Lecture 9: The Register File Design

Based on material prepared by prof. Visvesh S. Sathe

ELECTRICAL & COMPUTER ENGINEERING

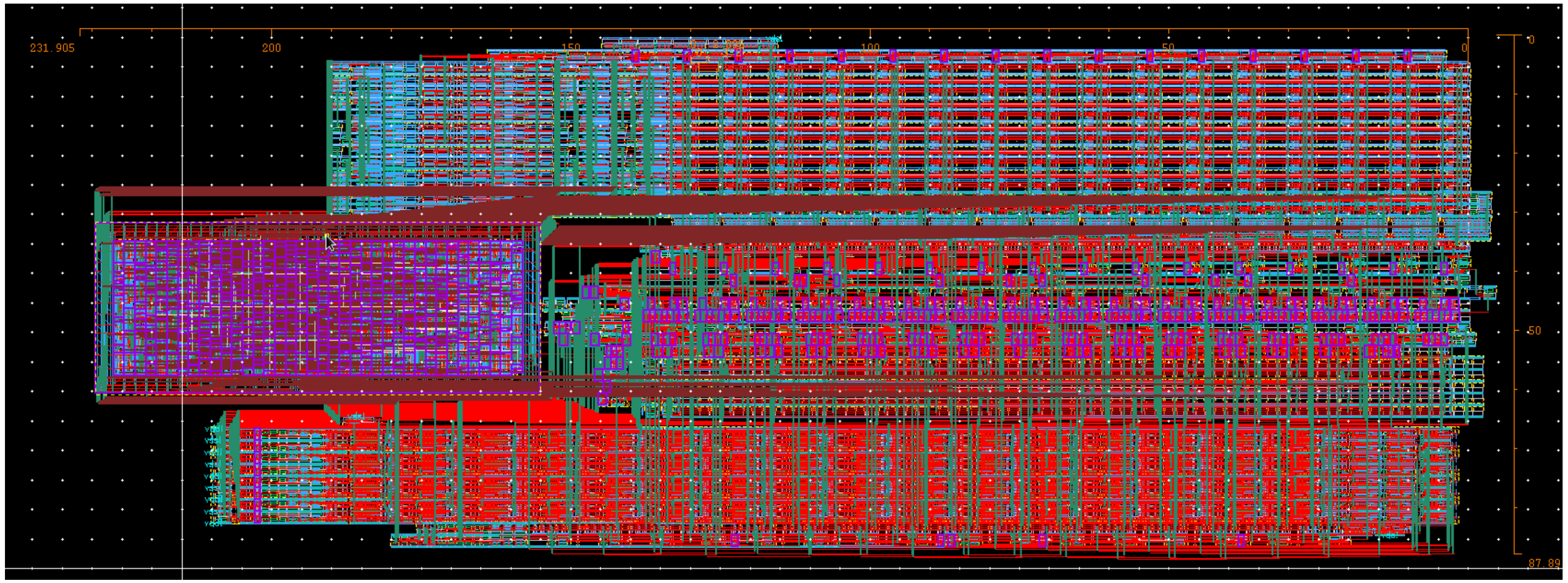
Acknowledgements

All class materials (lectures, assignments, etc.) based on material prepared by Prof. Visvesh S. Sathe, and reproduced with his permission

Visvesh S. Sathe
Associate Professor
Georgia Institute of Technology
https://psylab.ece.gatech.edu

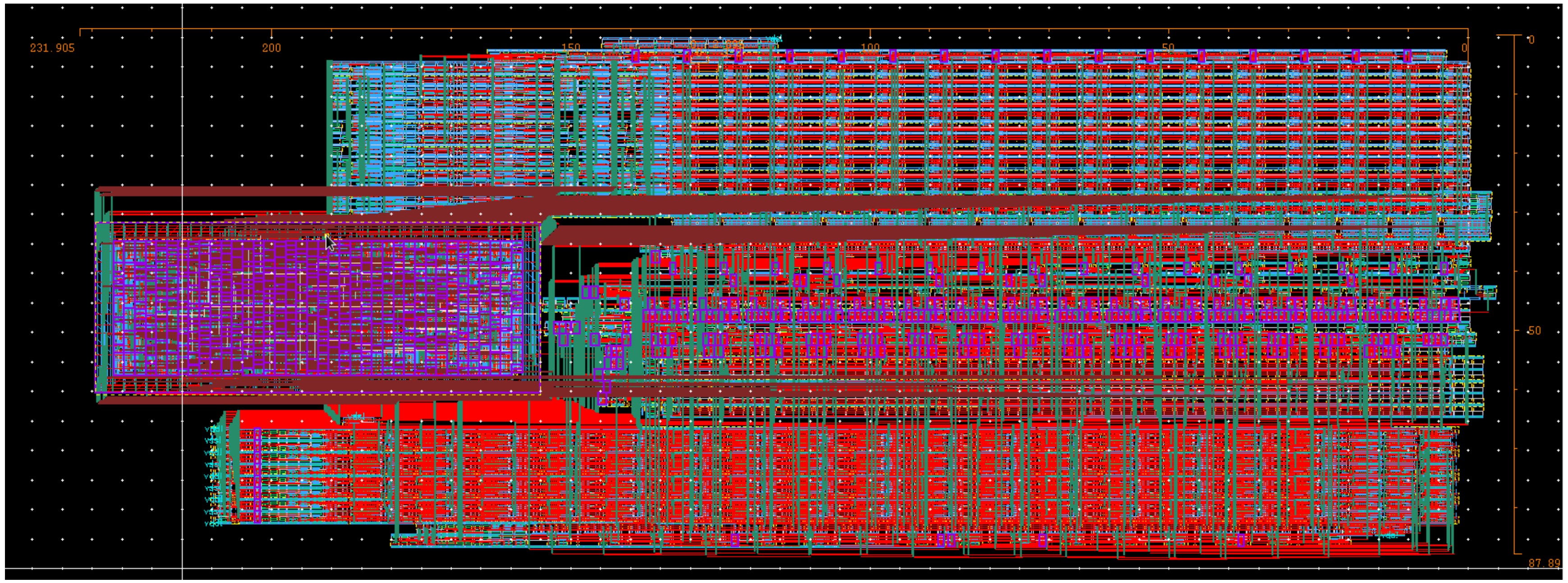
UW (2013-2022)
GaTech (2022-present)

Example Microprocessor Layout



- Datapath layed-out in a regular structure
- "Bitslice" is an important concept/metric
  - Aligning modules and wires along each of the 16 bits of your design will help
  - Data flow from regfile -> shift/alu/pc, from shift/alu ->regfile occur naturally along bit-boundaries.
  - E.g. regfile-out lines up with alu/shift/pc IN
  - Regfile-in lines up with alu/shift/ OUT

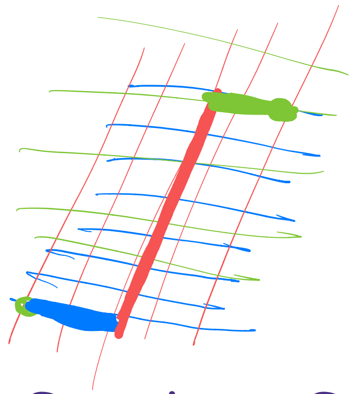
Example Microprocessor Layout

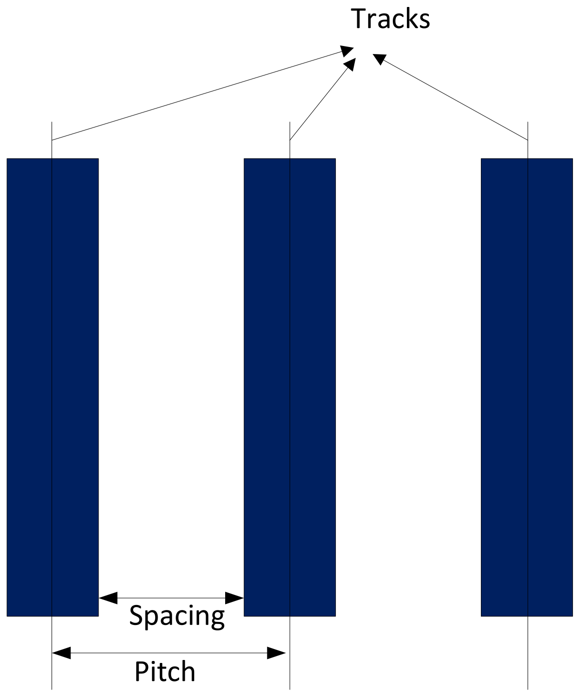


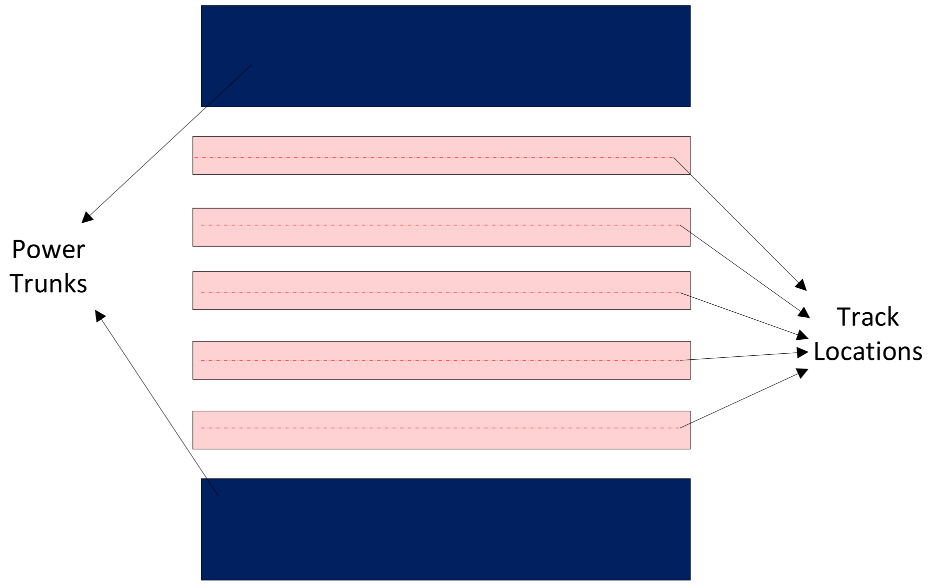
- Datapath layed-out in a regular structure
- "Bitslice" is an important concept/metric
  - Aligning modules and wires along each of the 16 bits of your design will help
  - Data flow from regfile -> shift/alu/pc, from shift/alu ->regfile occur naturally along bit-boundaries.
  - E.g. regfile-out lines up with alu/shift/pc IN
  - Regfile-in lines up with alu/shift/ OUT

COMMON BITSLICE!!!

Layout Design Conventions

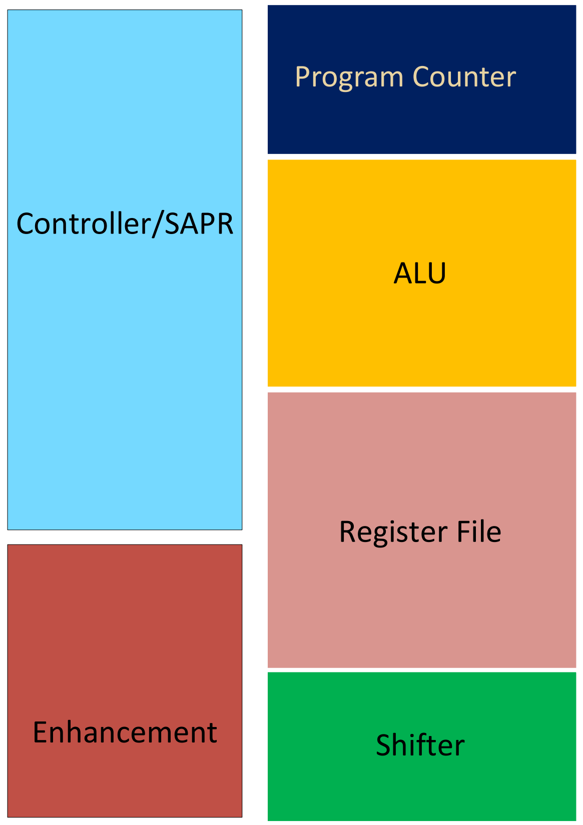






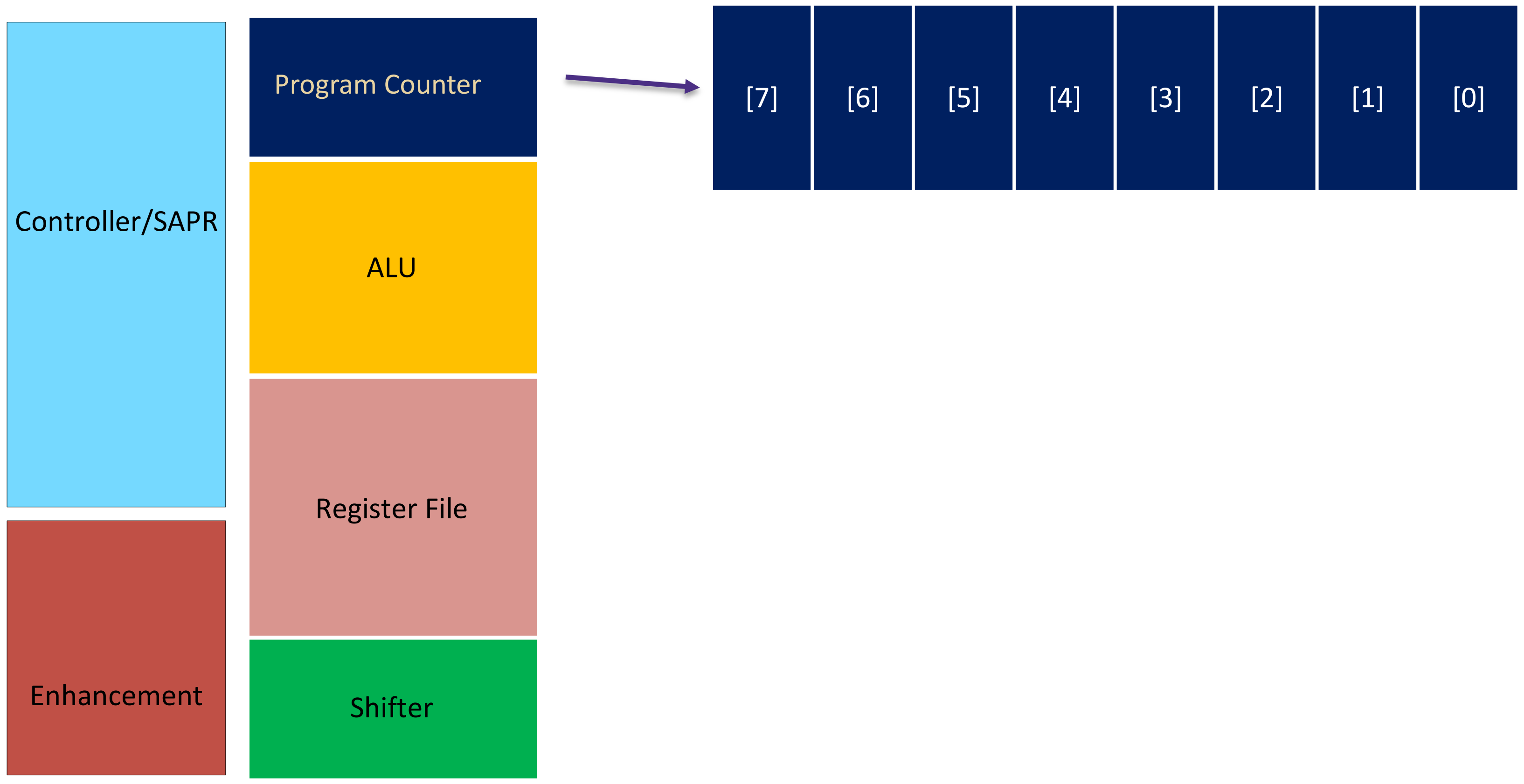
- Spacing : Space between object edges
- Pitch : Space between object centerlines
- Track : Centerline of region designated for metal route
- Cell-height : Height of cell from Vdd-centerline to Vss-centerline
- Bit-slice width : **Width** of design corresponding to 1 bit
  - Important to maintain uniform bit-slice width across your structures

Bit-slices and your Datapath



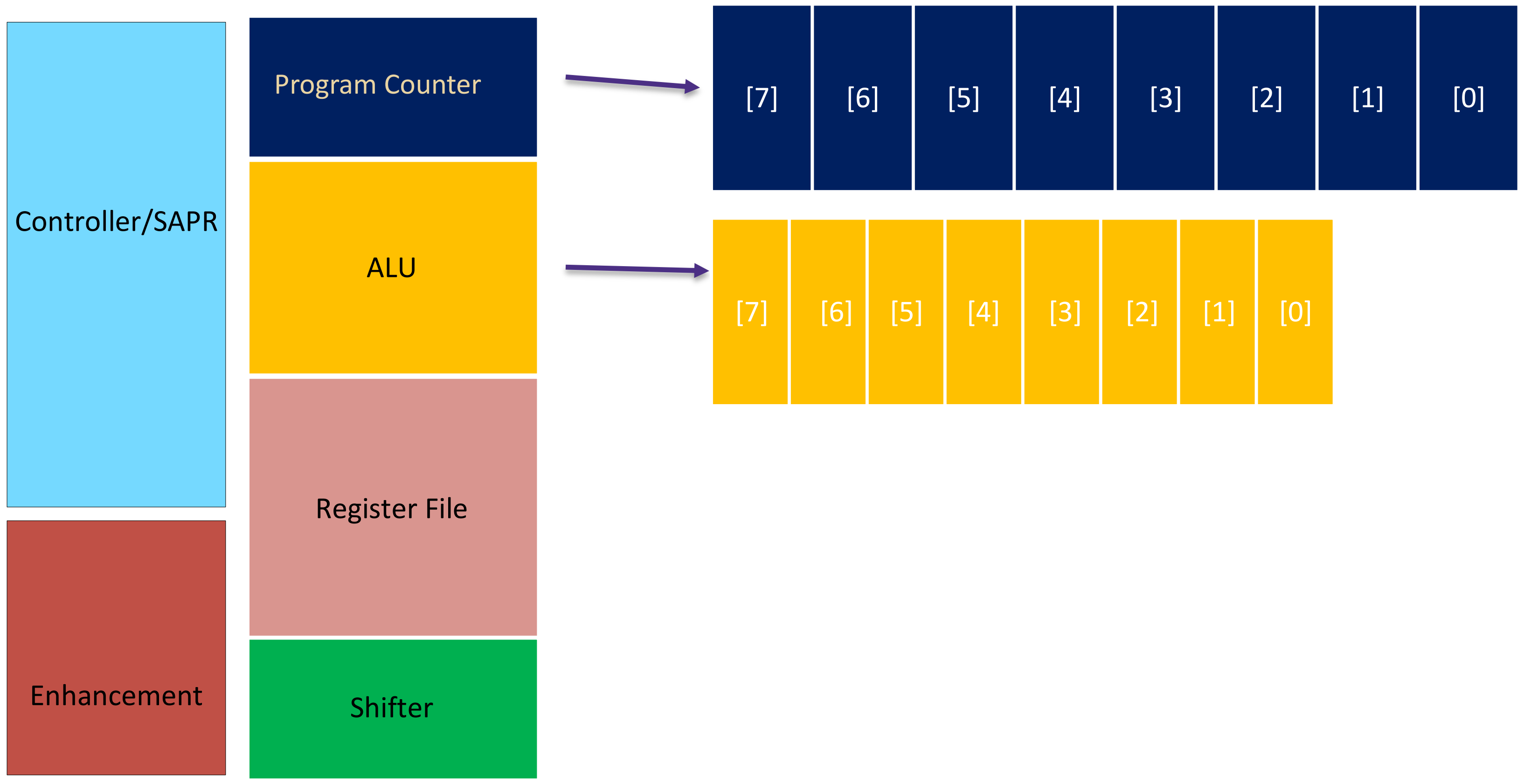
- Bit-slice: Unit section of your datapath corresponding to 1 bit
- Maintain uniform bit-slice across your modules
  - If the structure is narrow, insert space, or shorten height

Bit-slices and your Datapath



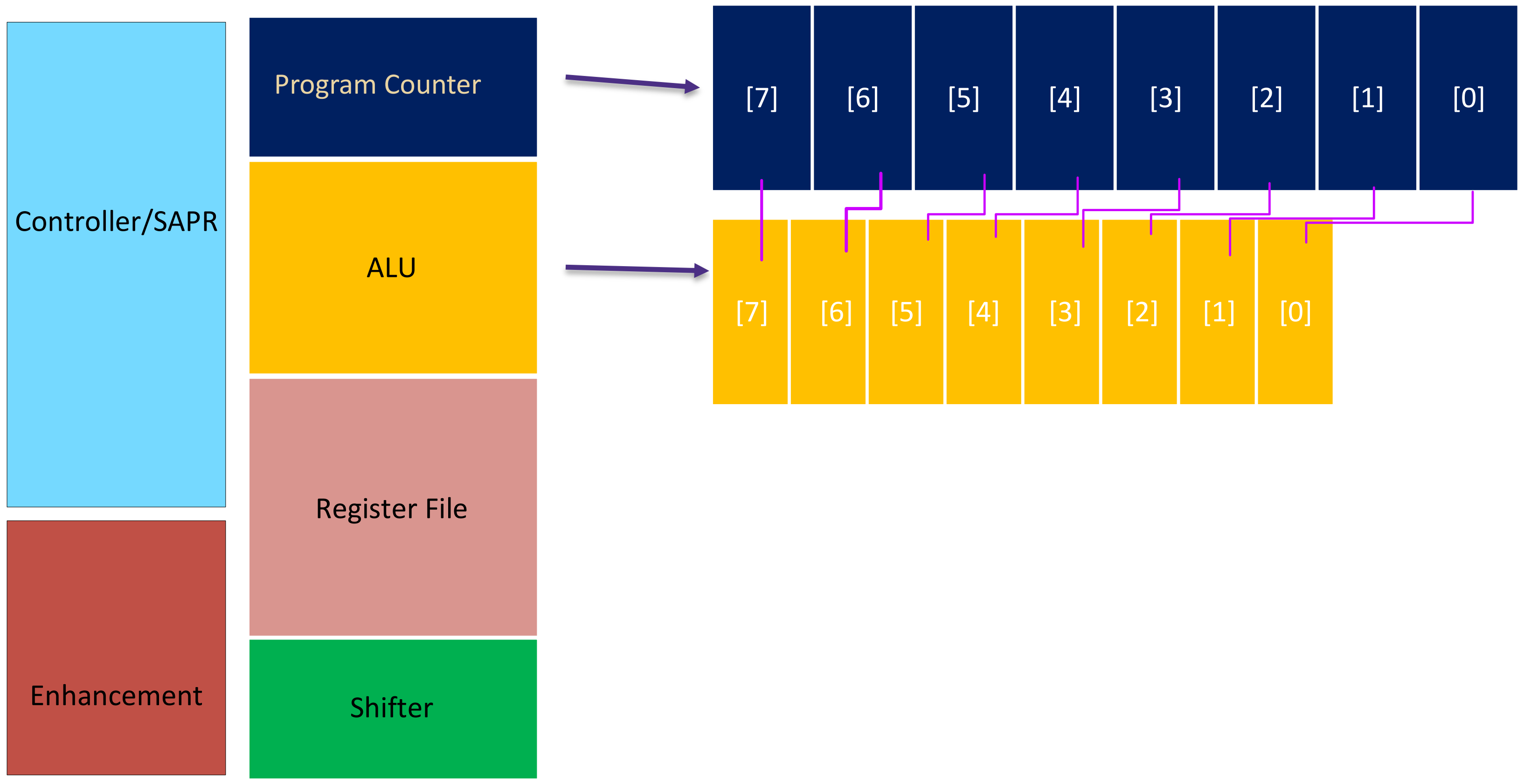
- Bit-slice: Unit section of your datapath corresponding to 1 bit
- Maintain uniform bit-slice across your modules
  - If the structure is narrow, insert space, or shorten height

Bit-slices and your Datapath



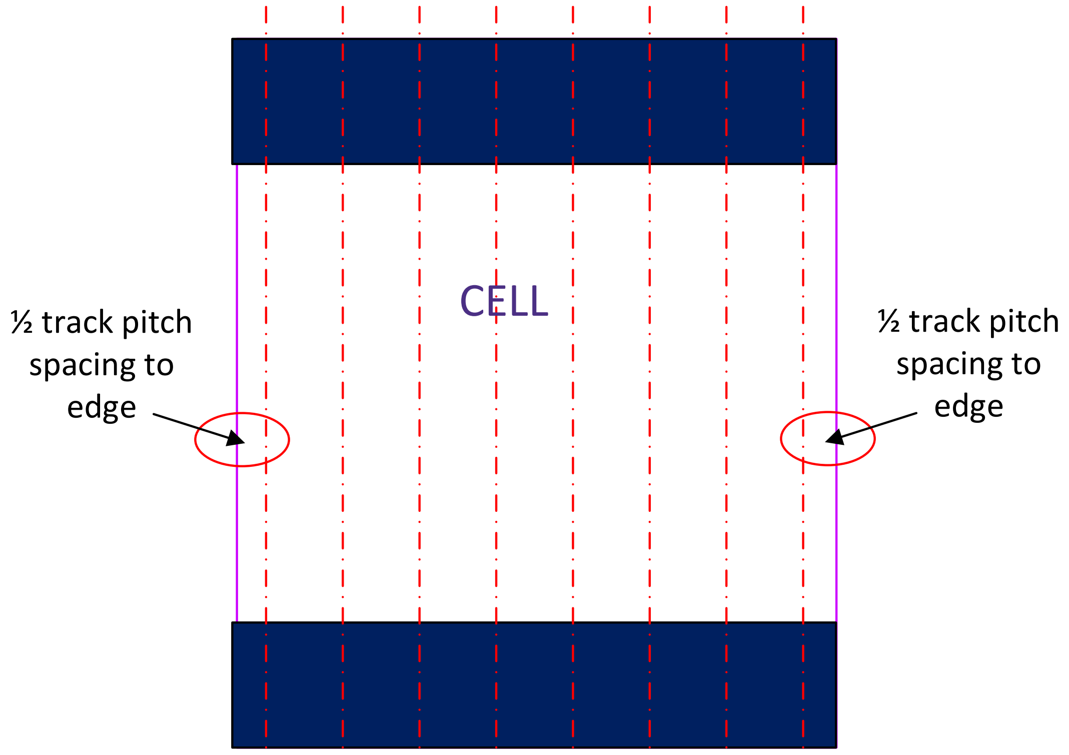
- Bit-slice: Unit section of your datapath corresponding to 1 bit
- Maintain uniform bit-slice across your modules
  - If the structure is narrow, insert space, or shorten height

Bit-slices and your Datapath



- Bit-slice: Unit section of your datapath corresponding to 1 bit
- Maintain uniform bit-slice across your modules
  - If the structure is narrow, insert space, or shorten height

Managing Power Trunks and Tracks



- For example: Min-width = 0.1um, Min-spacing = 0.1um
- Track pitch = 0.2um
- Recommended : Cell boundary 0.5 track pitches away from track location ➔ Cell width is a multiple of track pitch (0.2um in this example)
- **Route on track locations**. Number/Plan tracks

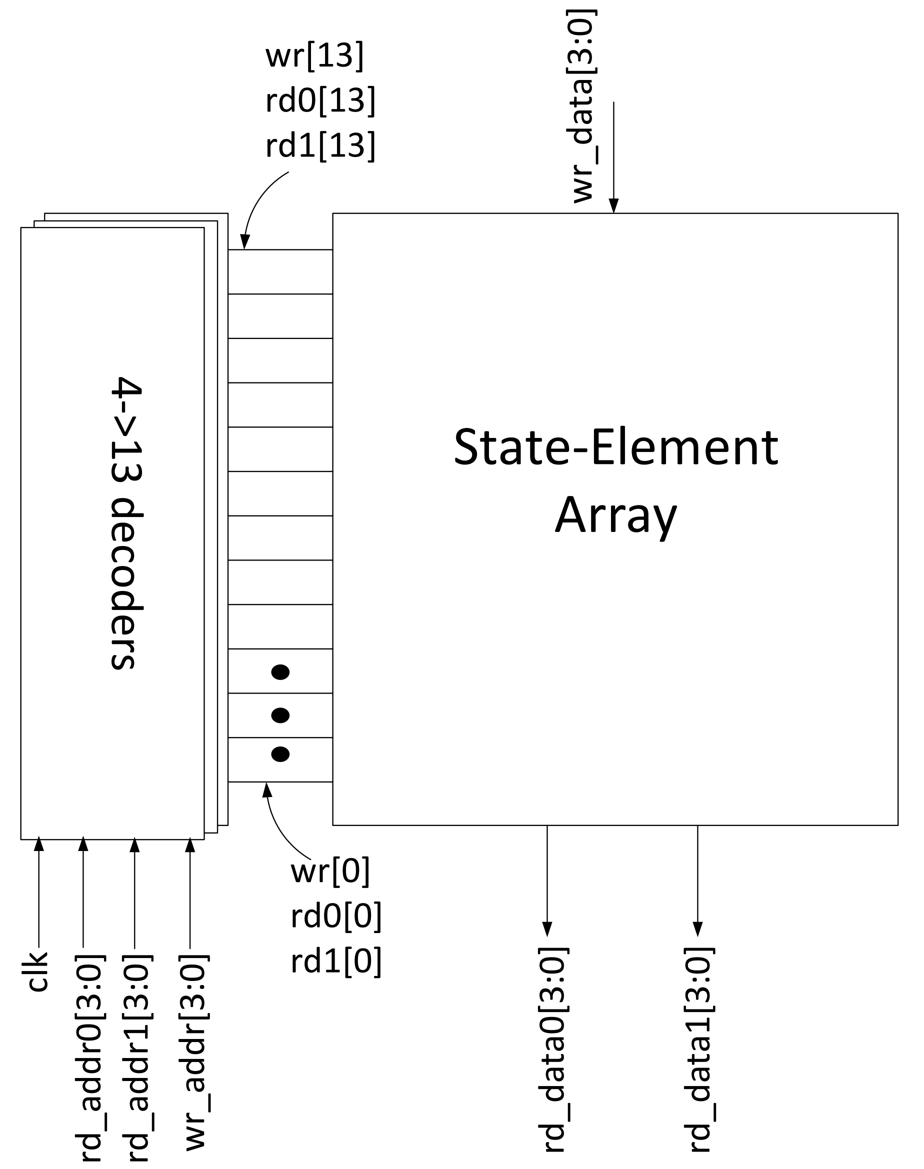
The Register File (Regfile)

- Store register values (operands for processor instructions)
  - 13-entry (R0-R12), 16-bit
  - 1 write port, 2 read ports
  - Write-before-read regfile operation

- Ports
  - VDD!
  - GND!
  - clk (synchronize the write operation)
  - read_addr_0 (Read address for port 0. 4-bits)
  - read_addr_1 (Read address for port 1. 4-bits)
  - wr_en (Write enable)
  - wr_data (write data, 16 bits)
  - regfile_data0 (Data from read port 0)
  - regfile_data1 (Data from read port 1)
  - wr_ address

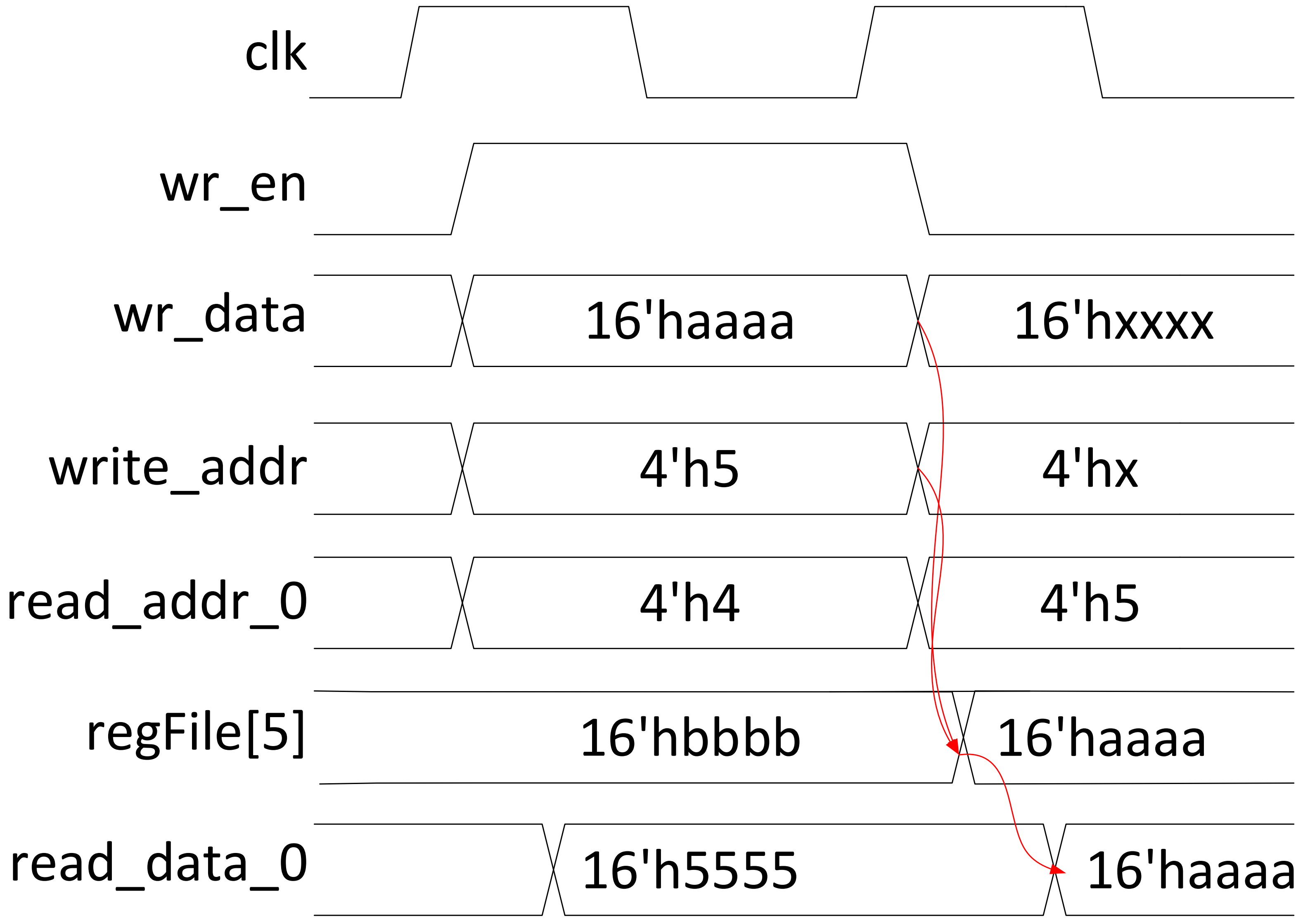
ELECTRICAL & COMPUTER ENGINEERING

Basic Structure



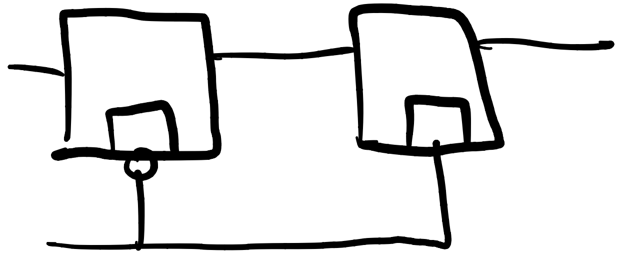
ELECTRICAL & COMPUTER ENGINEERING

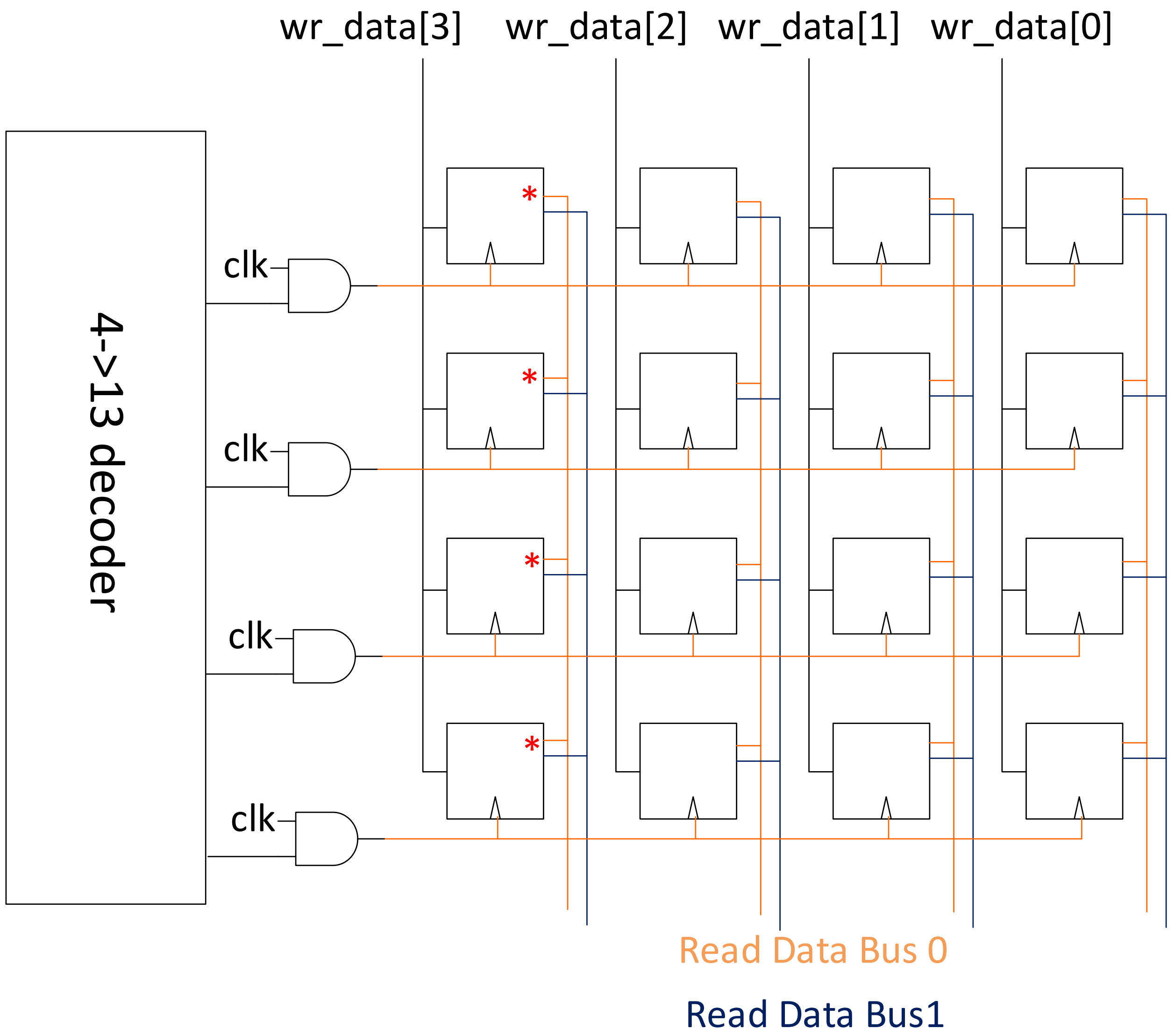
Timing Diagram Walkthrough



Naïve Implementation

- Array of Flip Flops

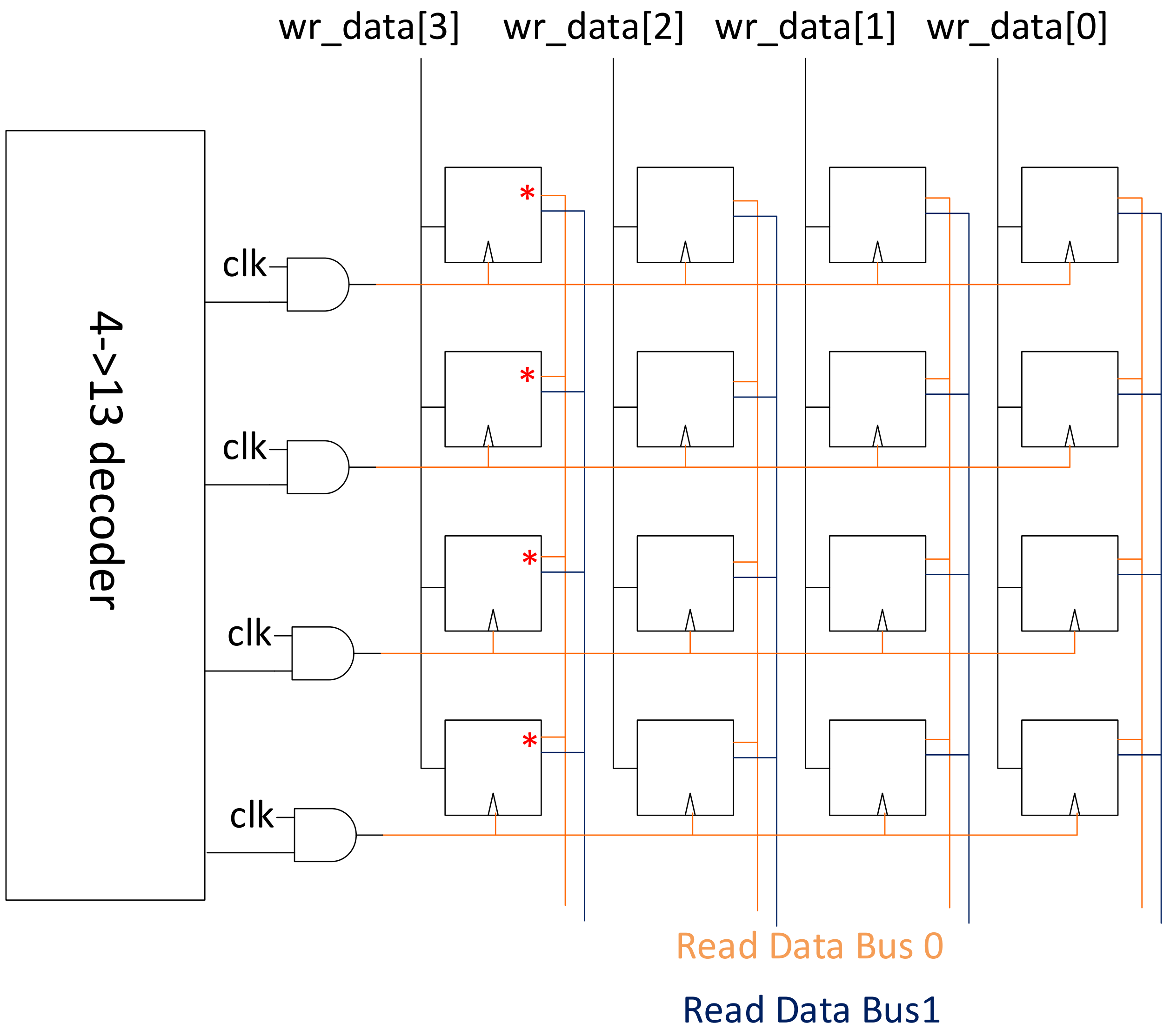




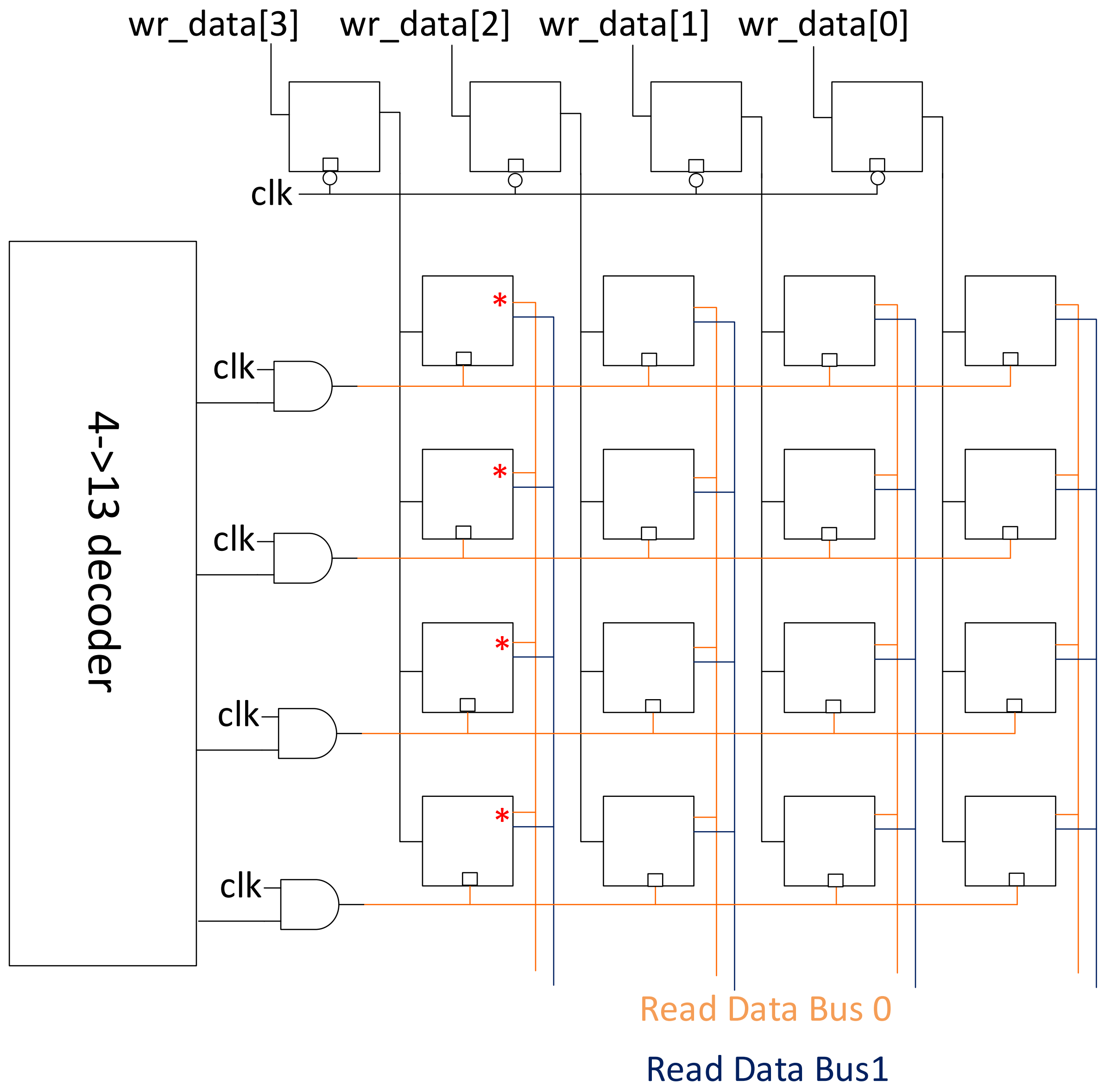
ELECTRICAL & COMPUTER ENGINEERING

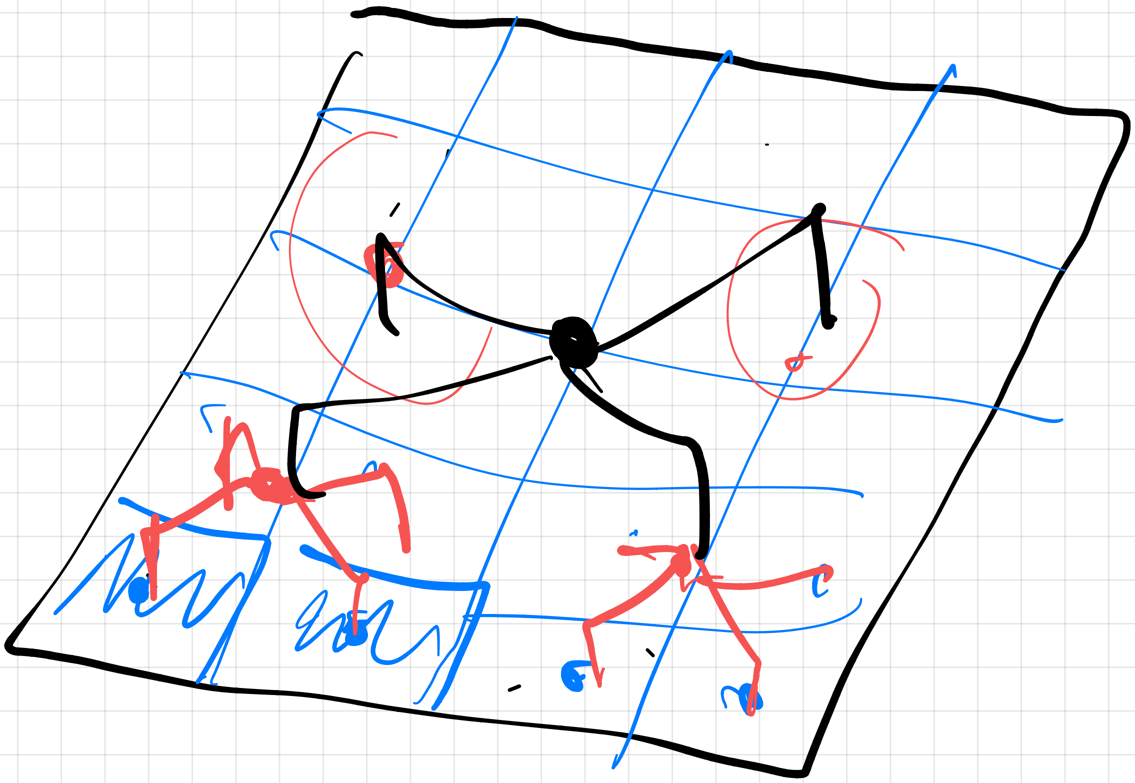
Naïve Implementation

- Array of Flip Flops
- BUT….All master latches are doing the same job!

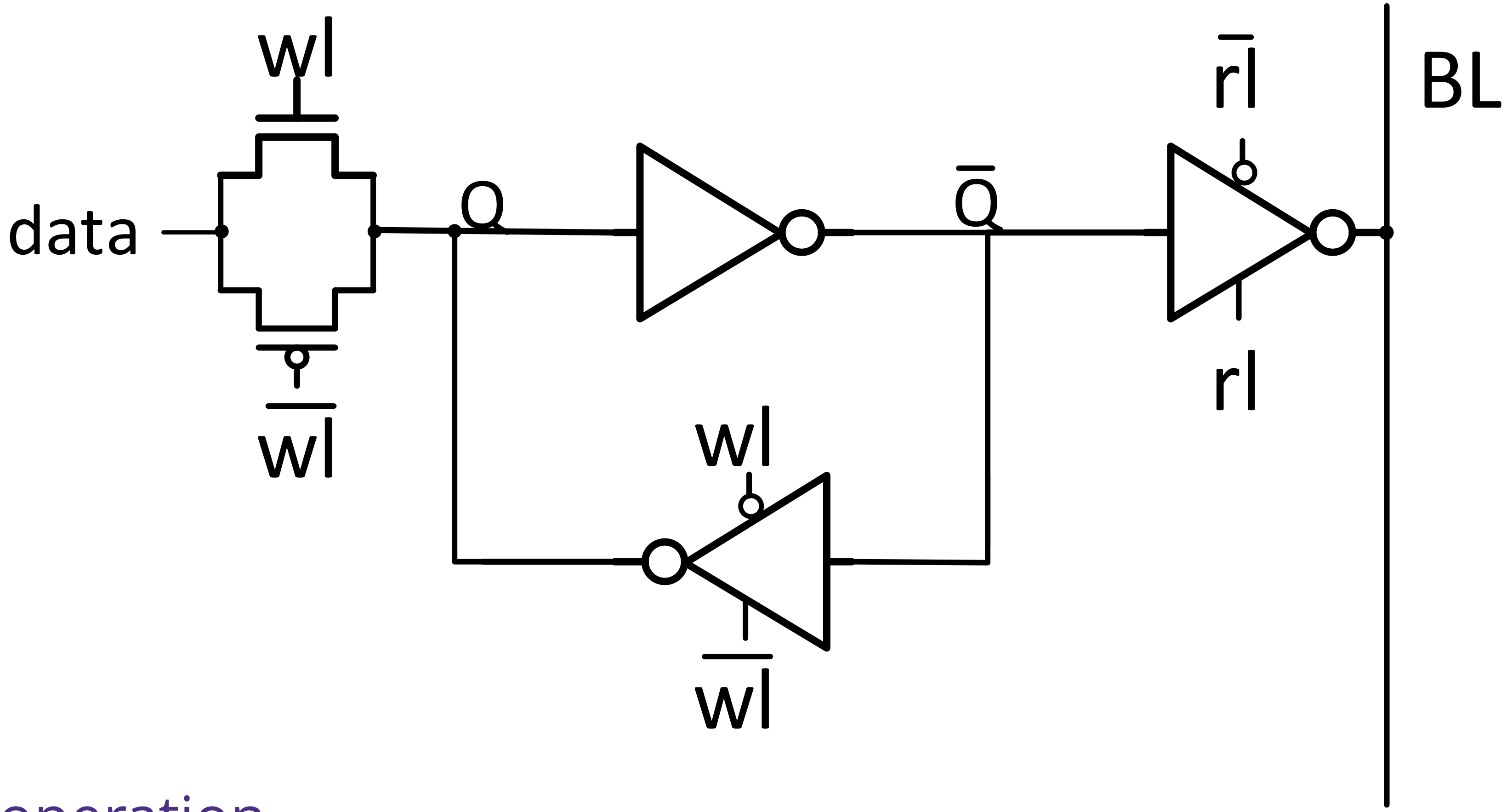


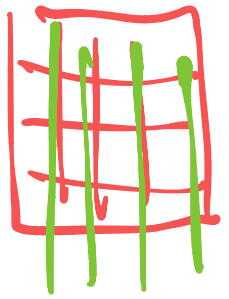
More Compact Alternative





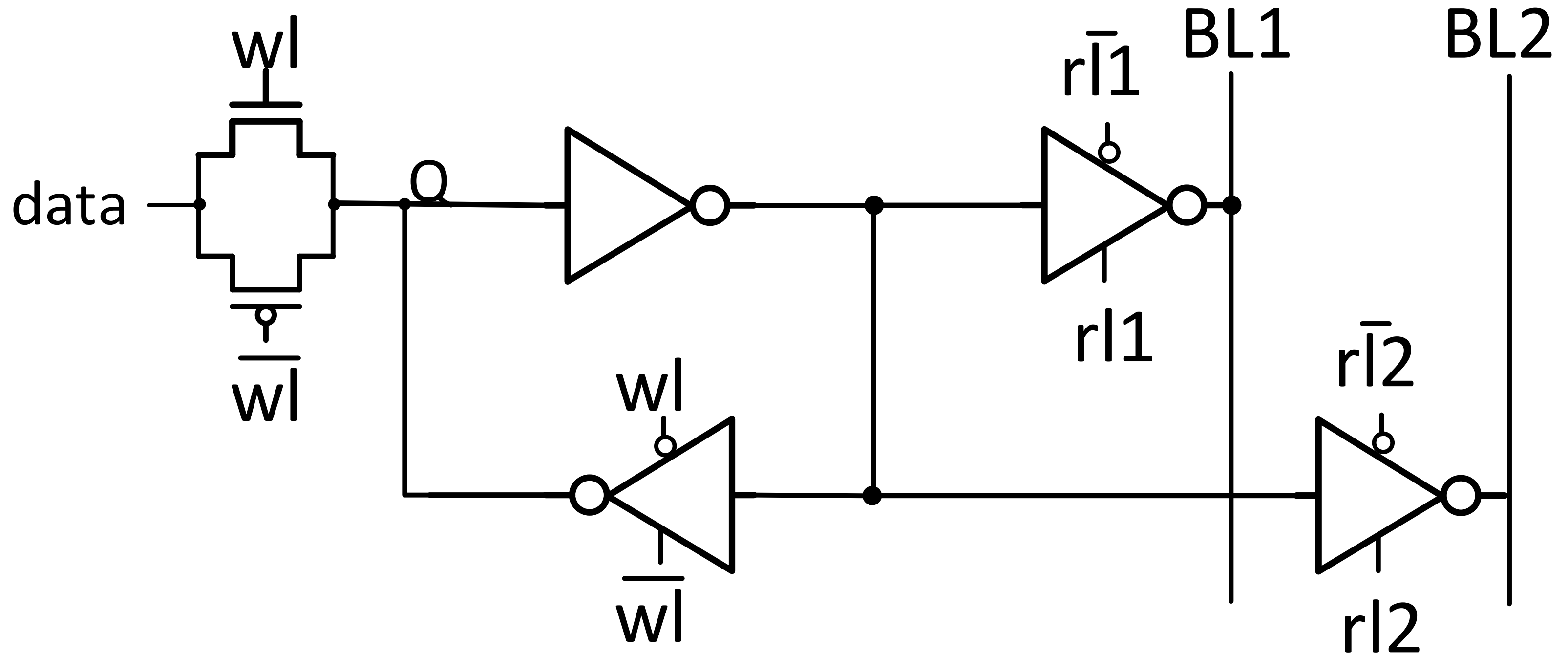
Basic Regfile cell





- 12T cell
- Fully static operation

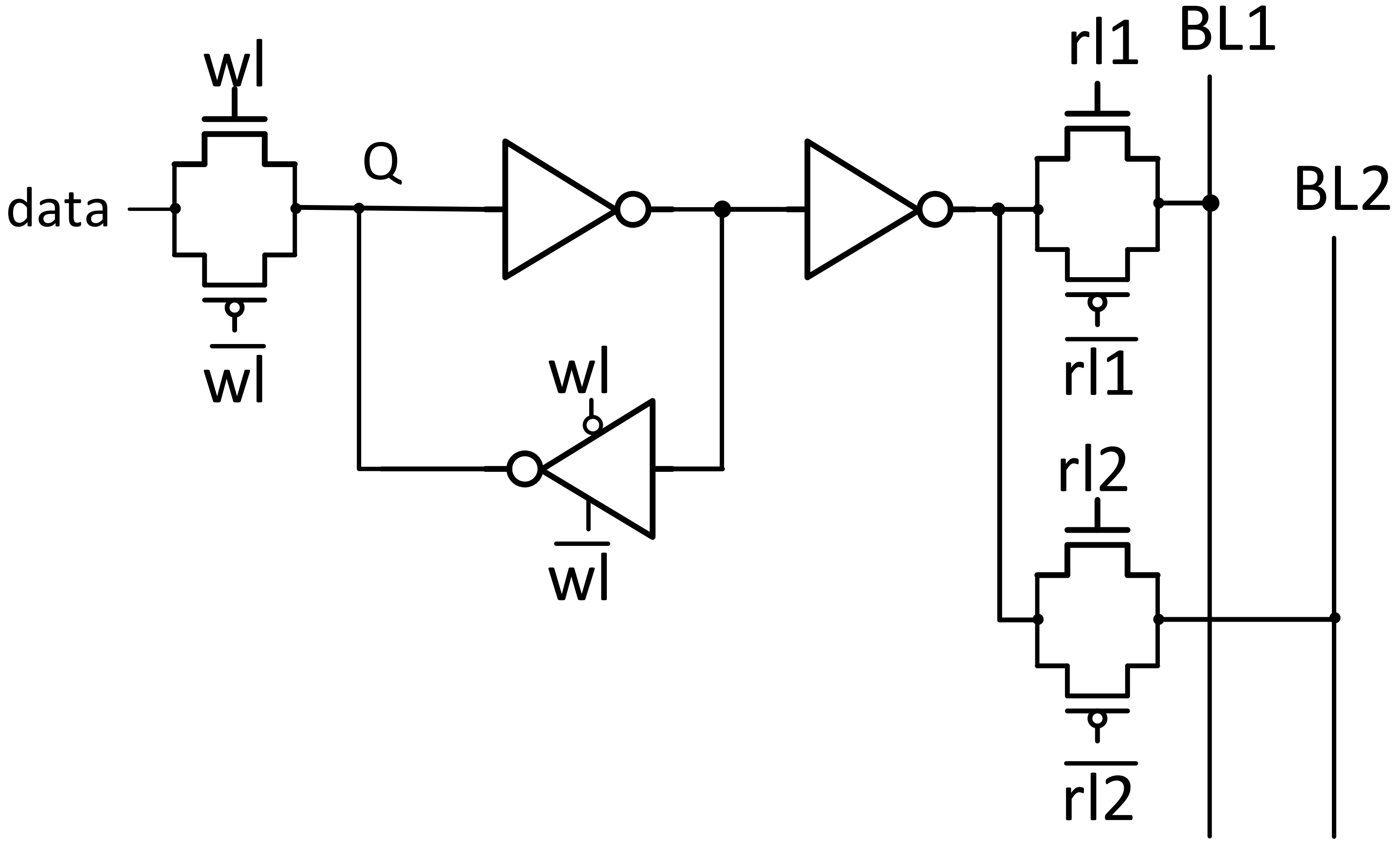
Supporting Multiple Bitlines



- Each bitline is the muxed output connecting all column registers
- What is the capacitance on each bitline?
- How does it impact the size of the drivers?

ELECTRICAL & COMPUTER ENGINEERING

Transmission Gate Alternative



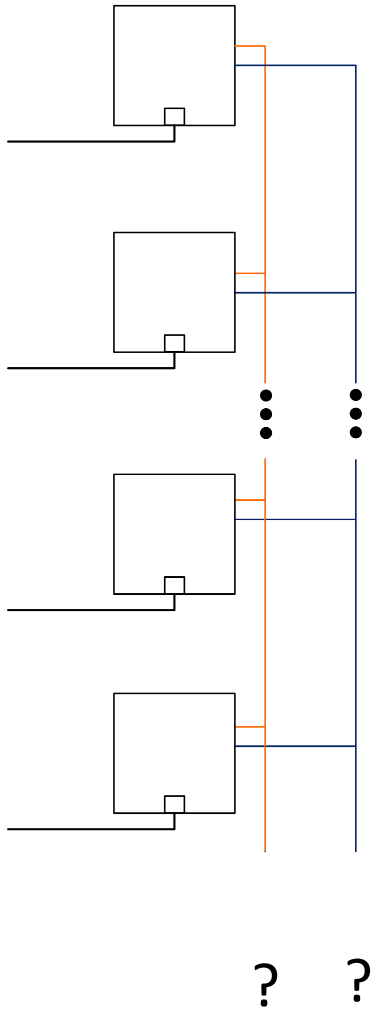
- Advantage: Transmission gate offers lower delay for large loads
- Disadvantage: Critical path loading!
- Is there a third approach to avoid this disadvantage?

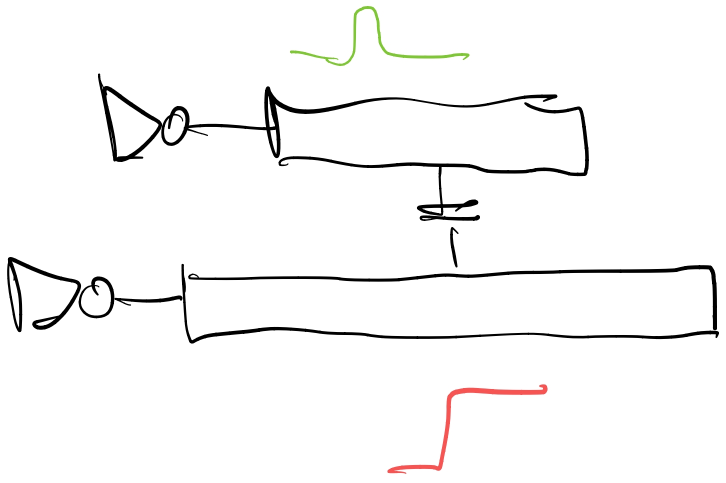
ELECTRICAL & COMPUTER ENGINEERING

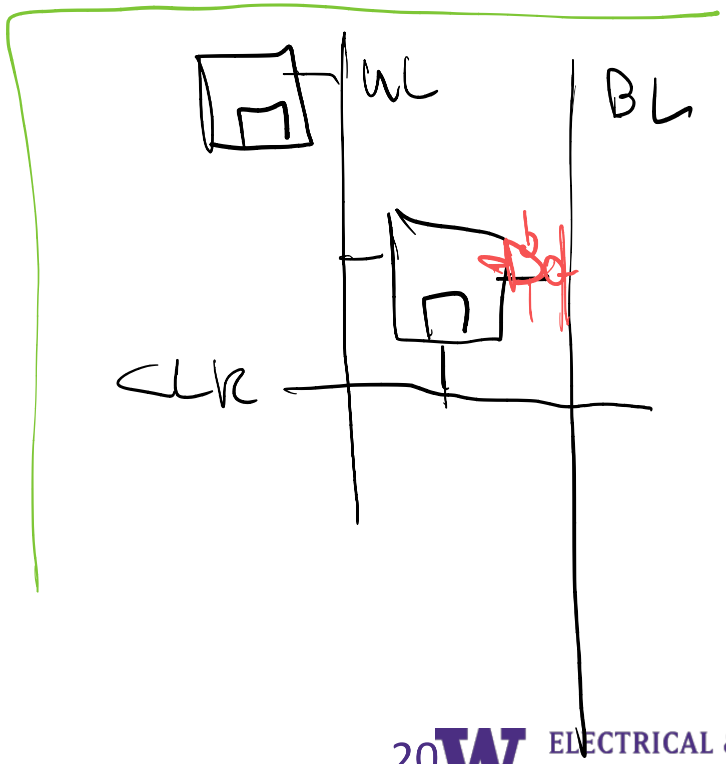
Read Delay Analysis

$$T_{read} = T_{clk-q} + T_{bl-drive}$$

- Delay depends on bitcell topology
- Coupling on bitline is important
- Exercise: Order the following delays
  - TX gate output. Bitlines read: (a) Same reg and (b) Different reg
  - Tri-state output. Bitlines read: (a) Same reg and (b) Different reg





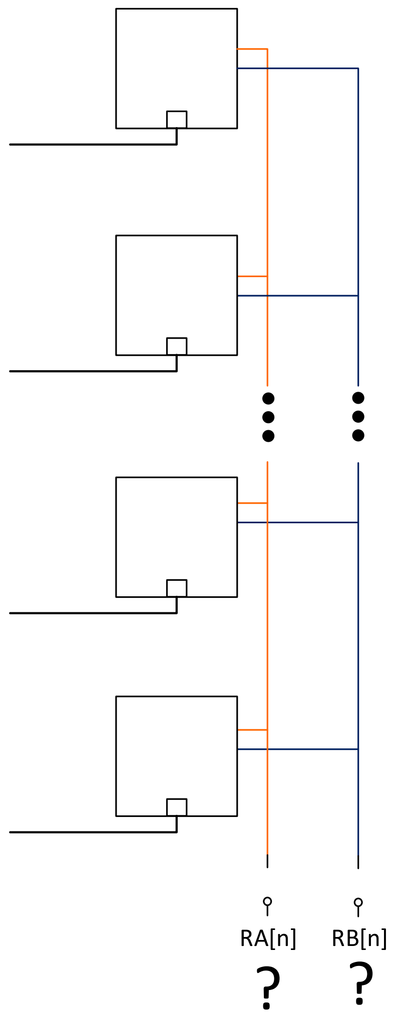


ELECTRICAL & COMPUTER ENGINEERING

Read Delay Analysis

$$T_{read} = T_{clk-q} + T_{bl-drive}$$

- Delay depends on bitcell topology
- Coupling on bitline is important
- Exercise: Order the following delays
  - TX gate output. Bitlines read: (a) Same reg and (b) Different reg
  - Tri-state output. Bitlines read: (a) Same reg and (b) Different reg

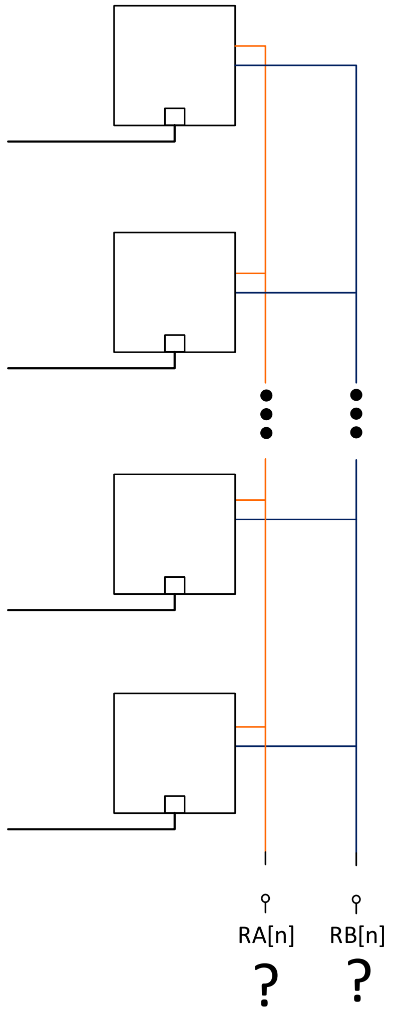


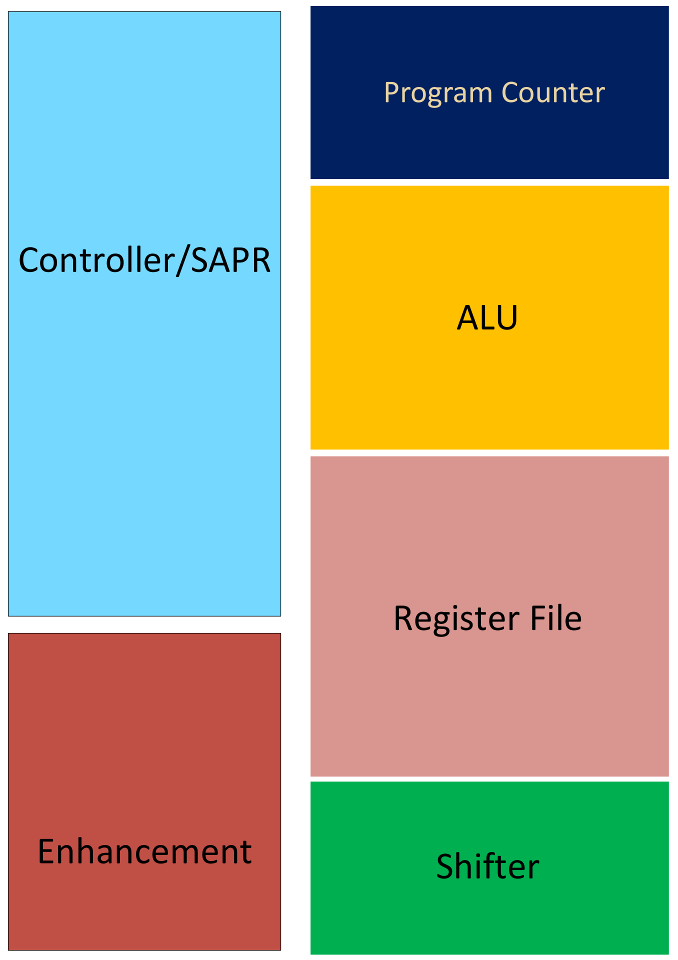
ELECTRICAL & COMPUTER ENGINEERING

Read Delay Analysis

$$T_{read} = T_{clk-q} + T_{bl-drive}$$

- Delay depends on bitcell topology
- Coupling on bitline is important
- Exercise: Order the following delays
  - TX gate output. Bitlines read: (a) Same reg and (b) Different reg
  - Tri-state output. Bitlines read: (a) Same reg and (b) Different reg

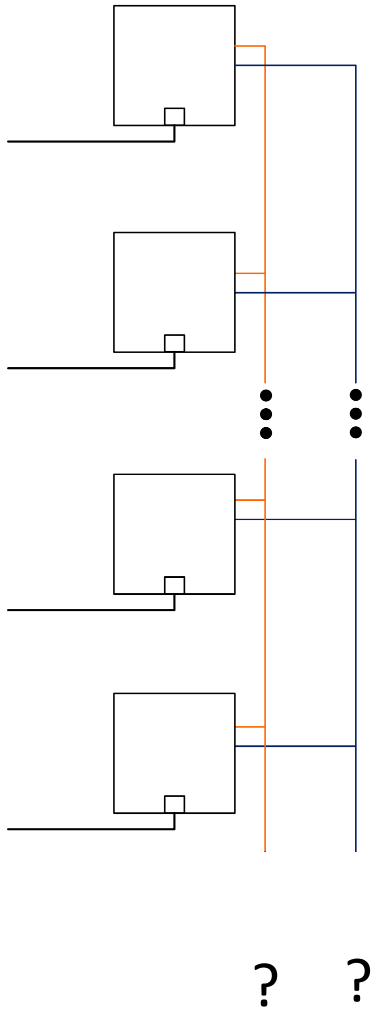


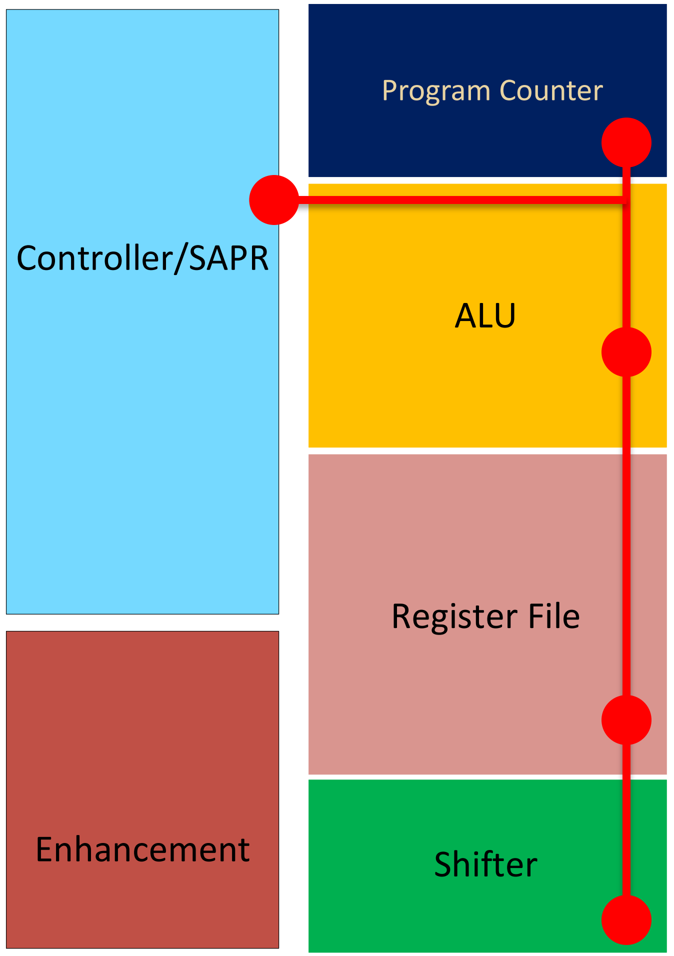


Read Delay Analysis

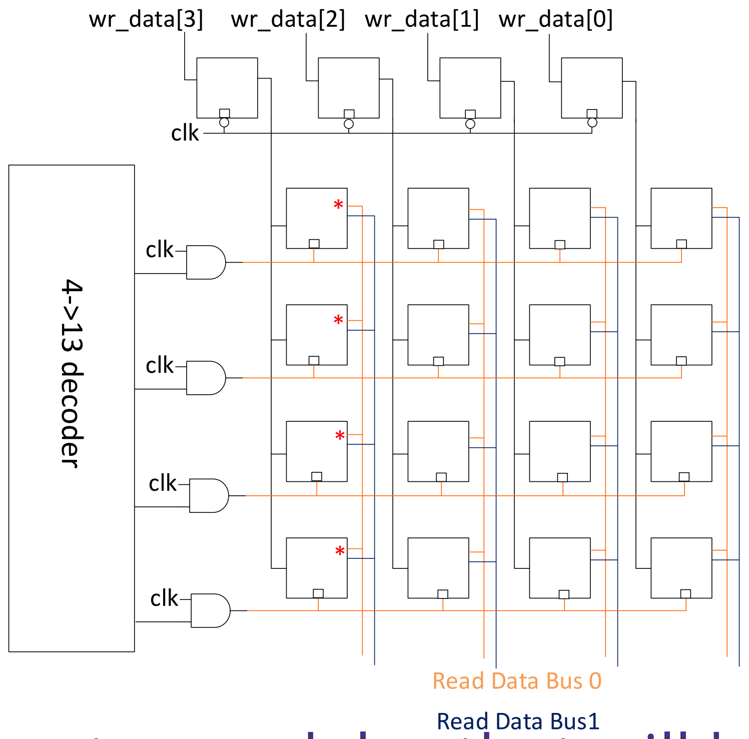
$$T_{read} = T_{clk-q} + T_{bl-drive}$$

- Delay depends on bitcell topology
- Coupling on bitline is important
- Exercise: Order the following delays
  - TX gate output. Bitlines read: (a) Same reg and (b) Different reg
  - Tri-state output. Bitlines read: (a) Same reg and (b) Different reg





Critical Delay Analysis

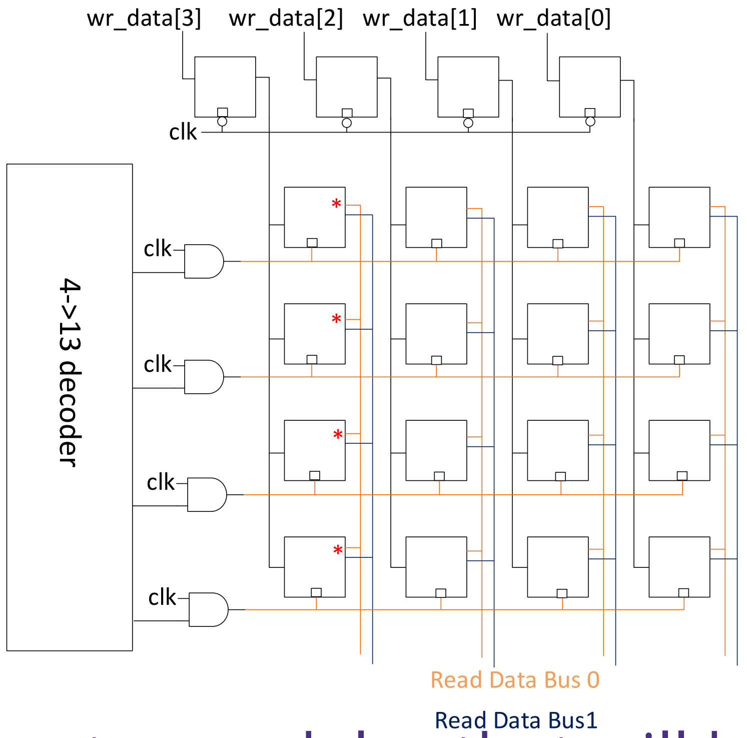


- Exercise: What is the worst-case delay that will be faced by this register file? (Hint: Data written in the previous cycle may be read in the current cycle)

$$T_{clk-out} =$$

- Note: This eqn. applies for when you have a flop-based capture of the write address

Critical Delay Analysis

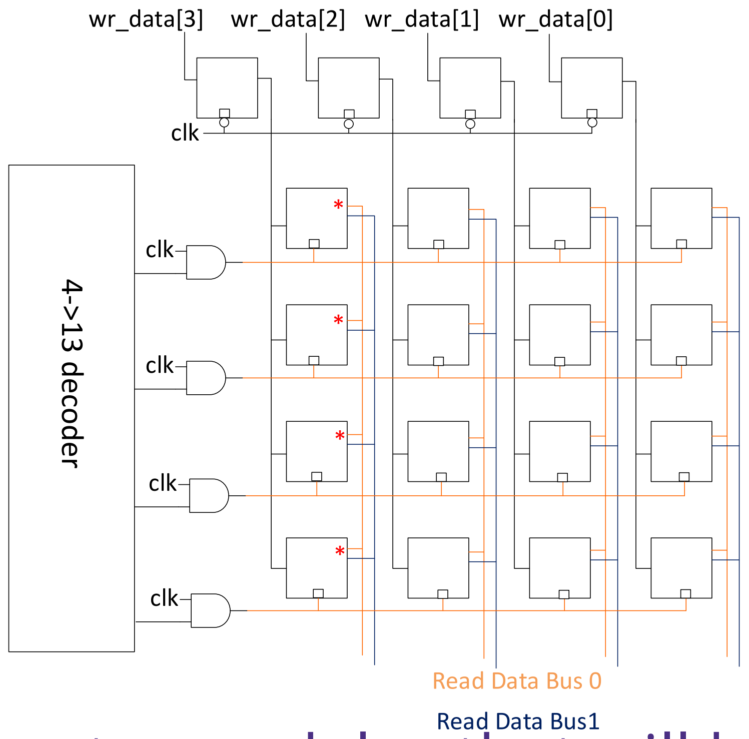


- Exercise: What is the worst-case delay that will be faced by this register file? (Hint: Data written in the previous cycle may be read in the current cycle)

$T_{clk-out} =$ $T_{DQ-slave} +$

- Note: This eqn. applies for when you have a flop-based capture of the write address

Critical Delay Analysis

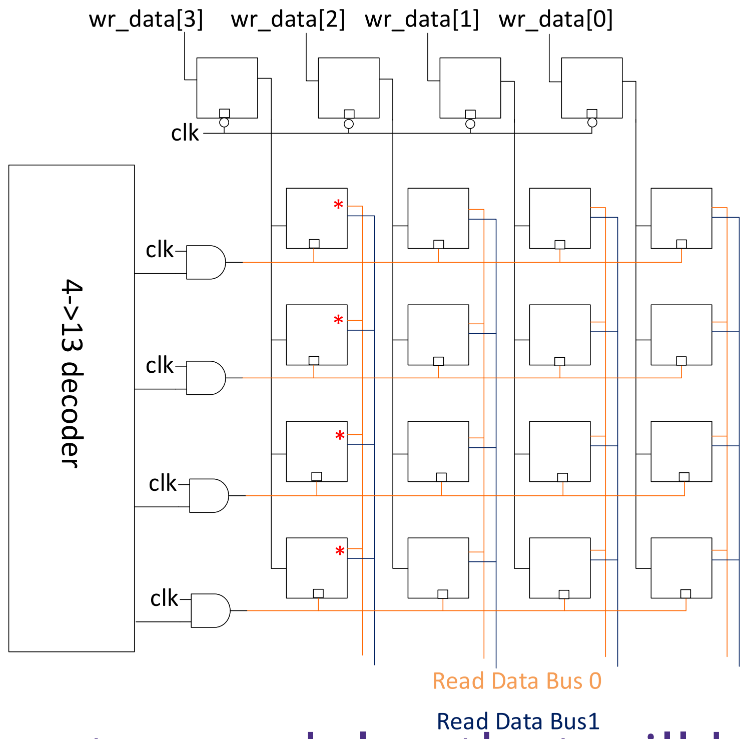


- Exercise: What is the worst-case delay that will be faced by this register file? (Hint: Data written in the previous cycle may be read in the current cycle)

$$T_{clk-out} = T_{clk-wr\_addr} + \qquad\qquad T_{DQ-slave} +$$

- Note: This eqn. applies for when you have a flop-based capture of the write address

Critical Delay Analysis

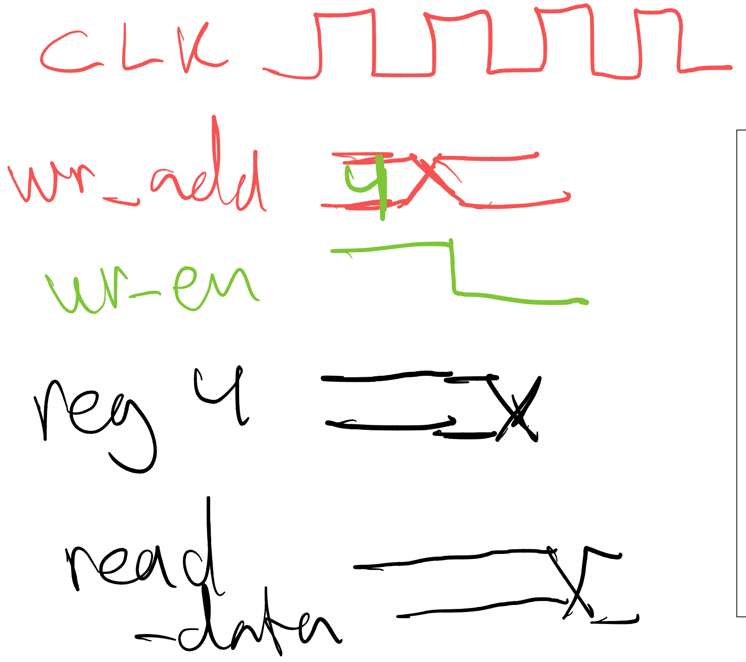


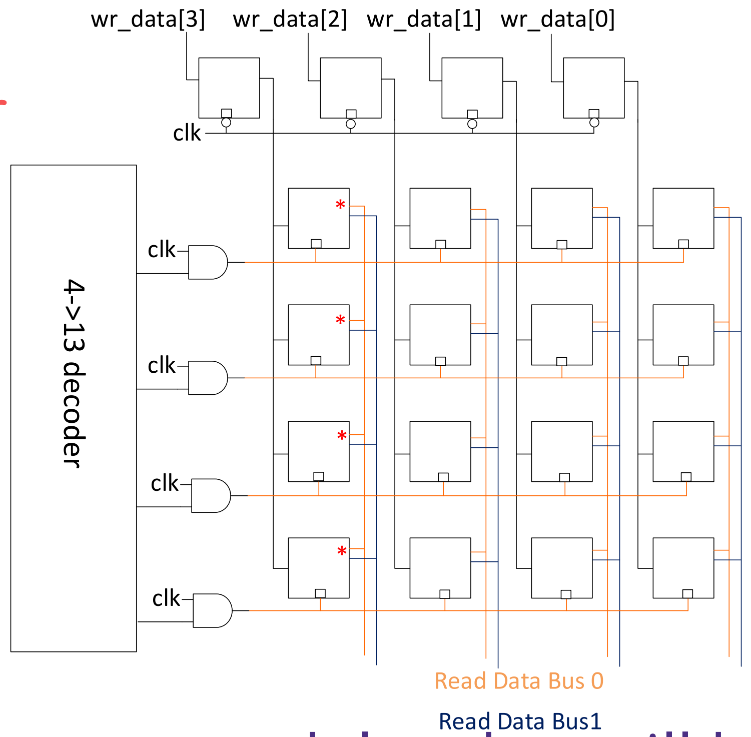
- Exercise: What is the worst-case delay that will be faced by this register file? (Hint: Data written in the previous cycle may be read in the current cycle)

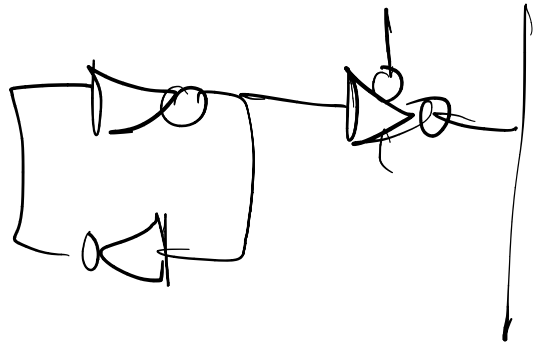
$$T_{clk-out} = T_{clk-wr\_addr} + \qquad\qquad T_{DQ-slave} + T_{bl-drive}$$

- Note: This eqn. applies for when you have a flop-based capture of the write address

Critical Delay Analysis







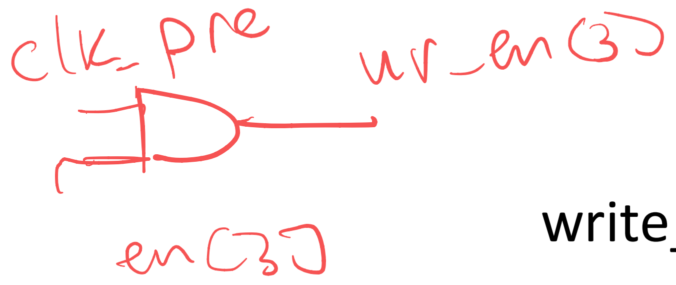
- Exercise: What is the worst-case delay that will be faced by this register file? (Hint: Data written in the previous cycle may be read in the current cycle)

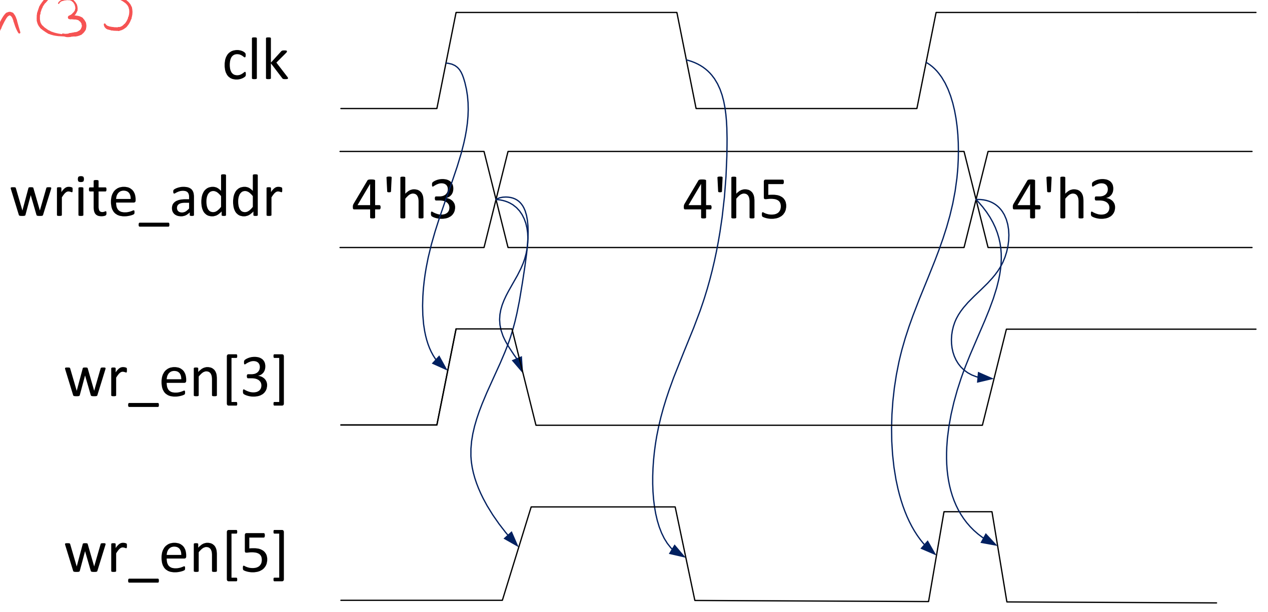
$$T_{clk-out} = T_{clk-wr\_addr} + T_{wr-addr-wl} + T_{DQ-slave} + T_{bl-drive}$$

- Note: This eqn. applies for when you have a flop-based capture of the write address

ELECTRICAL & COMPUTER ENGINEERING

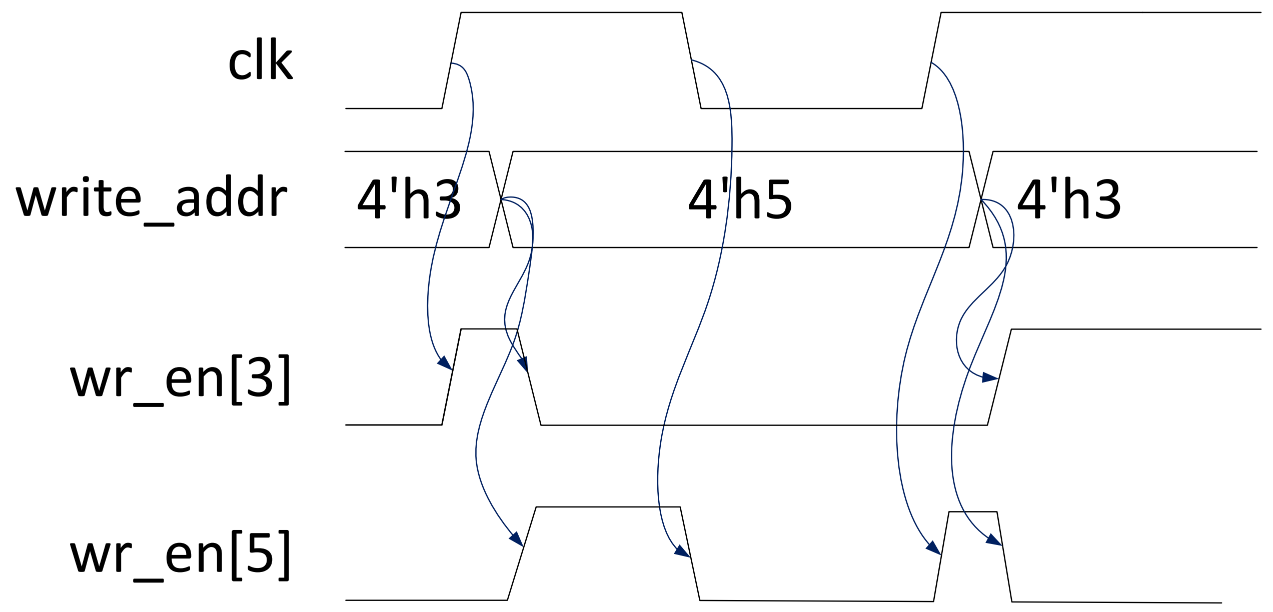
Pitfalls





- 1. What happens when slave clock arrives after master clock
- 2. Writing to multiple slaves due to glitches
  - Common problem if enabling logic is not properly done
- Run timing verification with the right state transitions for worst case delay and slew

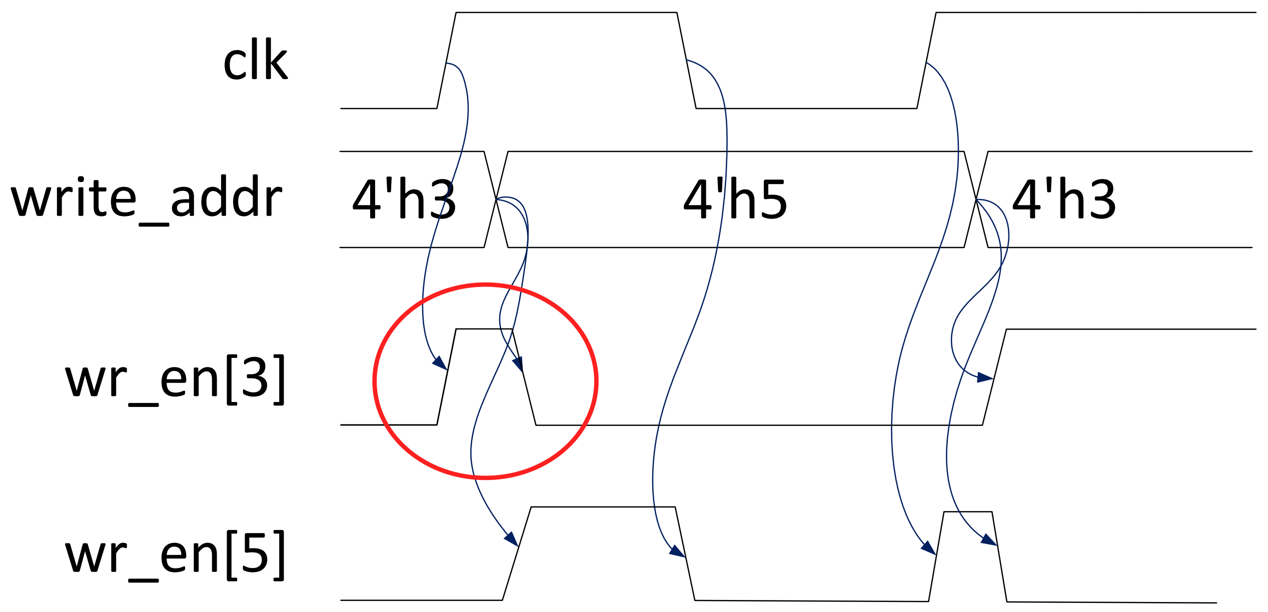
Pitfalls



Why must the write be synchronous?

- 1. What happens when slave clock arrives after master clock
- 2. Writing to multiple slaves due to glitches
  - Common problem if enabling logic is not properly done
- Run timing verification with the right state transitions for worst case delay and slew

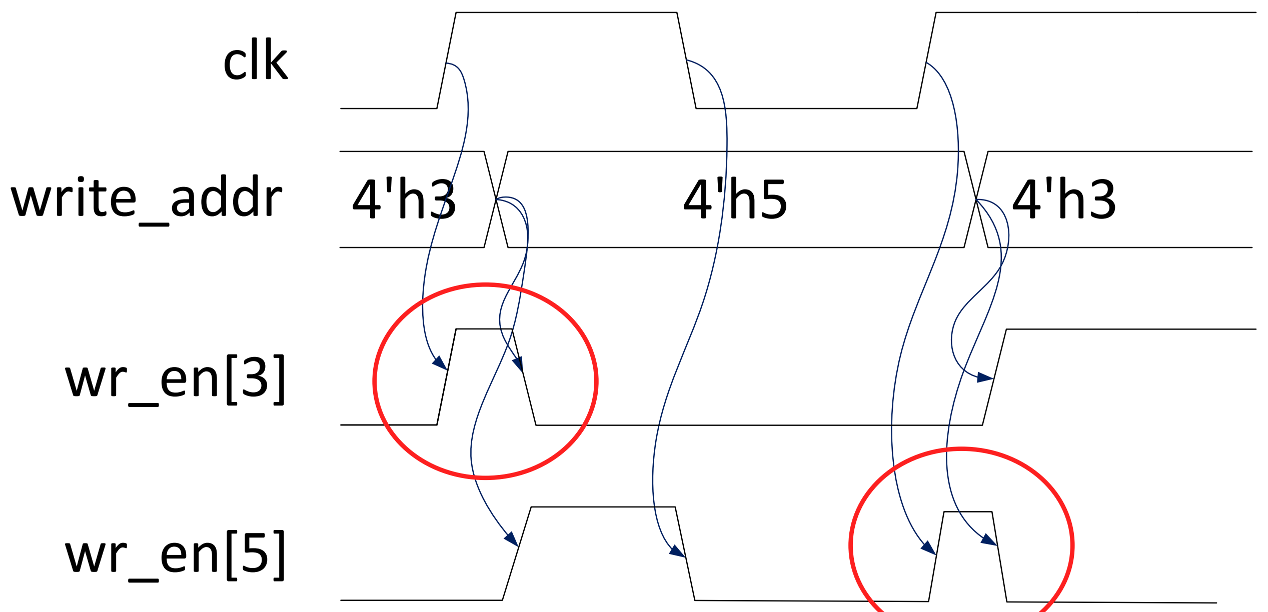
Pitfalls



Why must the write be synchronous?

- 1. What happens when slave clock arrives after master clock
- 2. Writing to multiple slaves due to glitches
  - Common problem if enabling logic is not properly done
- Run timing verification with the right state transitions for worst case delay and slew

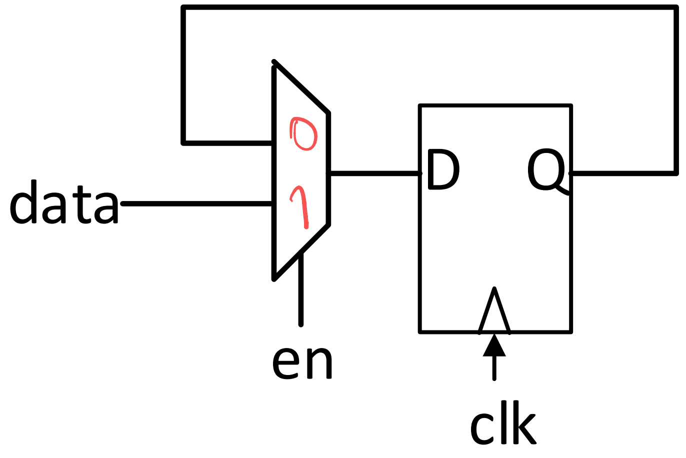
Pitfalls

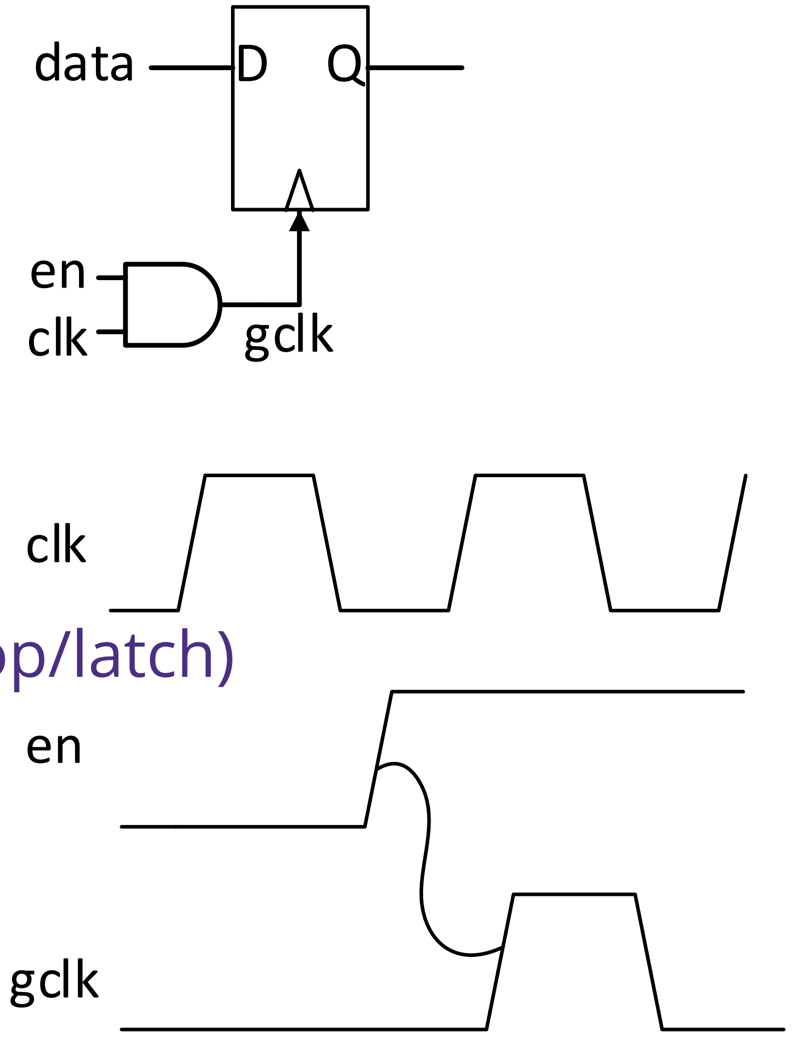


Why must the write be synchronous?

- 1. What happens when slave clock arrives after master clock
- 2. Writing to multiple slaves due to glitches
  - Common problem if enabling logic is not properly done
- Run timing verification with the right state transitions for worst case delay and slew

Related Basics : Clock-Gating

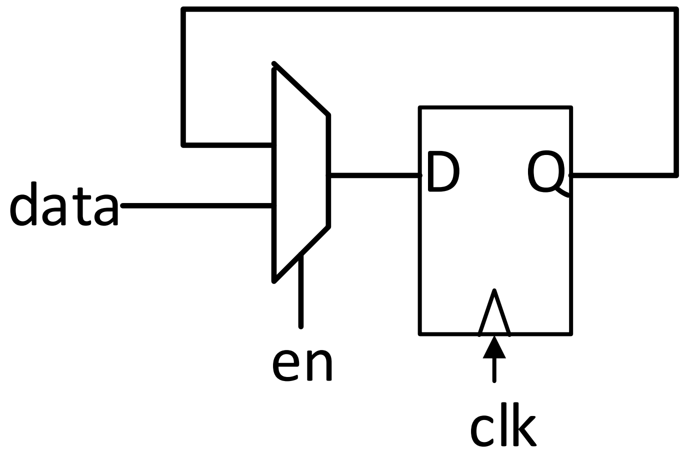


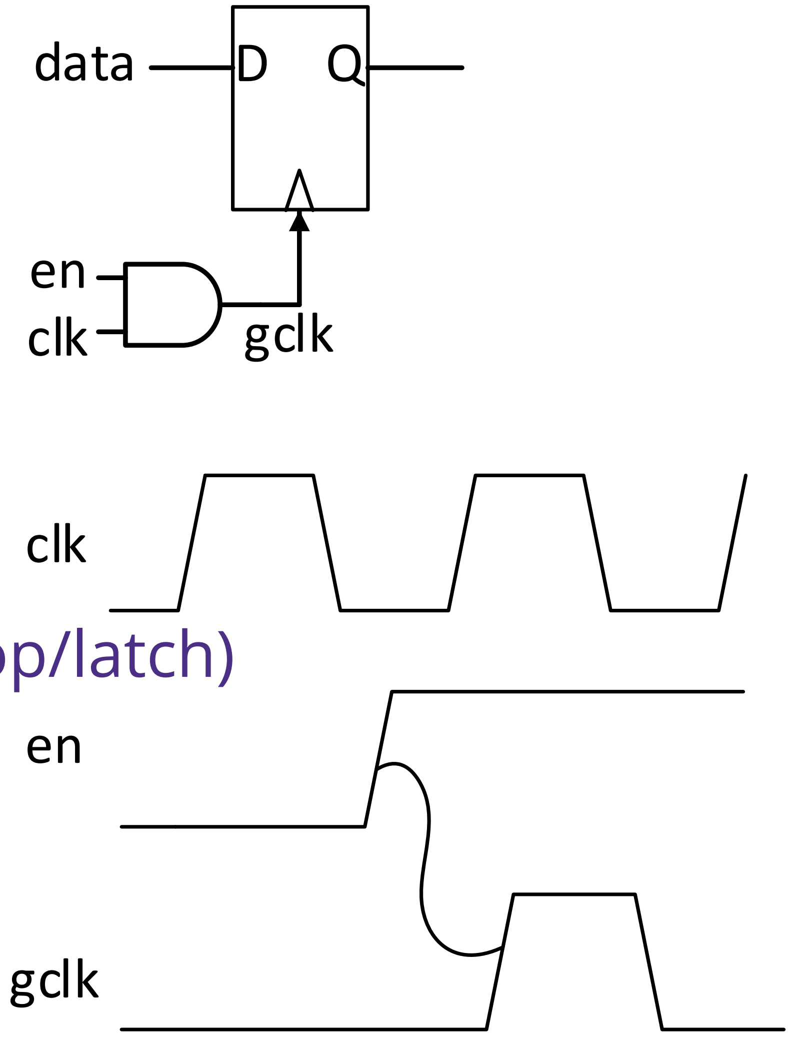


- Conditionally capture state into a timing element (flop/latch)
  - Reduce power

ELECTRICAL & COMPUTER ENGINEERING

Related Basics : Clock-Gating

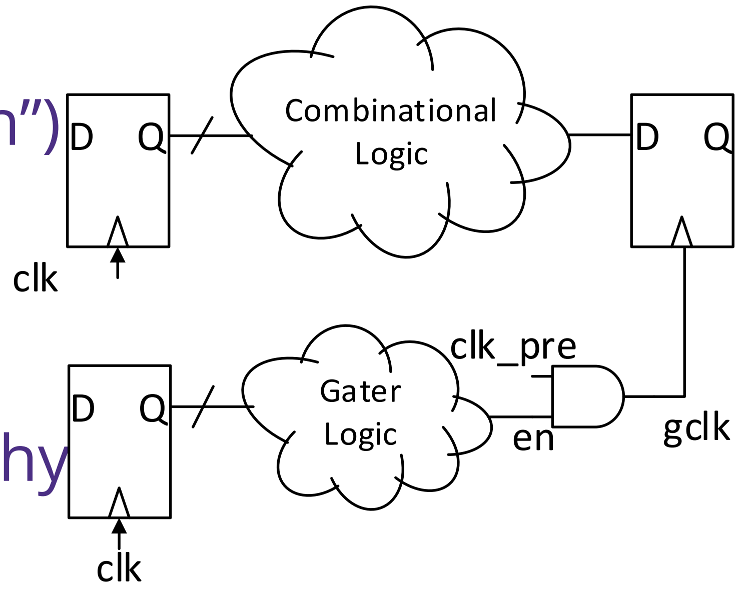


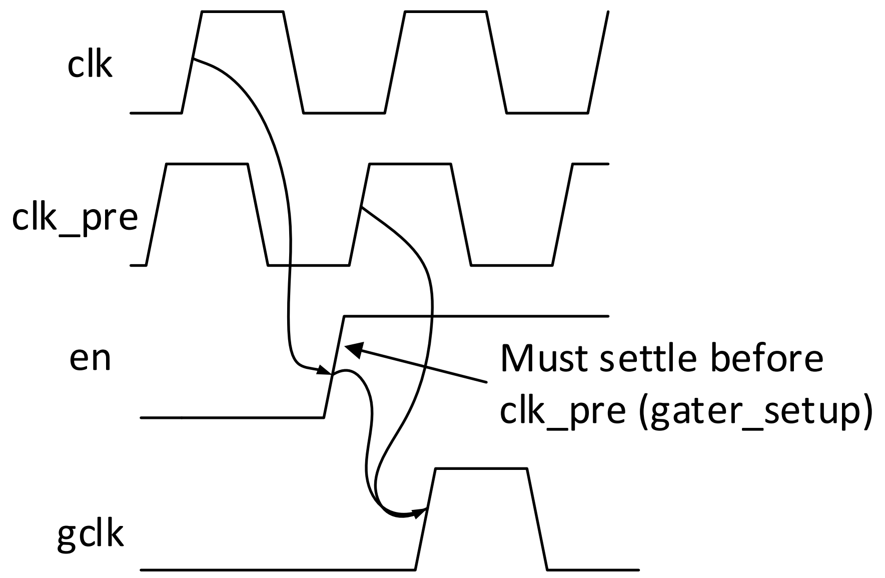


- Conditionally capture state into a timing element (flop/latch)
  - Reduce power
  - Potential area savings (if multiple flops are involved)

System-context

- State machine generates enable ("en")
- Clk, or clk_pre gated by en
  - Depends on context
  - You will end up using clk
- En must settle before clk_pre →1 (why)
  - Gater-setup

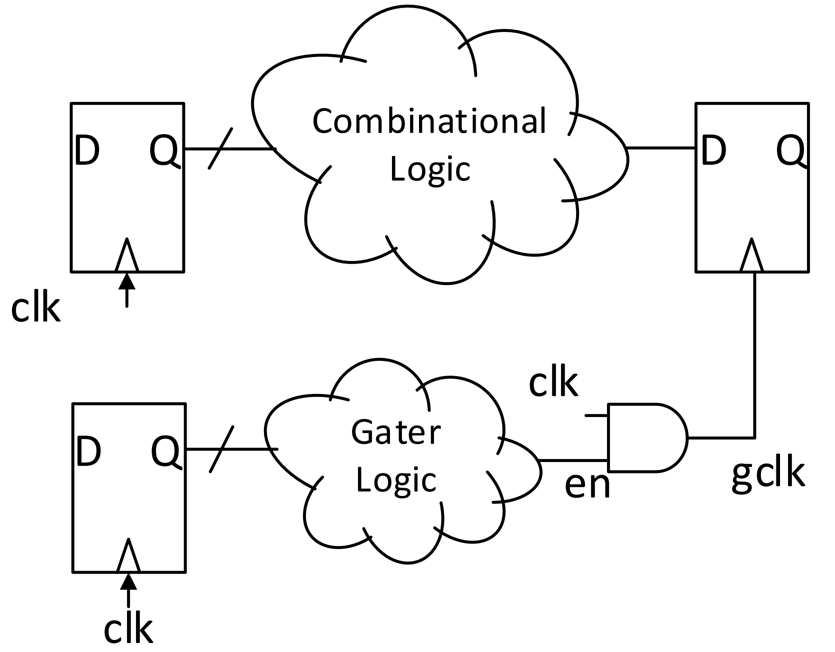


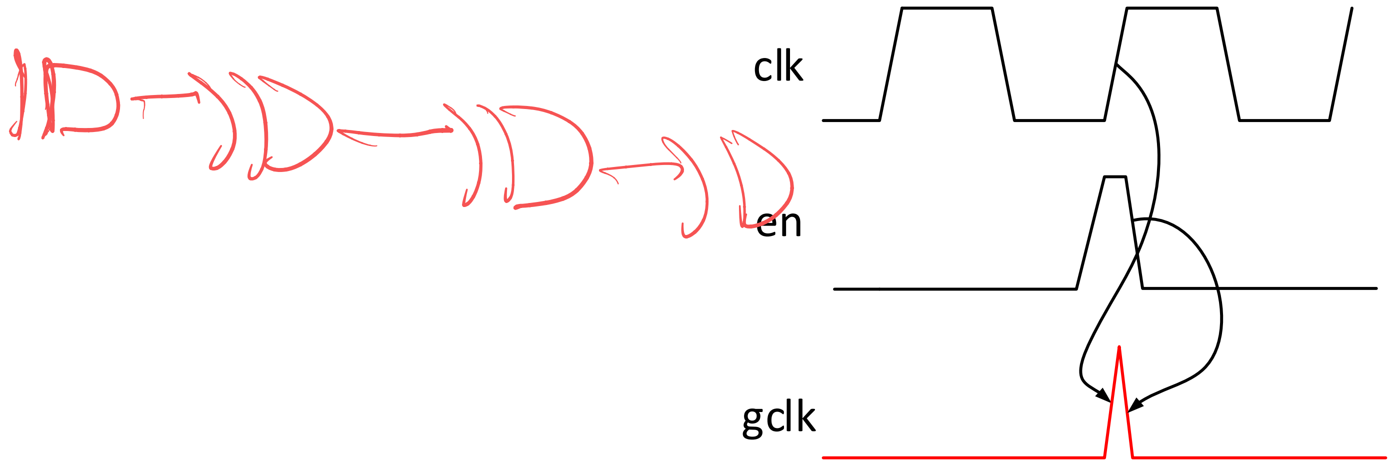


ELECTRICAL & COMPUTER ENGINEERING

Avoiding Glitches

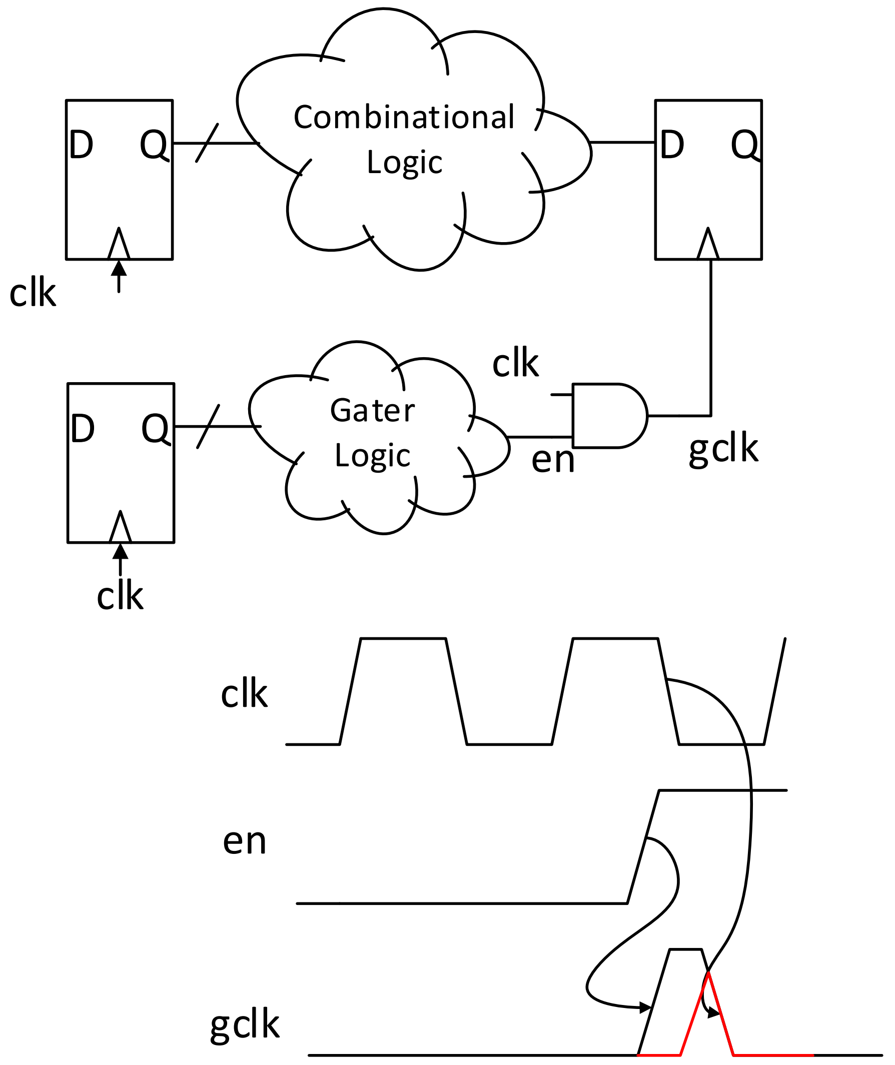
- En logic cone sees delay spread
- Early "en" evaluation causes runt pulse
  - Early de-assertion causes runt pulse





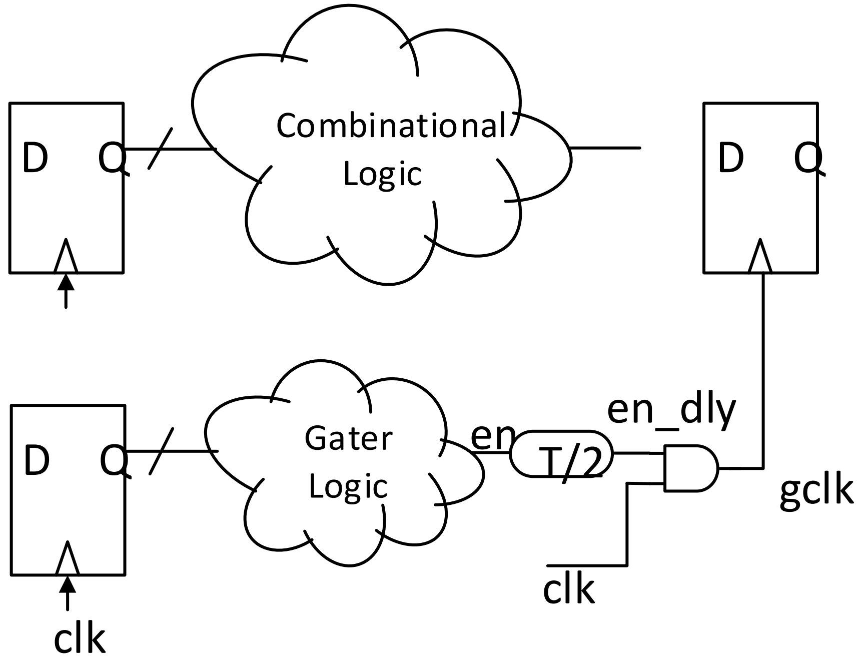
Avoiding Glitches

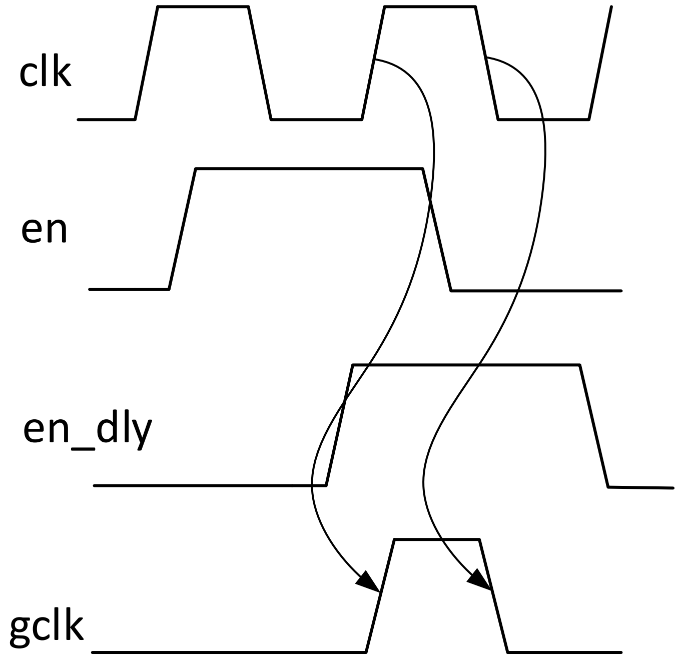
- En logic cone sees delay spread
- Early "en" evaluation causes runt pulse
  - Early de-assertion causes runt pulse
  - Early assertion causes premature clock pulse or premature runt pulse



Avoiding Glitches

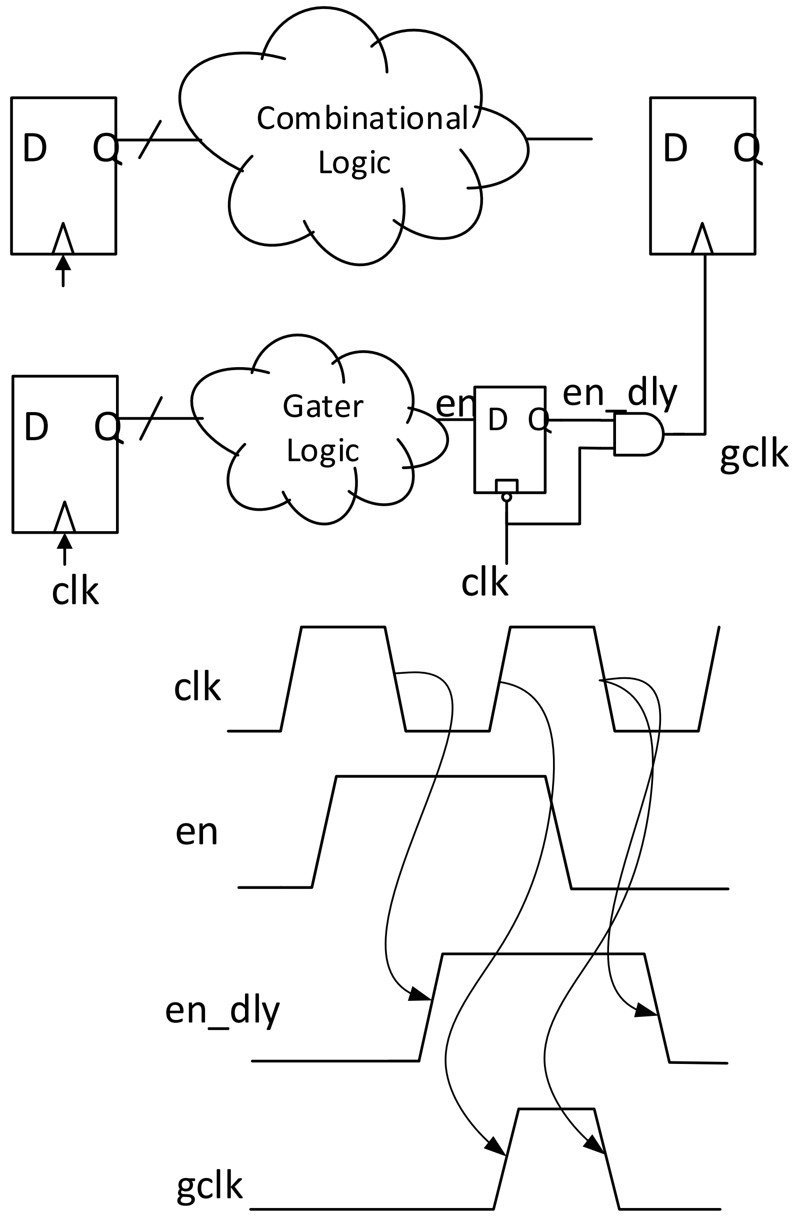
- En logic cone sees delay spread
- Early "en" evaluation causes runt pulse
  - Early de-assertion causes runt pulse
  - Early assertion causes premature clock pulse or premature runt pulse
- Inserting T/2 delay guarantees enable for current cycle will not transition until clk goes low.
- Strictly speaking, correct BUT
  - Adds to critical path!!
  - Dissipative, variable and inefficient





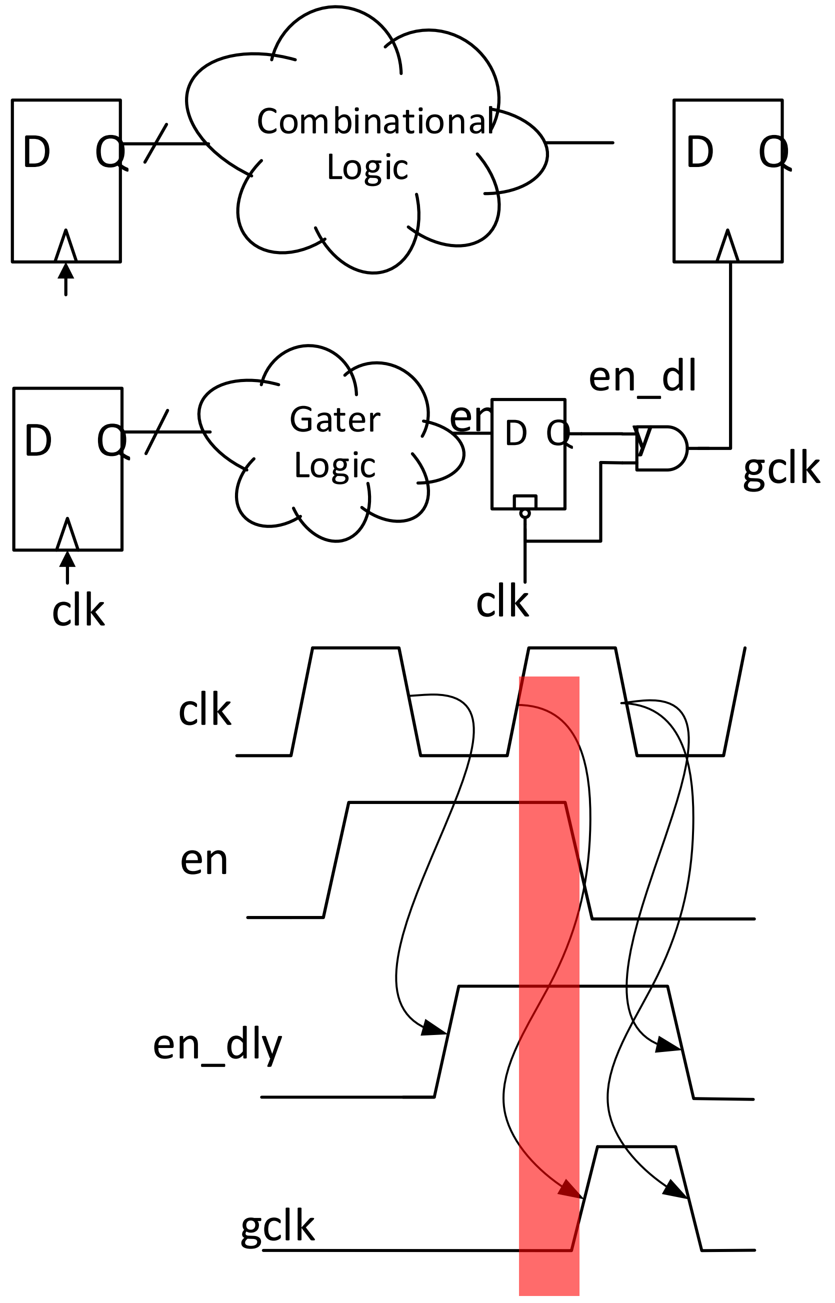
Avoiding Glitches

- En logic cone sees delay spread
- Early "en" evaluation causes runt pulse
    - Early de-assertion causes runt pulse
    - Early assertion causes premature clock pulse or premature runt pulse
- Inserting T/2 delay guarantees enable for current cycle will not transition until clk goes low.
- Strictly speaking, correct BUT
    - Adds to critical path!!
    - Dissipative, variable and inefficient
    - (Technically, you could design for min_delay > T/2 instead..)
- Instead,Use a latch!!



One more thing....

- Clock skew often reported for global clocks
  - Good way to control skew BUT most timing elements today are driven by gclk
  - gclk skew ultimately matters!
  - Skew between clk and gclk is important when either launch xor capture is ungated
    - If capture is flip flop



Breakout Session:

- Problem. Construct a clock gating methodology for gating a state machine triggered by the negative edge of the clock.