10/26/26

COMP 311: *Computer Organization!*

Lecture 18: Intro to Timing Analysis



The University
of North Carolina
at Chapel Hill

https://tinyurl.com/comp311-fa26

Logistics Update



- Lab 3 is out
  - Due 11/6
- WA4 will be released today.
- Quiz Corrections!

Quiz Corrections and Conceptual OH

- I want everyone to understand where they went wrong and get credit for things they've mastered by end of semester
- I also want you all to make use of office hours & the amazing TAs!
  - Nothing can replace working together in-person ☺
- Sign-up for a 20-minute appt.
  - https://tinyurl.com/311fa26-oh
- You can get up to 50% of missed points back
- You can also sign up to just talk through concepts

Quiz Corrections and Conceptual OH

- Choose 1–2 missed questions to focus on, per appointment.
- Be ready to explain your original mistake, why your correction works, and how to solve a similar problem.
- Credit is based on demonstrated understanding, not attendance alone.

Plan for Class Today

- 3:35-4:00: Combinational vs. Synchronous Logic
- 4:00-4:15: Intro to Basic Waveform Diagrams
- 4:15-4:20: BREAK
- 4:20-4:50: Waveform Diagram Practice

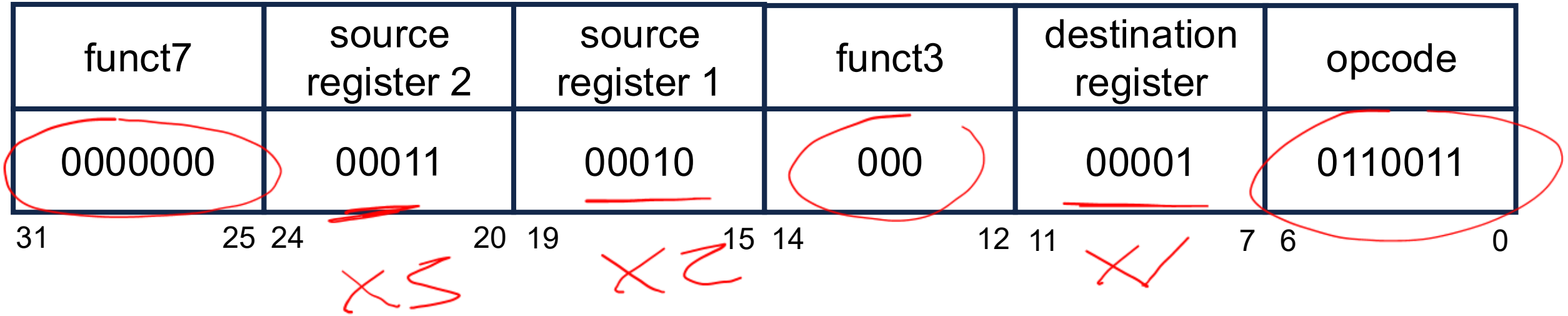
Guiding Question ☺

How does hardware remember information, and how can we predict when its outputs change?



Piecing Things Together: The ALU + Register File

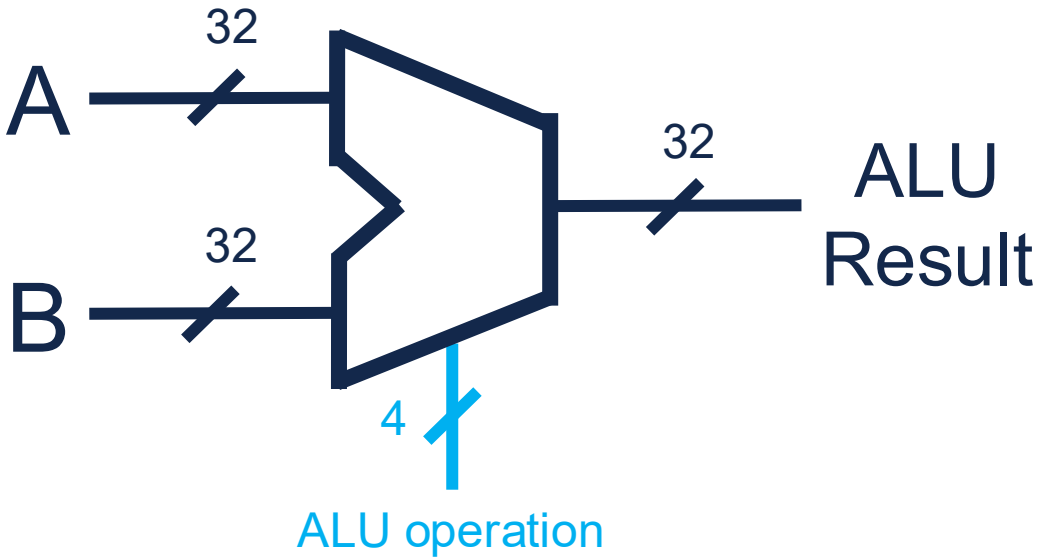
x1 = x3 + x2



00000000001100010000000010110011

Opcode is always 0110011 for integer R-type ALU instructions
funct3 is always the primary operation code group (e.g. 000 = add/sub. 100 = xor, 111 = and)
funct7 is used to further specify the operation (e.g. distinguishes add vs. sub. srl vs. sra)

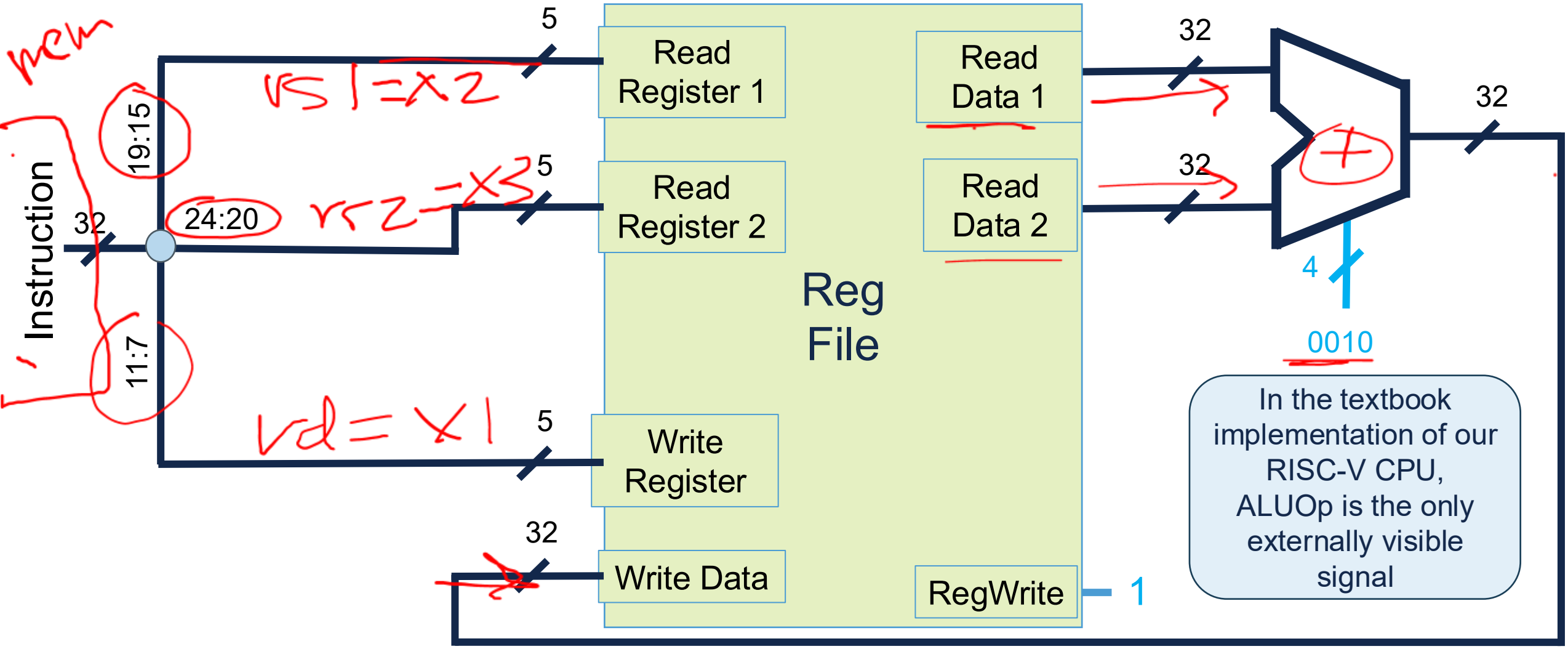
RISC-V 32I ALU



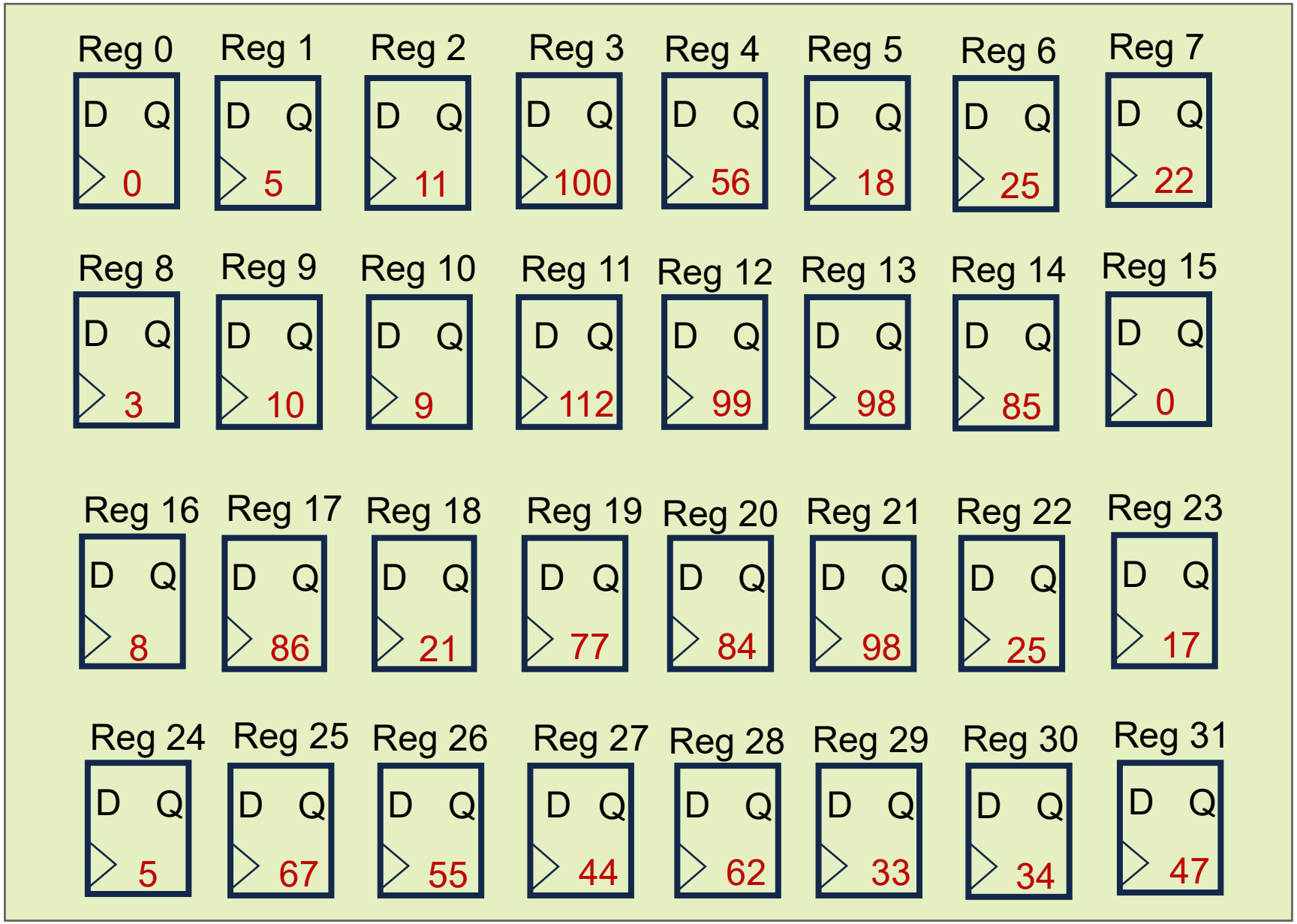
Function	ALU Operation
AND	0b0000
OR	0b0001
ADD	0b0010
SUB	0b0110
set on less than (SLT)	0b0111

Control Signal Input

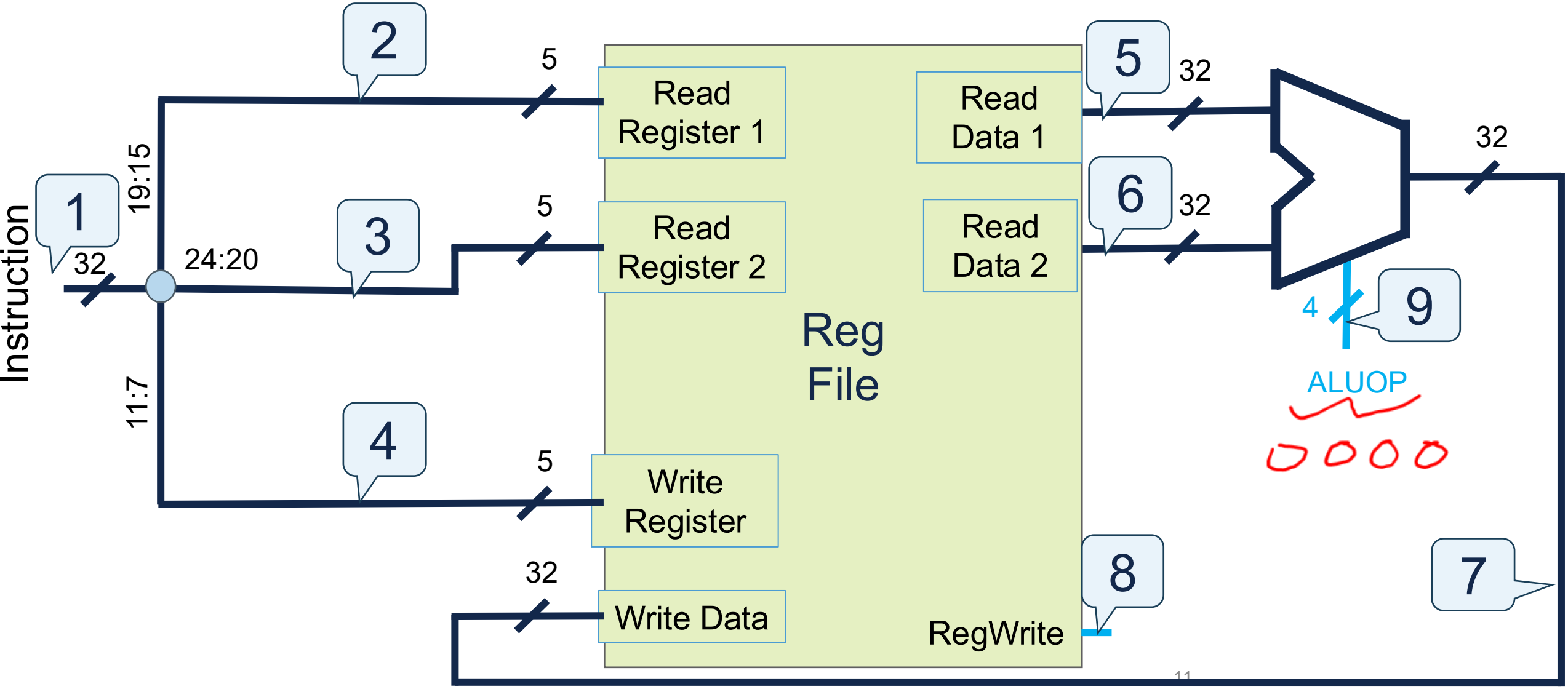
funct7	rs2	rs1	funct3	rd	opcode
0000000	00011	00010	000	00001	0110011
31 25	24 20	19 15	14 12	11 7	6 0



Contents of RegFile



What is the value of each signal when executing: `and x4, x2, x5`





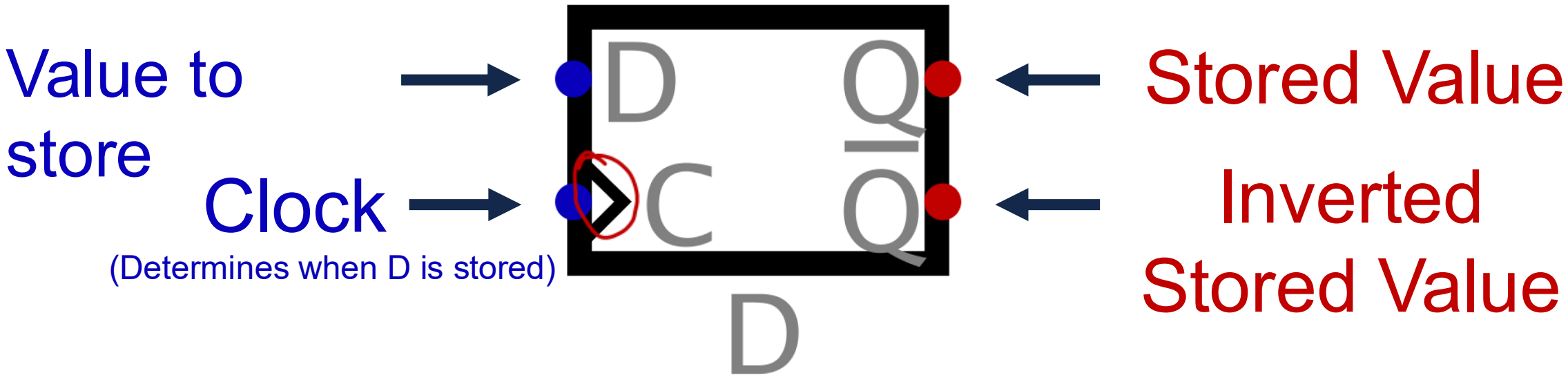
Flip Flops And Registers

Combinational vs. Synchronous Logic



- Combinational logic:
  - All the circuits we've seen so far!
  - Every element in the circuit is *recomputed* as soon as a change is detected on an input
  - Result dependent only on current input (no memory)
- Synchronous logic:
  - Circuits ~synced~ with a clock
  - Output depends on current input && past state
  - This past state is stored in a register!

D Flip-Flop


Value to store
Clock
(Determines when D is stored)
D
C
Q
Q
D
Stored Value
Inverted Stored Value

Digital Signals

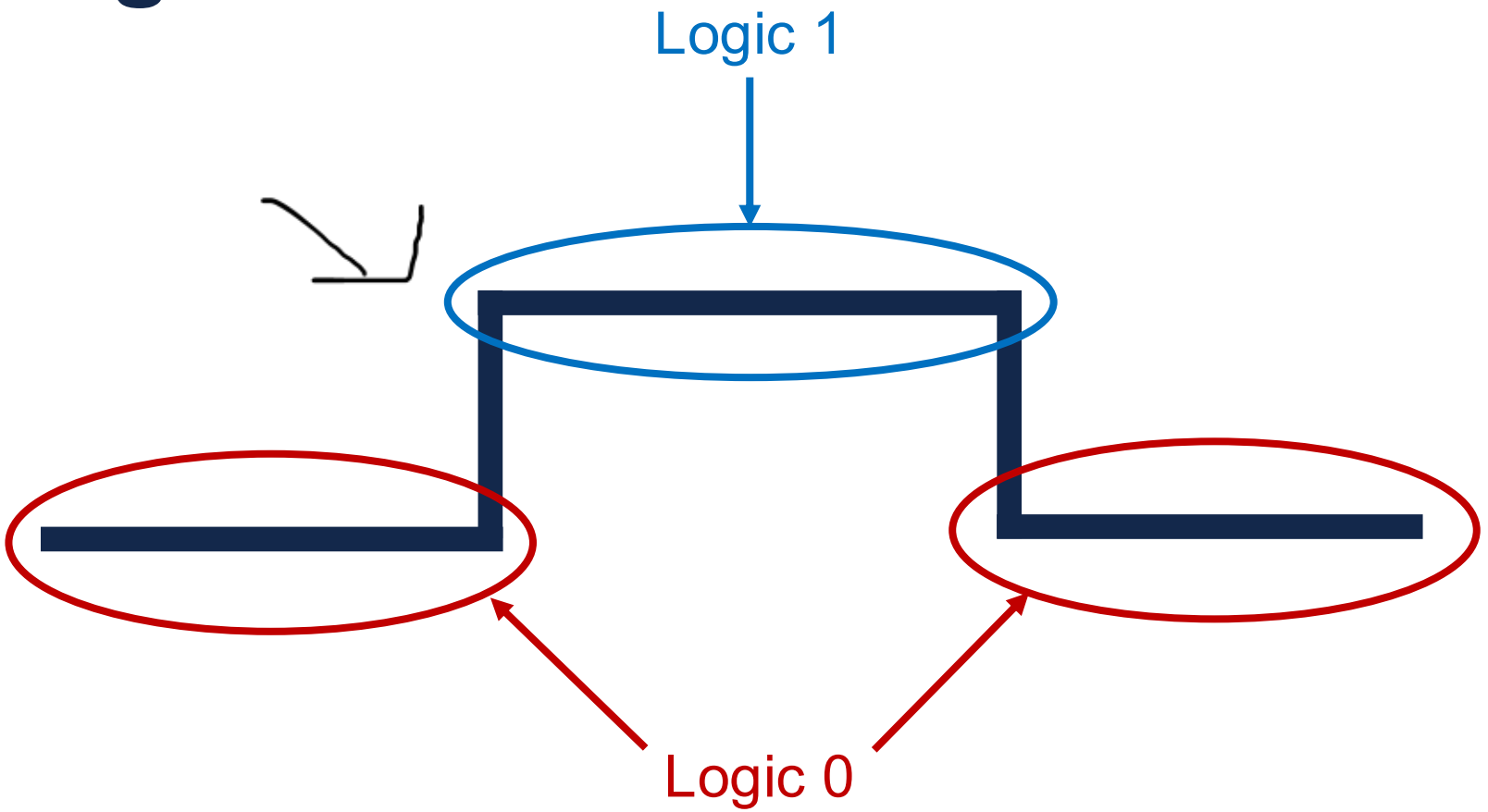


- Rising Edge
  - When a signal changes from logic low to logic high
  - You can think of the signal as *rising* from 0 to 1
- Falling Edge
  - When a signal changes from logic high to logic low
  - You can think of the signal as *falling* from 1 to 0

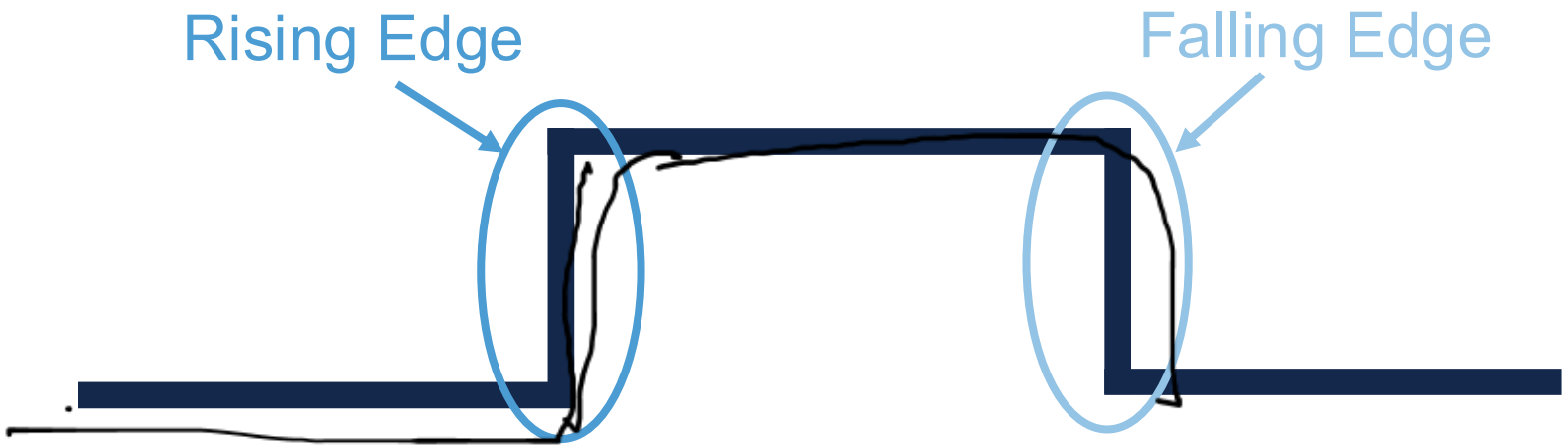
Digital Signals



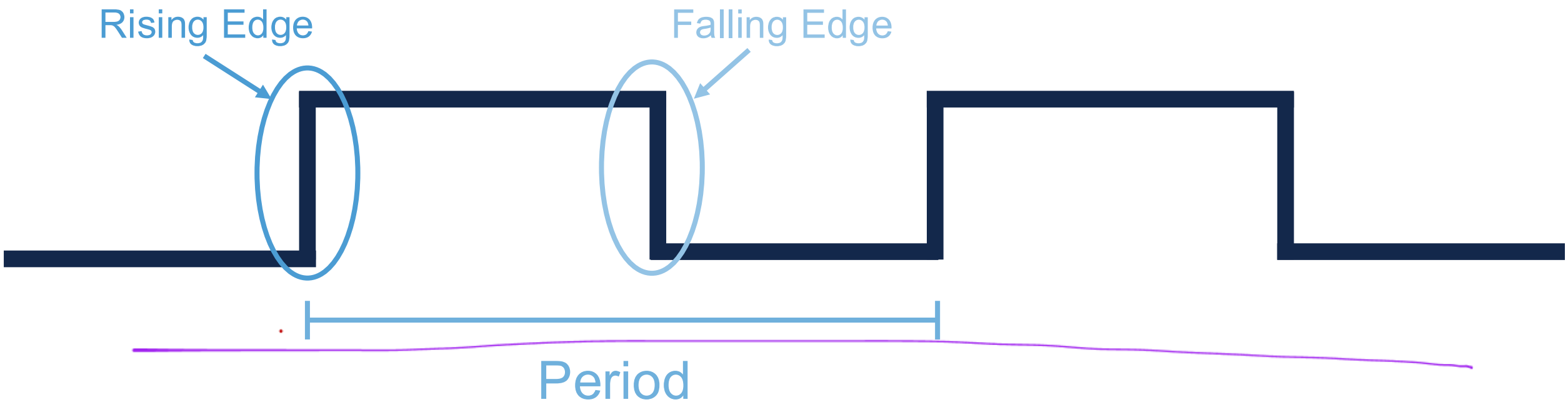
Digital Signals



Digital Signals



Clock



Clock

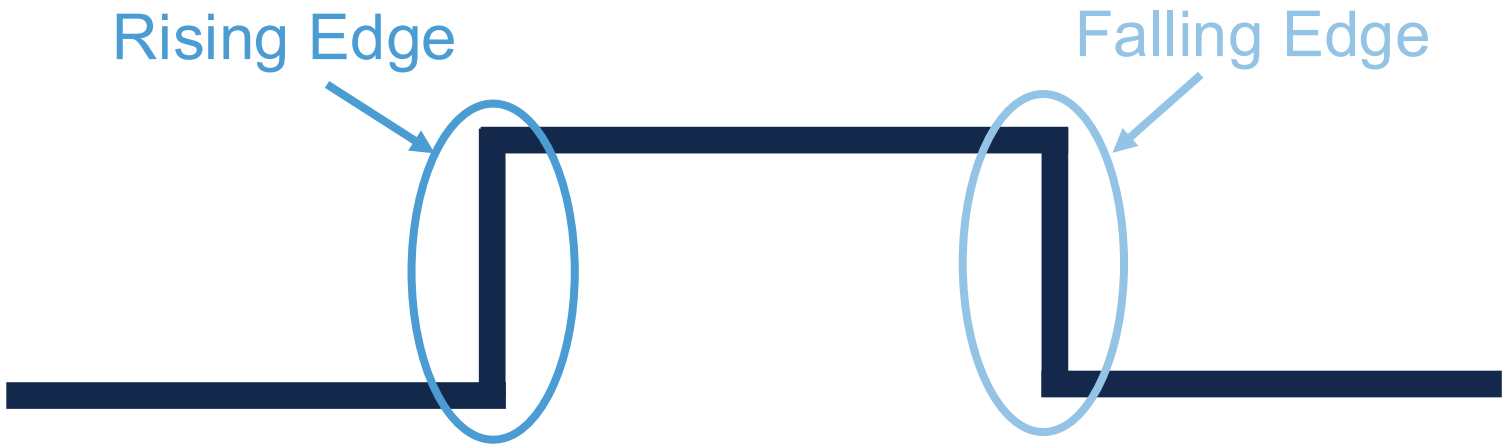


- The clock is a one-bit signal that oscillates (or toggles) back and forth between 0 and 1
- The purpose of the clock is to ensure that our circuits stay in sync
- The period of the clock is the time between one rising edge to the next. The clock spends an equal time in low and high states.
  - The period determines how fast our circuit runs
  - The shorter the period – or in other words the higher the frequency – the faster our circuit will run

D Flip-Flop

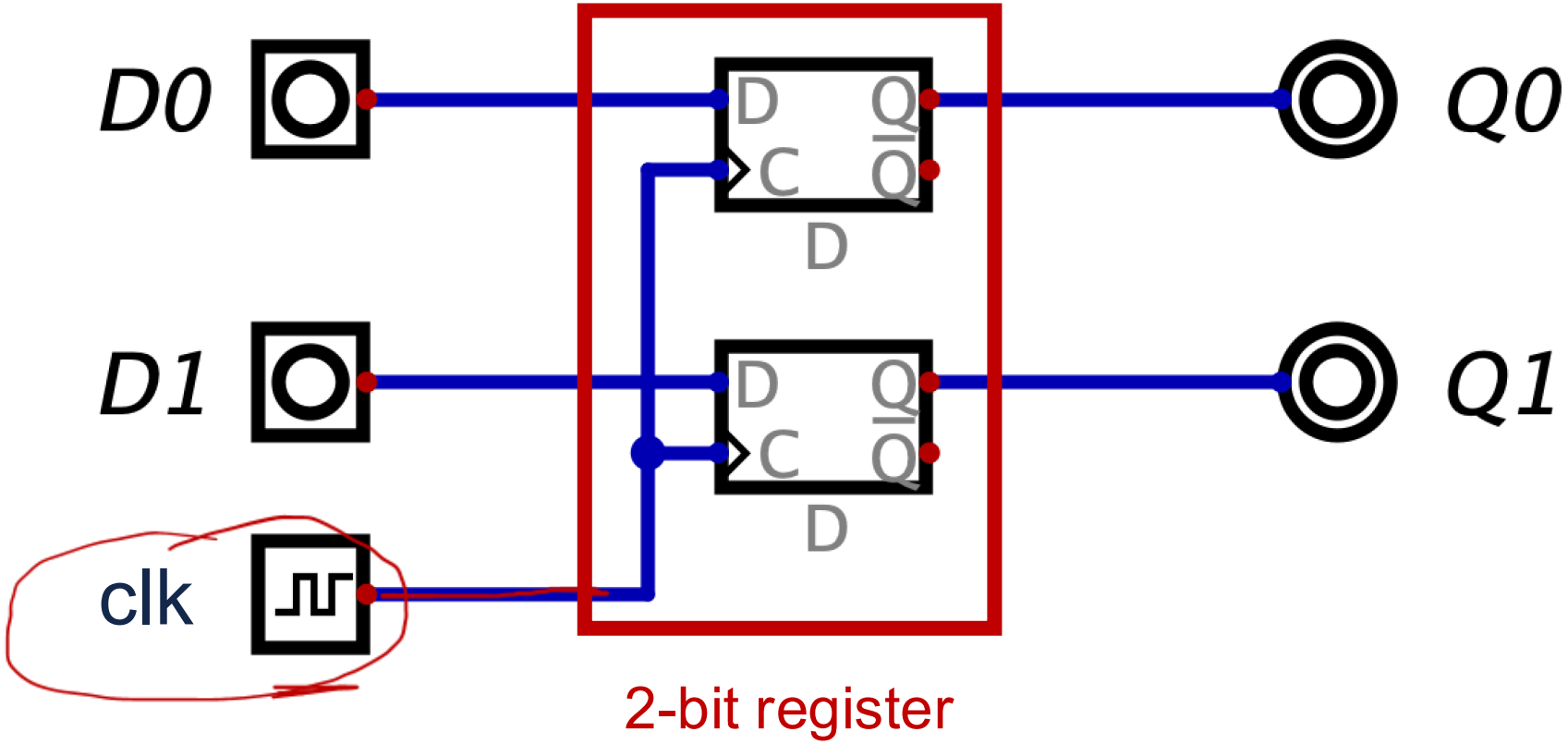


- Positive-edge triggered
  - Stores D when the clock goes from 0 to 1
- Negative-edge triggered
  - Stores D when the clock goes from 1 to 0

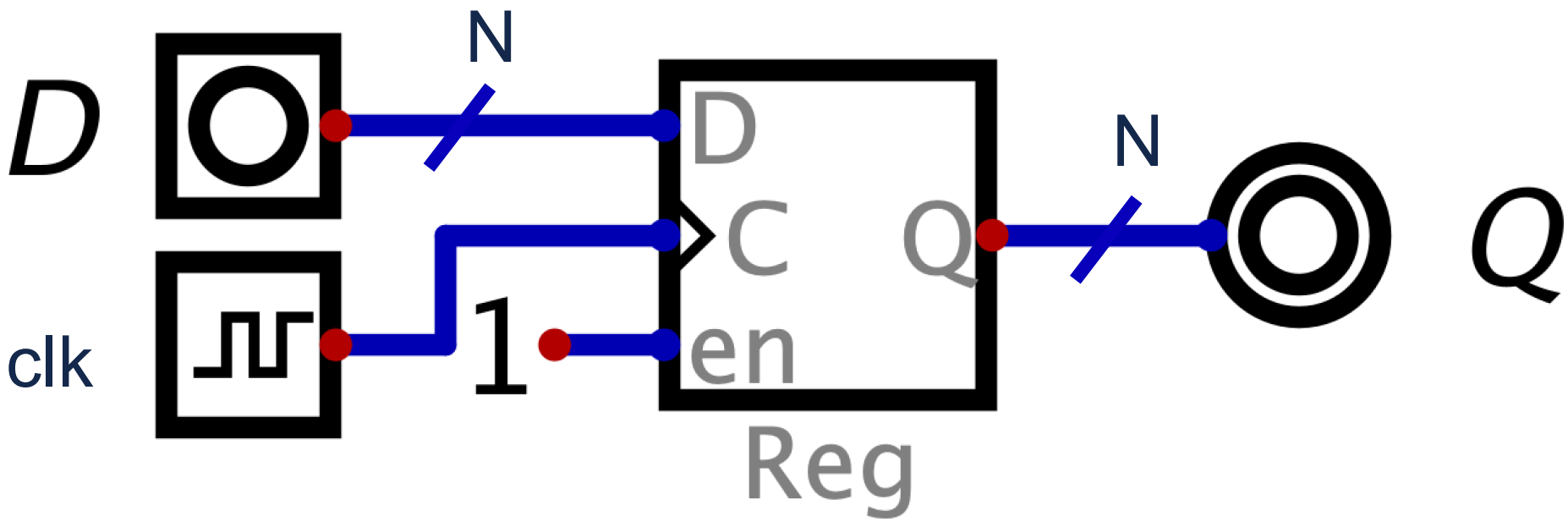


Registers

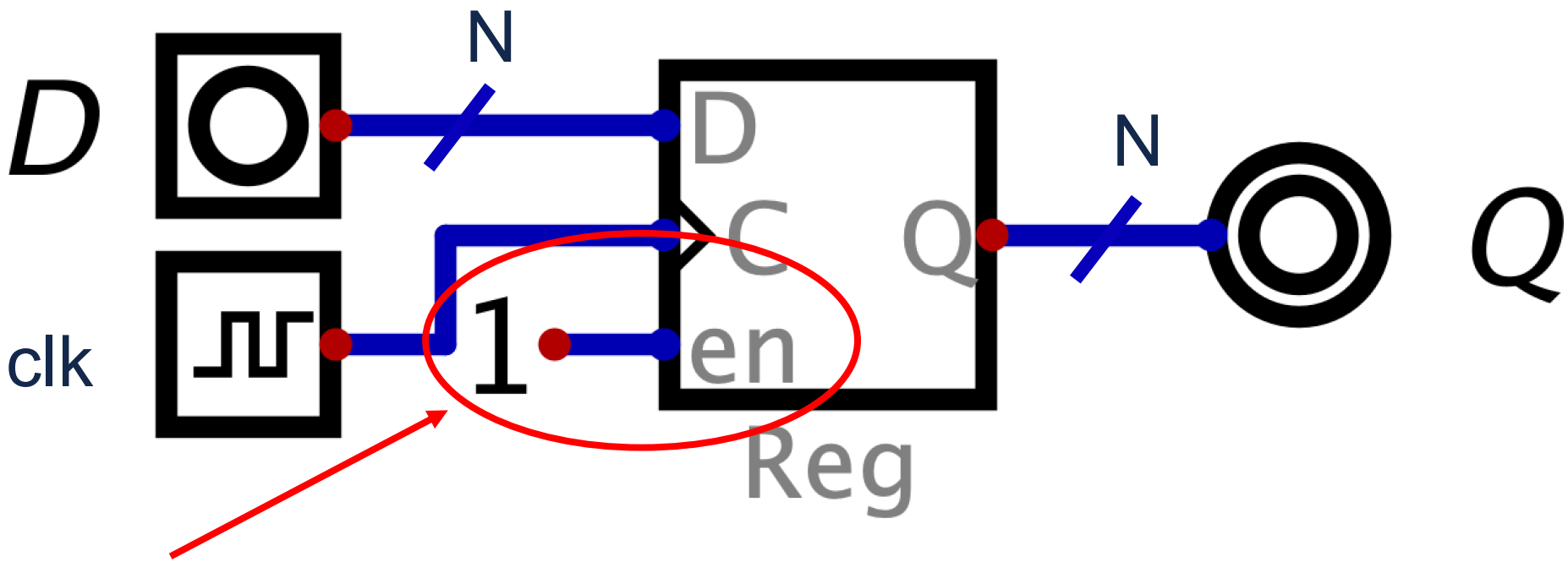
- To store multi-bit values, we can combine multiple FFs together into what is known as a register



N-bit Positive Edge Triggered Register



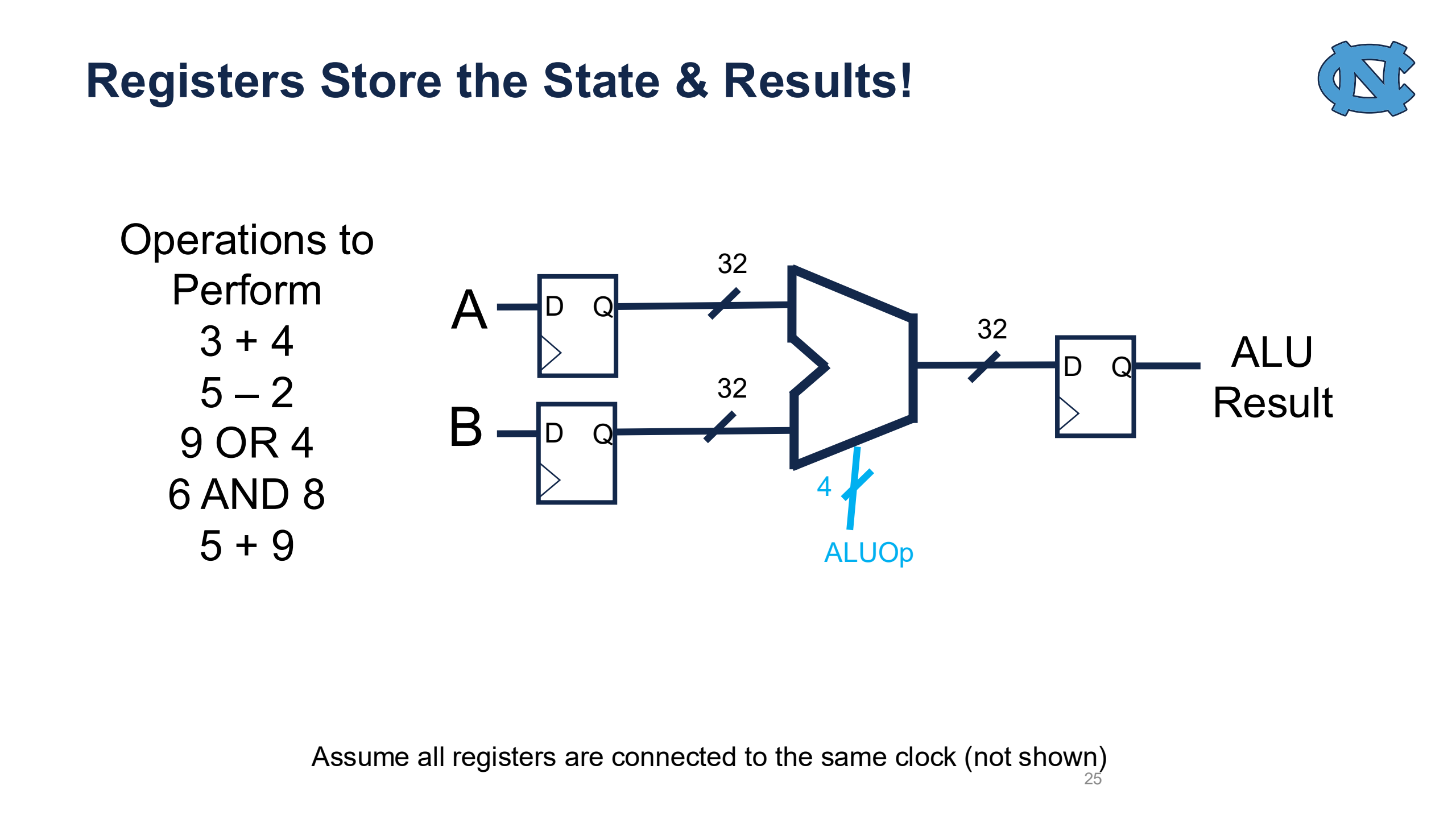
N-bit Positive Edge Triggered Register



Enable

- If 1, the register will read D on the rising edge of the clock
- If 0, the register will not read D

Registers Store the State & Results!



Operations to Perform

3 + 4

5 – 2

9 OR 4

6 AND 8

5 + 9

Assume all registers are connected to the same clock (not shown)

ALU



- The ALU will perform one operation per clock cycle
- To perform an operation, we have three inputs to set: A, B, and ALUOp
  - (ALU Op (ALU Control line from the textbook) is also synchronized with a register, but that is abstracted away for now)

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set less than
1100	NOR

	Operation	A	B	ALUOp
Cycle 0	3 + 4			
Cycle 1	5 - 2			
Cycle 2	9 OR 4			
Cycle 3	6 AND 8			
Cycle 4	5 + 9			

ALU



- The ALU will perform one operation per clock cycle
- To perform an operation, we have three inputs to set: A, B, and ALUOp
  - (ALU Op (ALU Control line from the textbook) is also synchronized with a register, but that is abstracted away for now)

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set less than
1100	NOR

	Operation	A	B	ALUOp
Cycle 0	3 + 4	3	4	0b0010
Cycle 1	5 - 2	5	2	0b0110
Cycle 2	9 OR 4	9	4	0b0001
Cycle 3	6 AND 8	6	8	0b0000
Cycle 4	5 + 9	5	9	0b0010

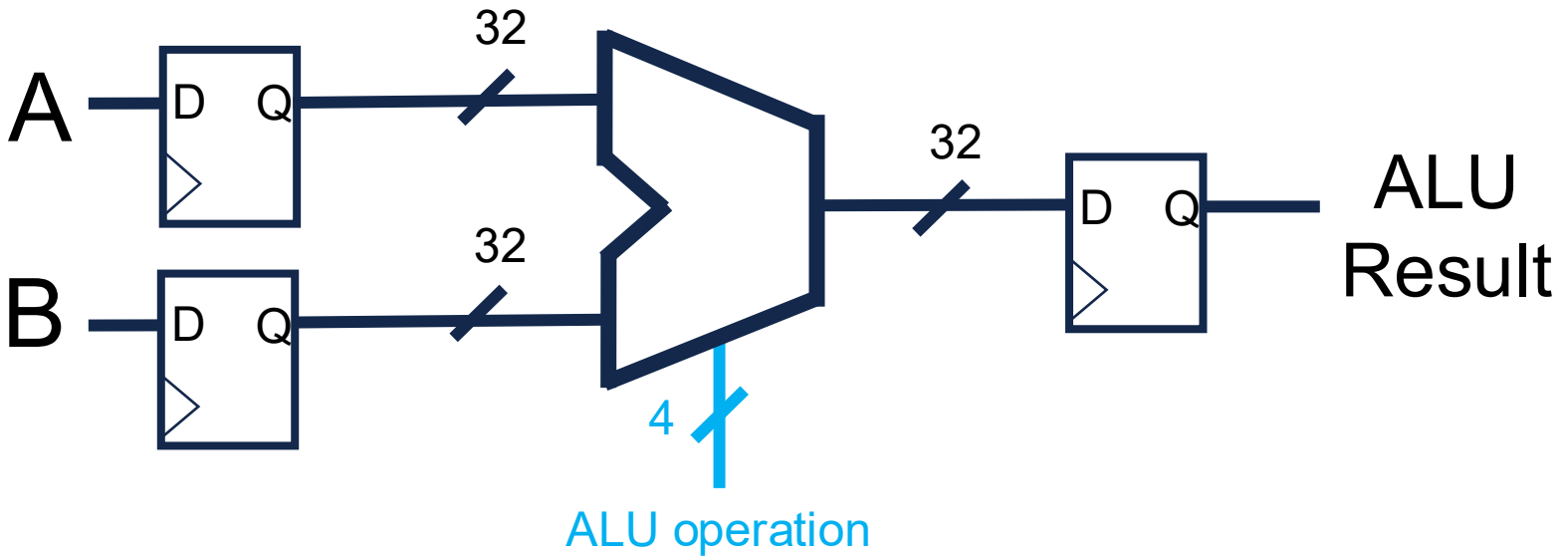
Dependent Operations

Operations to Perform

a = 5 + 4

b = 7 + 8

c = a + b



Assume all registers are connected to the same clock (not shown)



Our plan: We are going to zoom in for a moment to see how to analyze timing at the gate level, before popping back up to analyze instruction-level & CPU performance!

Timing Analysis

Waveform Diagrams

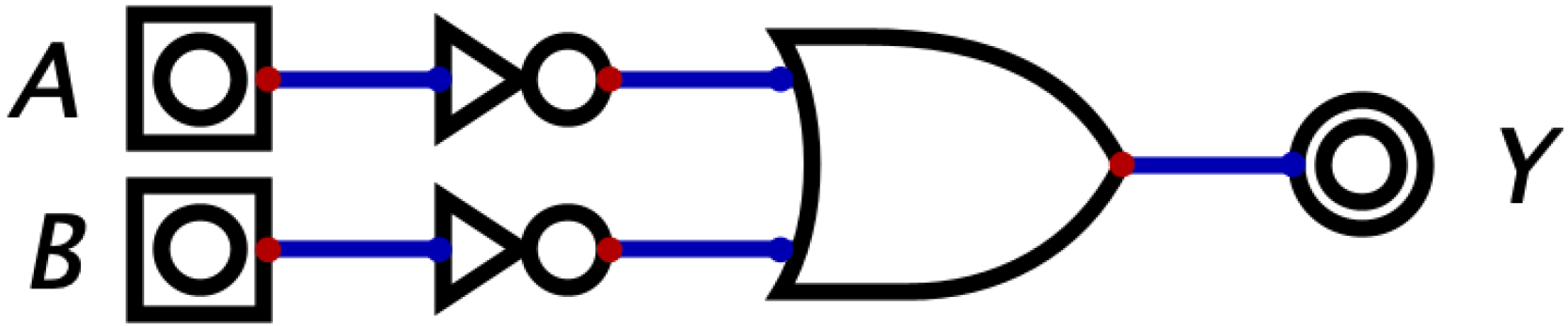


- We can represent the values of our signals over time using a waveform diagram
- Each one-bit signal will have its own waveform
- The waveforms are square waves where
  - a low value represents logic 0
  - a high value represents logic 1

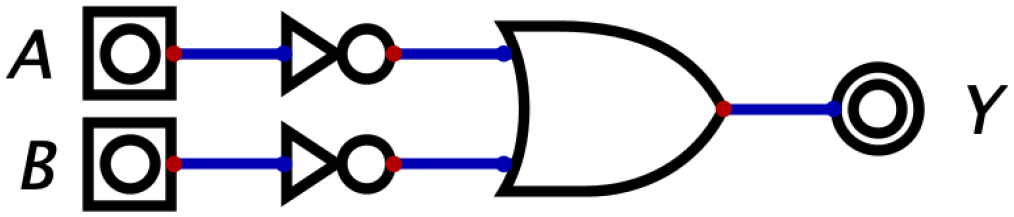
Ex1: Waveform Diagram

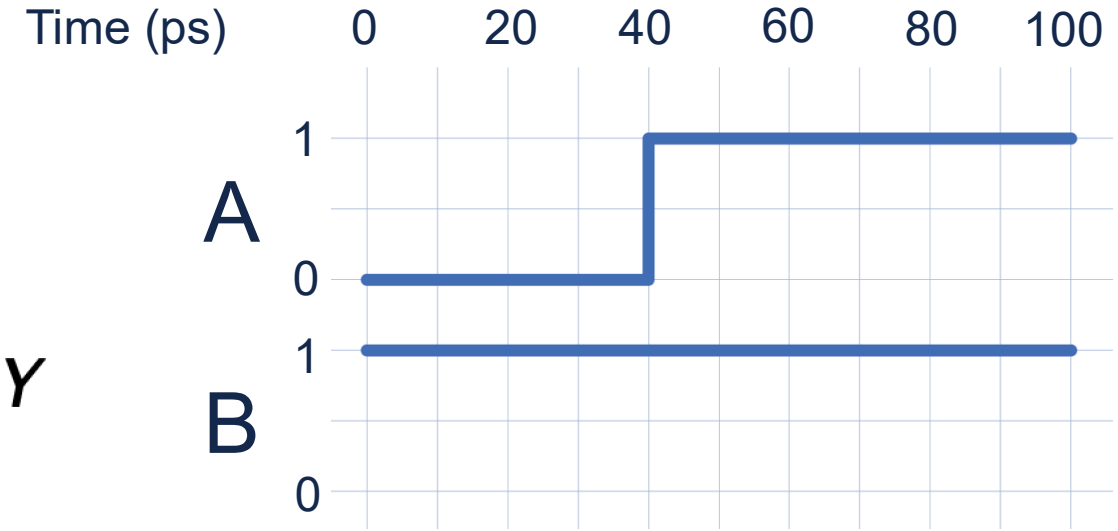


- We are going to analyze how the output of this circuit changes over time given the following changes to the circuit's inputs
  - At time t = 0ps, we will set the input A to 0 and the input B to 1
  - At time t = 40ps, we will change the input A to 1



Ex1: Waveform Diagram

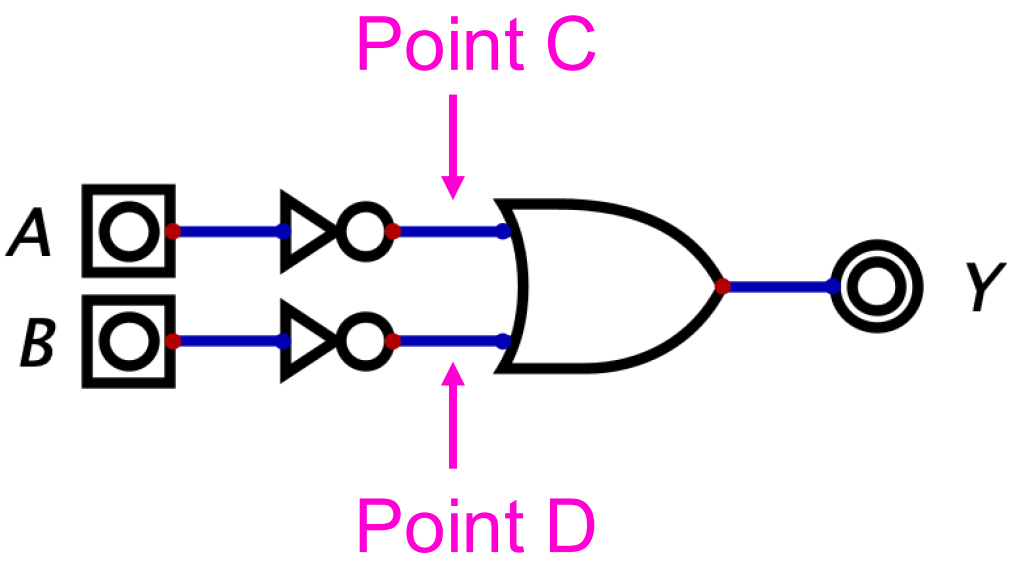


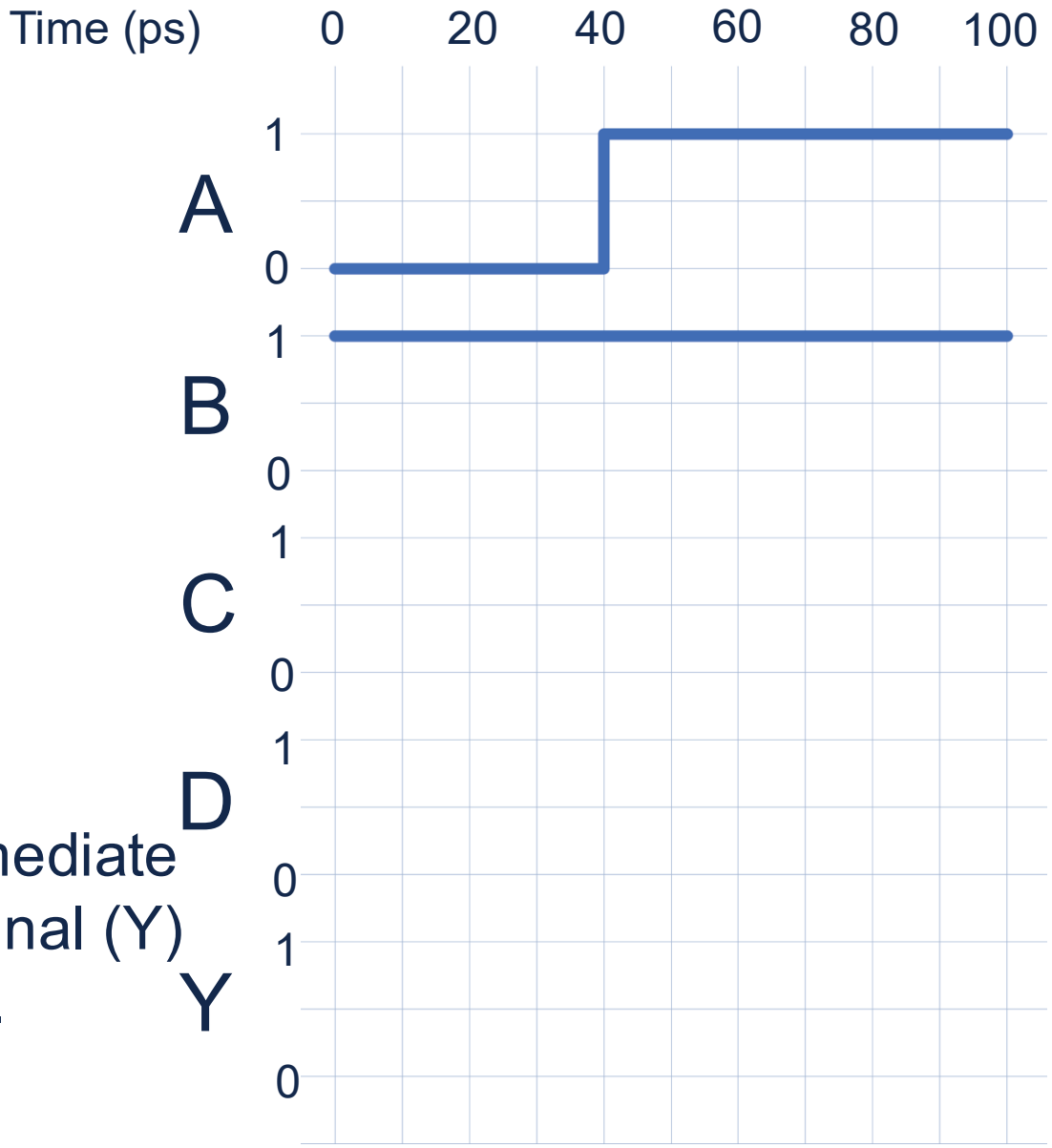


The waveform diagram above shows how we change the input values over time

- At time t = 0ps, we will set the input A to 0 and the input B to 1
- At time t = 40ps, we will change the input A to 1

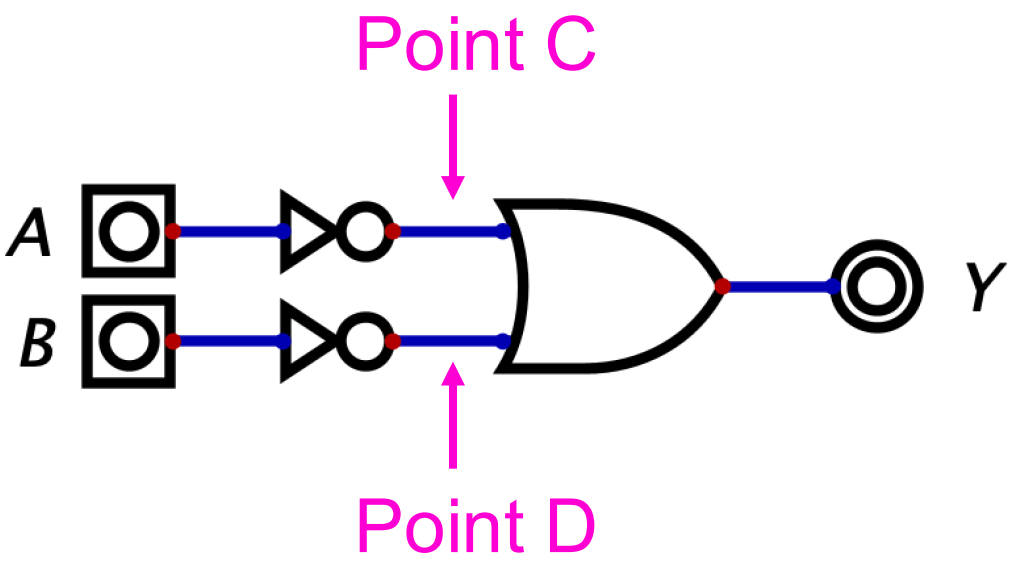
Ex1: Waveform Diagram

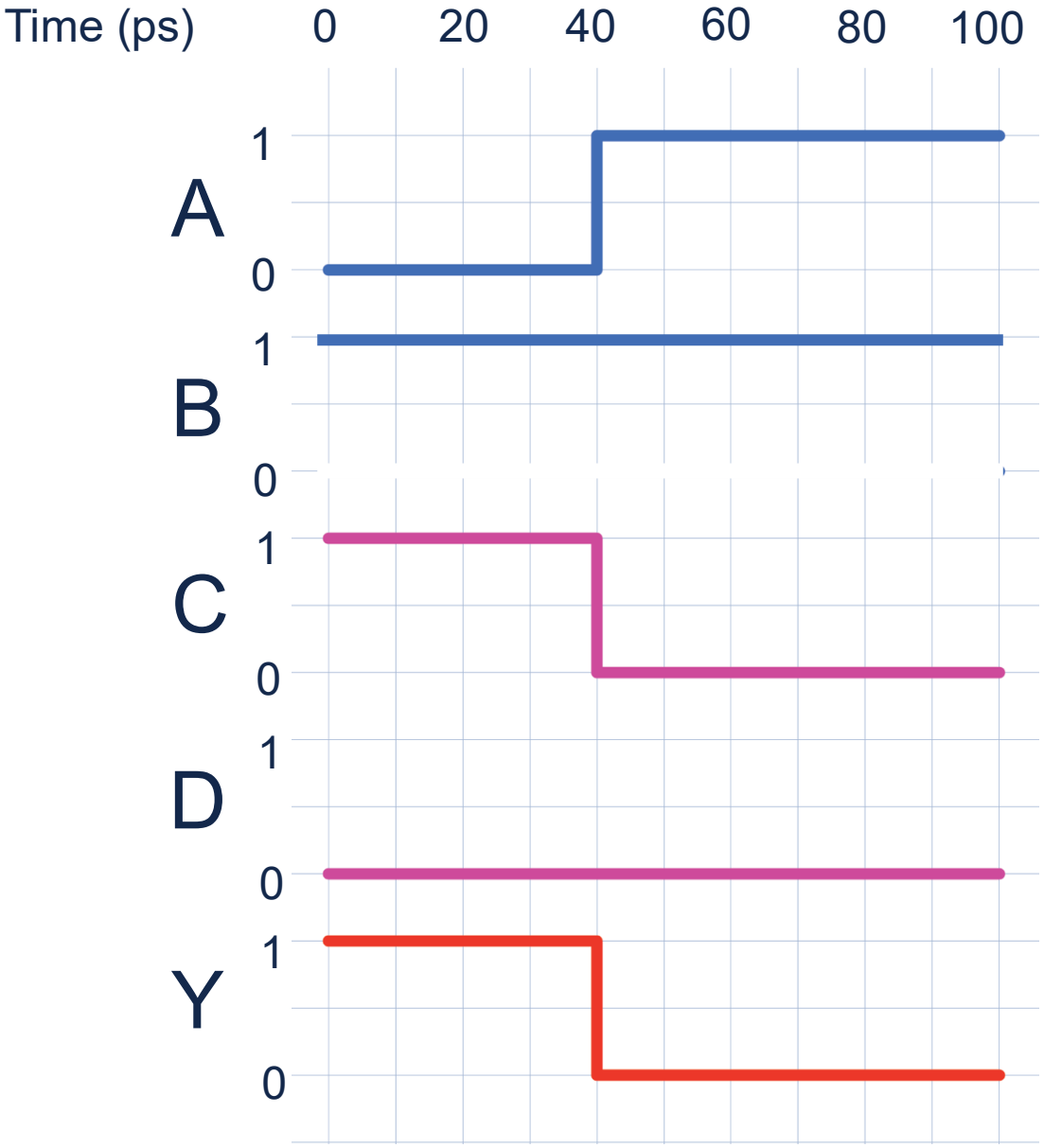




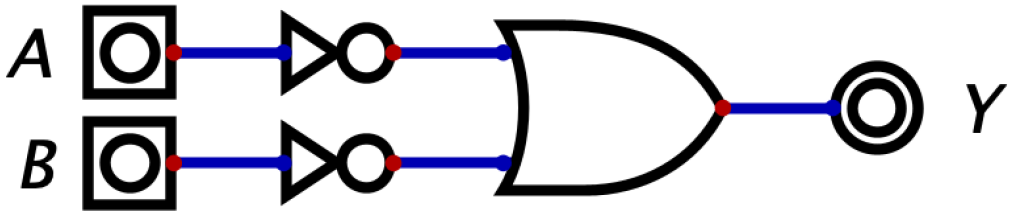
Now, we will analyze how the intermediate signals (C and D) and the output signal (Y) change as the inputs change.

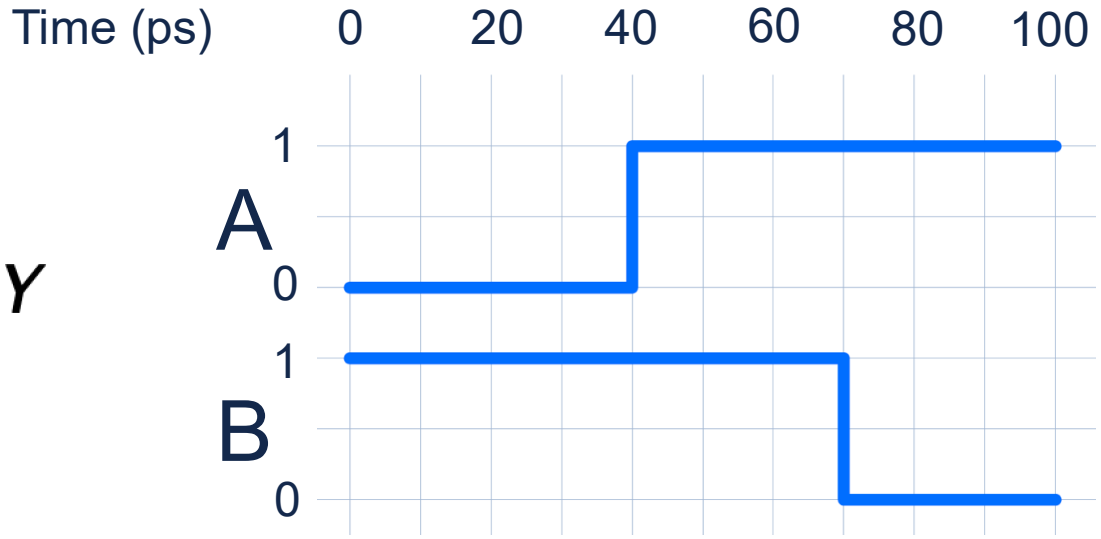
Ex1: Waveform Diagram





Ex2: Waveform Diagram

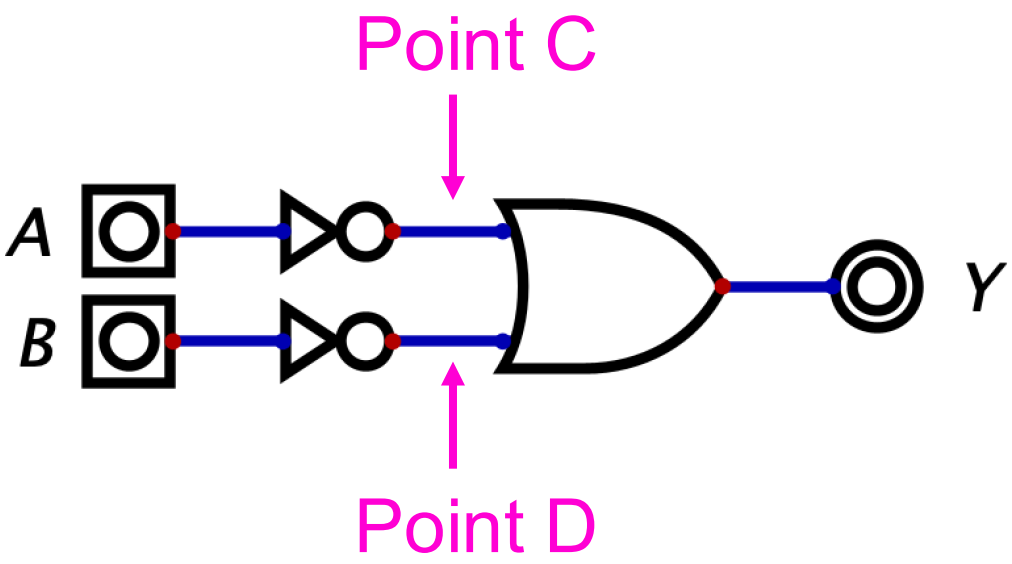


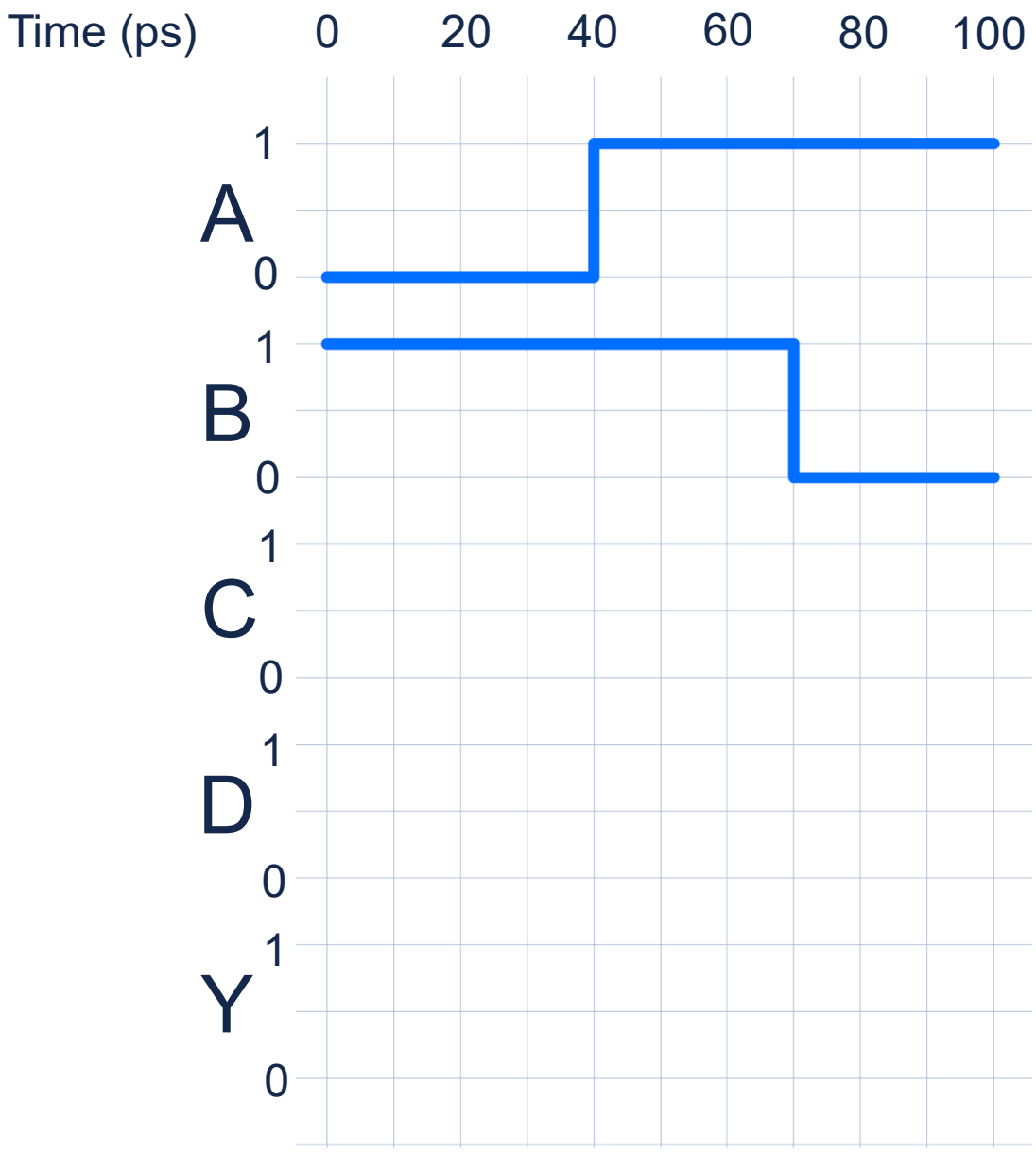


Let's look at another example with the same circuit

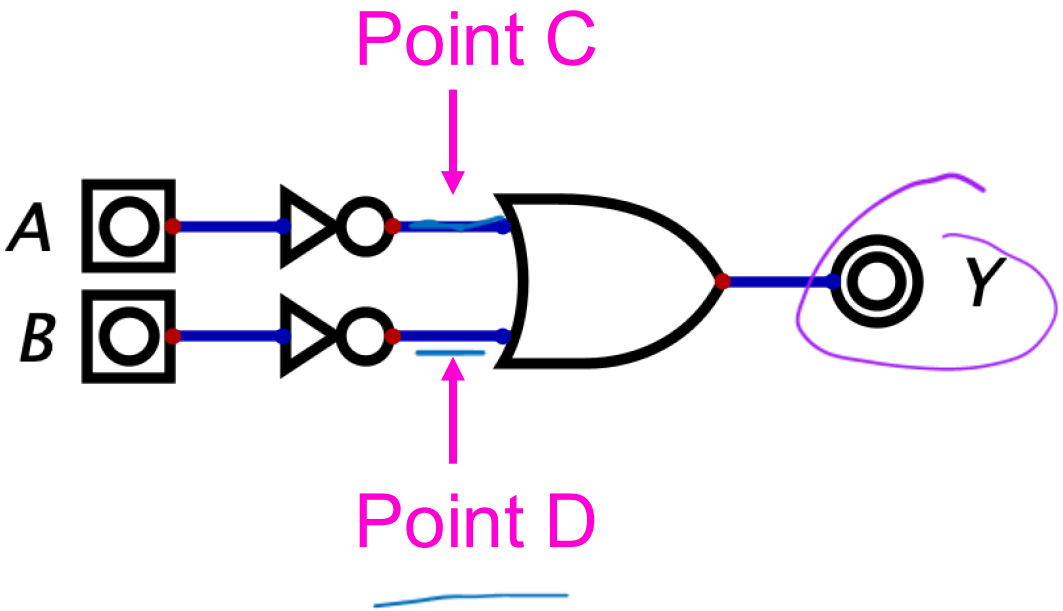
- At time t = 0ps, we will set the input A to 0 and the input B to 1
- At time t = 40ps, we will change the input A to 1
- At time t = 70ps, we will change the input B to 0

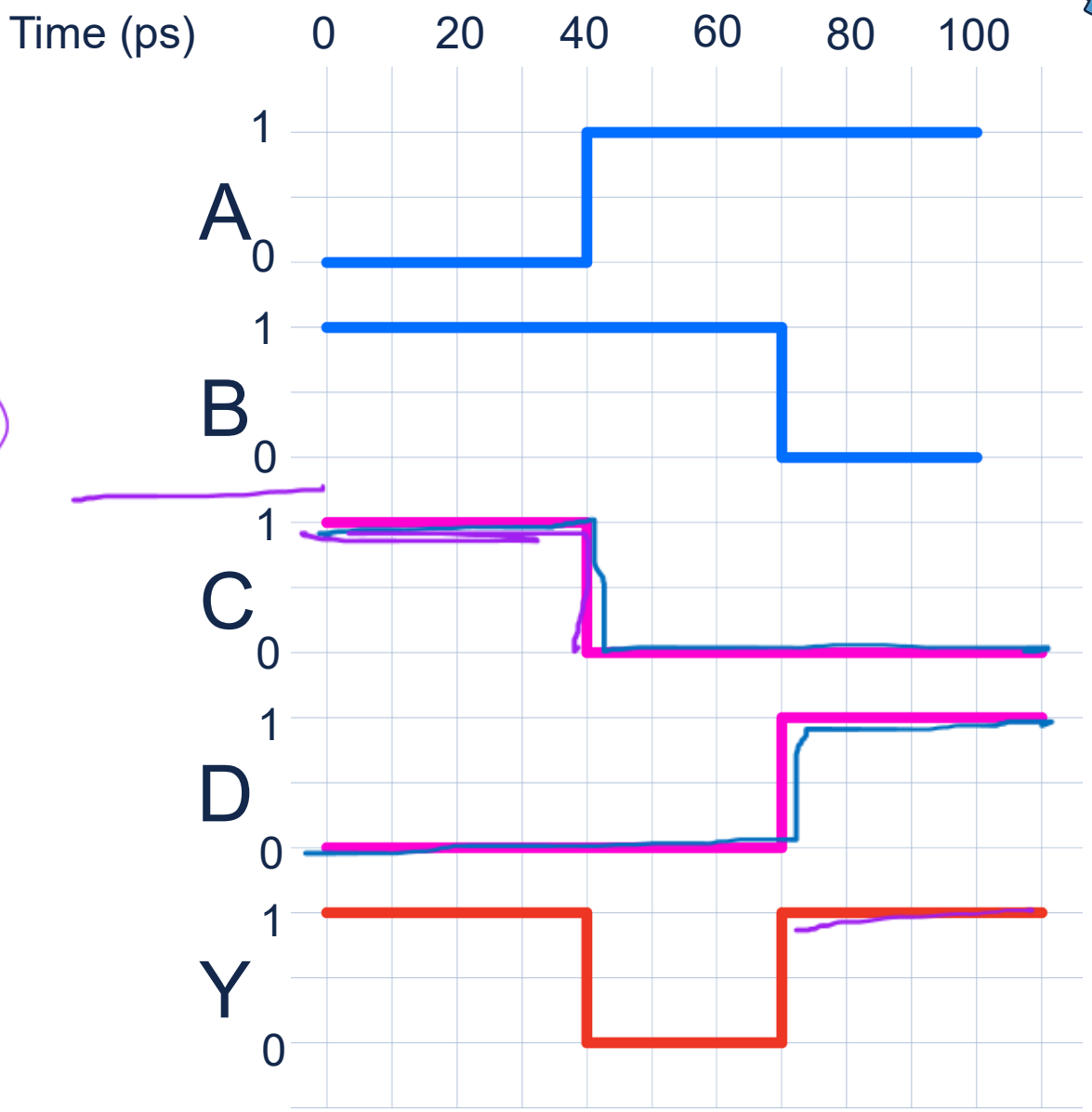
Ex2: Waveform Diagram



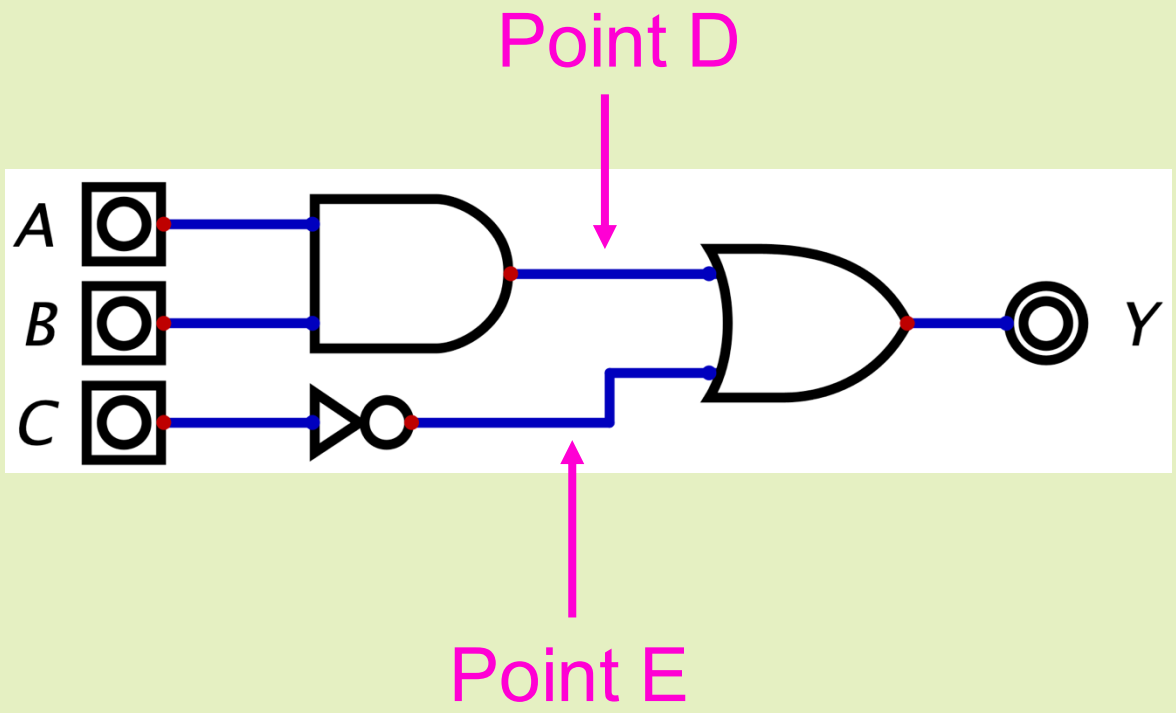


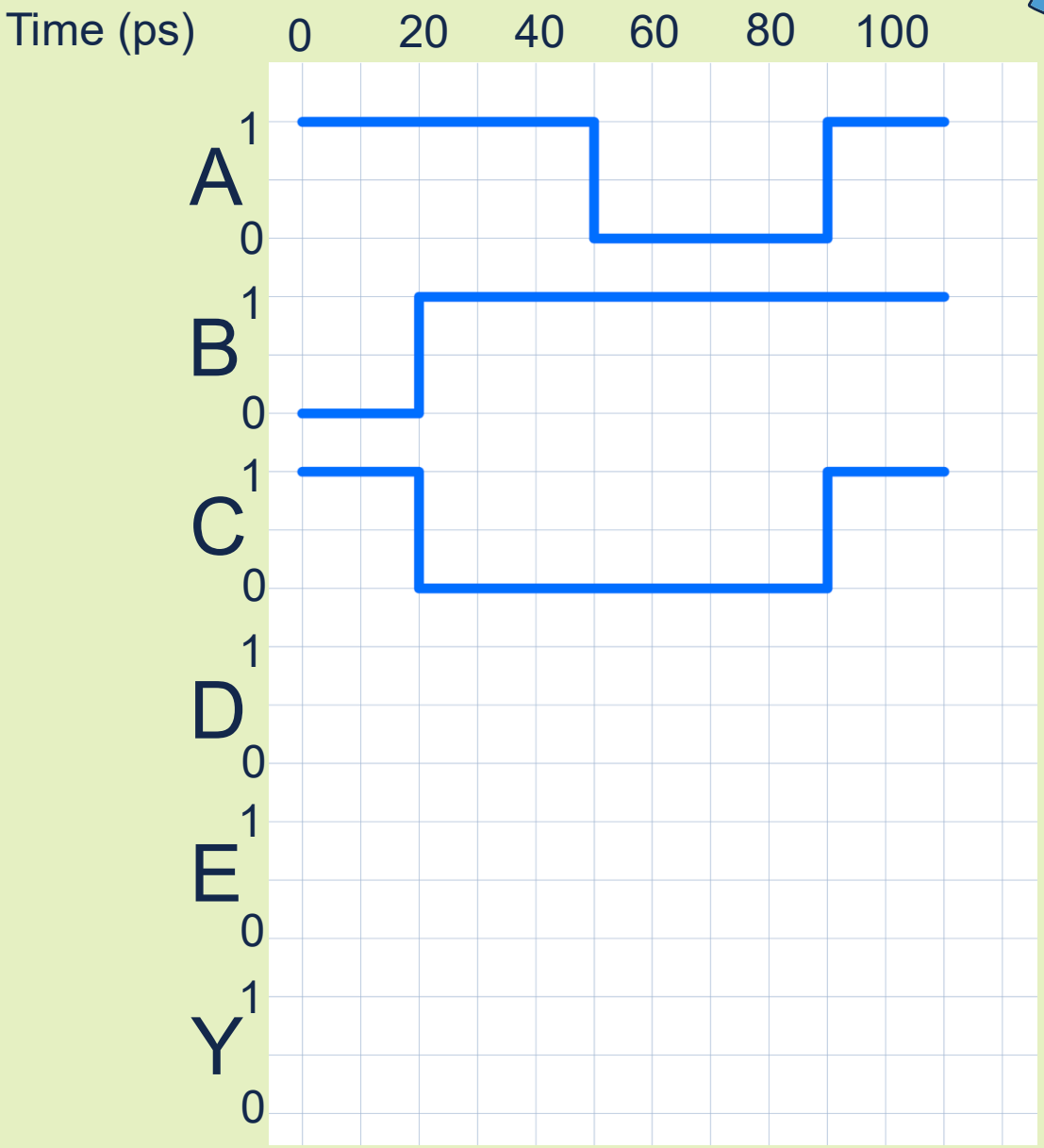
Ex2: Waveform Diagram





Q1: Waveform Diagram







Delays!

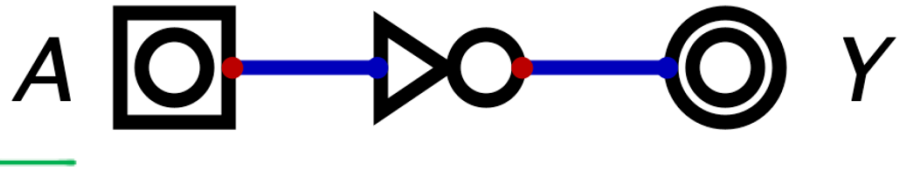
Propagation Delay



- The time taken for a signal to travel from an input to an output
- Causes
  - Time for electricity to travel through a wire
  - Capacitors charging/discharging
- The diagrams that we made on the previous slides were dealing with ideal circuits (i.e. circuits with no propagation delay)

Propagation delay (t_{pd})

- When we change the input to this inverter, the output will not change immediately.
- There will be a delay before the output changes



Propagation delay (t_{pd})



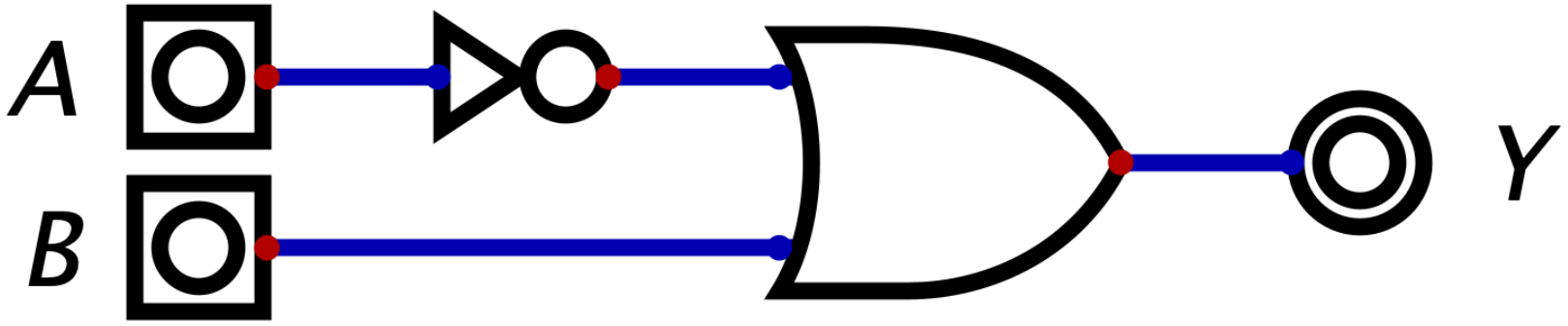
- Let's say the input to this inverter starts at 0.
- When we change the input to 1, the output will not become 0 immediately. There will be a delay.



A: 0 ➡ 1

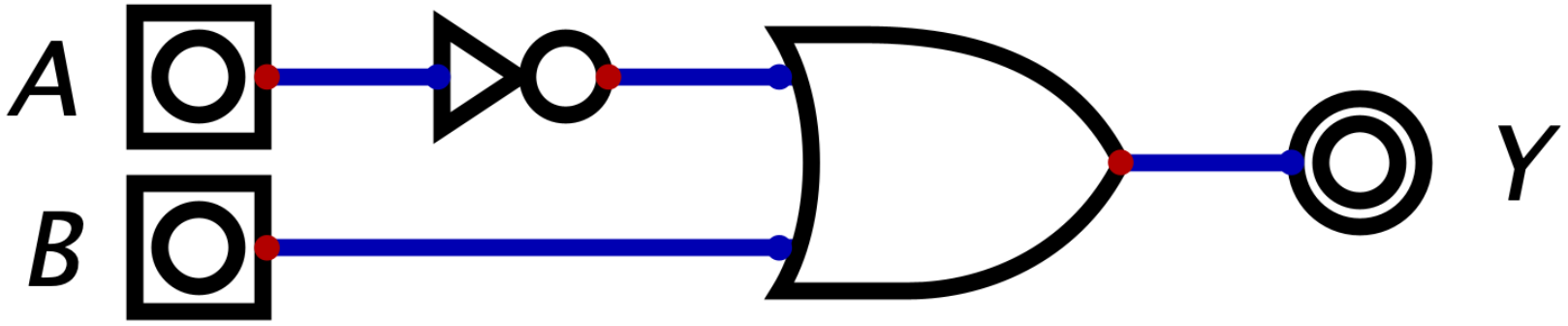
Y: 1 ➡ 0

Ex1: Computing Total Propagation Delay



Gate	t_{pd} (ps)
NOT	10
OR	20

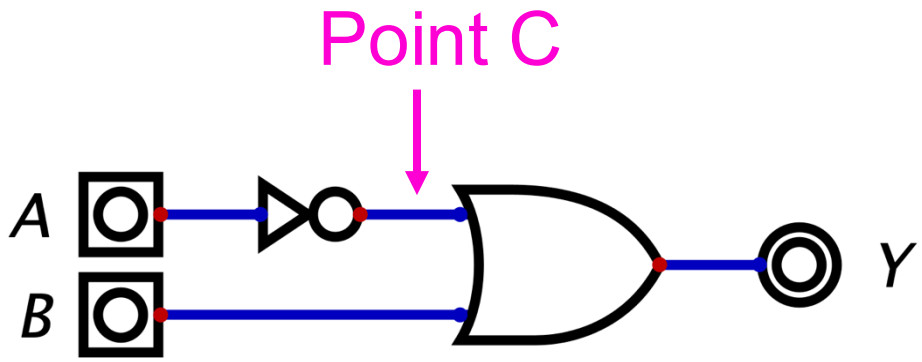
Ex1: Computing Total Propagation Delay



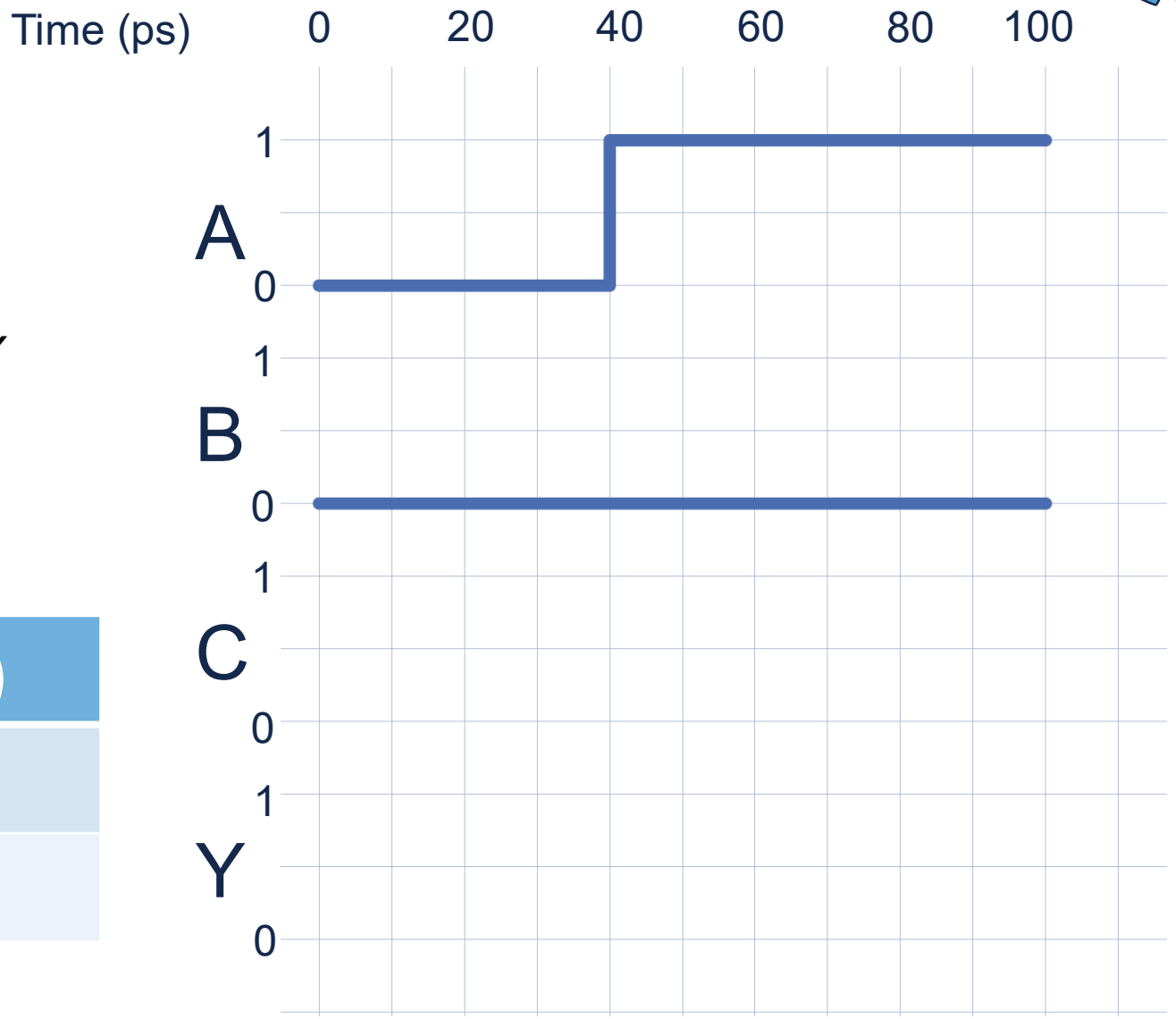
Gate	t_{pd} (ps)
NOT	10
OR	20

Delay from A to Y = 10ps + 20ps = 30ps
Delay from B to Y = 20ps

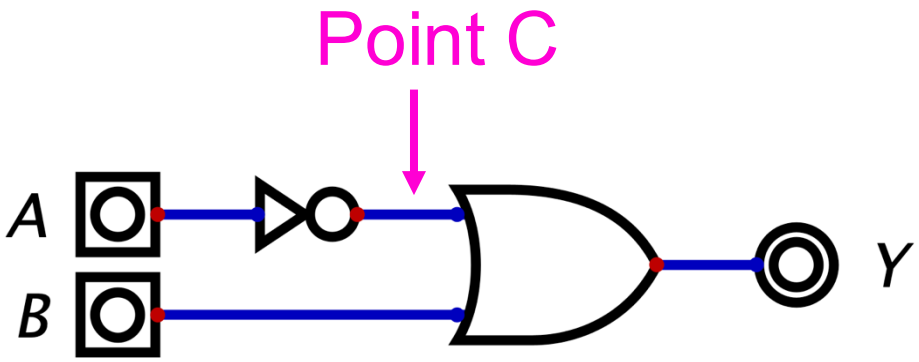
Ex1: Waveform Diagram with Propagation Delay



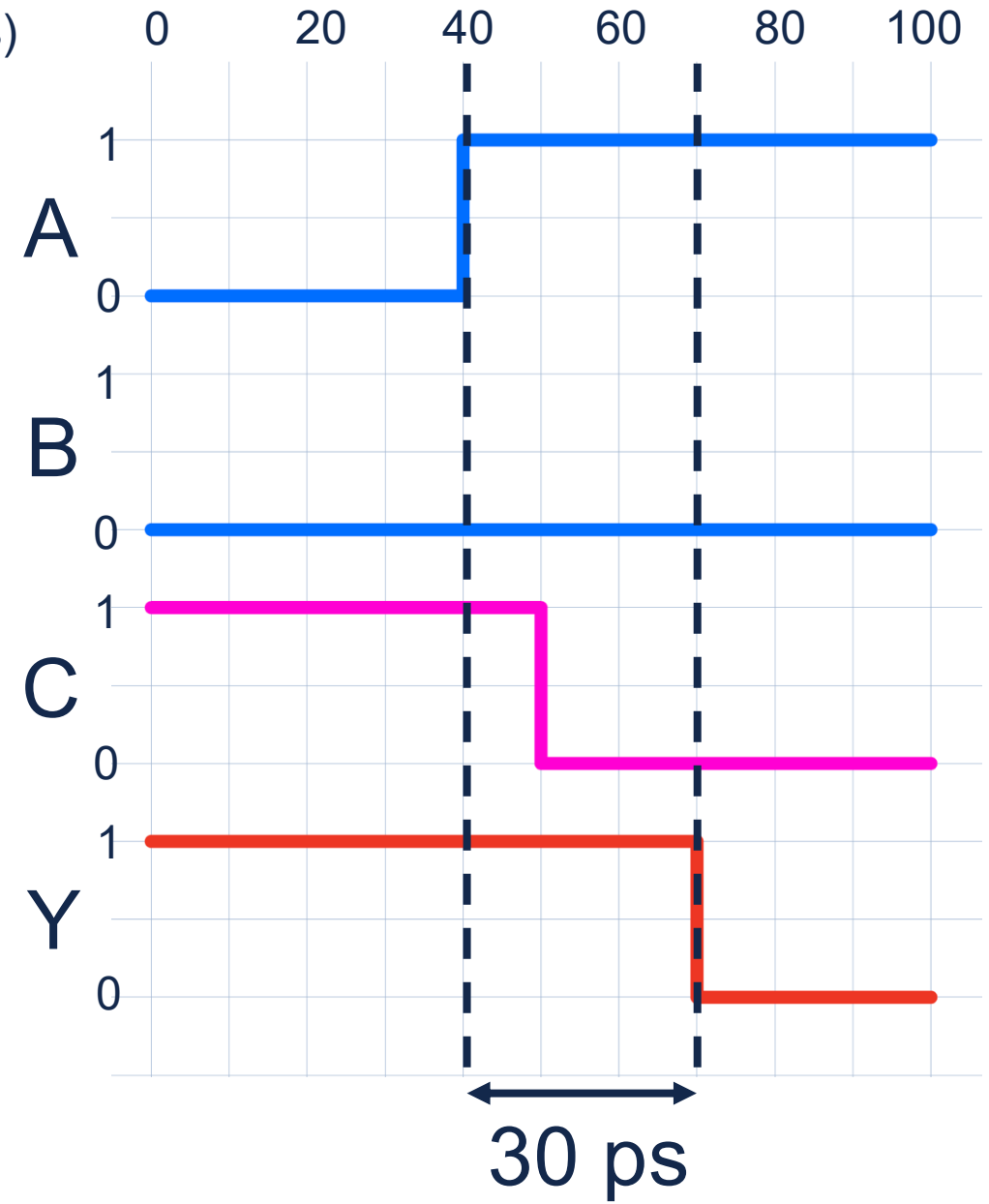
Gate	t_{pd} (ps)
NOT	10
OR	20



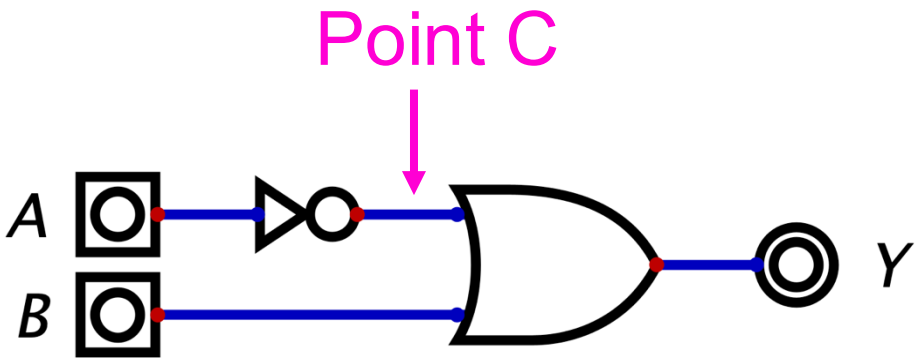
Ex1: Waveform Diagram with Propagation Delay



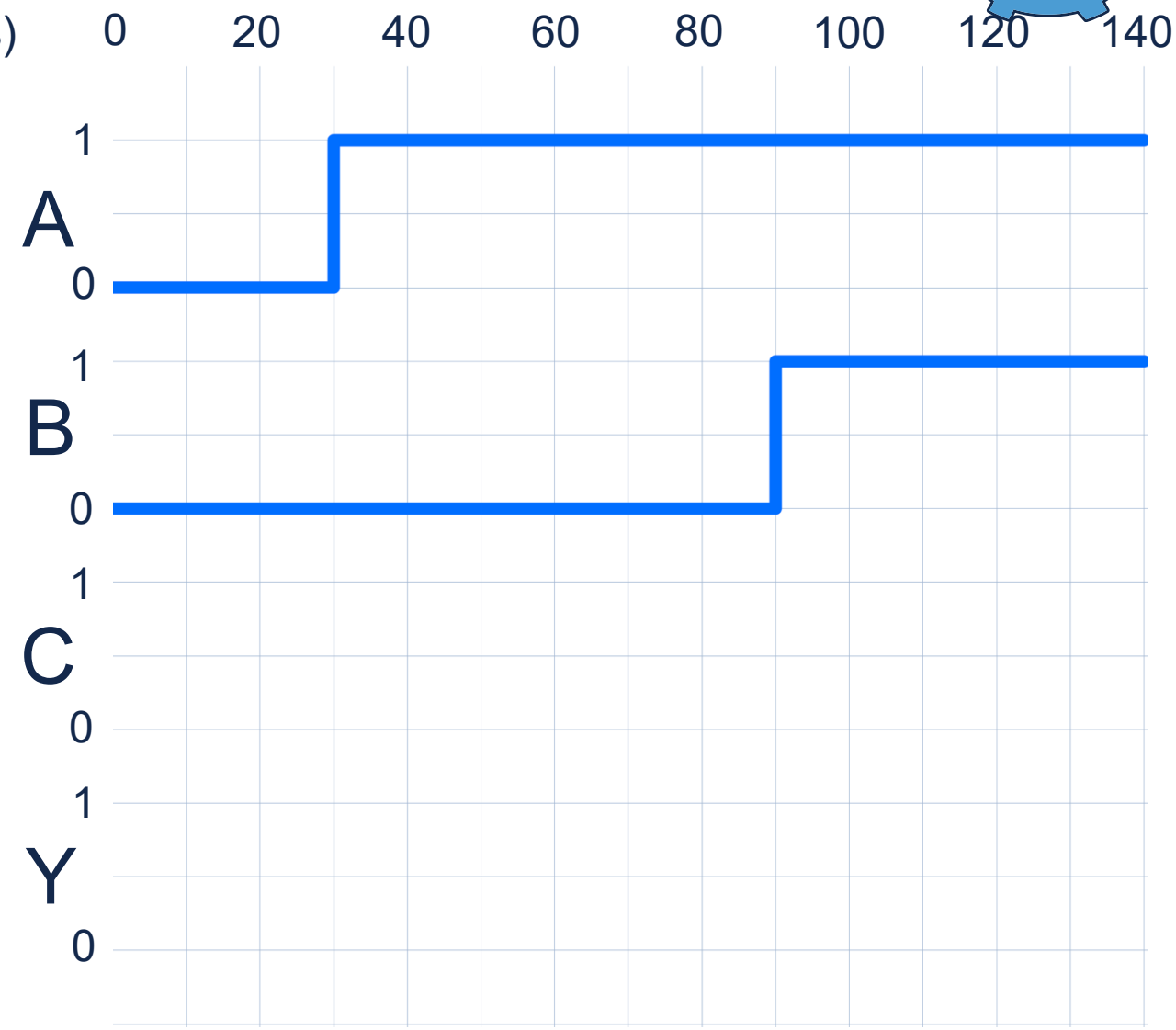
Gate	t_{pd} (ps)
NOT	10
OR	20



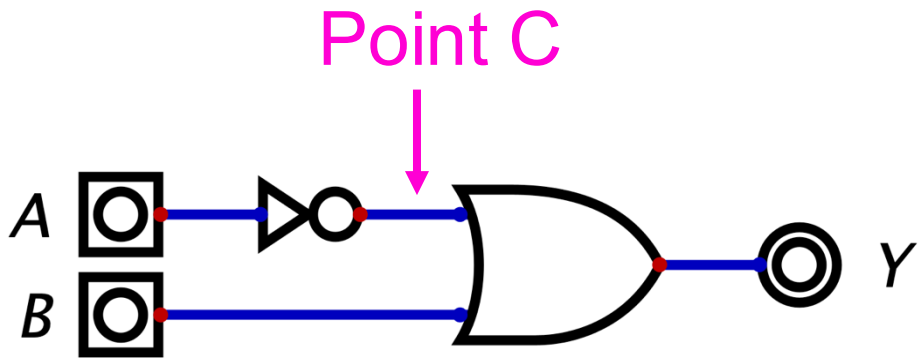
Ex2: Waveform Diagram with Propagation Delay



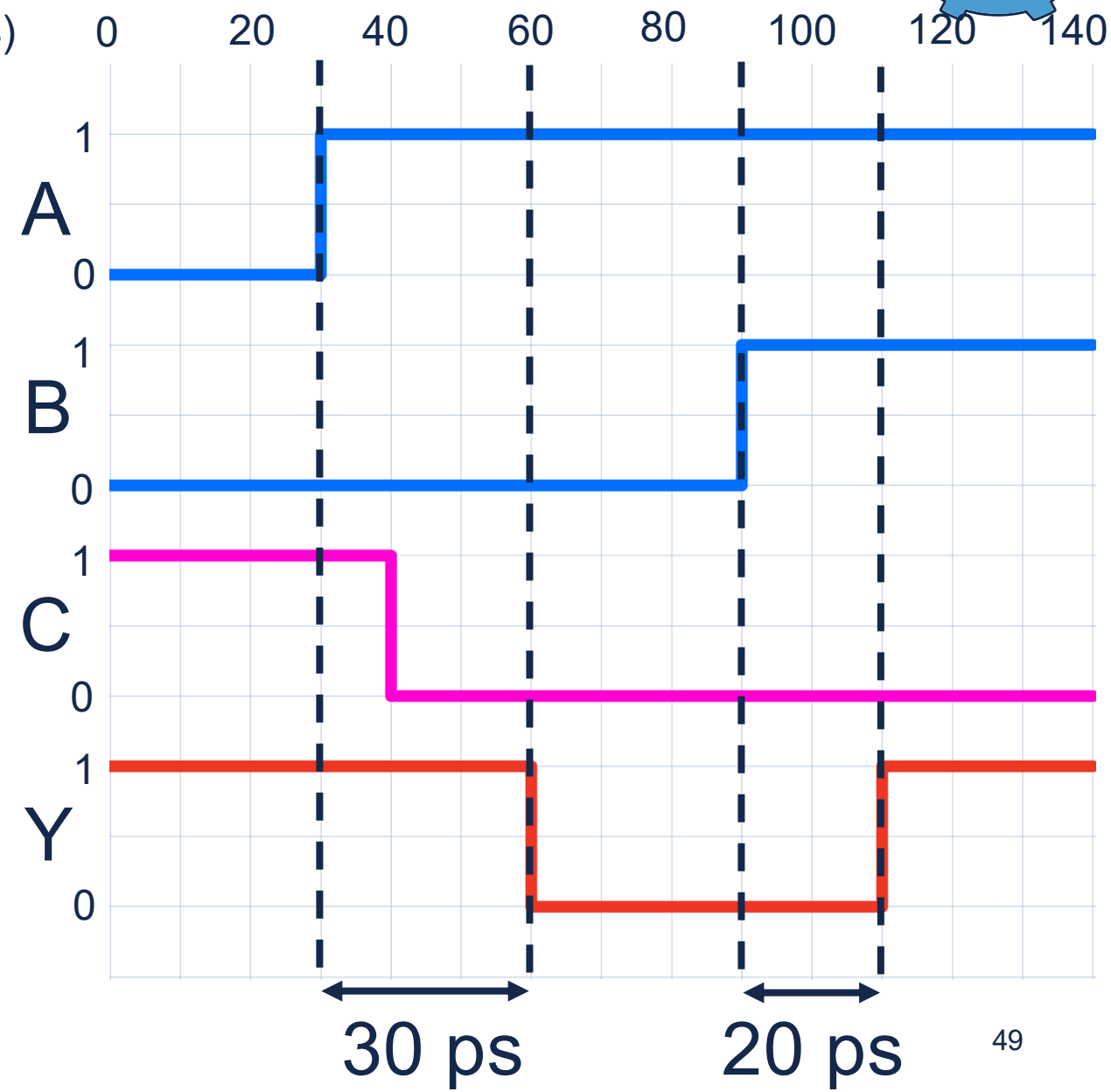
Gate	t_{pd} (ps)
NOT	10
OR	20



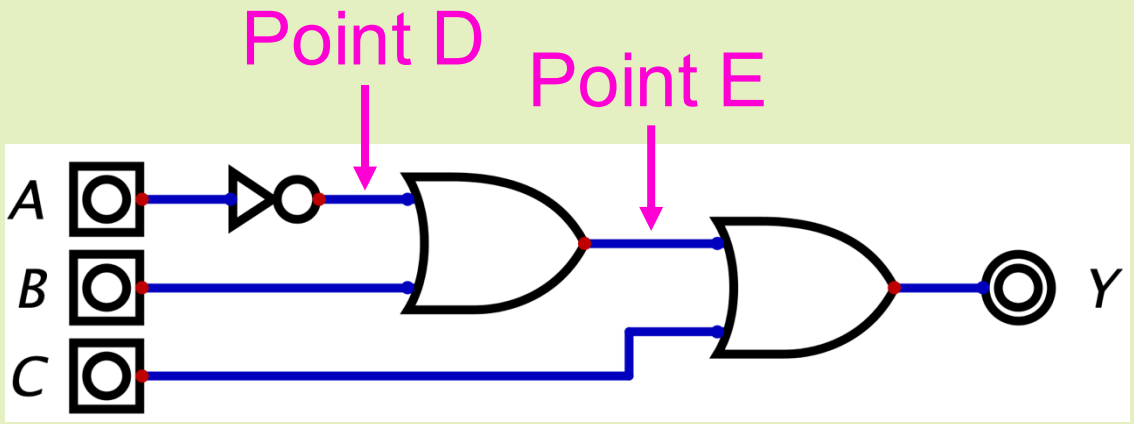
Ex2: Waveform Diagram with Propagation Delay



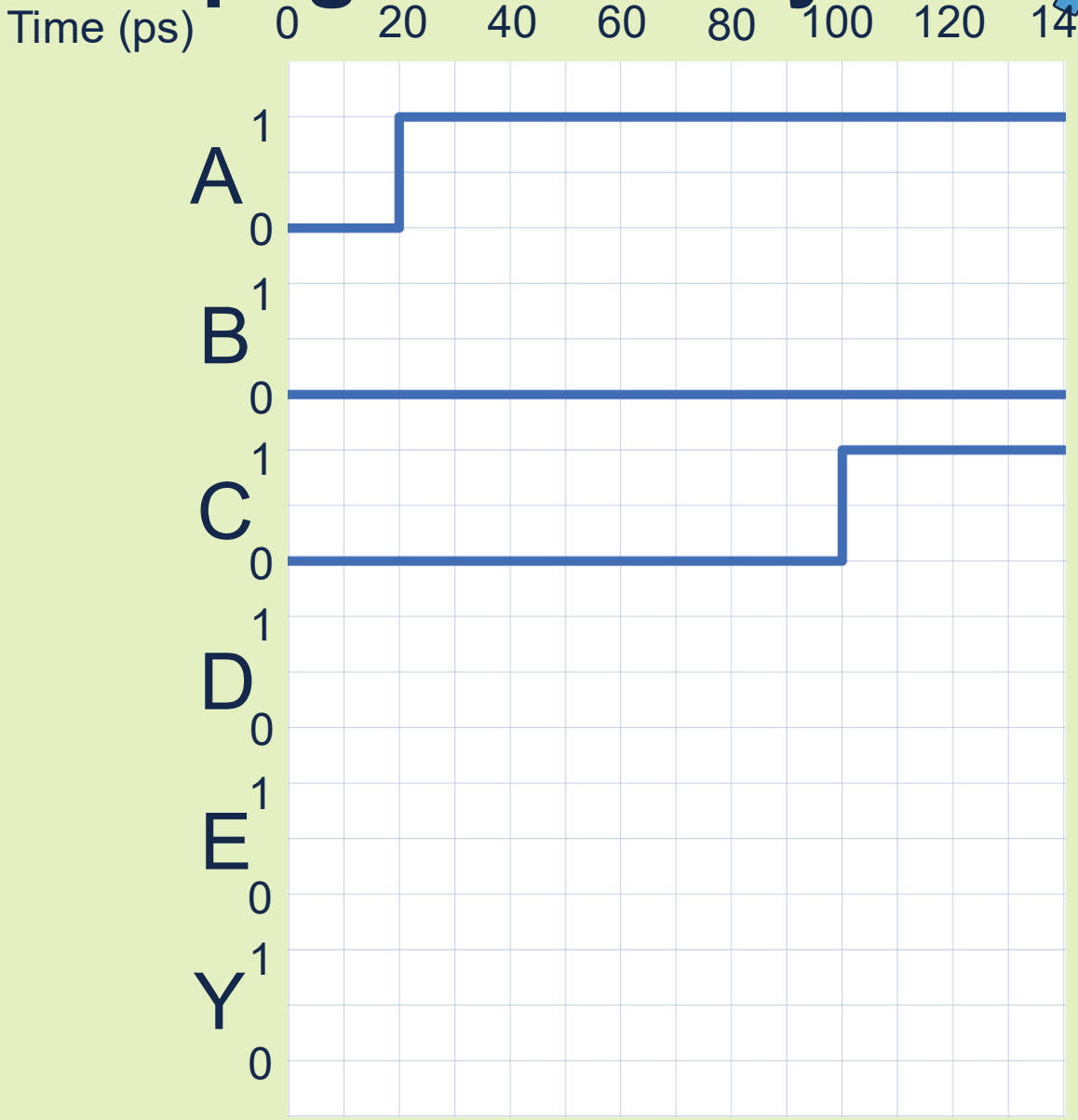
Gate	t_{pd} (ps)
NOT	10
OR	20



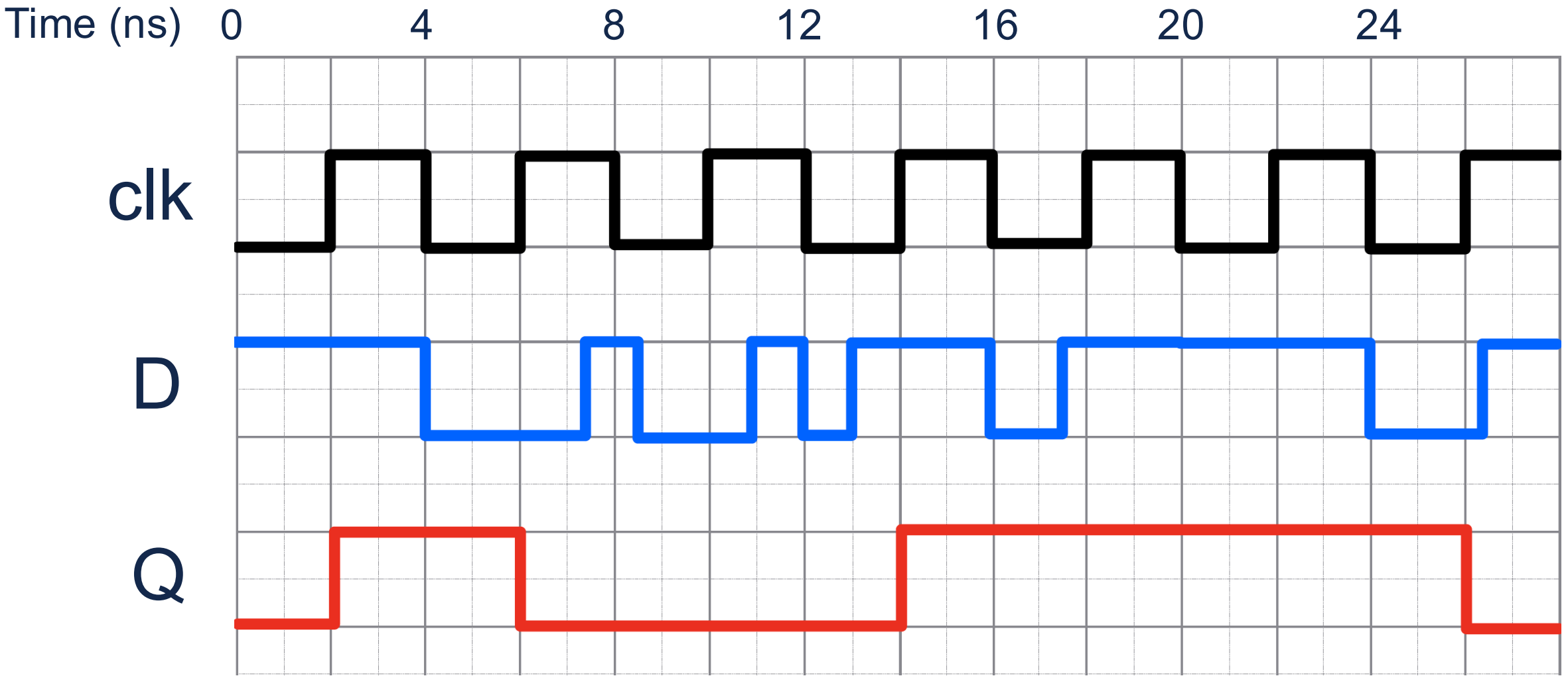
Waveform Diagram with Propagation Delay



Gate	t_{pd} (ps)
NOT	10
OR	30

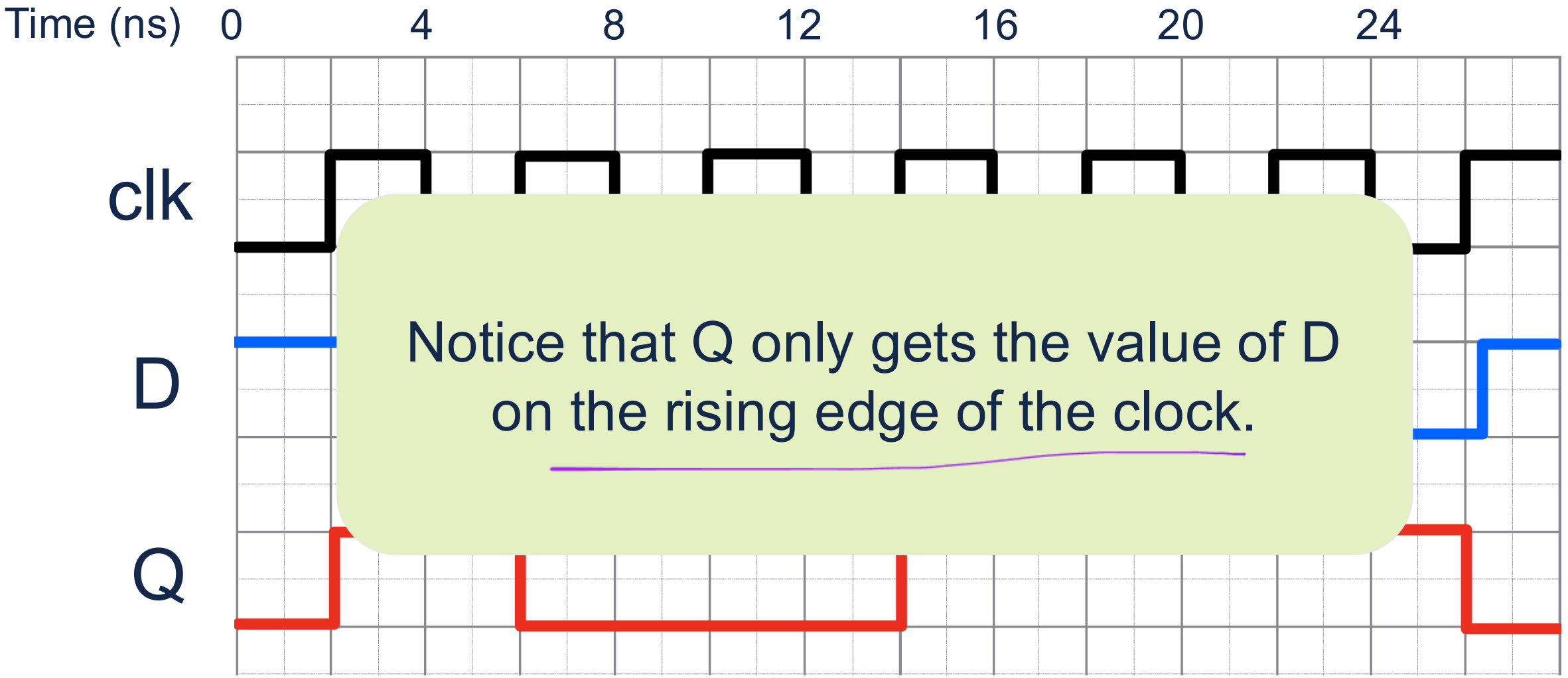


Positive Edge Triggered D Flip-Flop Waveform Diagram



assume Q starts at 0

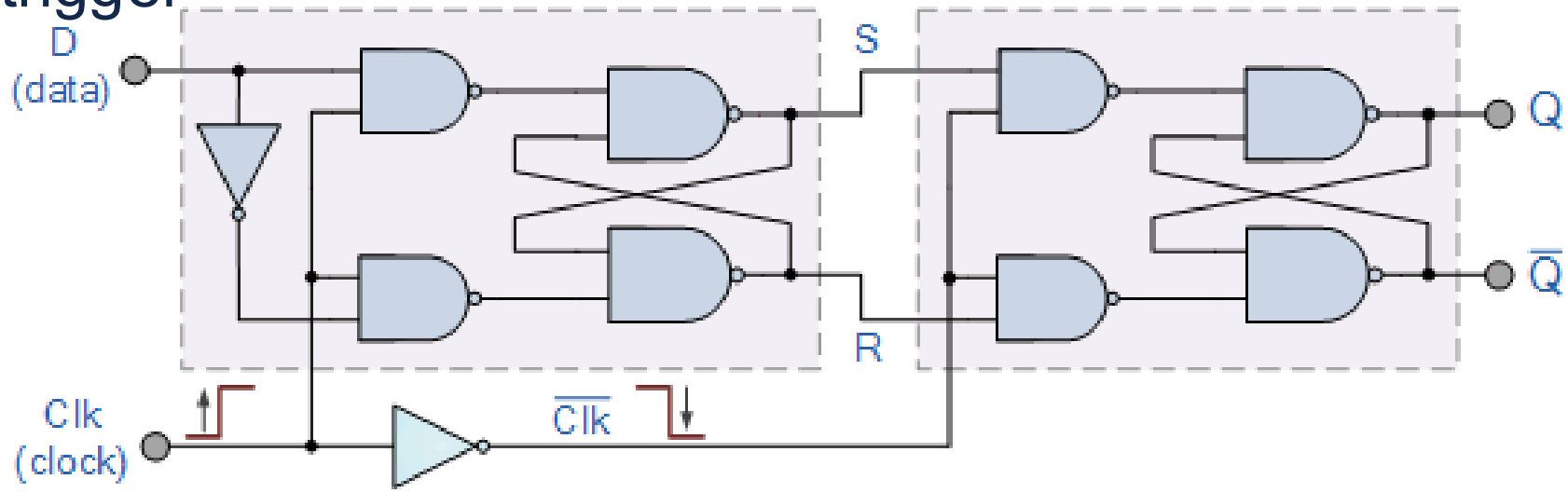
Positive Edge Triggered D Flip-Flop Waveform Diagram



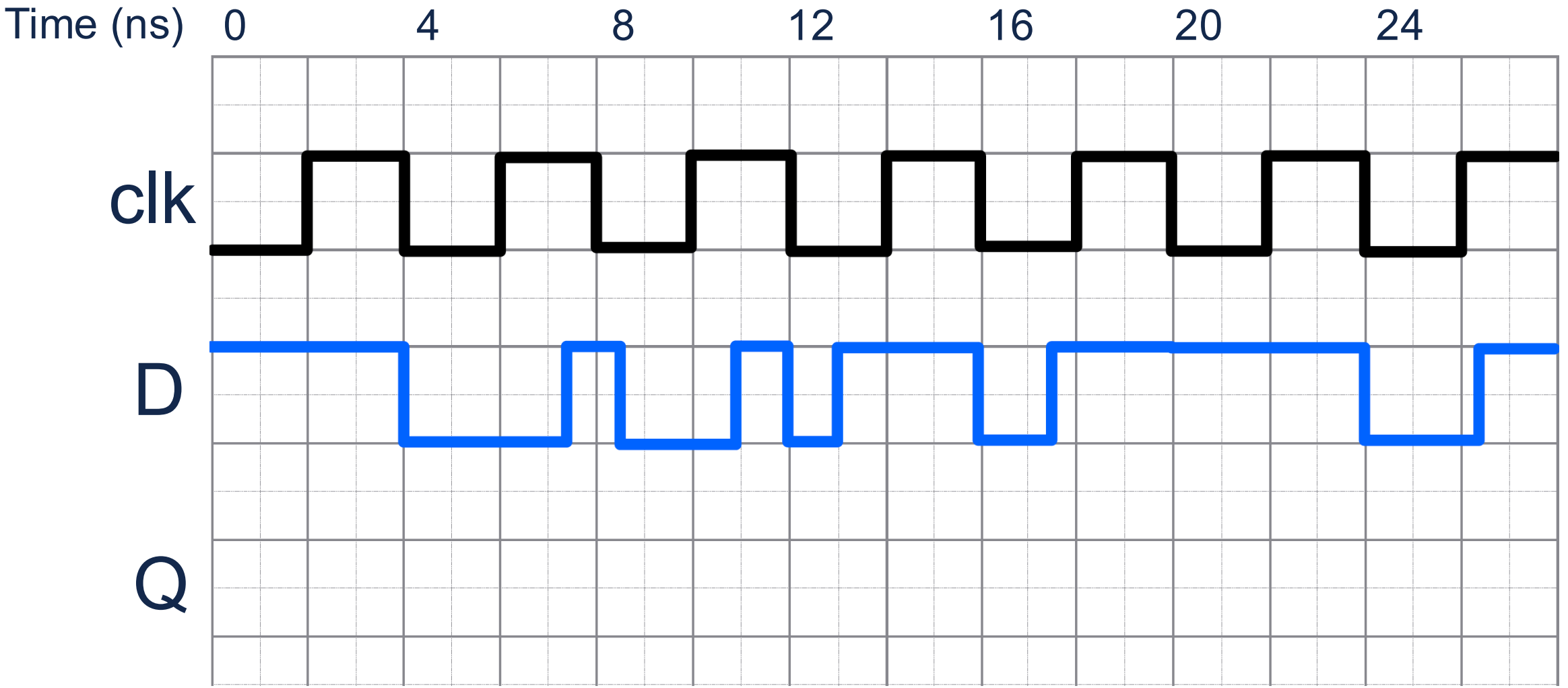
assume Q starts at 0

Clock to Q Delay

- The amount of time it takes for D to appear on the output after the clock trigger



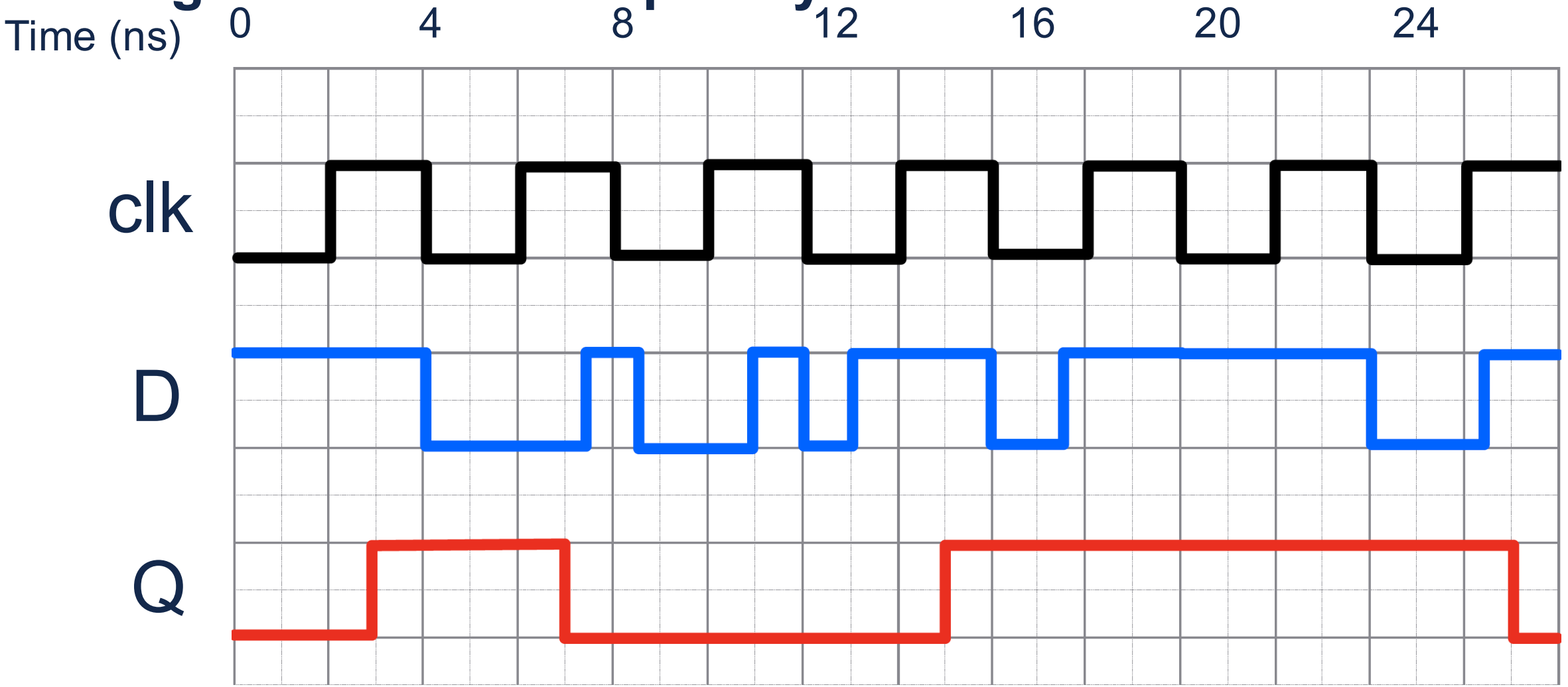
Positive Edge Triggered Flip-Flop Waveform Diagram with clk-to-q Delay



assume Q starts at 0

clk-to-q delay = 1 ns

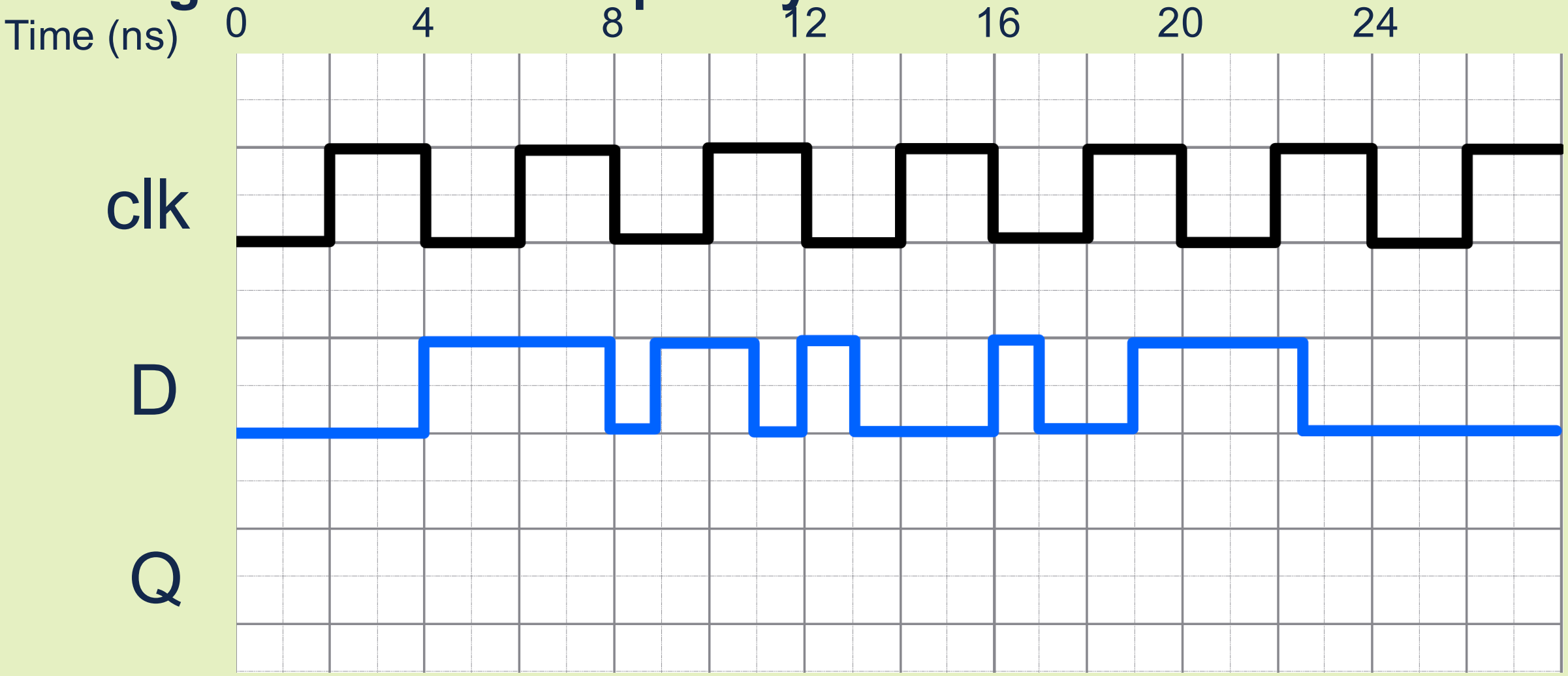
Positive Edge Triggered Flip-Flop Waveform Diagram with clk-to-q Delay



assume Q starts at 0

clk-to-q delay = 1 ns

Positive Edge Triggered Flip-Flop Waveform Diagram with clk-to-q Delay



assume Q starts at 0

clk-to-q delay = 1 ns