CS 61C:
Great Ideas in Computer Architecture

Lecture 10: *Sequential Logic*

Bernhard Boser & Randy Katz

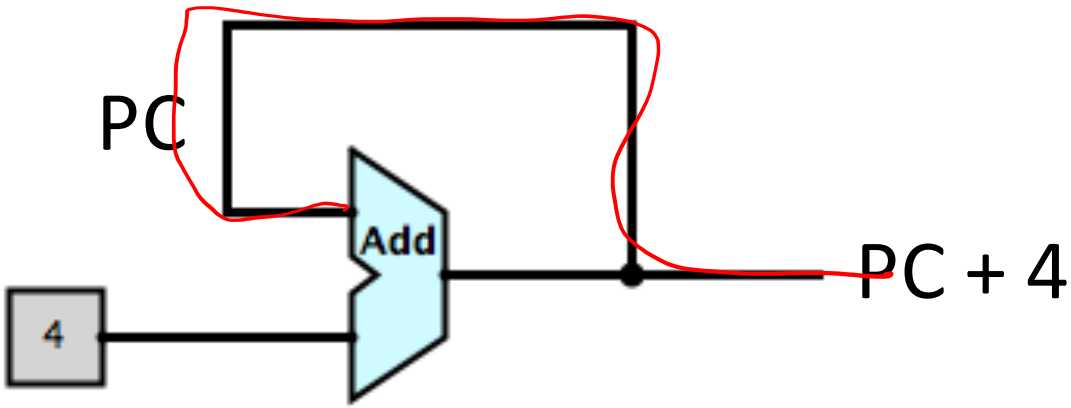
http://inst.eecs.berkeley.edu/~cs61c

Agenda

- **Sequential Circuits**
  - **Example: program counter**
  - **Latch (Flip-Flop)**
- Finite State Machine
- Memory
  - Register file
  - Data memory
- Logic Delay
  - Example: adder
- And in Conclusion, …

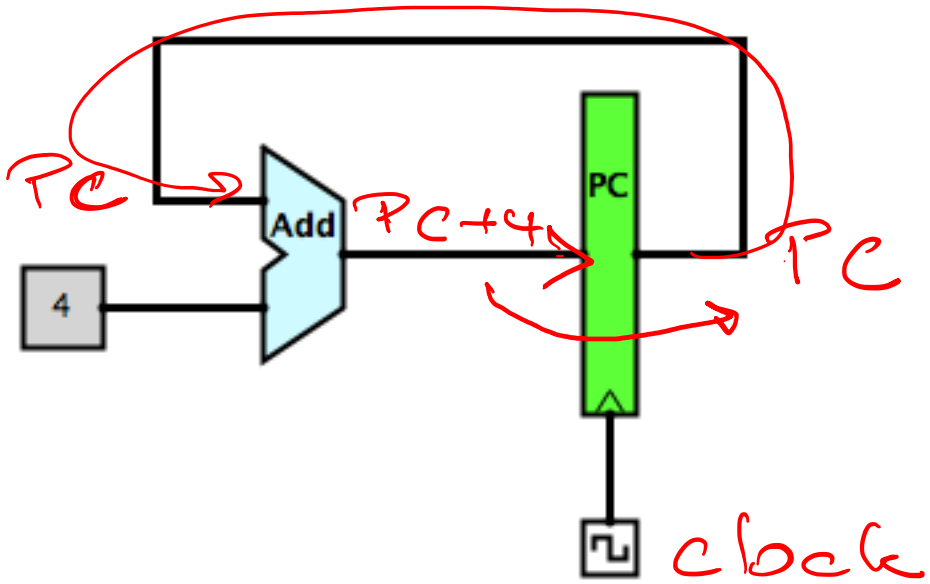
Program Counter: First Design

- Program Counter "PC"
  - Instruction address
  - Next PC: add 4 to current value
  - Let's try …



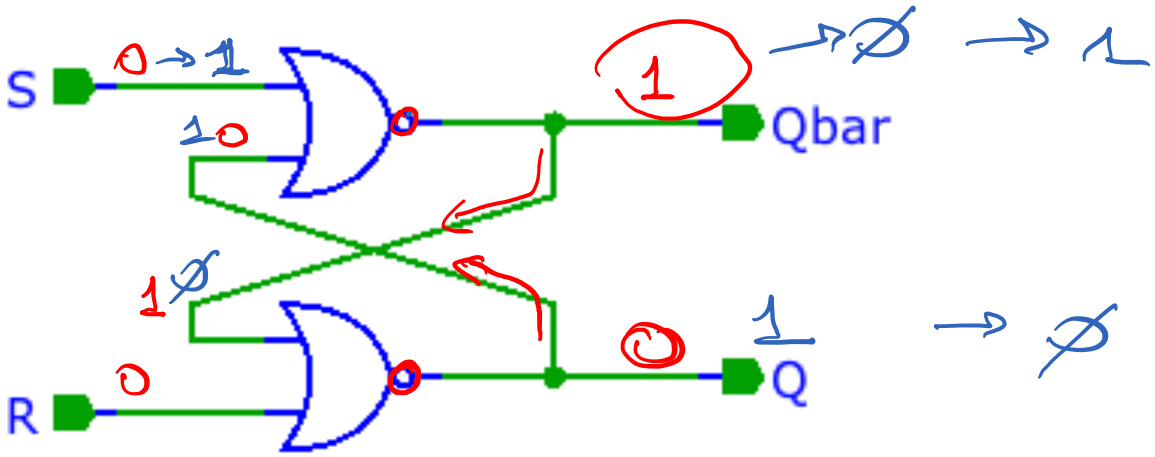
- Something is not quite right:
  - PC and PC+4 simutaneously on same wire???

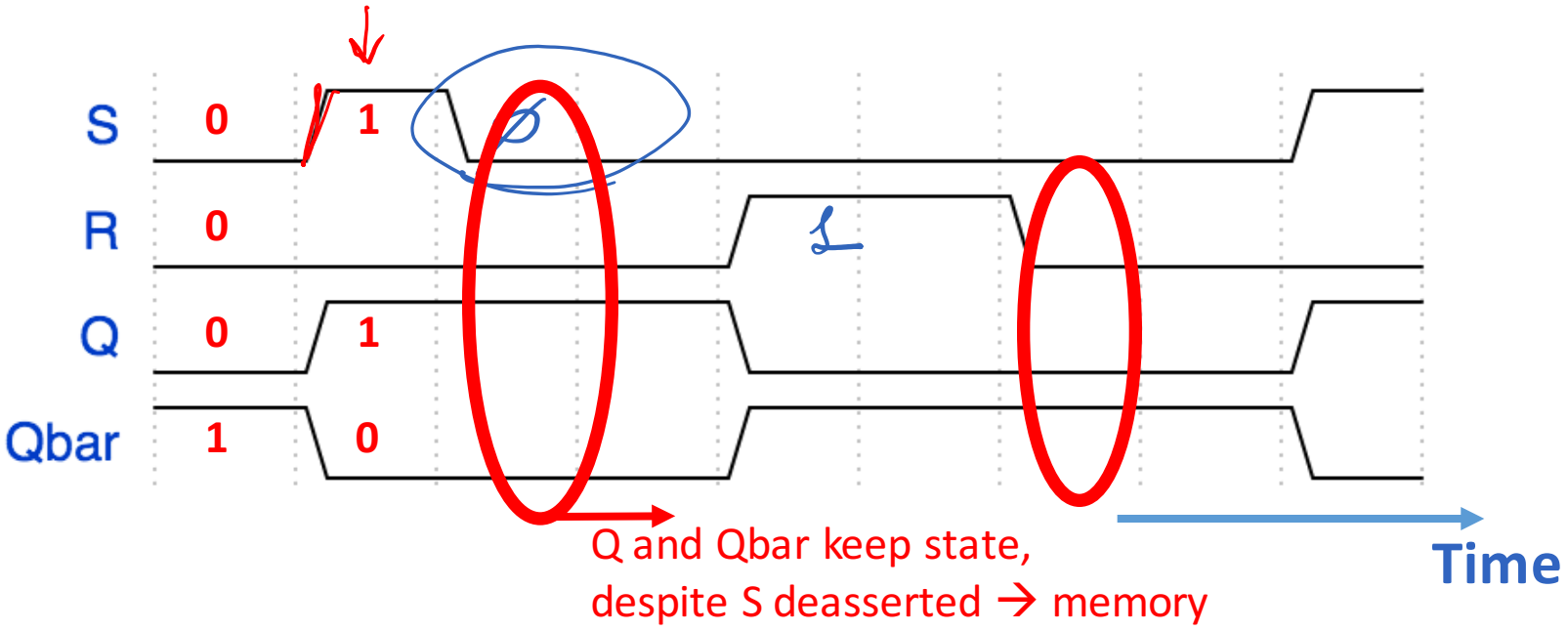
Fix



- Memory element breaks the feedback loop
- How does it work?

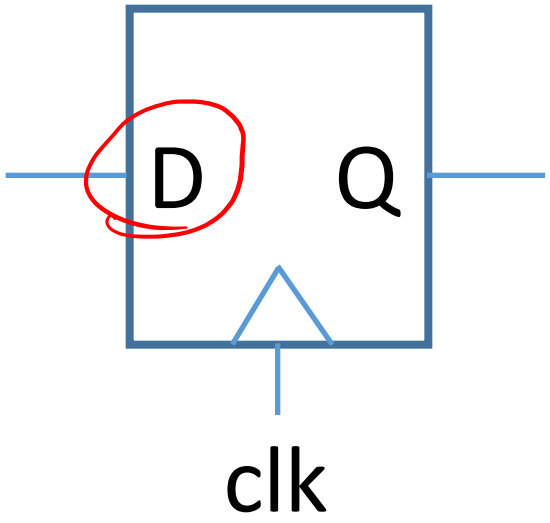
RS Flip-Flop





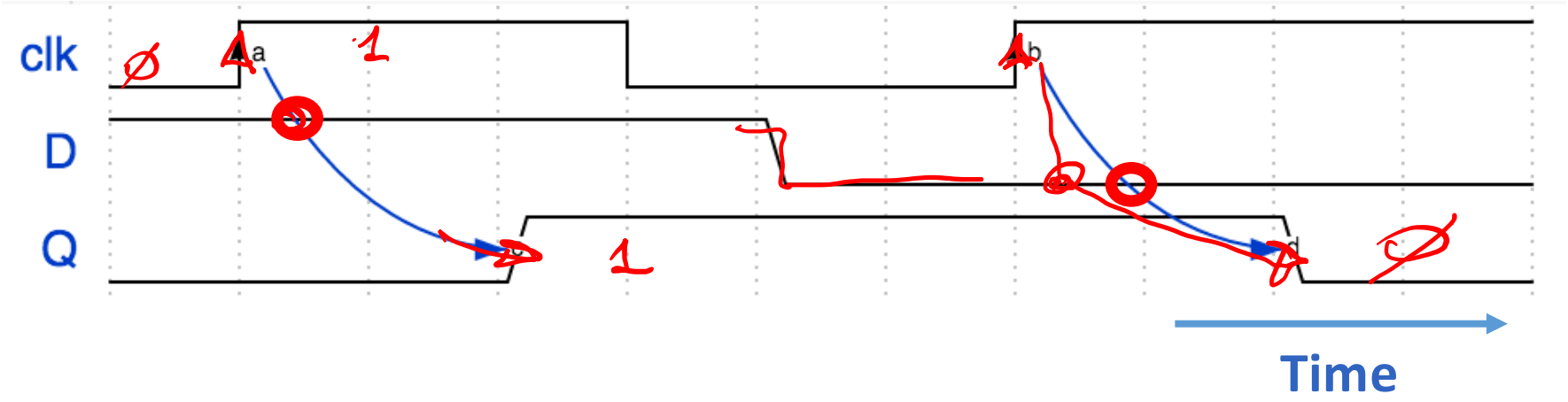
Q and Qbar keep state,
despite S deasserted → memory

(Positive) Edge Triggered Flip-Flop

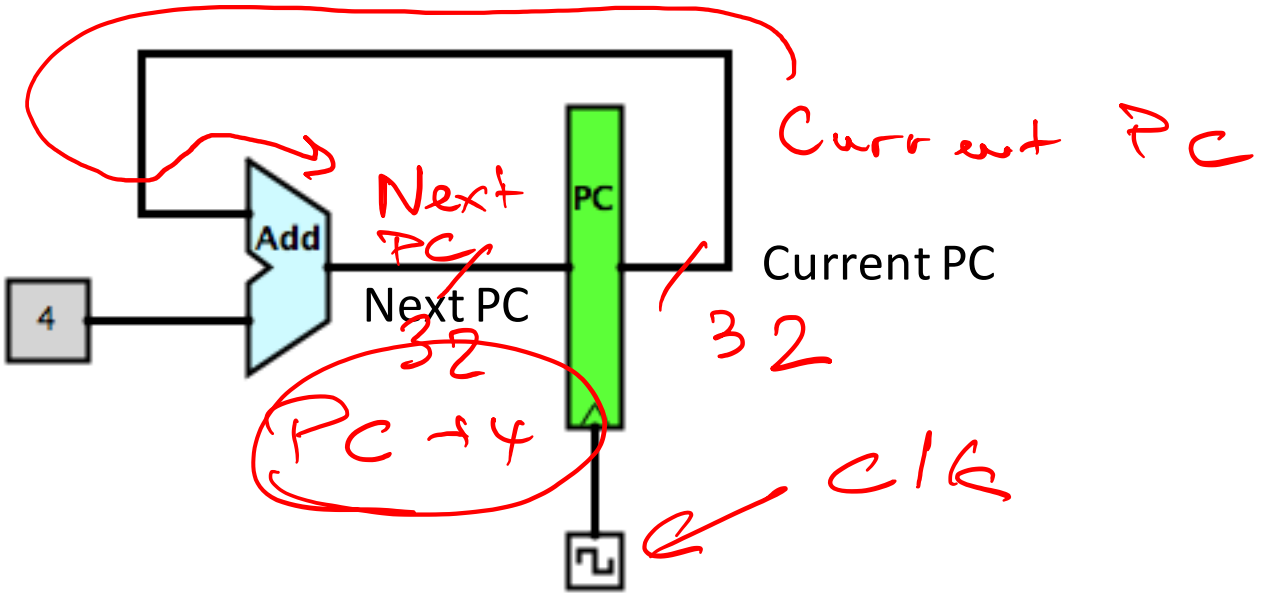


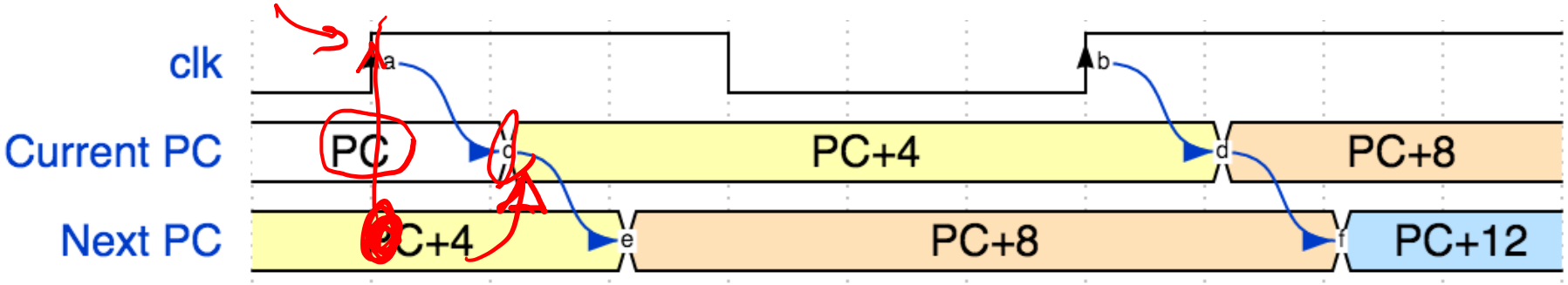
(see P&H B.8)

Clock "clk"

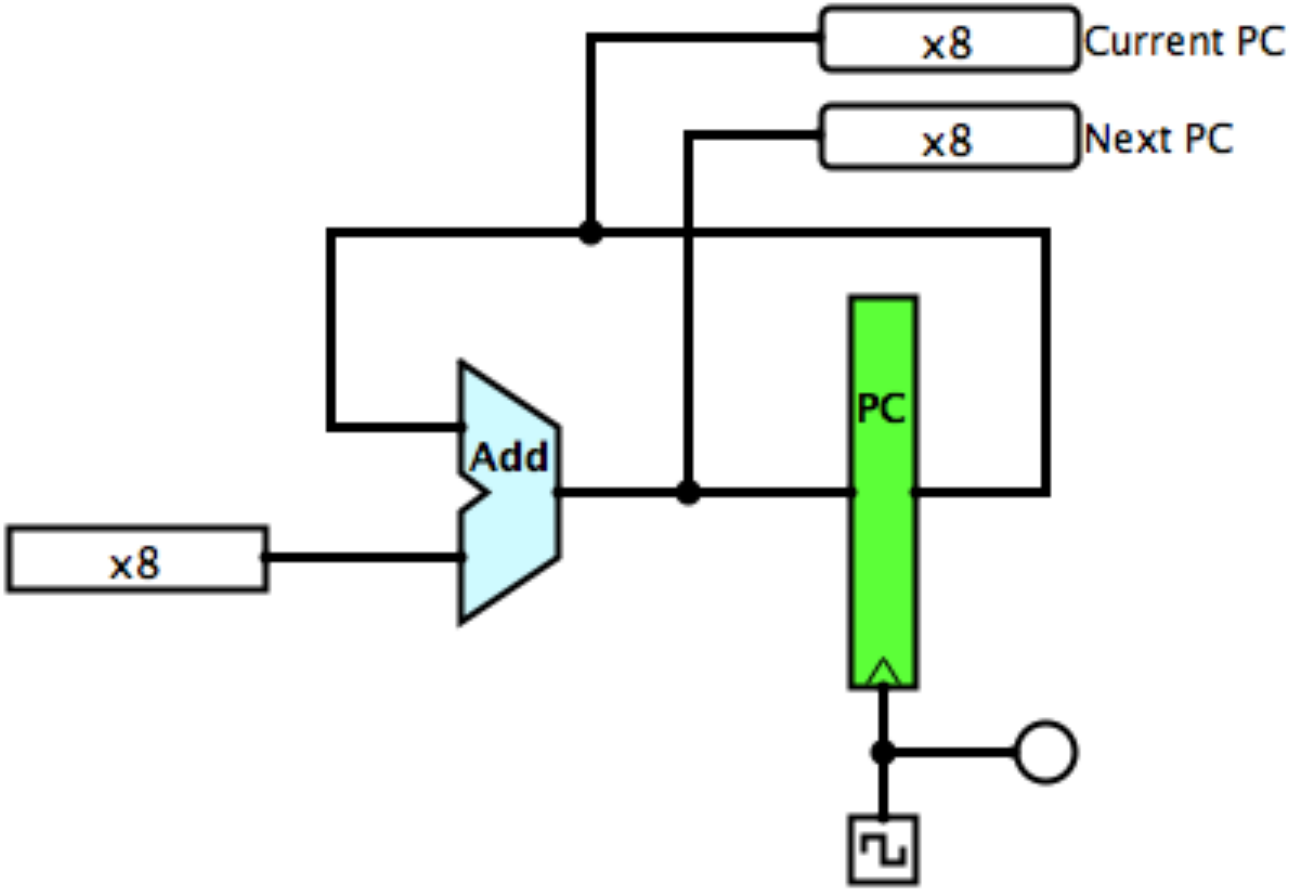


Program Counter: Improved Design



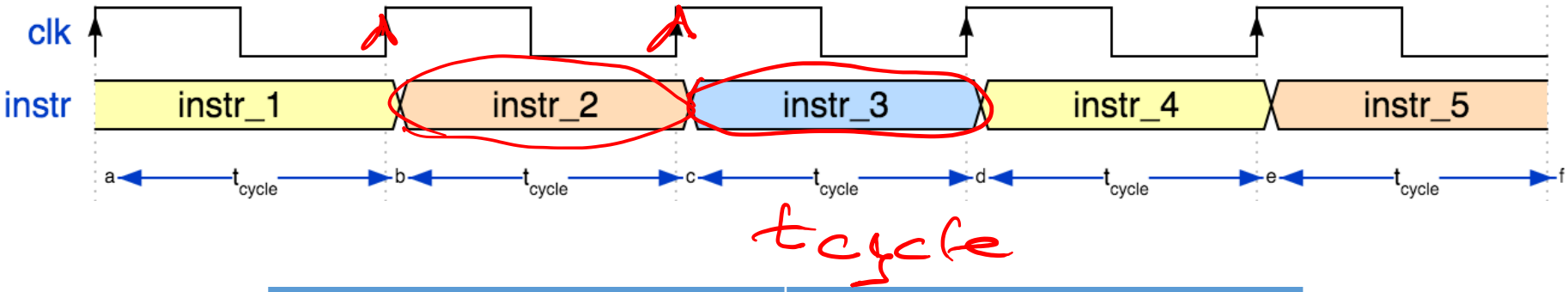


Program Counter Demo



Clock

$$f_s = 1 / t_{cycle}$$



tcycle

Clock frequency f_s	Period $t_{cycle} = 1/f_s$
CPU 2.5 GHz	400 ps
Drum/hart beat, 1Hz	1 s

Light facts …

Physics: Information cannot travel faster than light

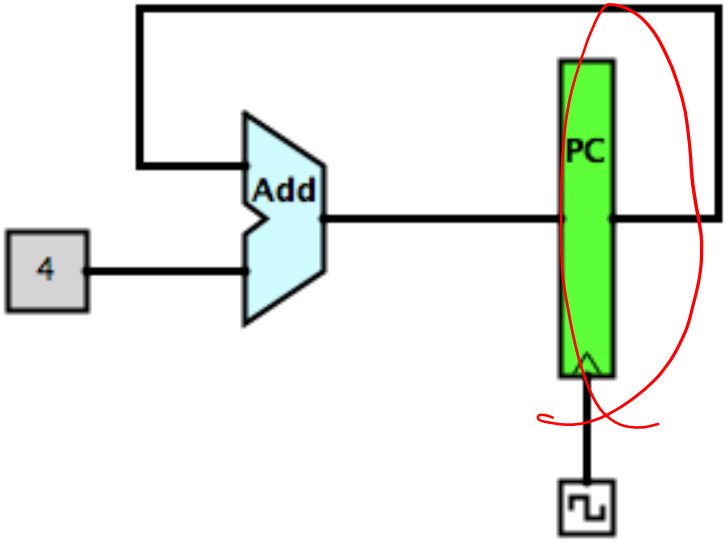
- Speed of light in vacuum: 300,000 km/s
- In computer chip: ~ 150,000 km/s
- Distance traveled in 400 ps: 60 mm
- Size of computer chip: 15 x 15 mm^2

Light travels only twice back and forth chip in one clock cycle!

→ Do not place ALU and memory on opposite sides!

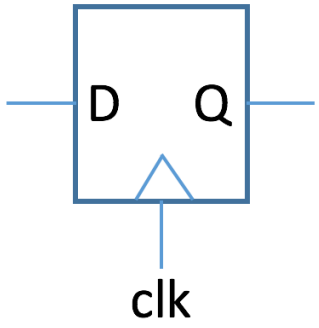
Maximum Clock Speed

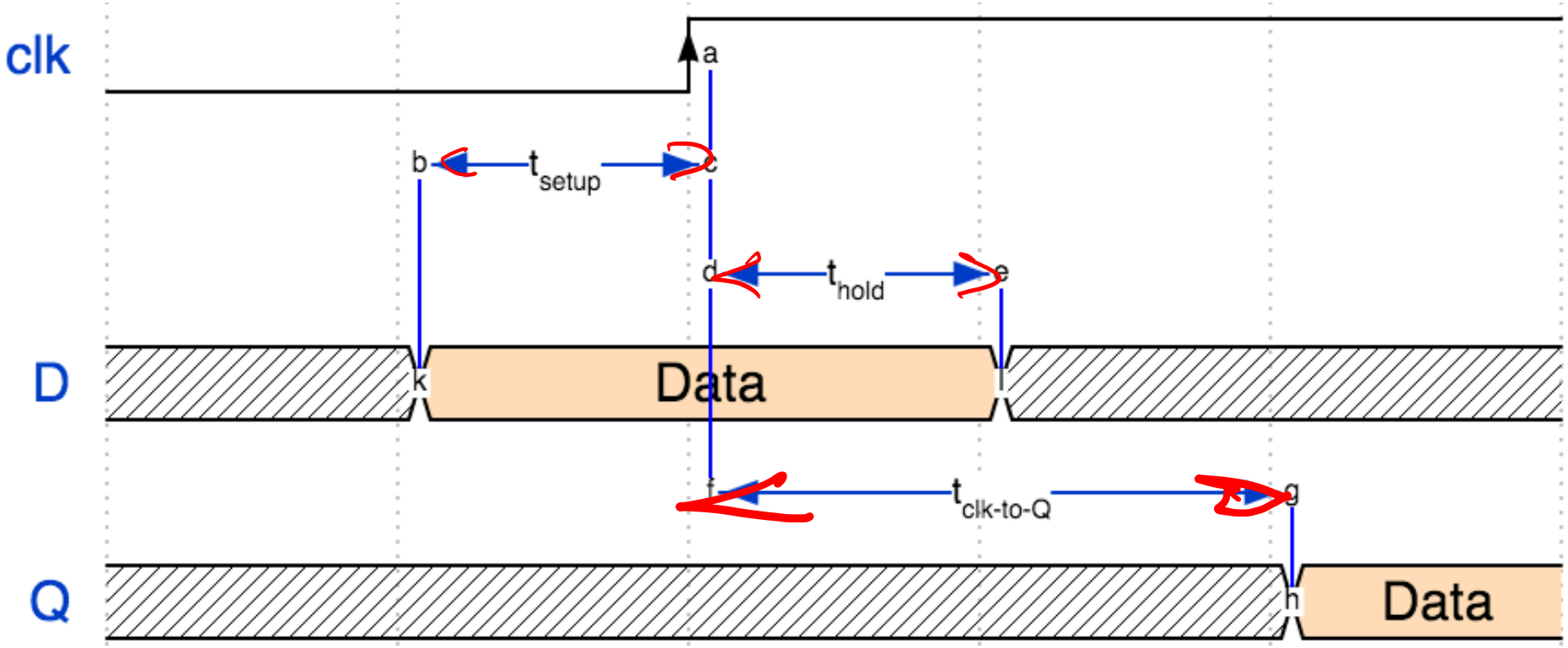
- How fast can the PC circuit operate?
  - I.e. what is the maximum clock frequency?



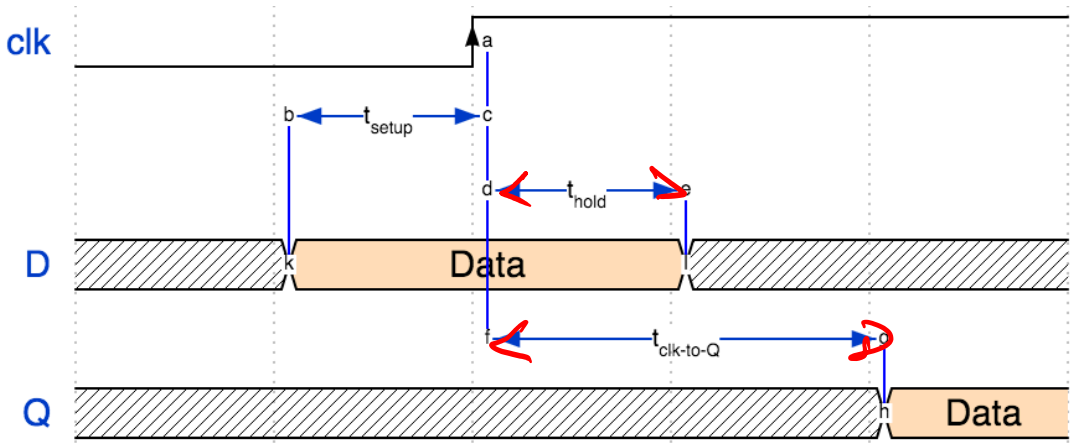
- Let's look at the timing requirements of each part
  - Let's start with the flip-flop

Flip-flop Timing



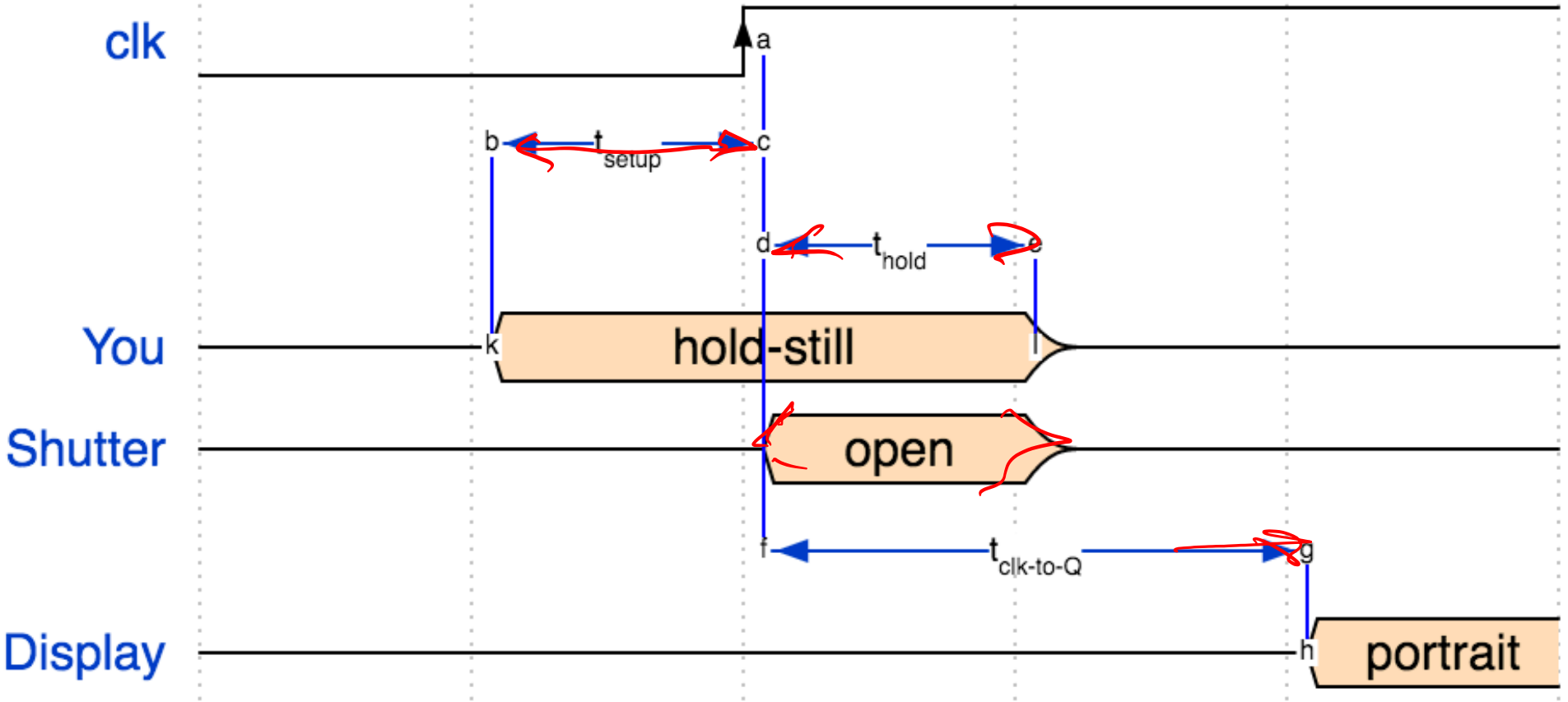


Flip-flop Timing

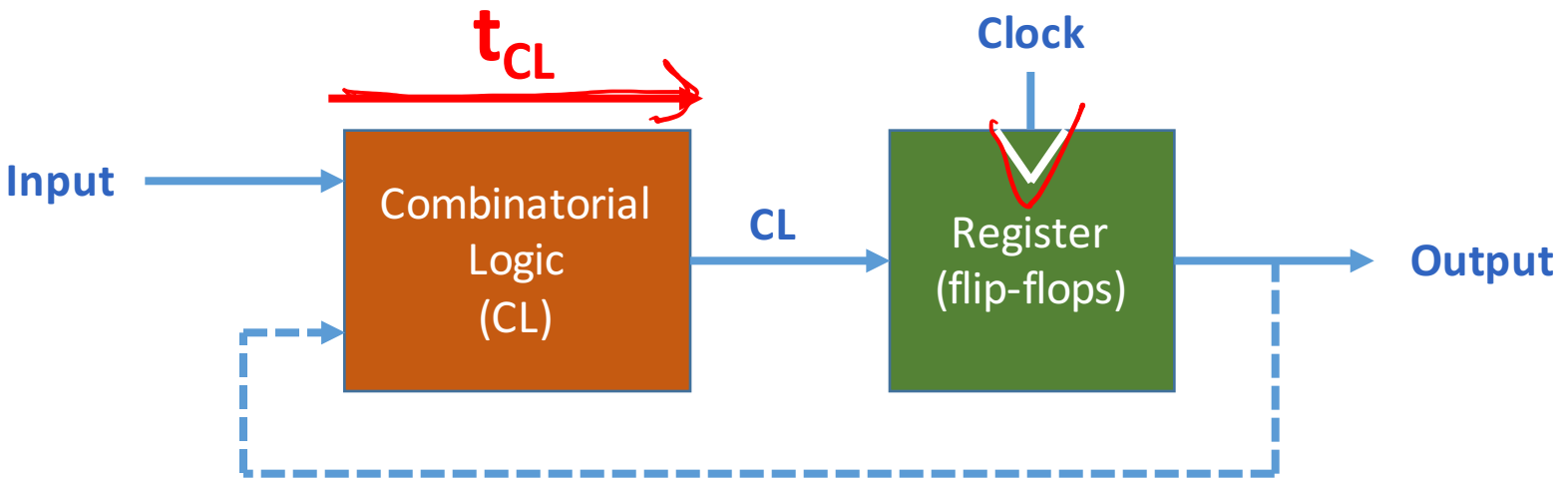


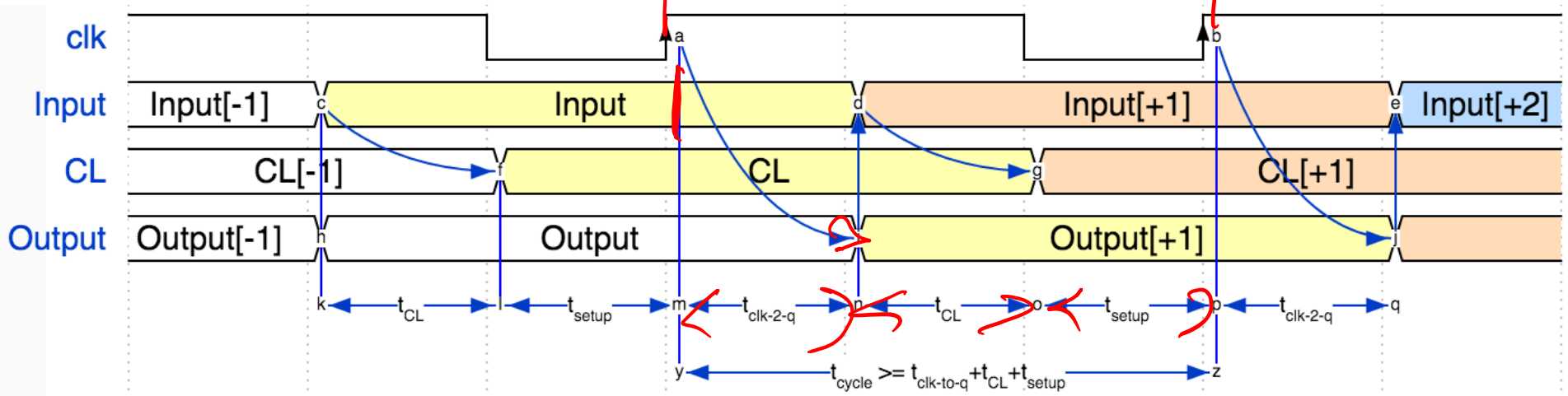
Time	Description
t_{setup}	time during which D must not change *before* positive clock edge
t_{hold}	time during which D must not change *after* positive clock edge
$t_{clk\text{-}to\text{-}Q}$	delay after positive clock edge after which D appears at Q output

Camera Analogy

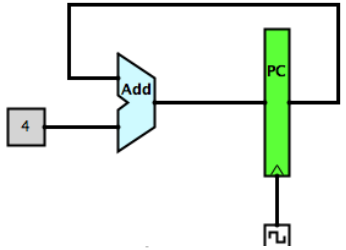


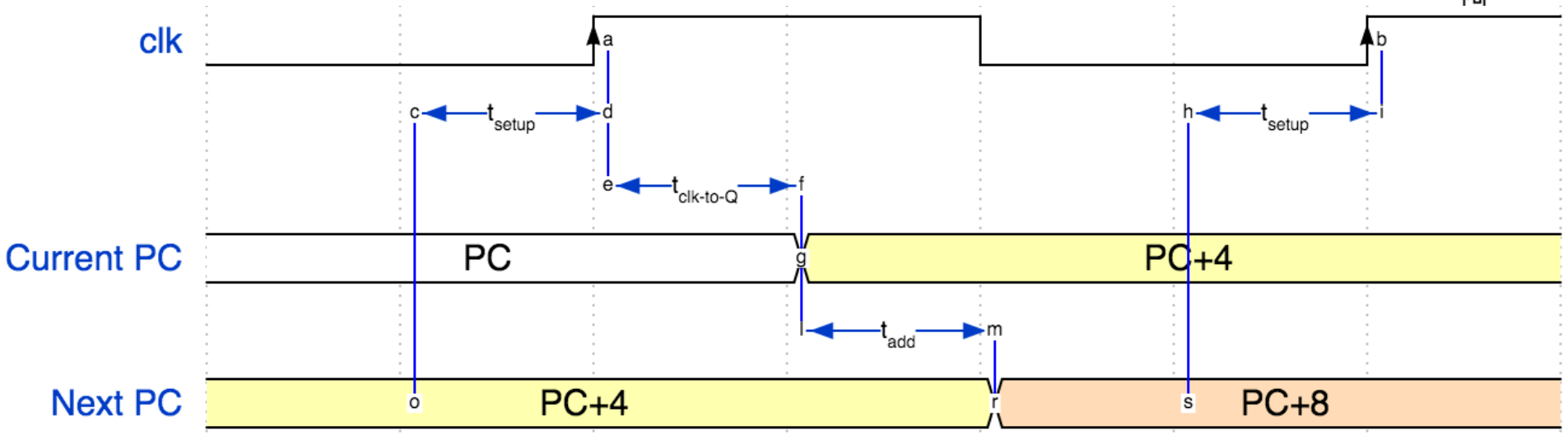
Maximum Clock Speed





Maximum PC Clock Speed





- Minimum cycle time: $t_{min} = t_{setup} + t_{add} + t_{clk-to-Q}$
- Maximum clock rate: $f_{clk-max} = \frac{1}{t_{min}}$

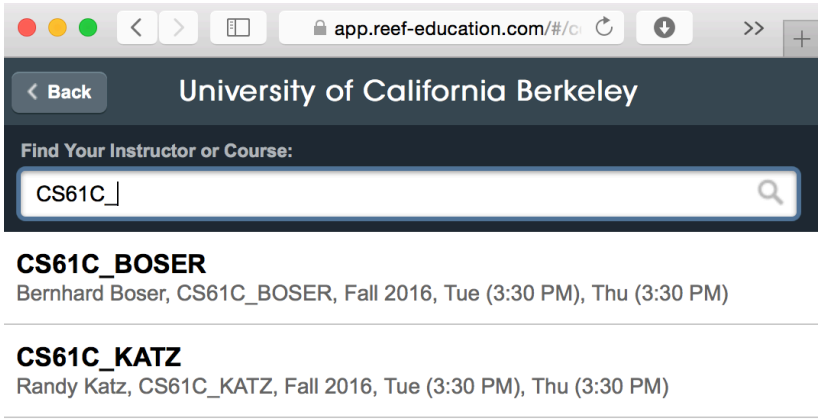
Example: $t_{setup} \geq 15ps$, $t_{add} = 75ps$, $t_{clk\text{-}toQ} \geq 10ps$

$t_{min} \geq 100ps$

$f_{clk\text{-}max} \leq 10GHz$

iClicker Registration

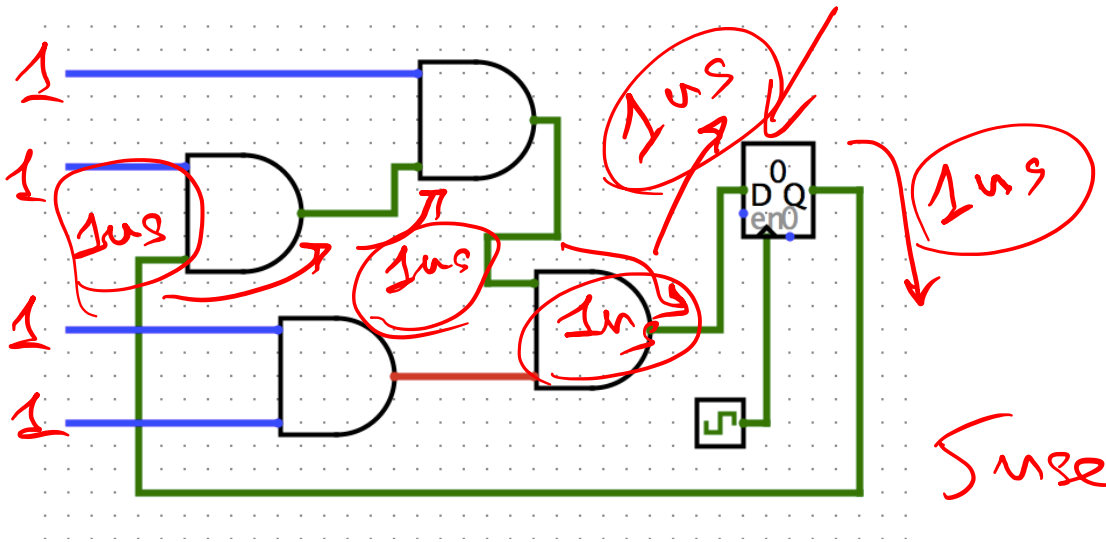
- Physical i>Clicker
  - https://www.ets.berkeley.edu/discover-services/clickers/students-getting-started
  - Register at https://bcourses.berkeley.edu, **not** REEF website!
- REFF
  - Register for *CS61C_BOSER* **and** *CS61C_KATZ* at https://reef-education.com



  - **Important**: use same ***name and SID*** for REEF registration and bcourses
    - Otherwise your scores won't be registered on bcourses

Debug on Piazza (pinned note)

Your Turn …



$t_{setup} \geq 1$ ns

$t_{hold} \geq 1$ ns

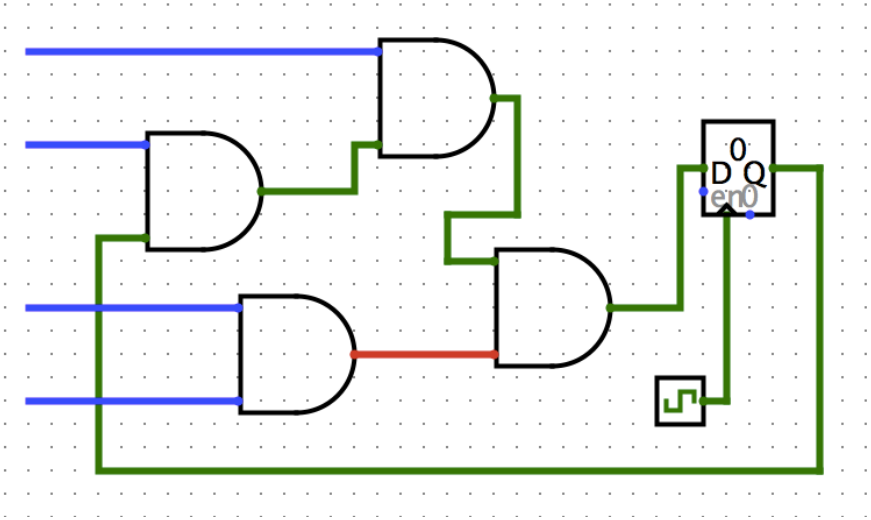
$t_{and} = 1$ ns

$t_{clk\text{-}toQ} = 1$ ns

What is maximum clock frequency?
(assume all unconnected inputs are constant)

Answer	Maxium Clock Frequency
A	5 GHz
B	200 MHz
C	500 MHz
D	1/7 GHz
E	1/6 GHz

Your Turn …



$t_{setup} \geq 1$ ns

$t_{hold} \geq 1$ ns

$t_{and} = 1$ ns

$t_{clk\text{-}toQ} = 1$ ns

What is maximum clock frequency?
(assume all unconnected inputs are constant)

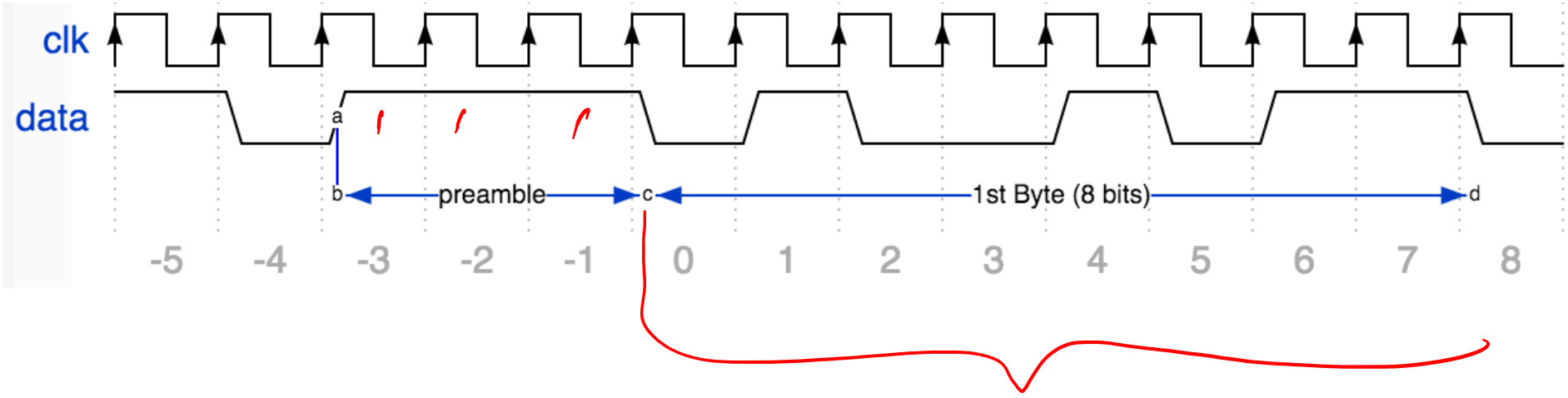
Answer	Maxium Clock Frequency
A	5 GHz
B	200 MHz
C	500 MHz
D	1/7 GHz
E	1/6 GHz

Agenda

- Sequential Circuits
  - Example: program counter
  - Latch (Flip-Flop)
- **Finite State Machine (FSM)**
- Memory
  - Register file
  - Data memory
- Logic Delay
  - Example: adder
- And in Conclusion, …

Example: Serial Communication

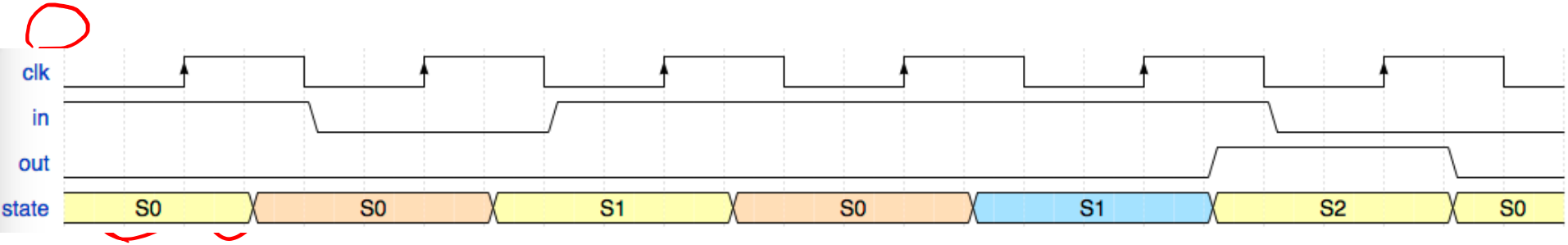
- Wifi sends data “1 bit at a time”
- How do we know where a byte “starts”?
- Send “preamble …”
  - E.g. 3 one’s in a row



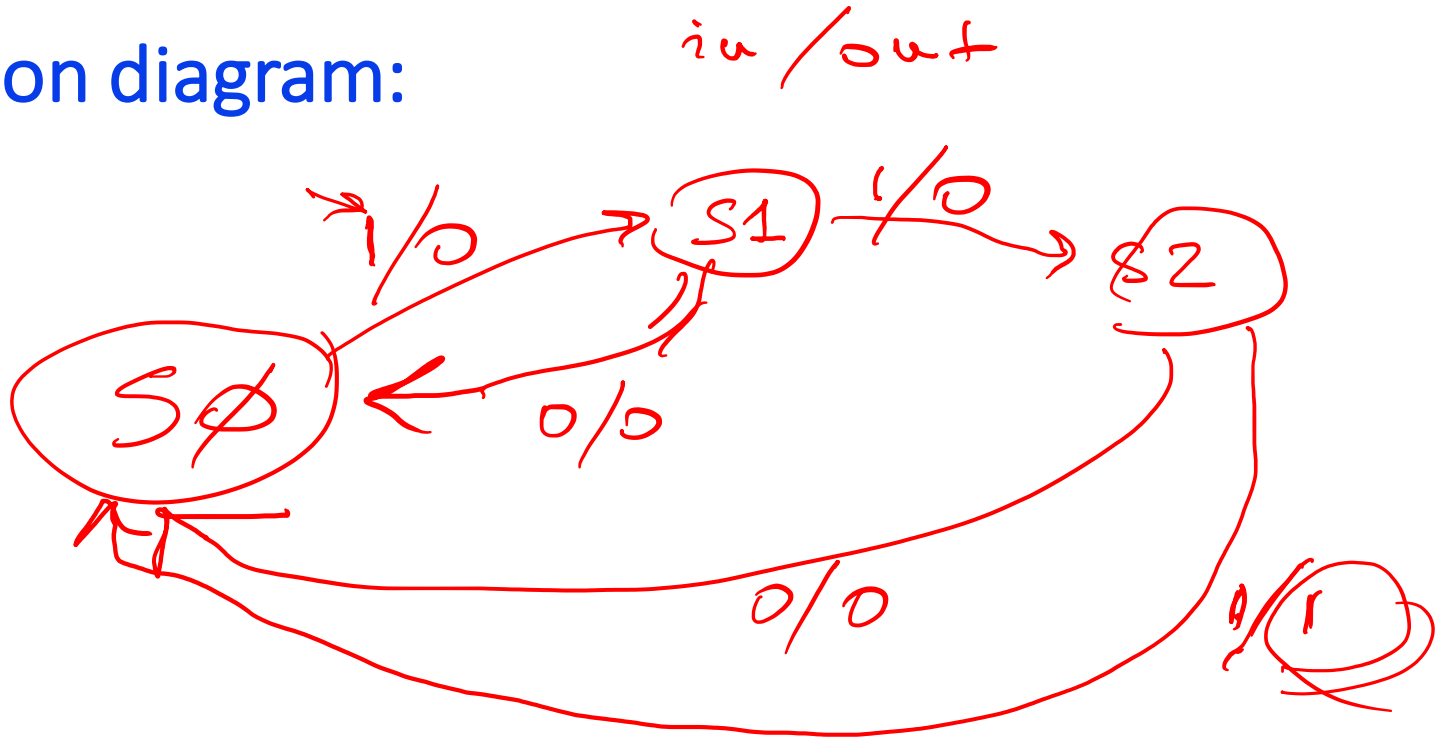
FSM* to detect 3 one's

* FSM = Finite State Machine

FSM to detect the occurrence of 3 consecutive 1's in the input.

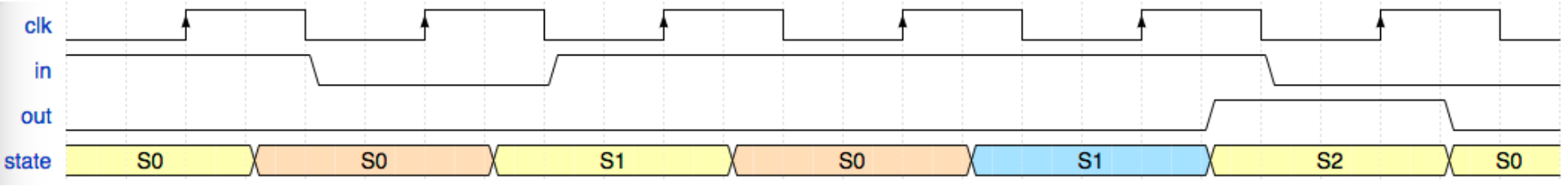


State transition diagram:

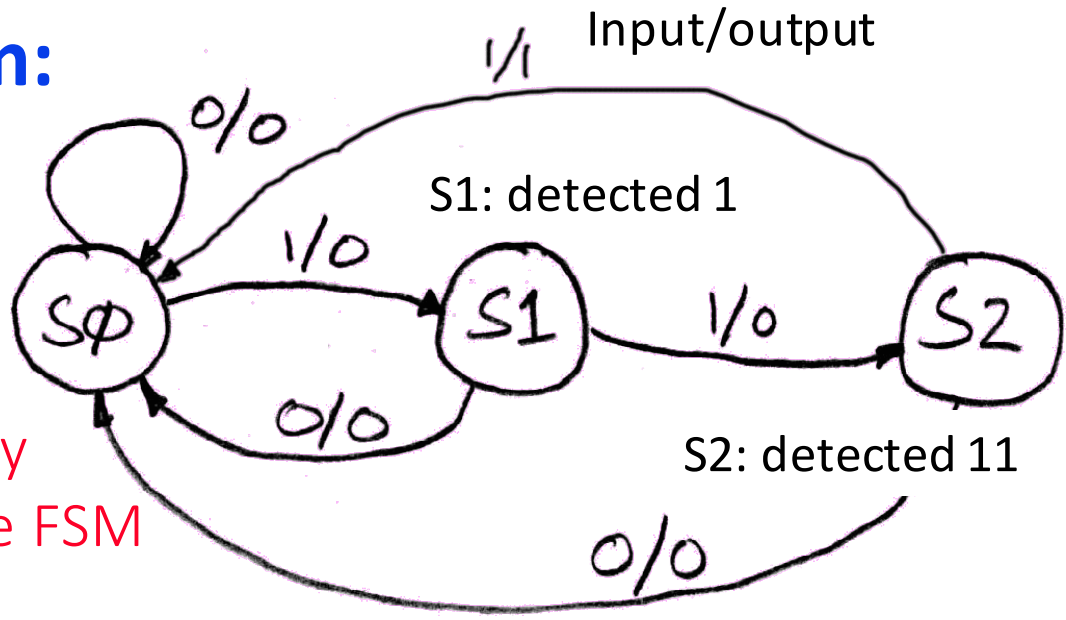


FSM to detect 3 one's

FSM to detect the occurrence of 3 consecutive 1's in the input.



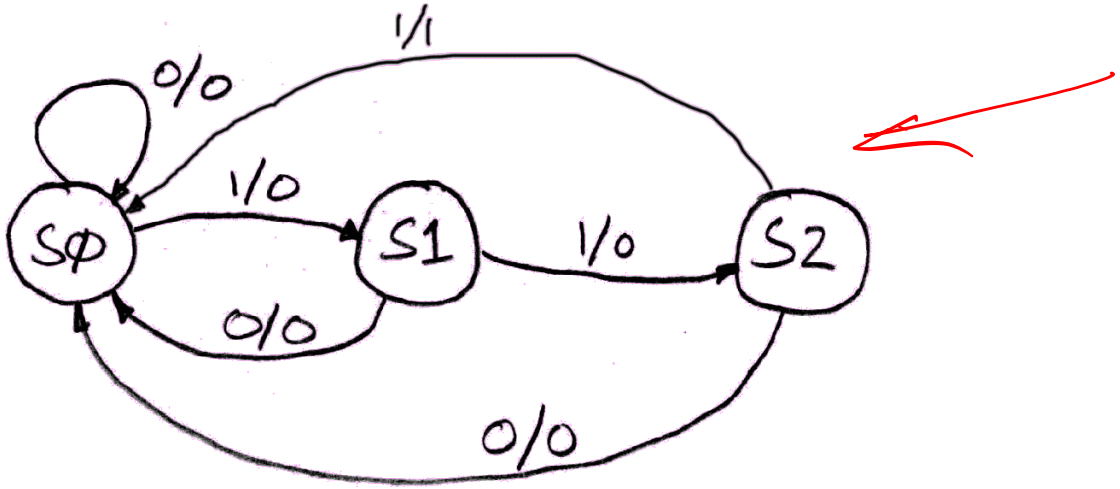
State transition diagram:



State transitions are controlled by the clock: on each clock cycle the FSM

- checks the inputs,
- transitions to a new state, and
- produces a new output …

FSM Combinatorial Logic



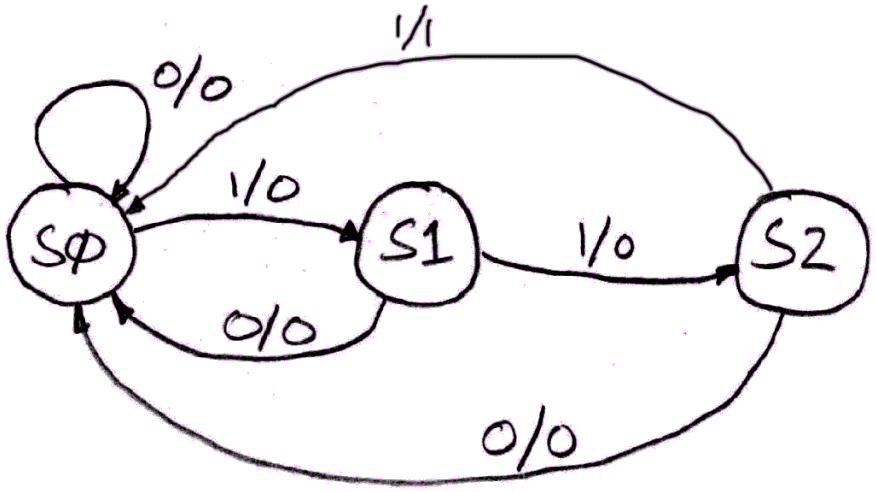
State Encoding

State	Code
S0	00
S1	01
S2	10
—	11

Truth Table

Current State	Input	Next State	Output
S0 = 00	1	S1 = 01	0
S1 = 01	1	S2 = 10	0
S2 = 10	1	S0 = 00	1
XX	0	S0 = 00	0
11	1	S0 = 00	0

FSM Combinatorial Logic



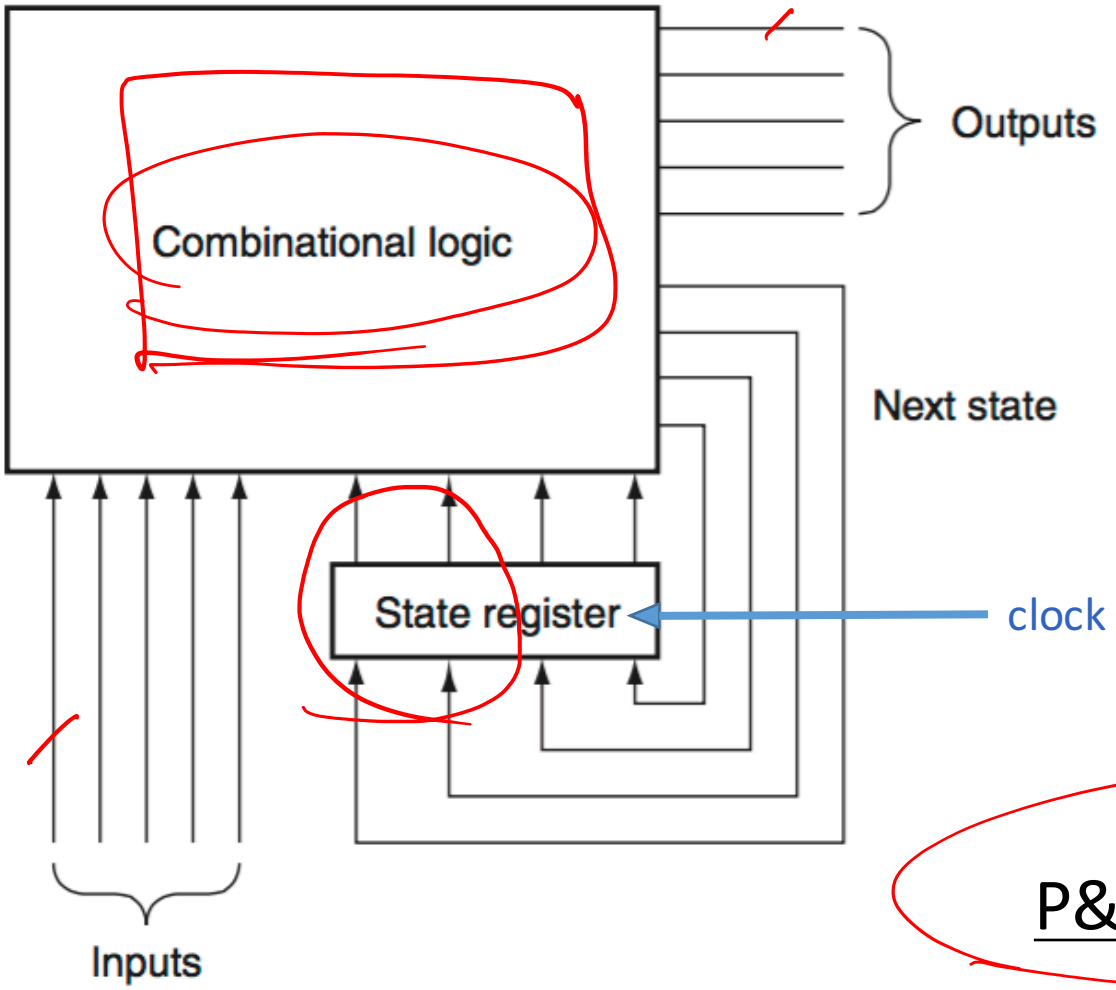
State Encoding

State	Code
S0	00
S1	01
S2	10
unused	11

Truth Table

PS	Input	NS	Output
XX	0	00	0
00	1	01	0
01	1	10	0
10	1	00	1
11	X	00	0

Hardware Implementation of FSM

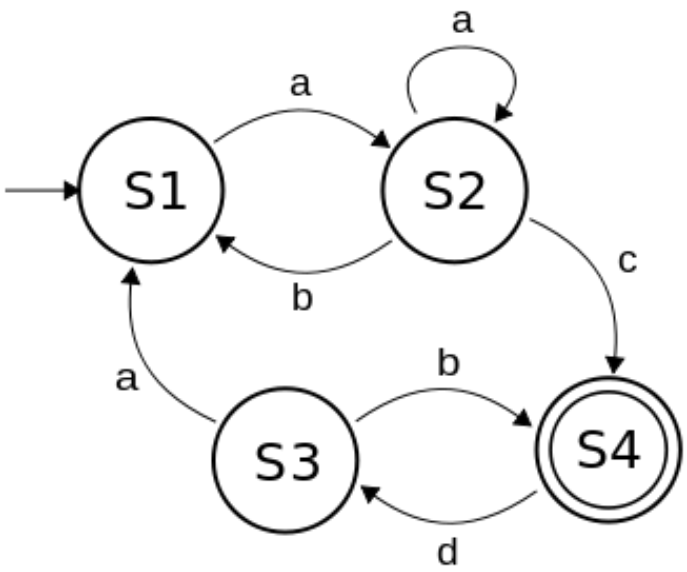


P&H Section B.10

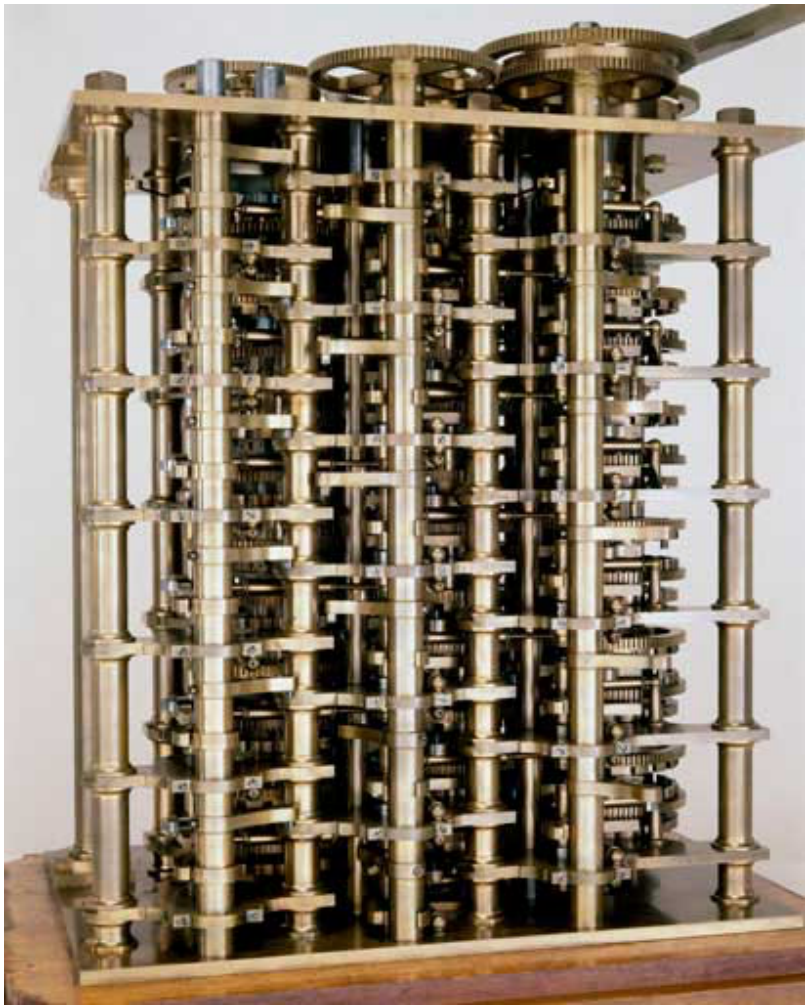
Finite State Machines (FSM) - Summary

- Describe computation over time
- Represent FSM with “state transition diagram”
- Start at given state and input, follow some edge to next (or same) state
- With combinational logic and registers, any FSM can be implemented in hardware

state transition diagram



Break

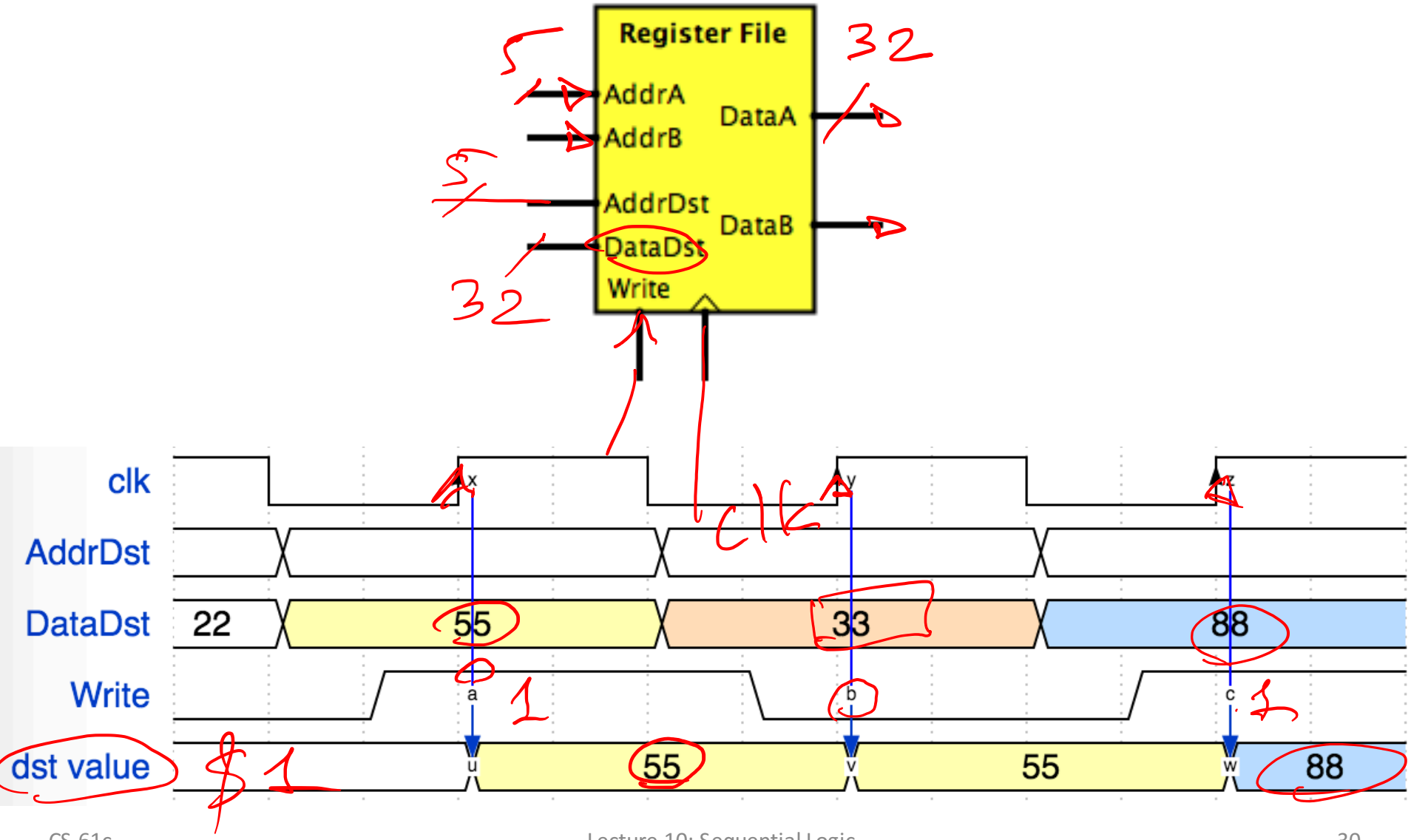


Babbage analytical engine, 1871

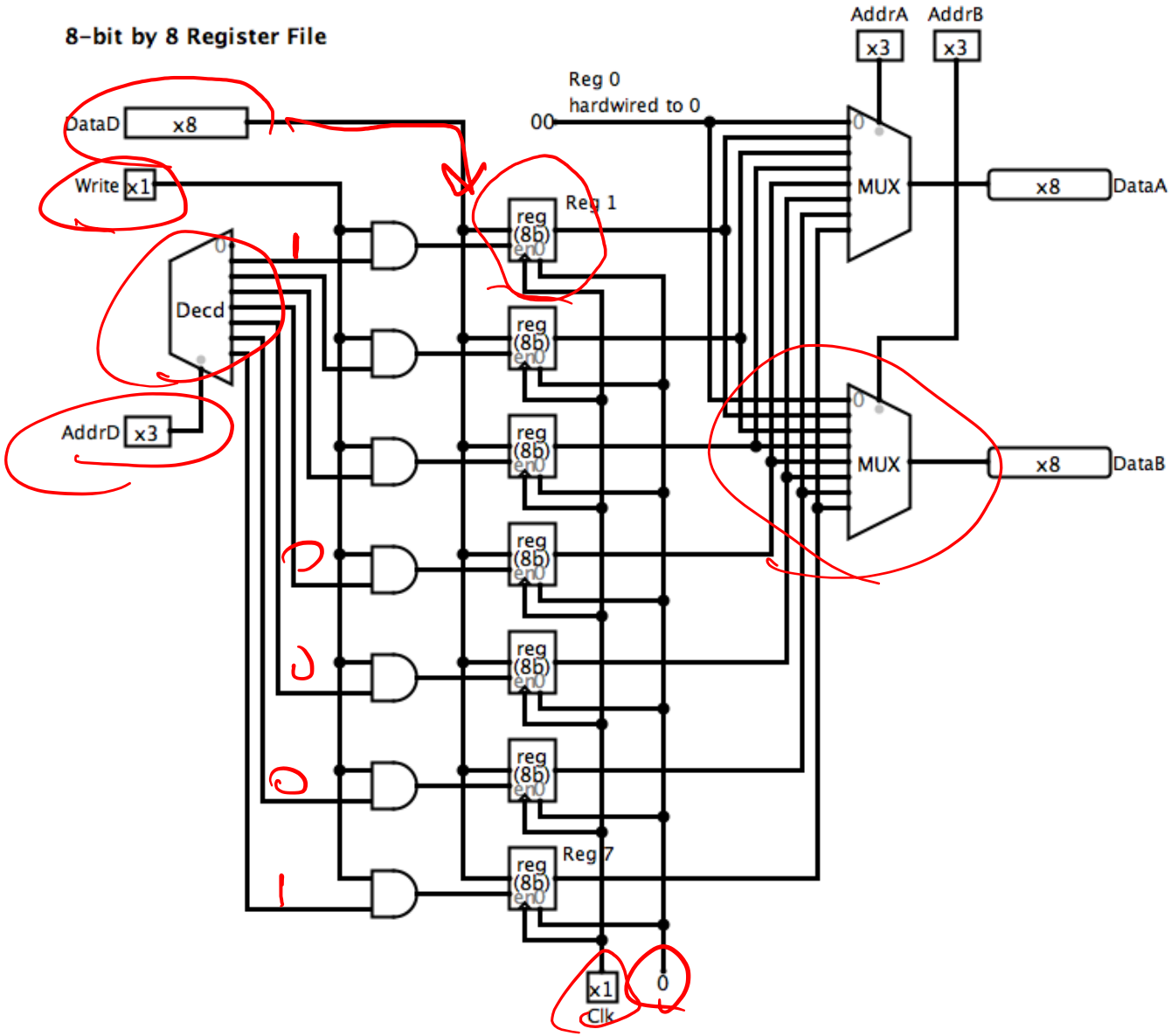
Agenda

- Sequential Circuits
  - Example: program counter
  - Latch (Flip-Flop)
- Finite State Machine
- **Memory**
  - **Register file**
  - **Data memory**
- Logic Delay
  - Example: adder
- And in Conclusion, …

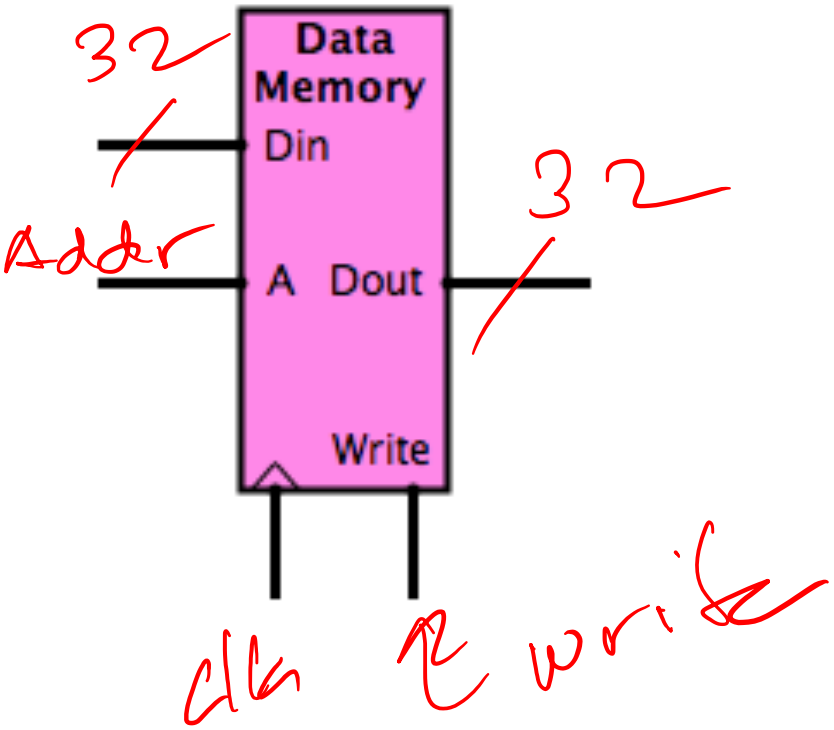
Register File

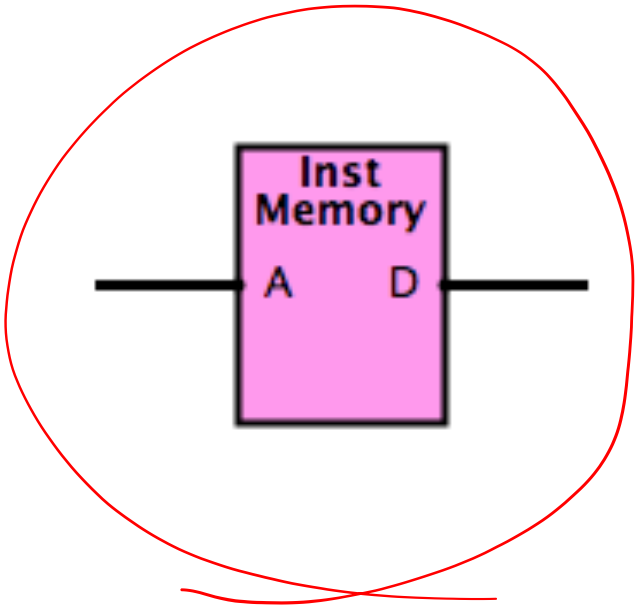


Register File Implementation

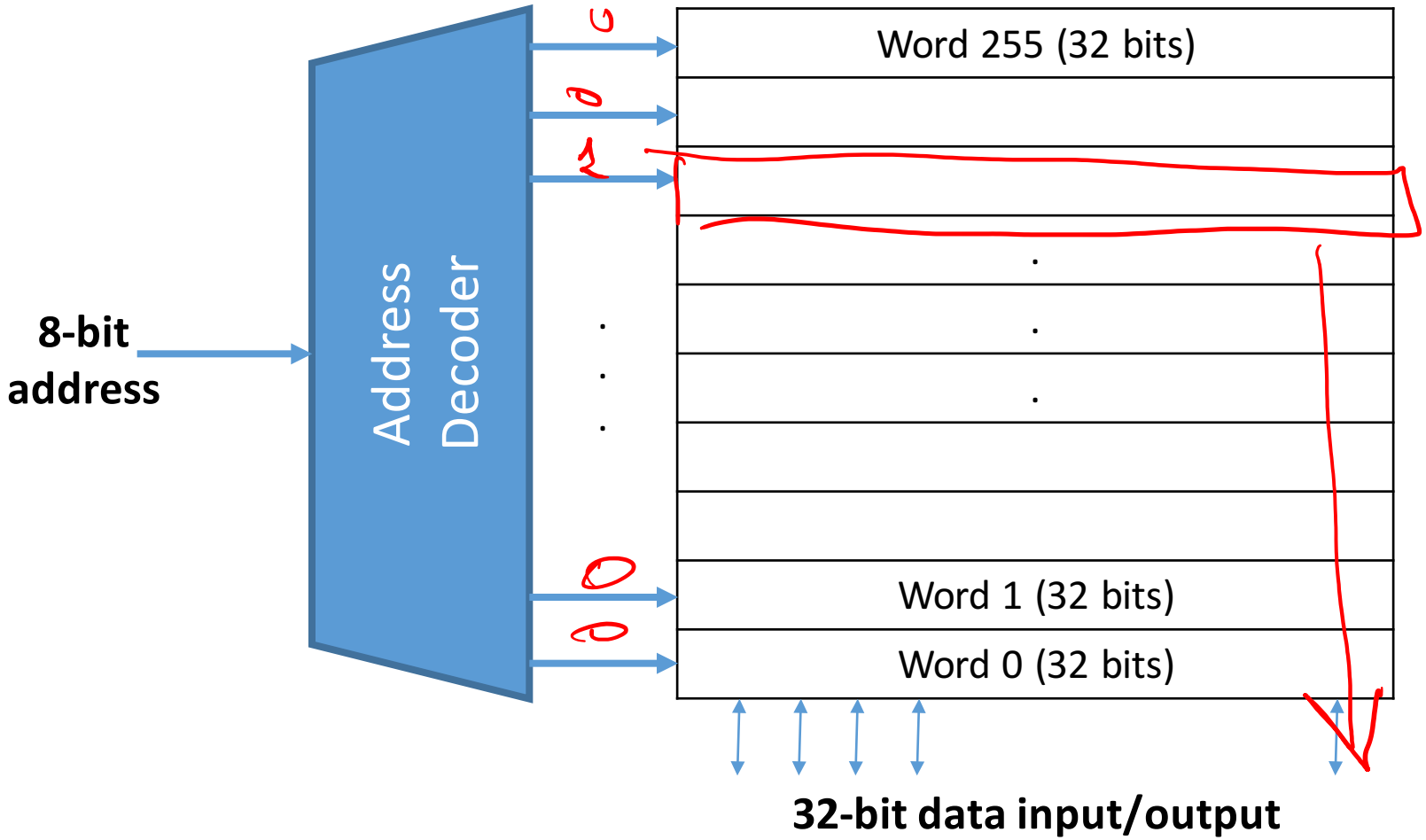


Data & Instruction Memory



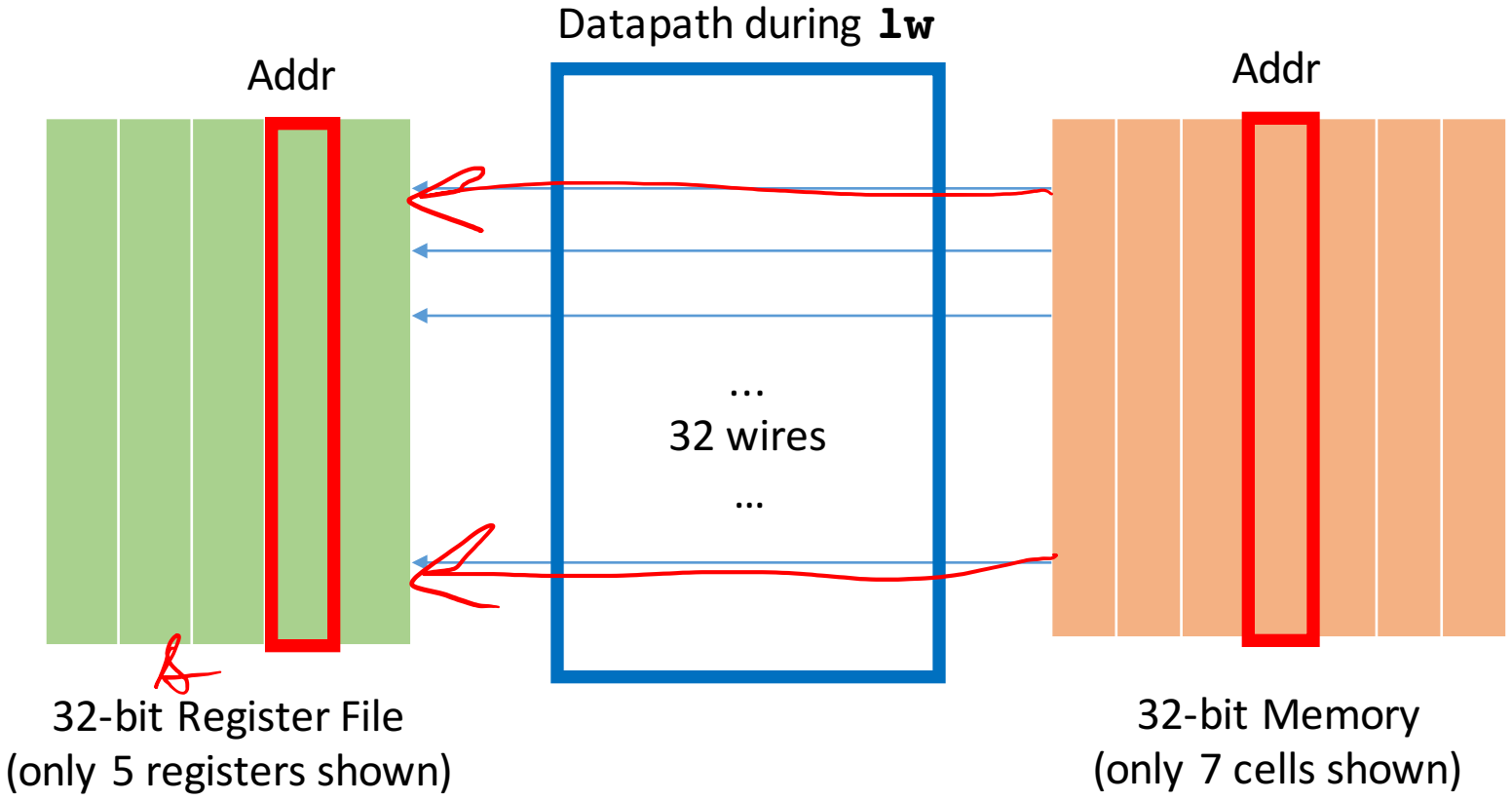


Memory Implementation (256 x 32-bit)



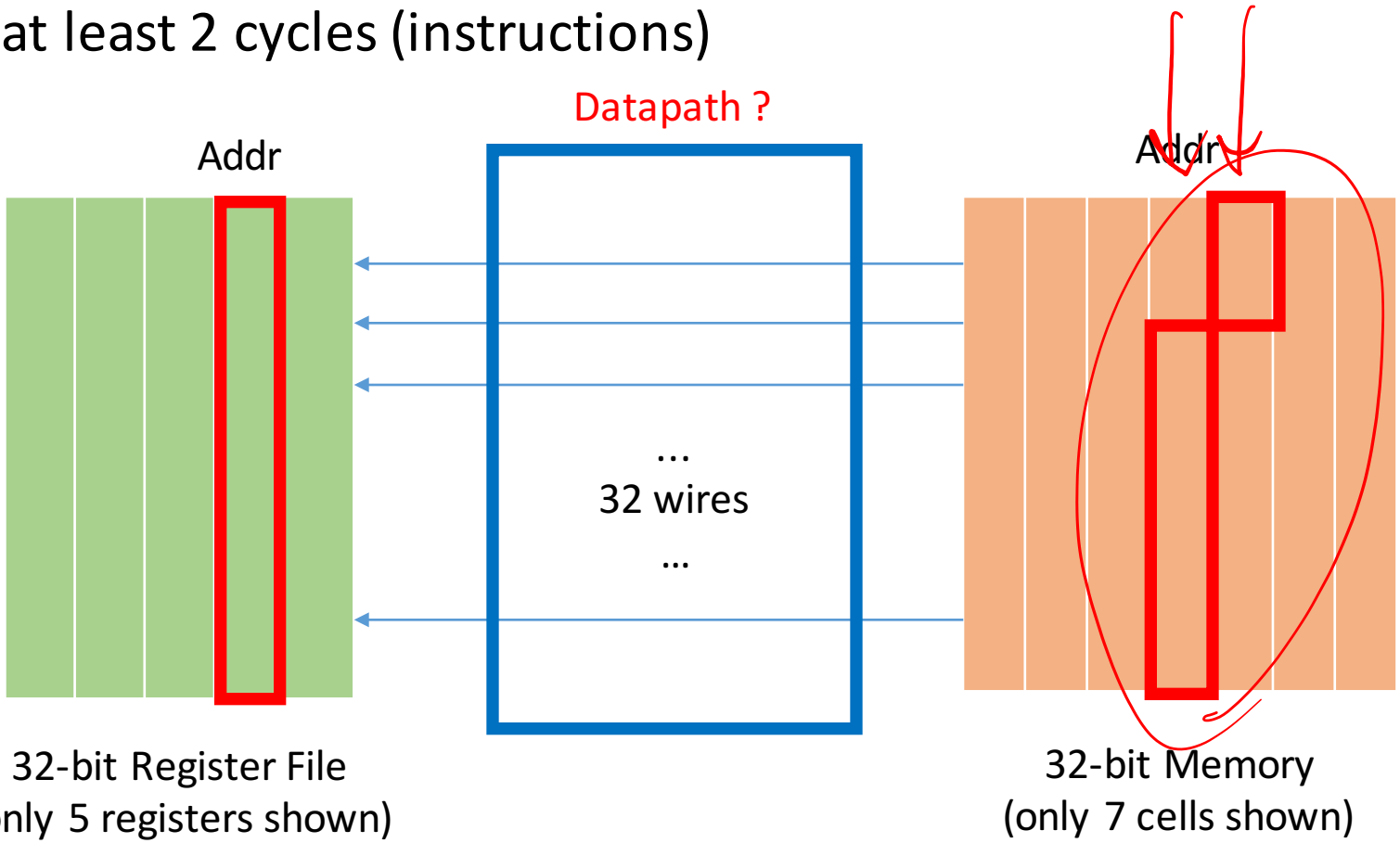
Why "aligned" loads?

- **`lw`** (32-bit loads) must be aligned on word boundary
  - i.e. two LSB's = 0



Unaligned Access

- Need 2 loads
  - at least 2 cycles (instructions)



Memory Types (P&H pp. 378)

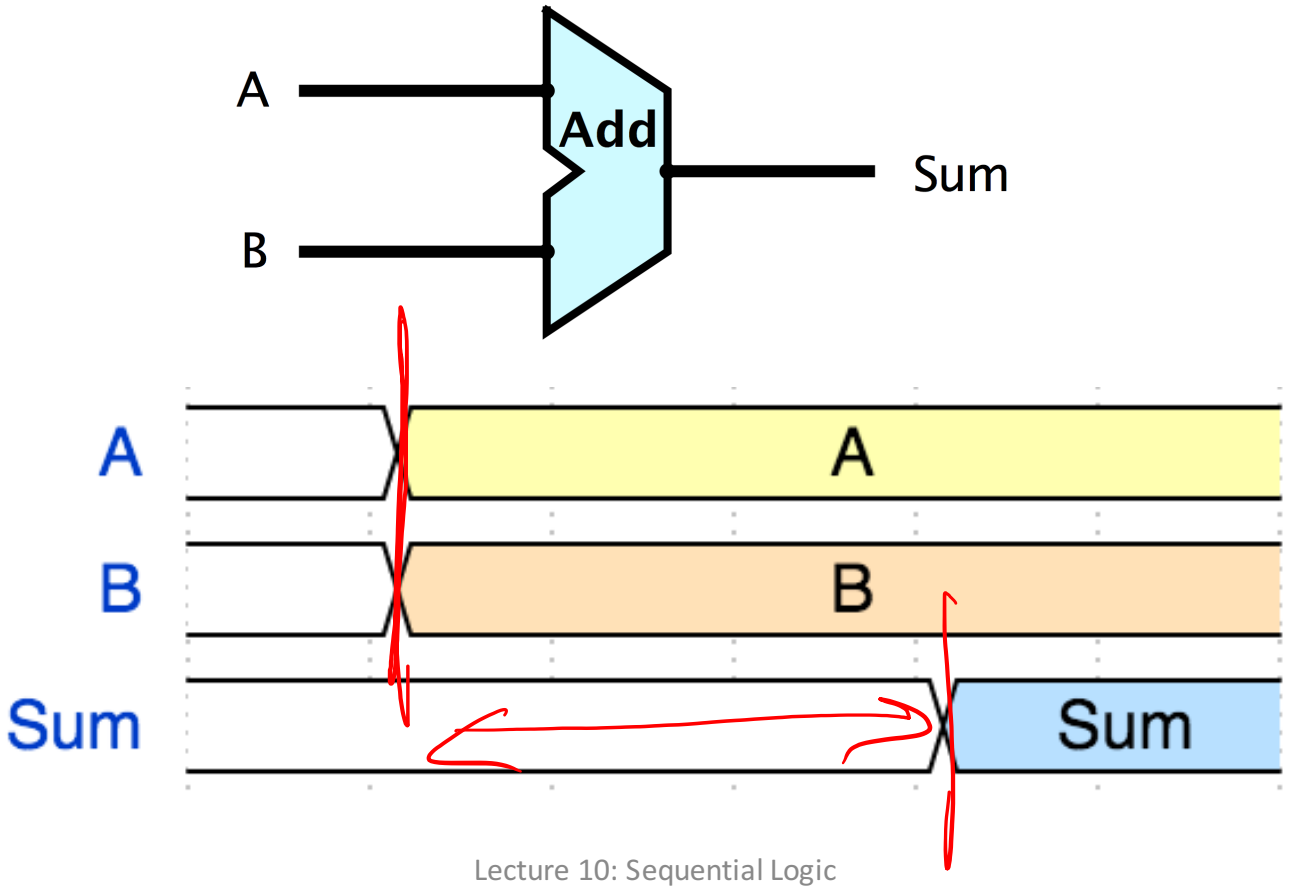
Type	Access Time (typical)	$ / GiB (2012)	Volatile	Applications
Flip-flop	< 100 ps	NA	needs power	register file
SRAM	0.5 … 3 ns	$500 … $1k	needs power	cache
DRAM	50 .. 70 ns	$10 … $20	clock & power	prog & data
Flash	5 … 50 μs	< $1	non-volatile	permanent
Disk	5 … 20 ms	< $0.1	non-volatile	permanent

Agenda

- Sequential Circuits
  - Example: program counter
  - Latch (Flip-Flop)
- Finite State Machine
- Memory
  - Register file
  - Data memory
- **Logic Delay**
  - **Example: adder**
- And in Conclusion, …

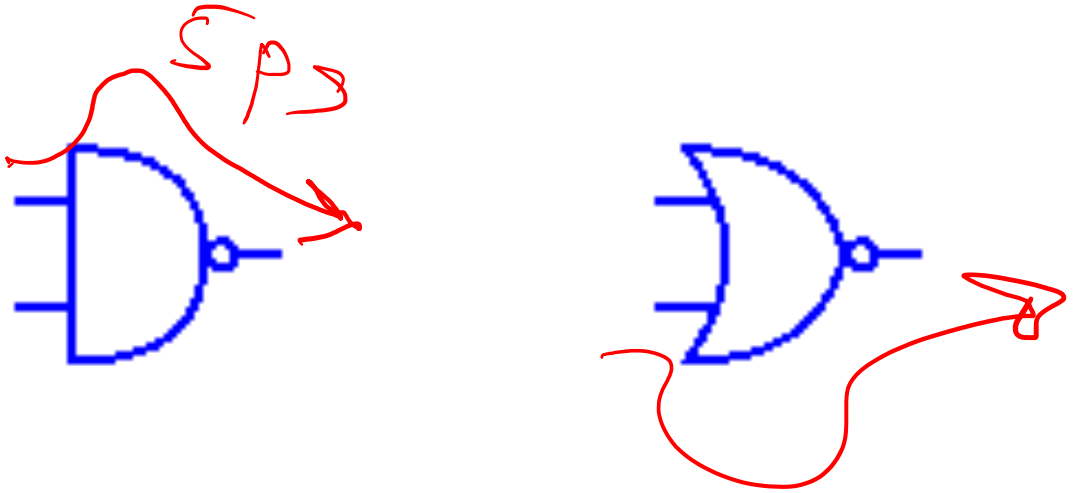
Logic Delay

- If we present new inputs **`A, B`** to an adder,
  - How long does it take for the **`Sum`** to appear at the output?



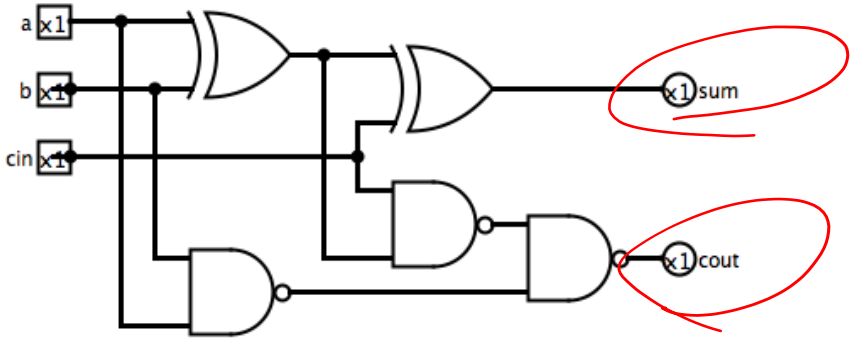
Gate Delay

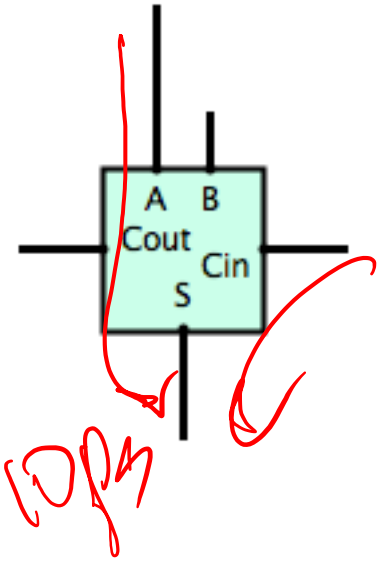
- The delay through a gate is approximately 5 ps
  - Rough estimate
  - Depends on technology (e.g. 14nm is faster than 28nm), supply voltage, and other parameters (see EE/CS 151)



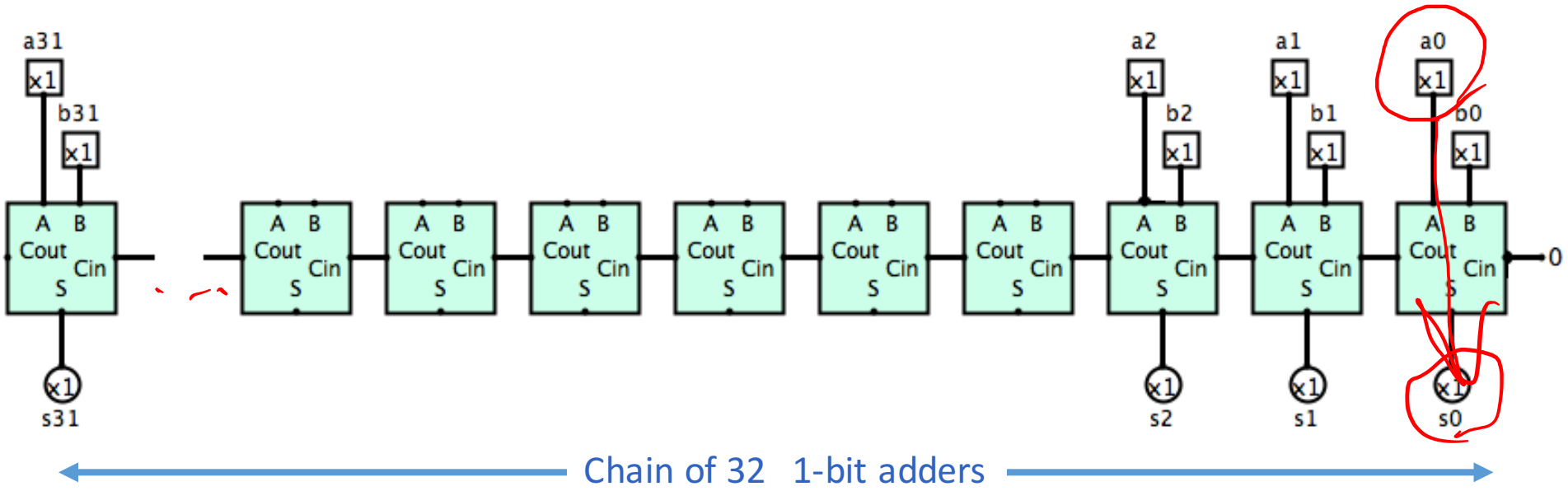
1-bit Adder Delay

- 2 … 3 gates from any input (**a, b, c_{in}**) to either output (**sum, c_{out}**)
  - Delay is approximately 10 ps





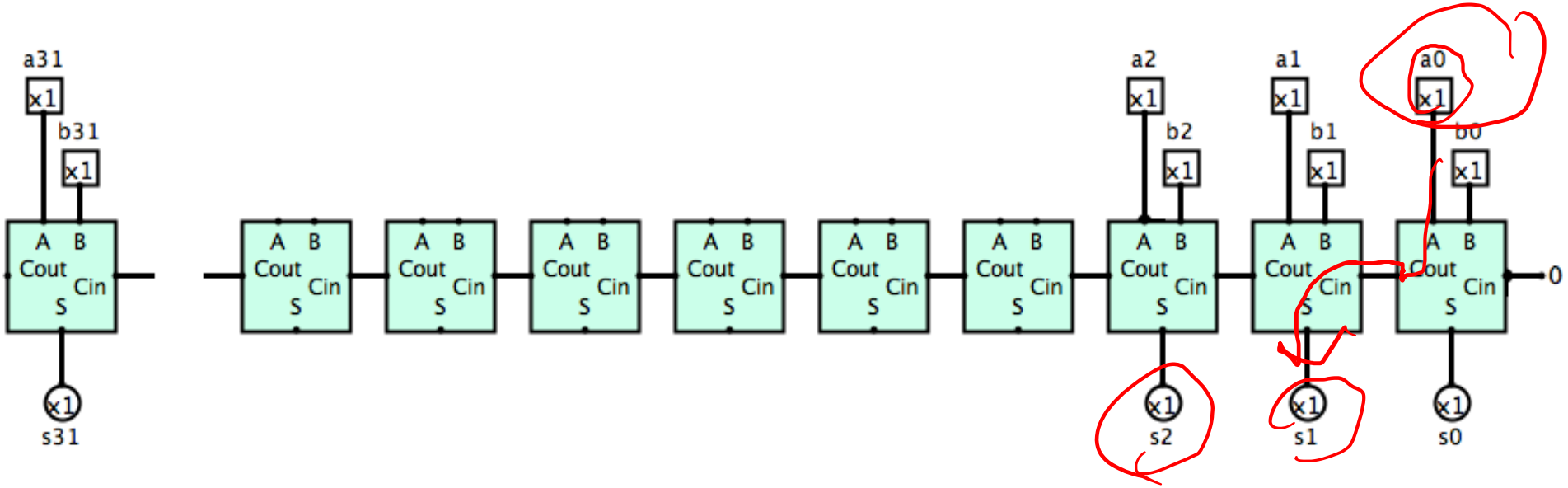
32-bit Adder Delay



- What is the delay from
  - a_0 to s_0?
  - C_{in} to s_0?

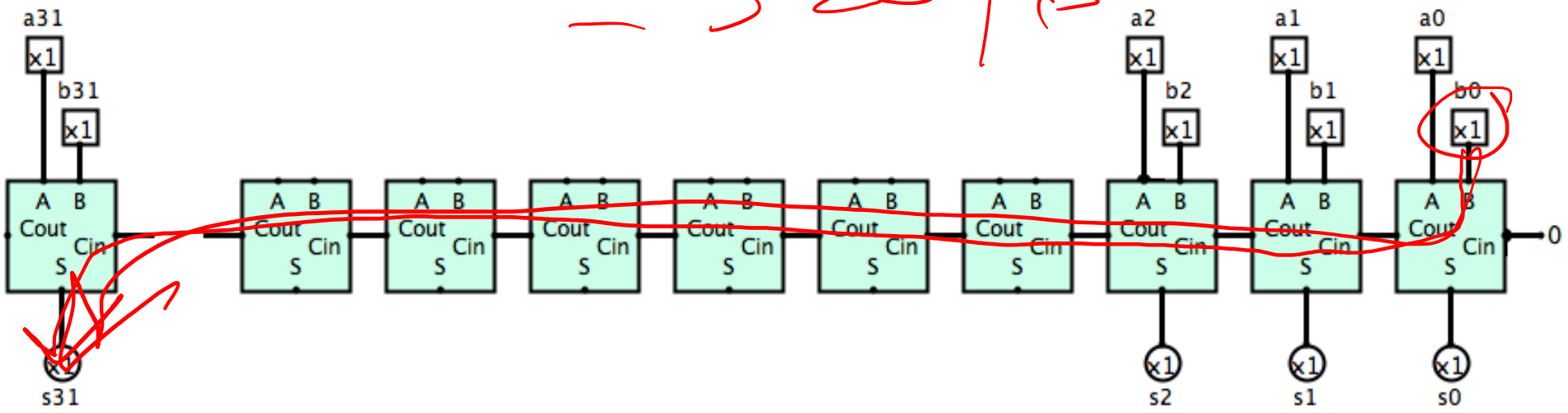
32-bit Adder Delay

- What is the delay from
  - a_0 to s_1? 20ps
  - a_0 to s_2? 30ps



Critical Path

- What are the
  - longest ("critical") path and
  - its delay?
- What about a 64-bit adder?



Faster Adder

- How can we reduce the delay of an adder?

- Need to reduce the length of the critical path!
  - But how?

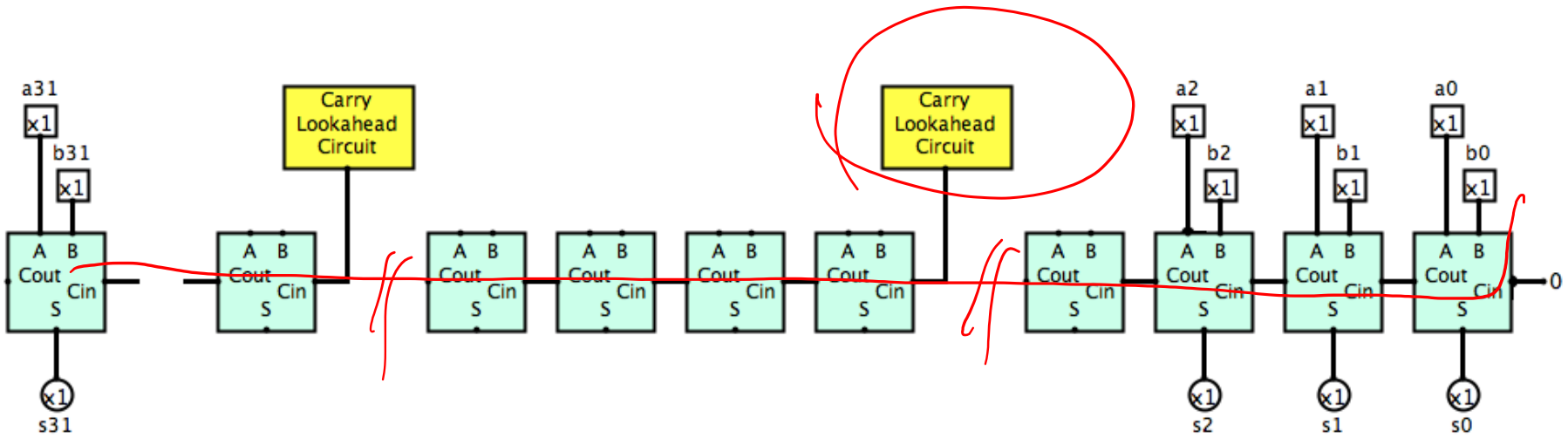
First Attempt

a0	b0	a1	…	b31	s0	s1	…	s31
0	0	0	…	0	0	0	…	0
0	0	0	…	0	0	0	…	0
…					…			
1	1	1	…	1	1	1	…	1

- s_0 = …
  - Sum of products of all a_i, b_i
  - How many rows?
- Ditto for s_1 … s_{31}
- Only 2 gate delays
  - 10 ps delay!?
  - But 2^{64} rows!
  - Such “big” gates are slooow (delay is much longer than 5 ps)

Carry Look-ahead Adder

- Idea:
  - Break carry chain between every 4 … 8 adders (4 in picture below)
  - I.e. break 32-bit adder into eight 4-bit adders (40 ps delay)
  - Carry Look-ahead Circuit computes correct carry fast
    - How?



Carry of 1-bit Adder

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Carry of 1-bit Adder

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

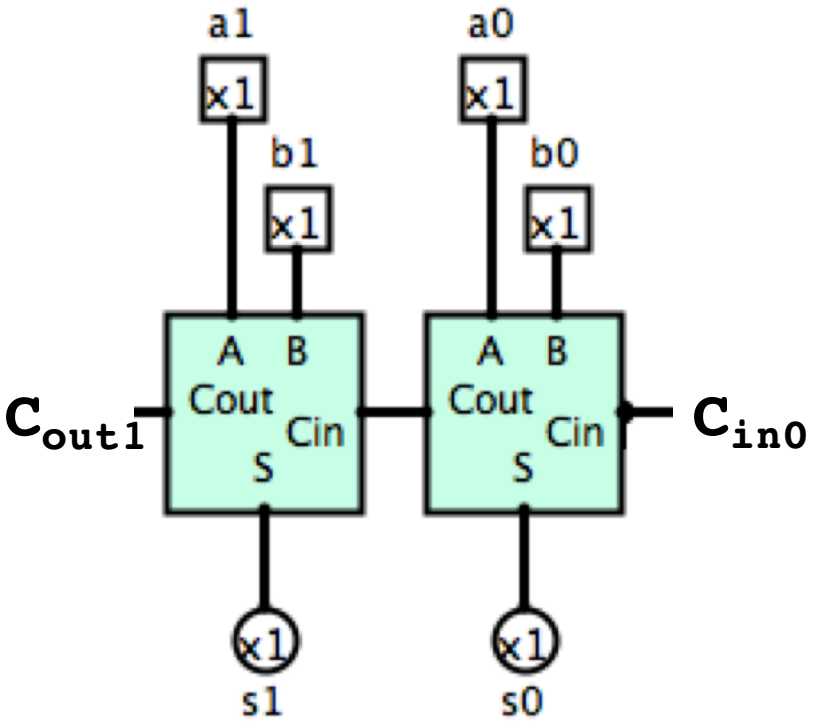
$$C_{out} = A \cdot B + (A + B) \cdot C_{in}$$

$$g = A \cdot B \qquad p = A + B$$

$$C_{out} = g + p \cdot C_{in}$$

Your Turn

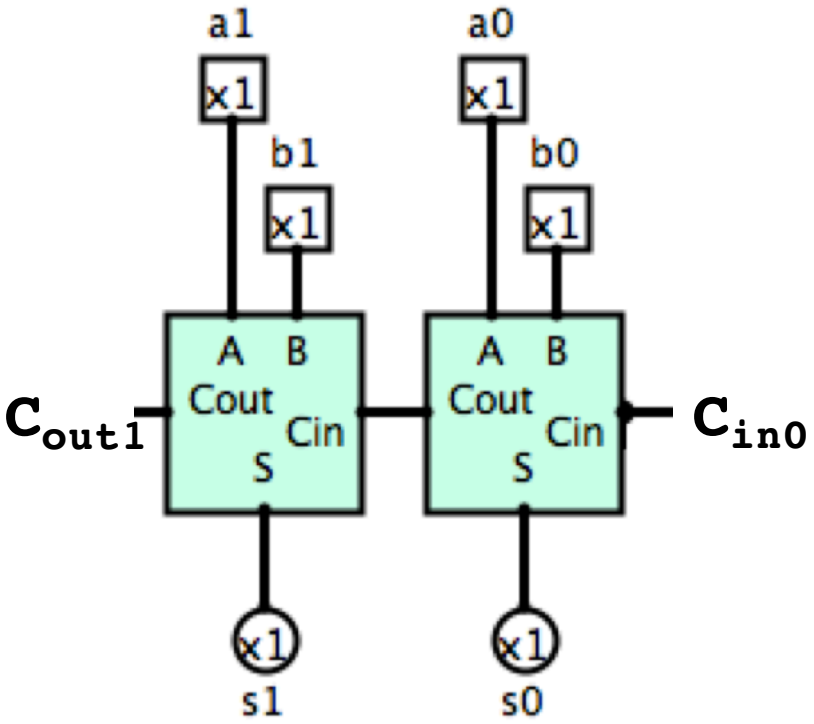
Which expression for C_{out1} is correct?



Ans	C_{out1}
A	$A_1 + P_1 G_0 + A_0 B_0 C_{in0}$
B	$G_1 + P_1 G_0 + C_{in0}$
C	$G_1 + P_1 G_0 + P_1 P_0 C_{in0}$
D	$G_1 + P_1 P_0 C_{in0}$
E	$P_1 G_0 + P_1 P_0 C_{in0}$

Your Turn

Which expression for C_{out1} is correct?



Ans	C_{out1}
A	$A_1 + P_1 G_0 + A_0 B_0 C_{in0}$
B	$G_1 + P_1 G_0 + C_{in0}$
C	$G_1 + P_1 G_0 + P_1 P_0 C_{in0}$
D	$G_1 + P_1 P_0 C_{in0}$
E	$P_1 G_0 + P_1 P_0 C_{in0}$

Carry of 4-bit Adder

$$C_1 = G_0 + P_0C_0$$

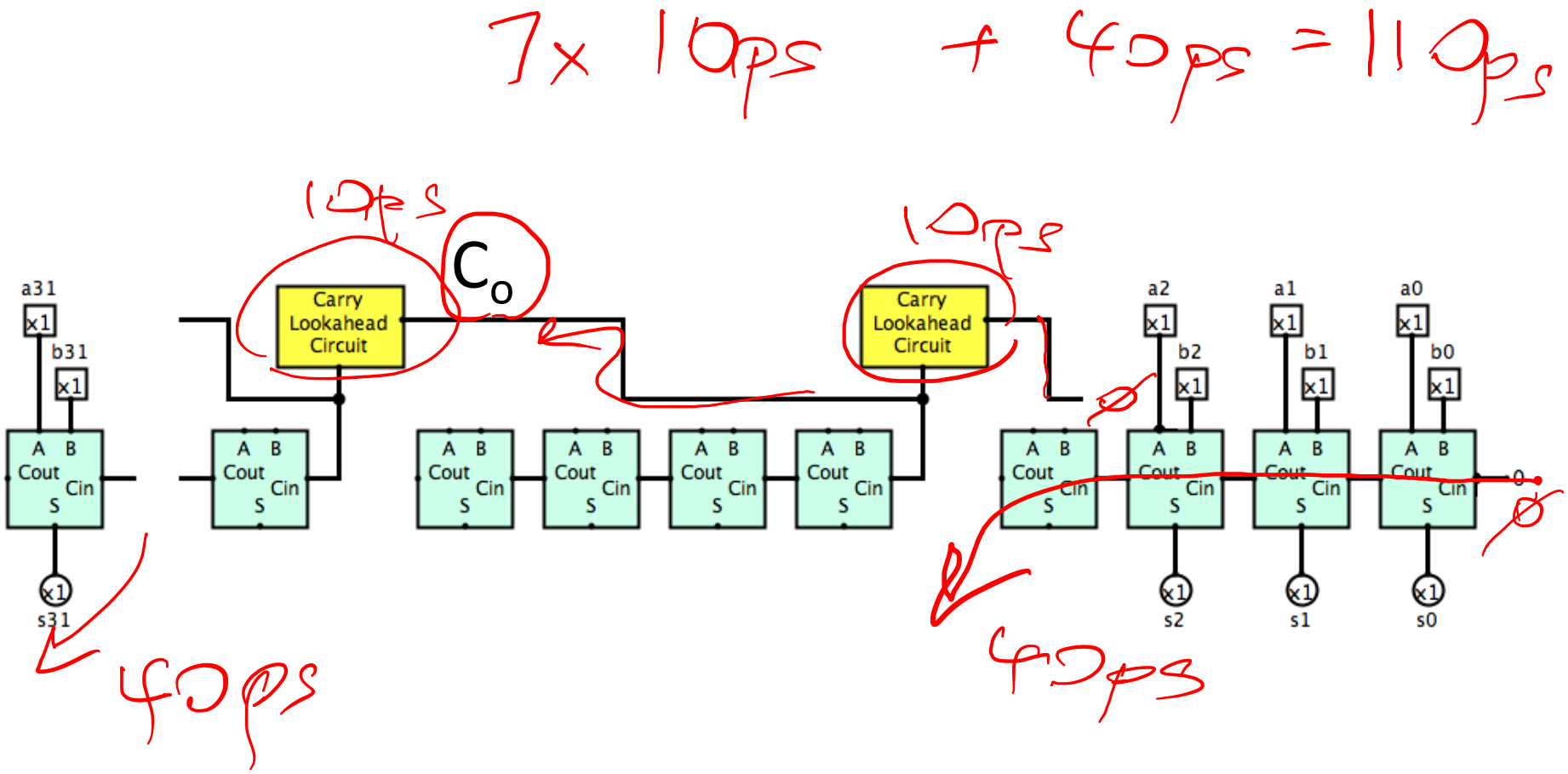
$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$$

$$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

$$C_4 = G_3 + P_3C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$$

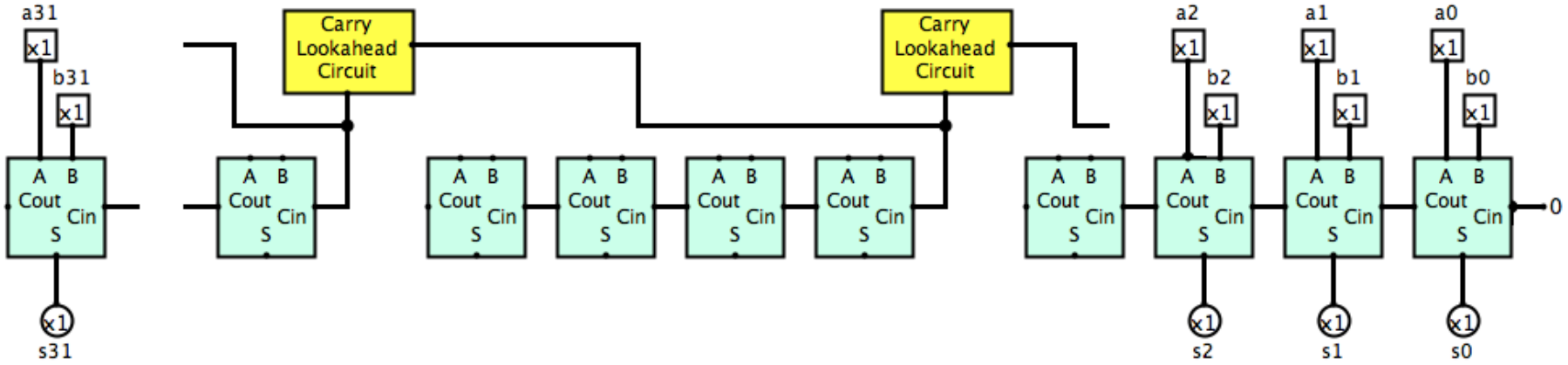
- Last equation "predicts" carry out of 4-bit adder
  - 4-bit Carry Look-ahead Circuit!
  - Delay: 2 gates → 10 ps

Implementation of Look-ahead Adder



Critical Path of Carry Look-ahead Adder

- Total delay is sum of
  - Delay of 4-bit adder (4 x 10 ps)
  - Delay of 7 carry look-ahead circuits (7 x 10 ps)
  - 110 ps << 320 ps!
- Price: additional gates (transistors)



Carry Look-ahead Adder Delay

- For N bits
  - $t_{d\_ripple} \propto N$
  - $t_{d\_look\_ahead} \propto \log(N)$
- Works well for up to about 100 bits
- Other structures for > 100 bits
  - E.g. encryption
- See EE/CS 151 for more on this

Agenda

- Sequential Circuits
  - Example: program counter
  - Latch (Flip-Flop)
- Finite State Machine
- Memory
  - Register file
  - Data memory
- Logic Delay
  - Example: adder
- **And in Conclusion, …**

And in Conclusion, …

- Clock
  - orchestrates state update
- Use register
  - as memory element
  - to break feedback loops
- Maximum clock frequency
  - limited by sum of delays of all elements in feedback loop
- Finite state machine (FSM)
  - describe computation over time
- Logic delay
  - trade additional transistors for higher speed