signaling requires slower logic
  - Need need to return to zero
- **Other protocols used**

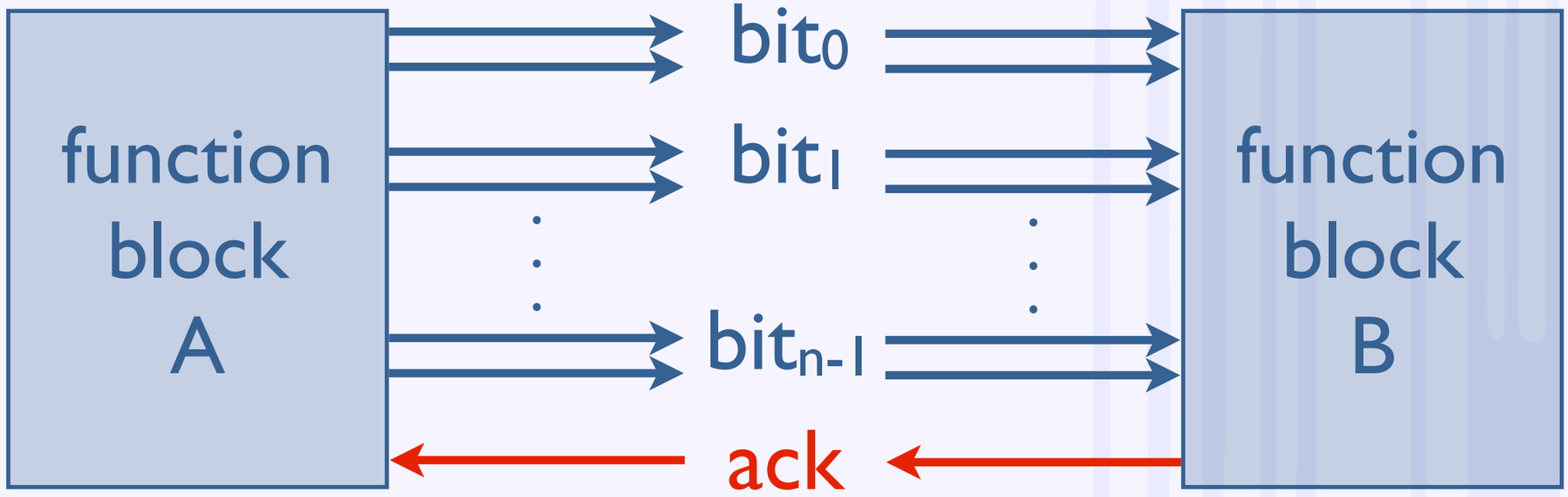
Asynchronous Logic
4-phase protocol



Asynchronous Logic
Data + Handshaking

- **Numerous combinations possible**
  - dual-rail 4-phase, dual-rail 2-phase, bundled-data 4-phase, bundled-data 2-phase, ...

Example: dual-rail 4-phase

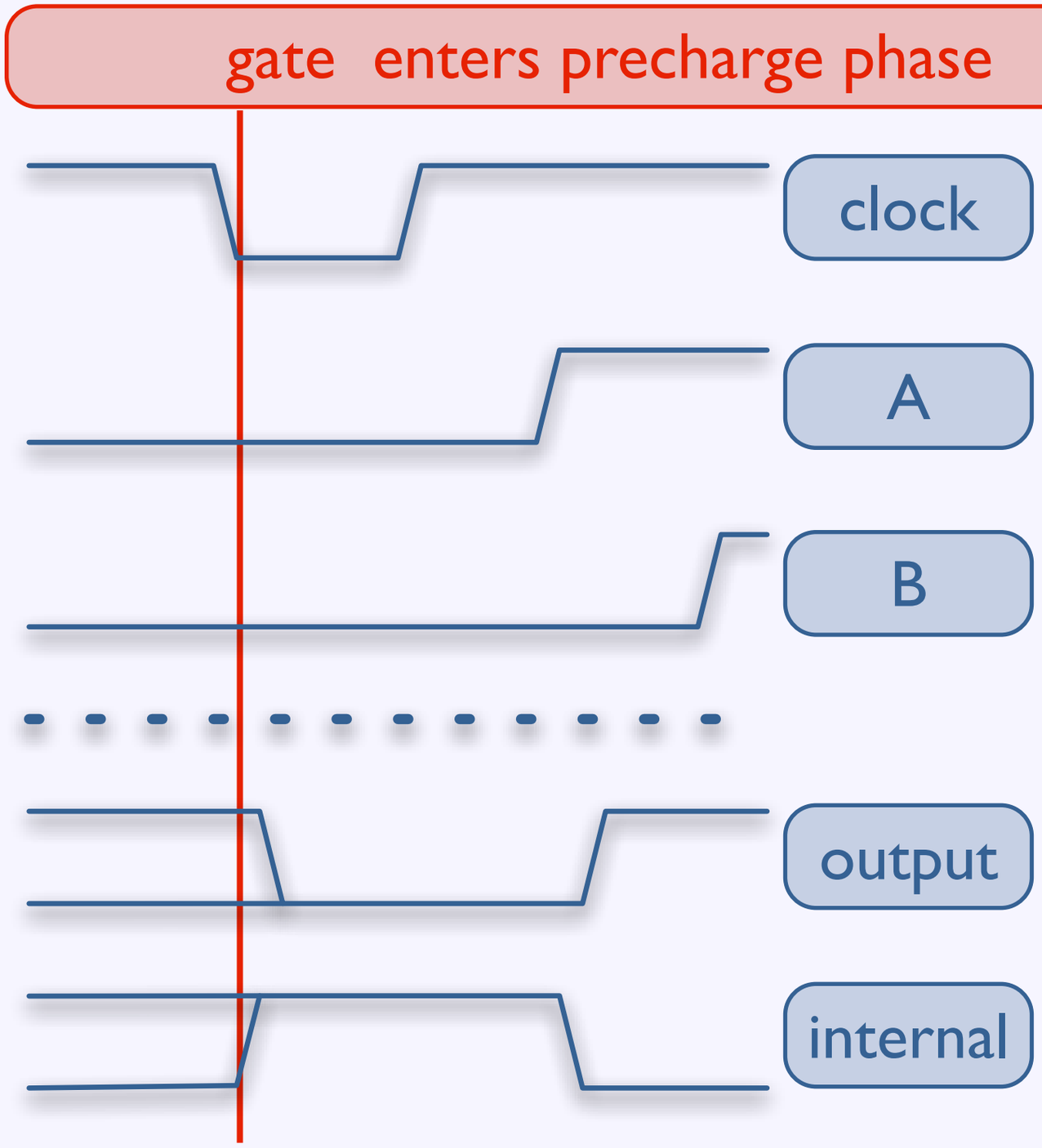


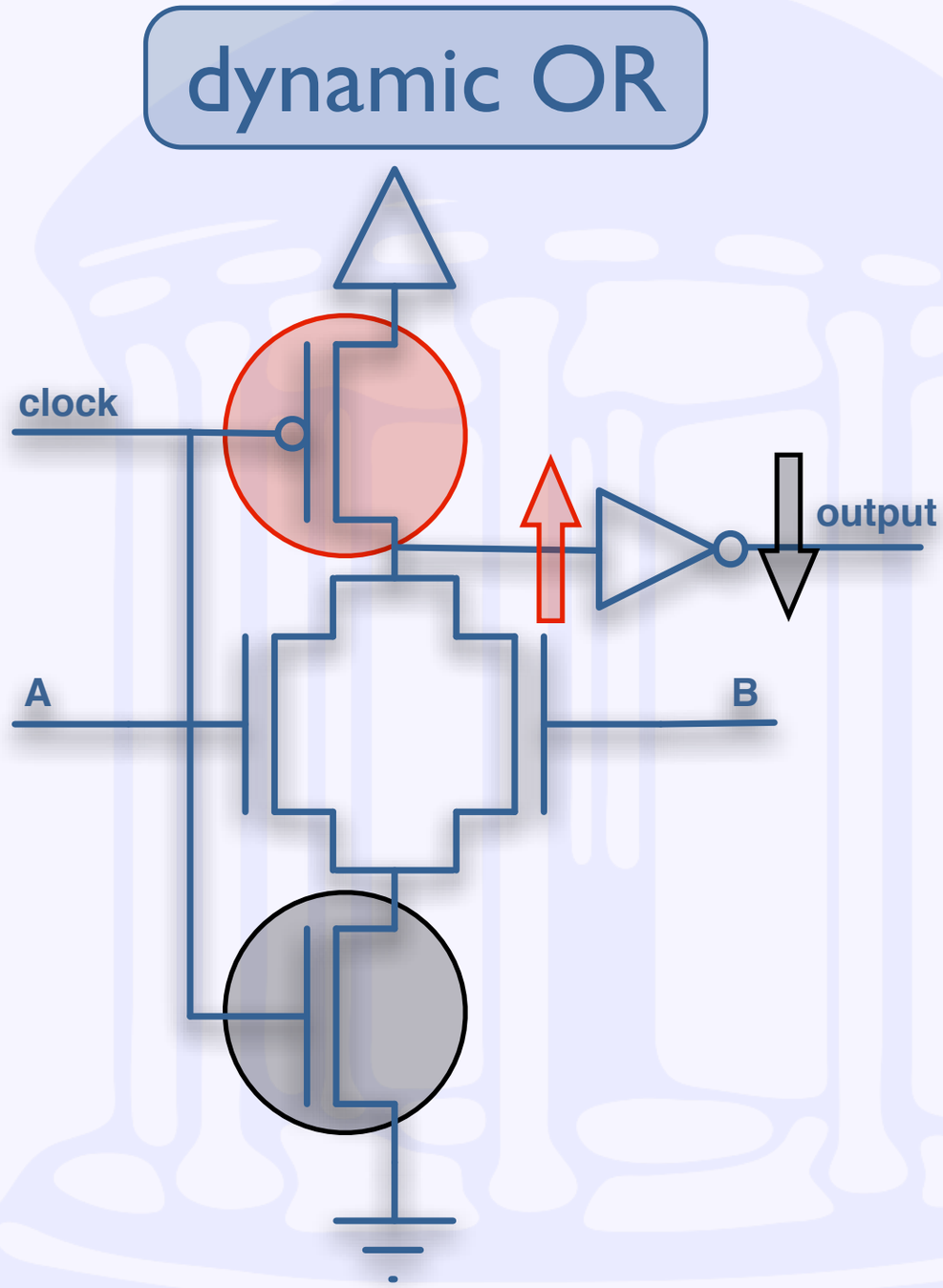
- dual-rail data acts as an ***implicit*** request
- 4-phase cycle between implicit request and acknowledge

Dynamic Logic Review (1/4)

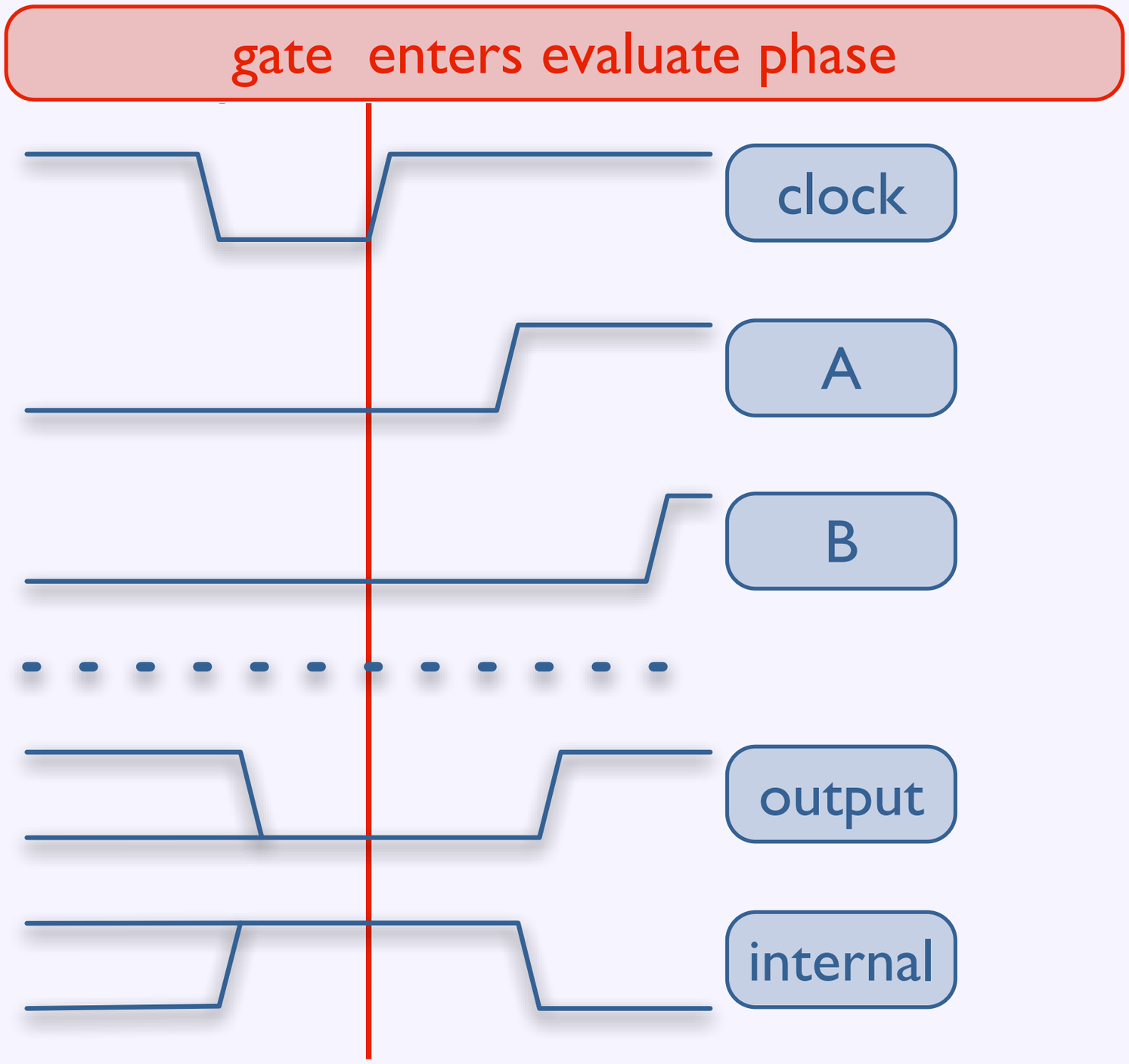
- **Also know as precharge or Domino logic**
- **Does not require complementary NMOS and PMOS networks**
  - PMOS slower due to hole motility
  - Function is implemented only in NMOS
- **Non-inverting logic**
  - Typically need to keep both the signal and its complement

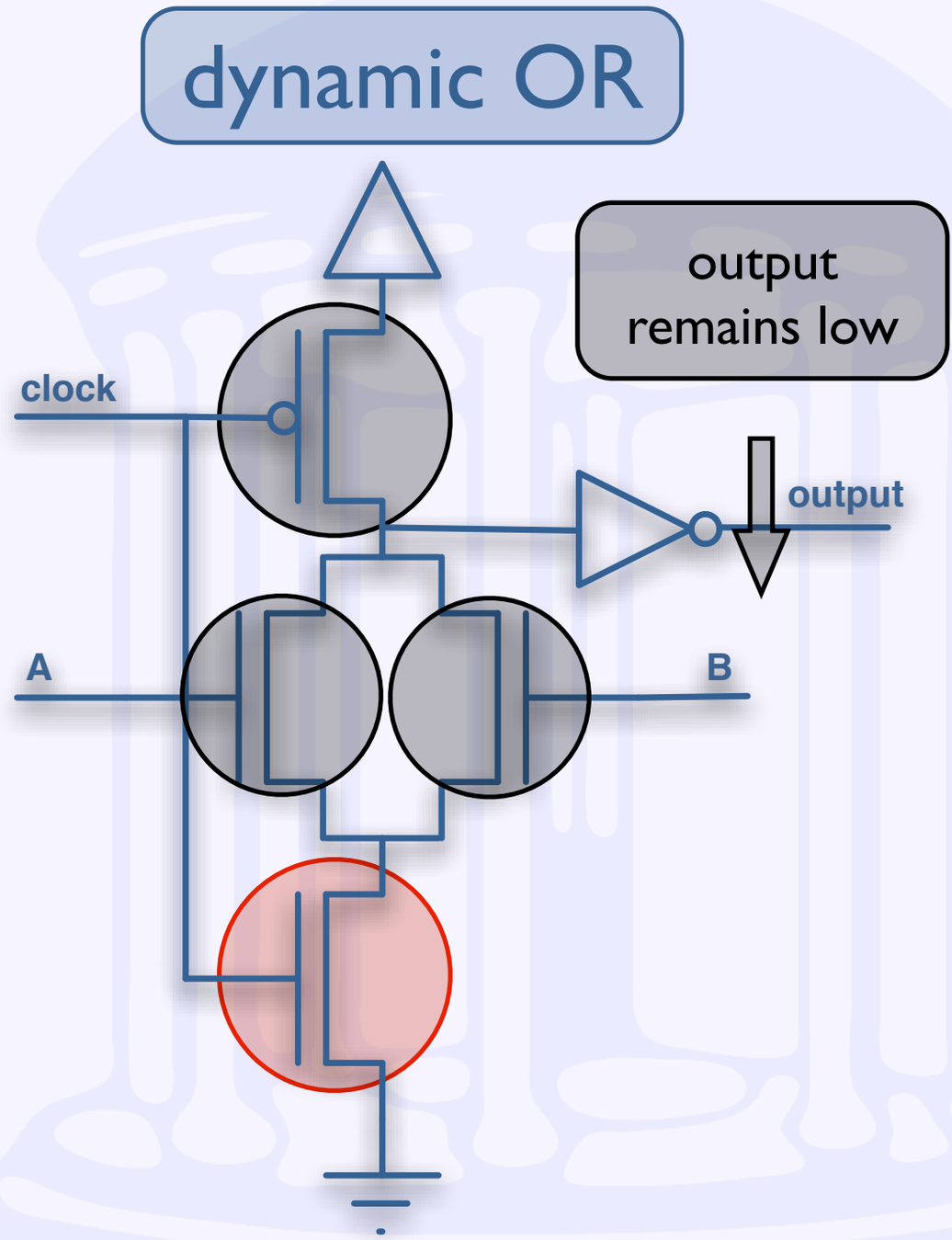
Dynamic Logic Review (2/4)



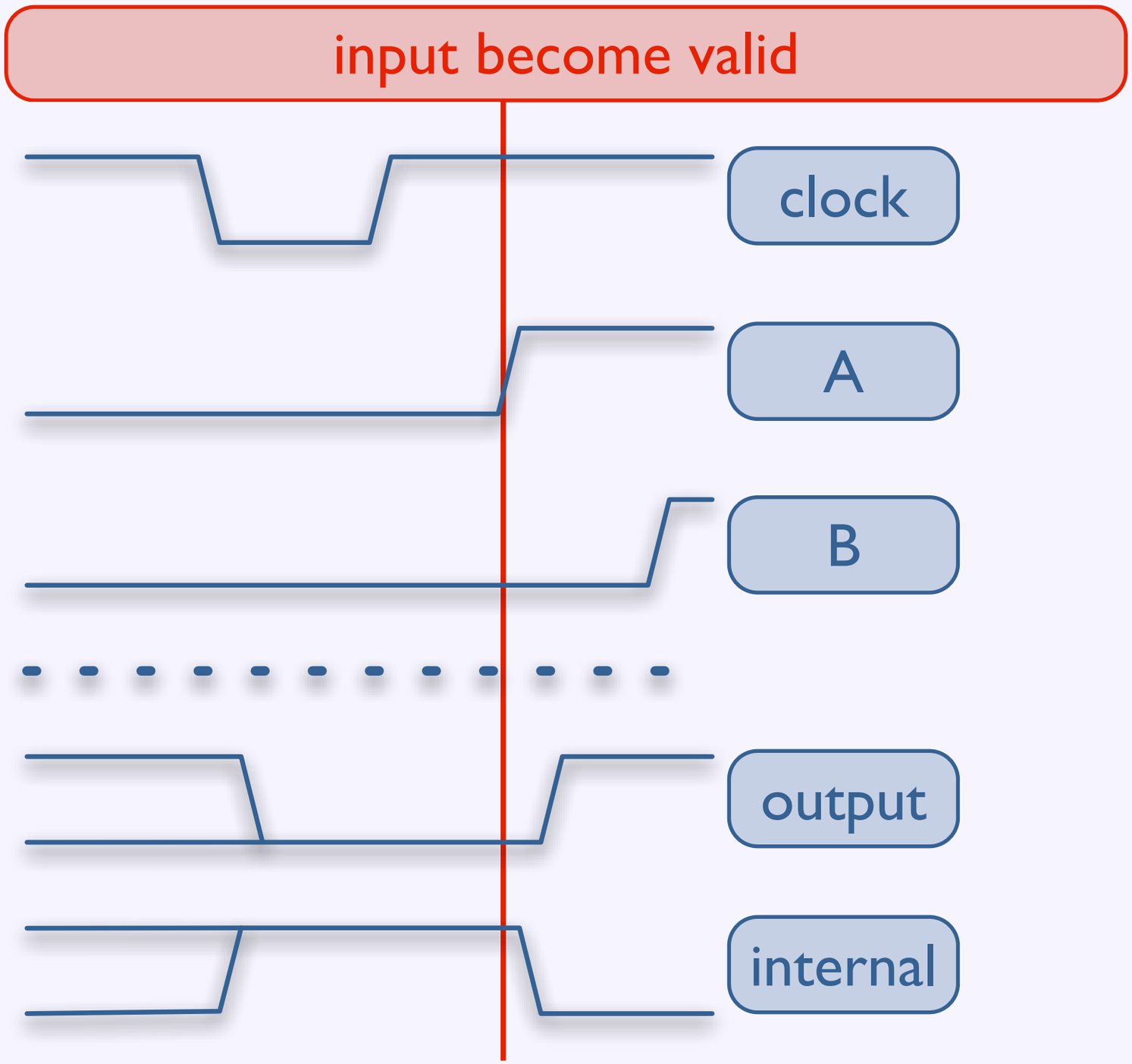


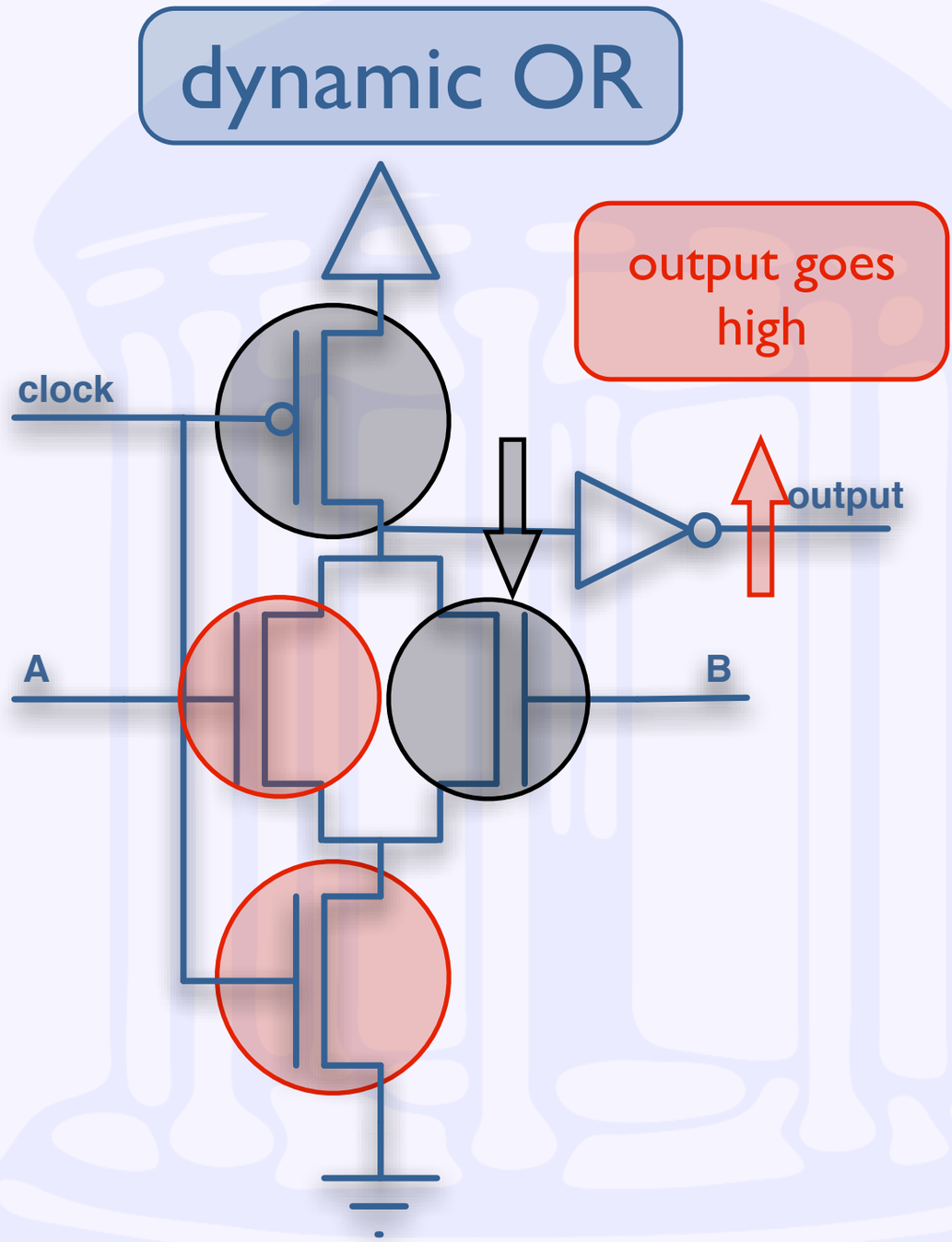
Dynamic Logic Review (3/4)



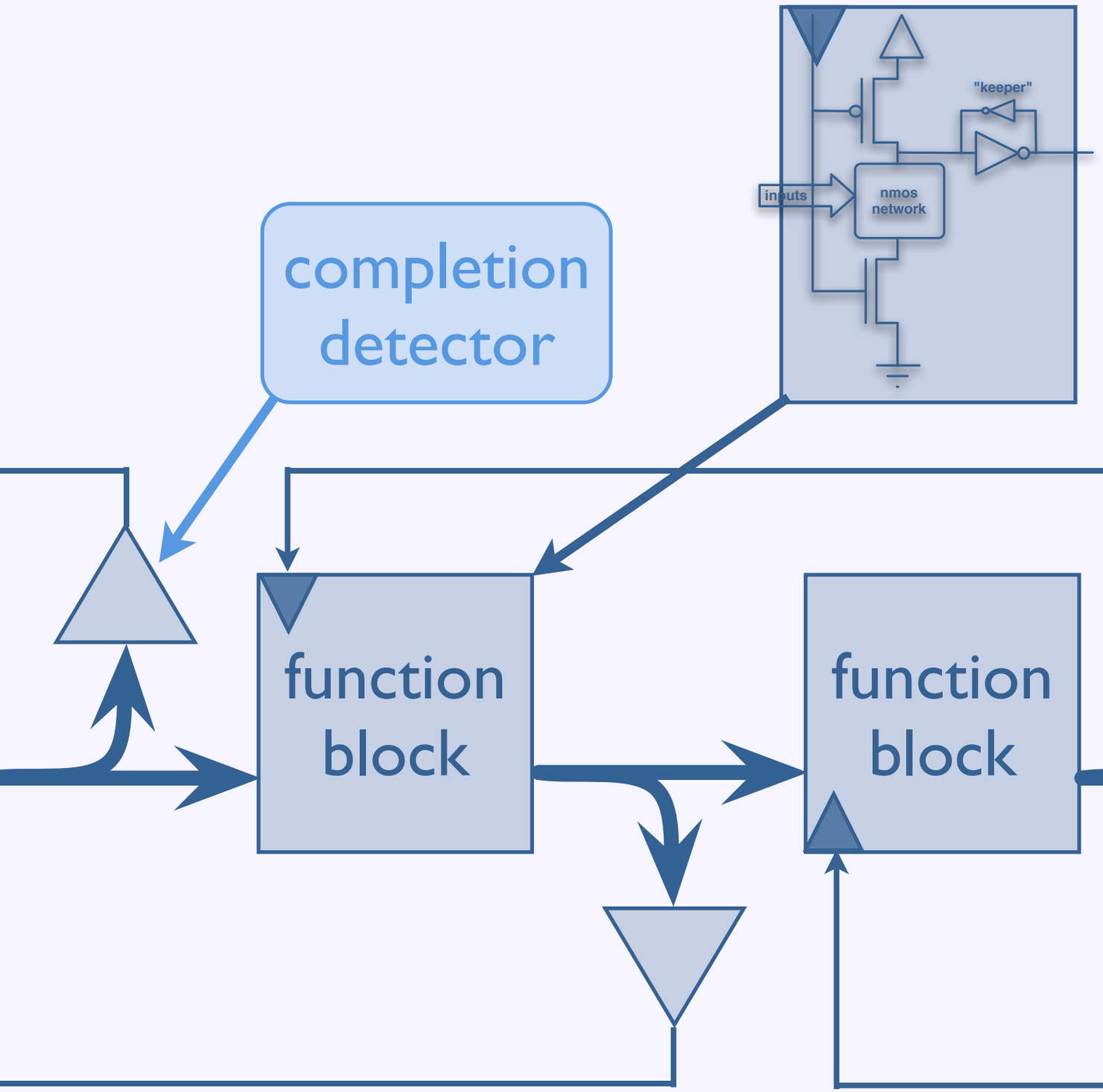


Dynamic Logic Review (4/4)





Putting it Together - PS0



completion detector

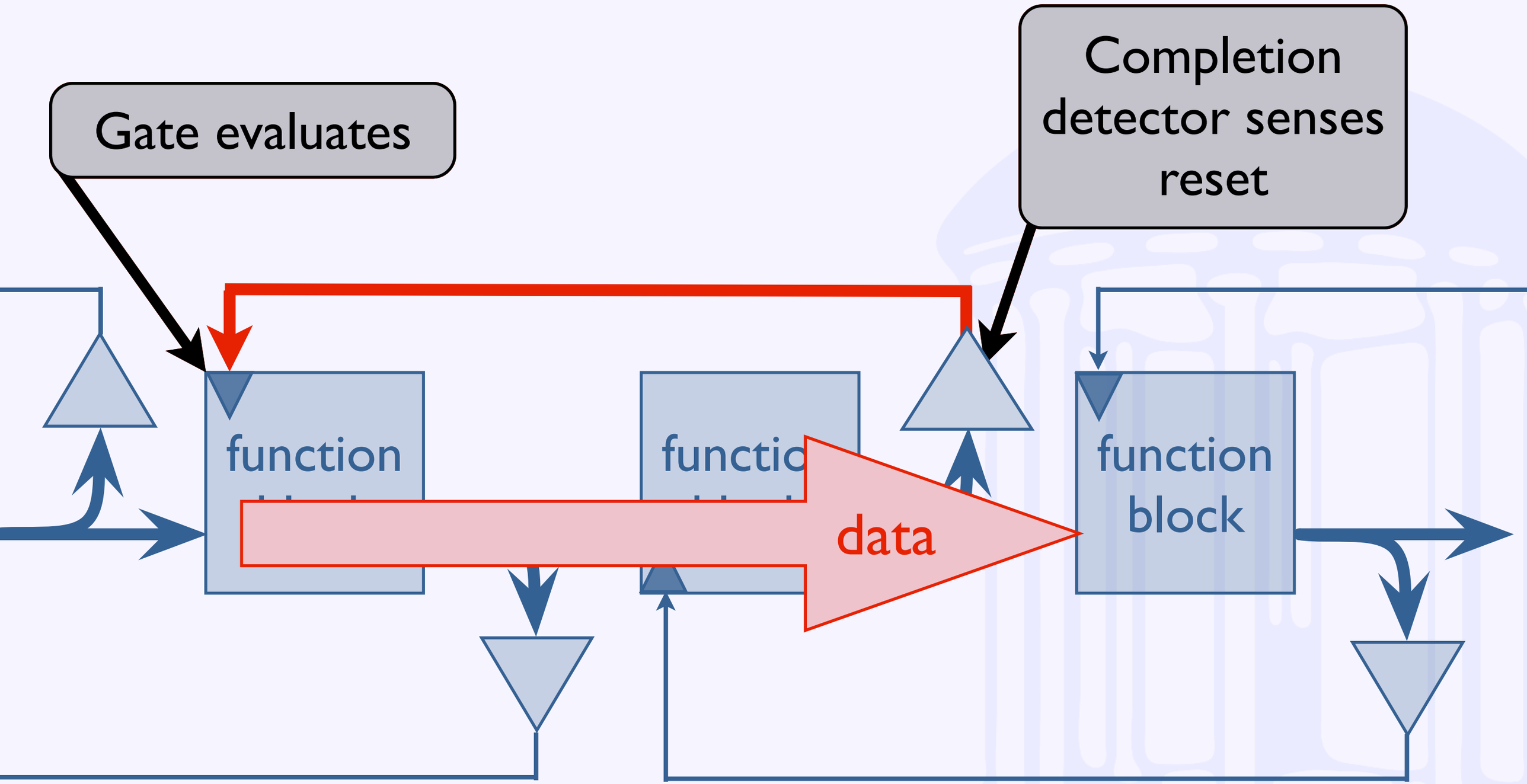
dual-rail datapath

function block

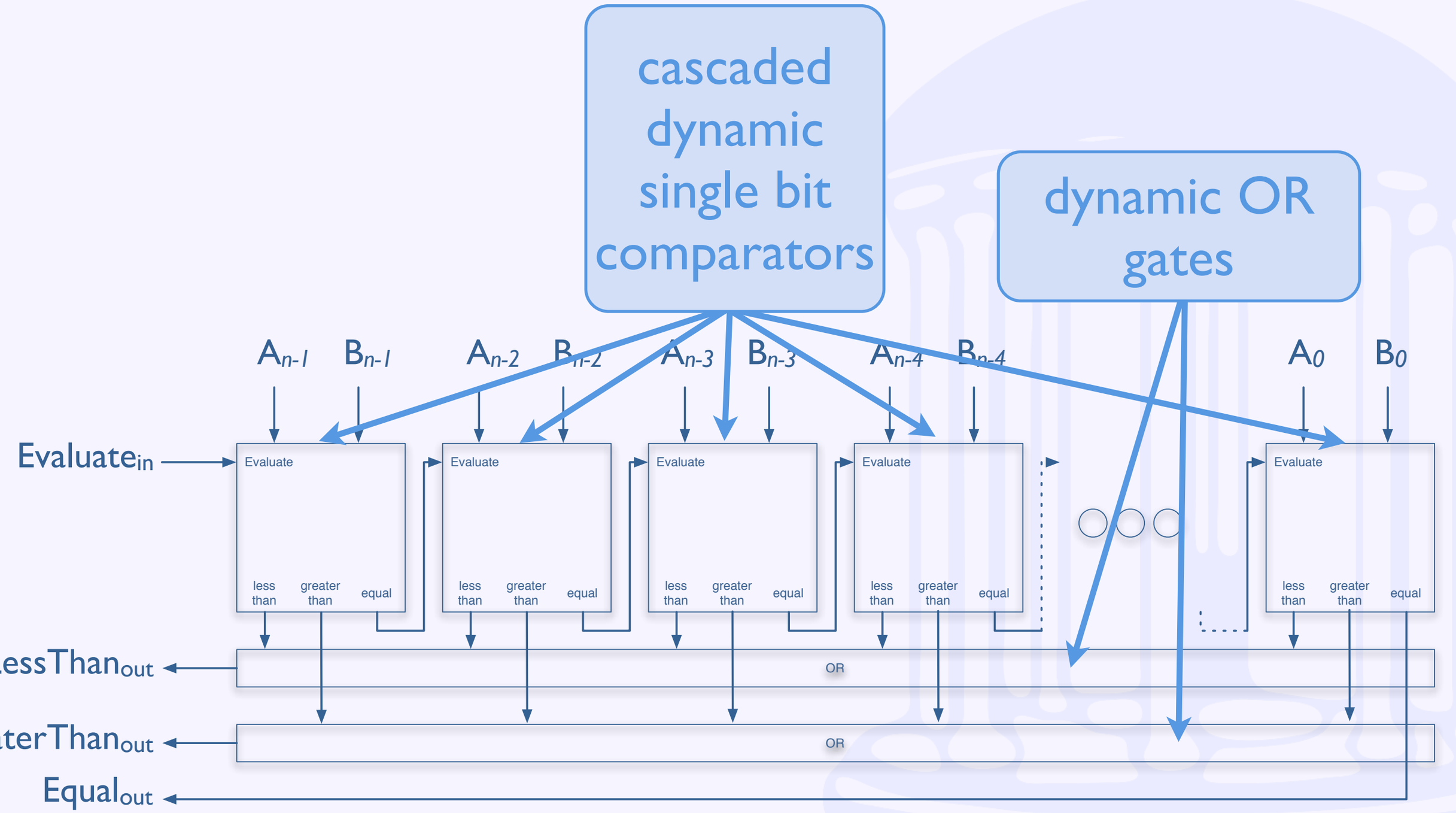
function block

function block

Putting it Together - PS0

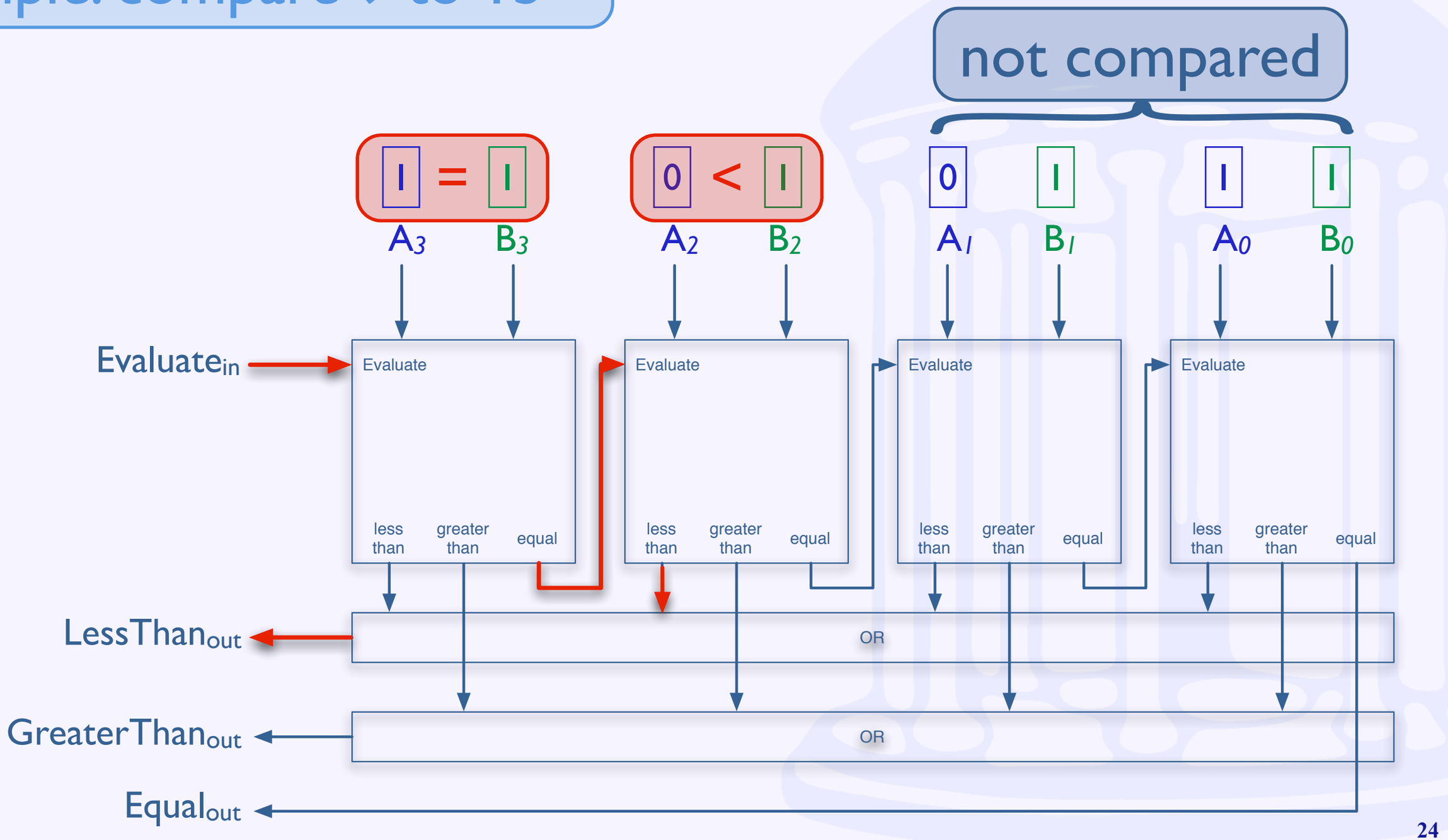


Comparator Architecture



Comparator Operation

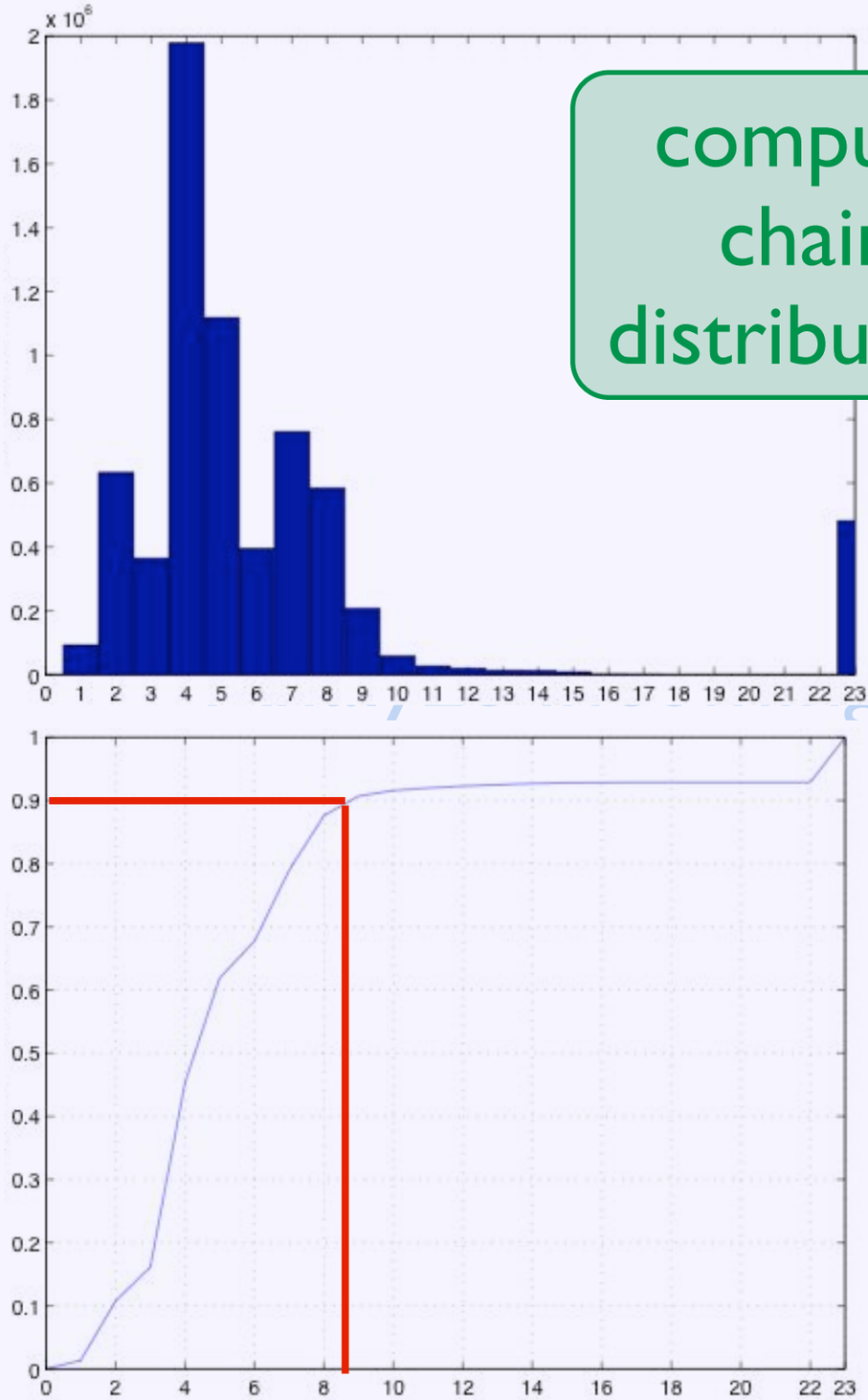
Example: compare 9 to 15



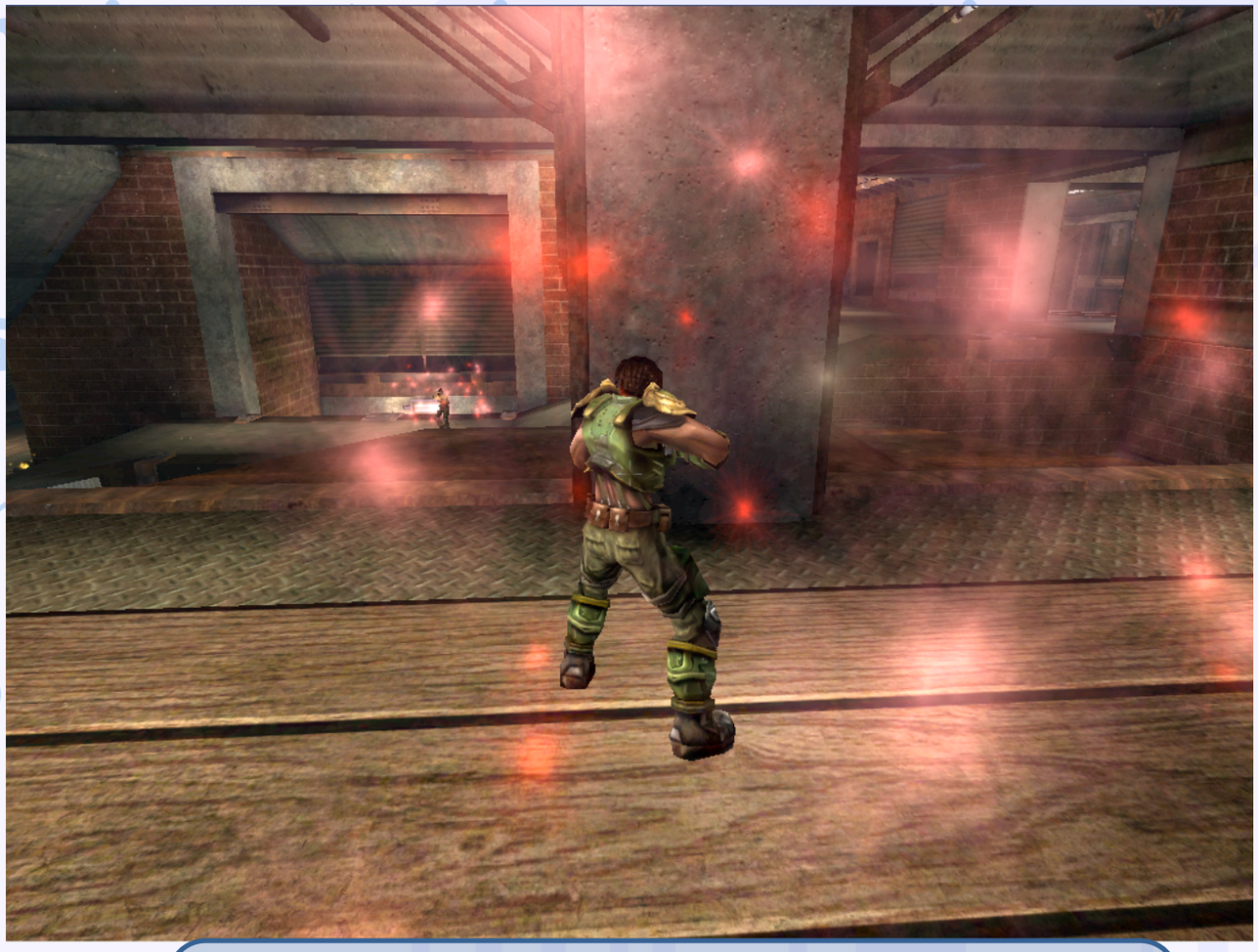
Comparator Results

- On average only 3 most significant bits needed for random inputs, irrespective size [Yun'97]
- Instrumented Mesa3D
  - Analyze incoming fragment depth value with depth value stored in the Z-buffer
  - Calculate number of most significant bits needed to perform comparison

Comparator Results



compute chain distribution



6,768,766 Z-comparisons
(frame from Unreal Tournament 2004)

Comparator
Simulation summary

Cadence SPICE simulation
180 *nm* at 300K with 1.8V
power supply

	Asynchronous	Synchronous
delay	**0.55 ns - 7.24 ns**	4.16 ns
average case delay (7 bit compare)	**2.49 ns**	4.16 ns
energy	**1.4 pJ - 12.89 pJ**	17.36 pJ - 22 pJ

Comparator
Previous Work

- **Similar to Knittel et. al. [Knittel'95]**
  - Computation proceeds from most-significant bit to least-significant bit
- **Data dependent completion times only when result is *true***
- **Uses alternating PMOS and NMOS stages**
- **Broadcast enable signal**

Asynchronous Logic for Graphics Hardware

- **Comparator on its own is insignificant**
- **Entire precision not always needed in the pipeline**
- **Take advantage of average-case performance**
- **Use "compute-on-demand" paradigm**
  - Reduce power consumption and increase performance for average case
- **Possible to mix asynchronous and synchronous systems**

Future Work

- Extend asynchronous multiplier of [Hensley/Singh'04] to handle variable precision operands
- Develop larger GPU components to use variable precisions arithmetic and compute on demand paradigm

Questions?

Acknowledgments:

- ATI Research
- National Science Foundation
  - CCF-0306478, CCF-0205425, CNS-0303590
- IBM faculty award
- Cadence Design Systems