COMP311: *COMPUTER ORGANIZATION!*

Lecture 13: Intro to Timing Analysis

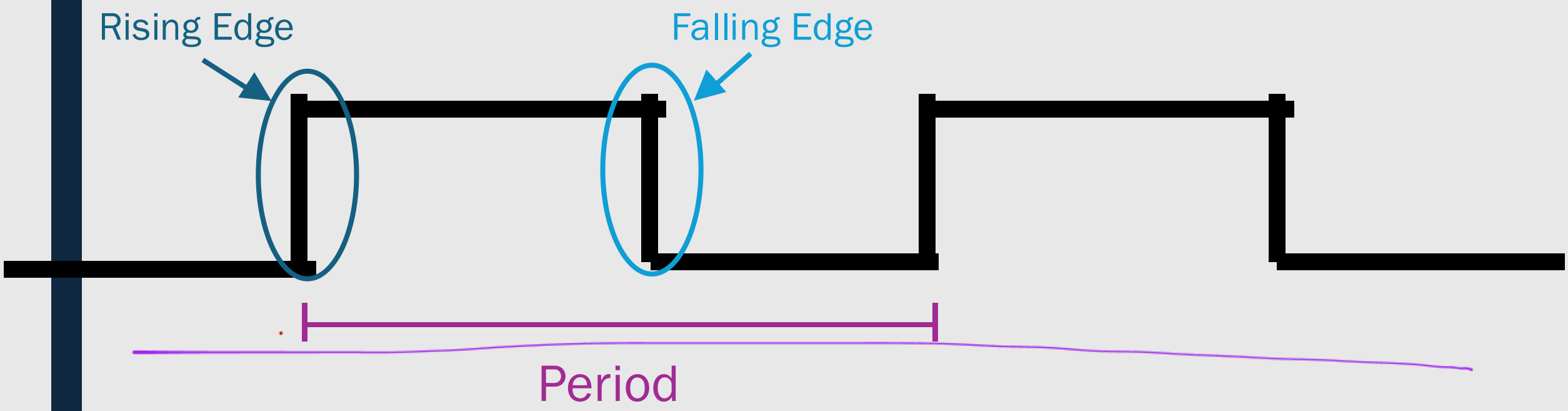
tinyurl.com/comp311-sp26

FLIP FLOPS AND REGISTERS: *MORE DETAILS*

Combinational vs. Sequential 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)*
- **Sequential logic:**
  - *Output depends on current input && past state*
  - *This past state is stored in a register!*
  - *Synchronous logic: a type of sequential logic where combinational logic blocks ~synced~ with a **global clock***

Clock

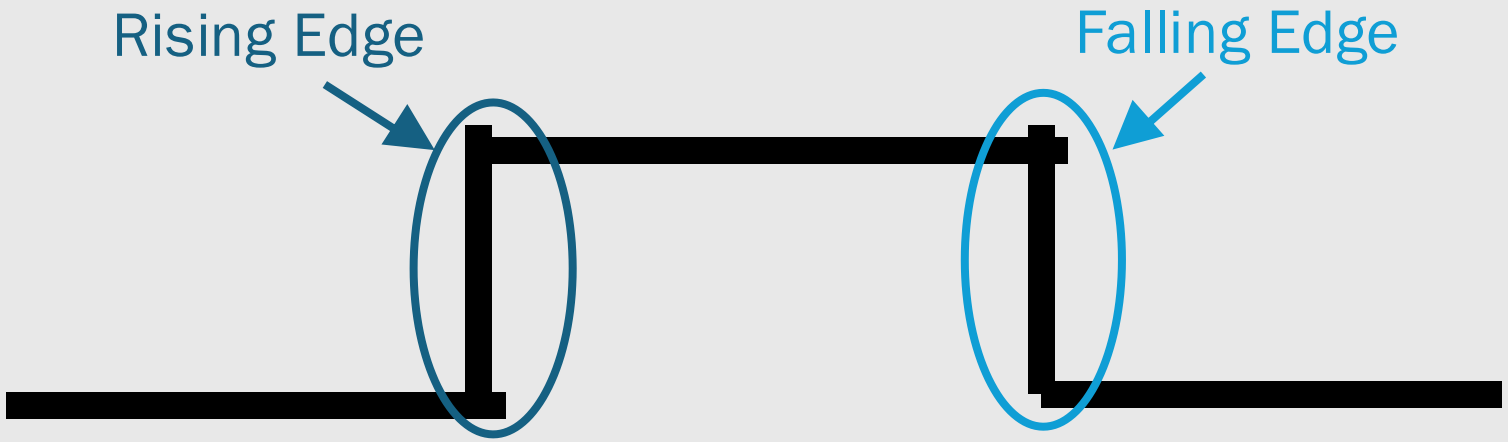


Clock

- 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*

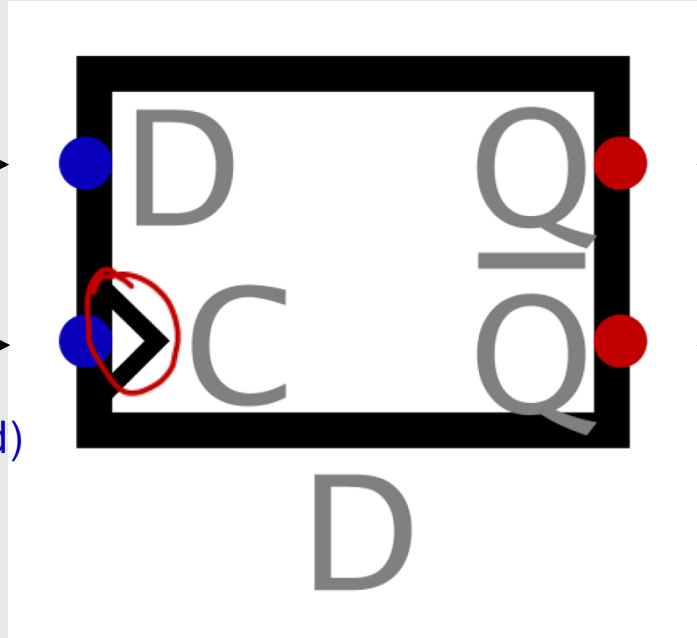


D Flip-Flop

Value to store

Clock

(Determines when D is stored)

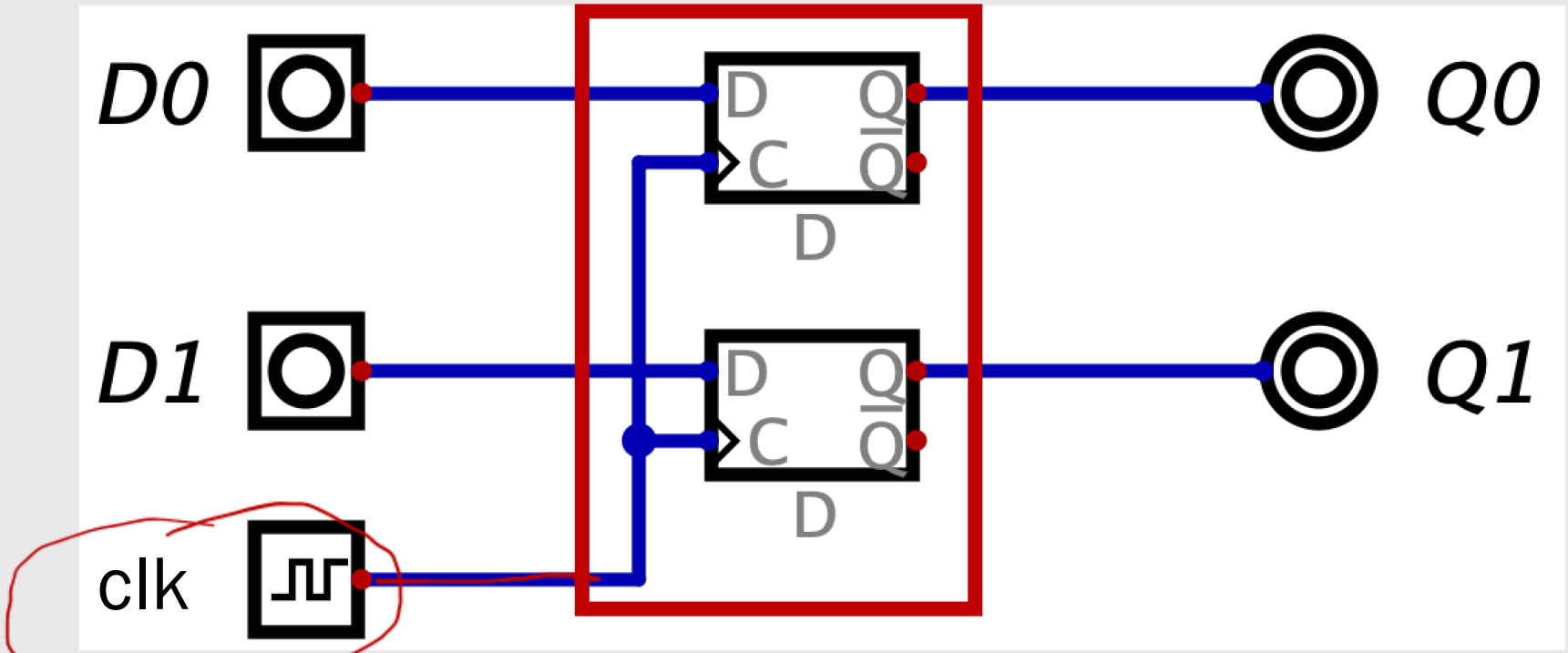


Stored Value

Inverted Stored Value

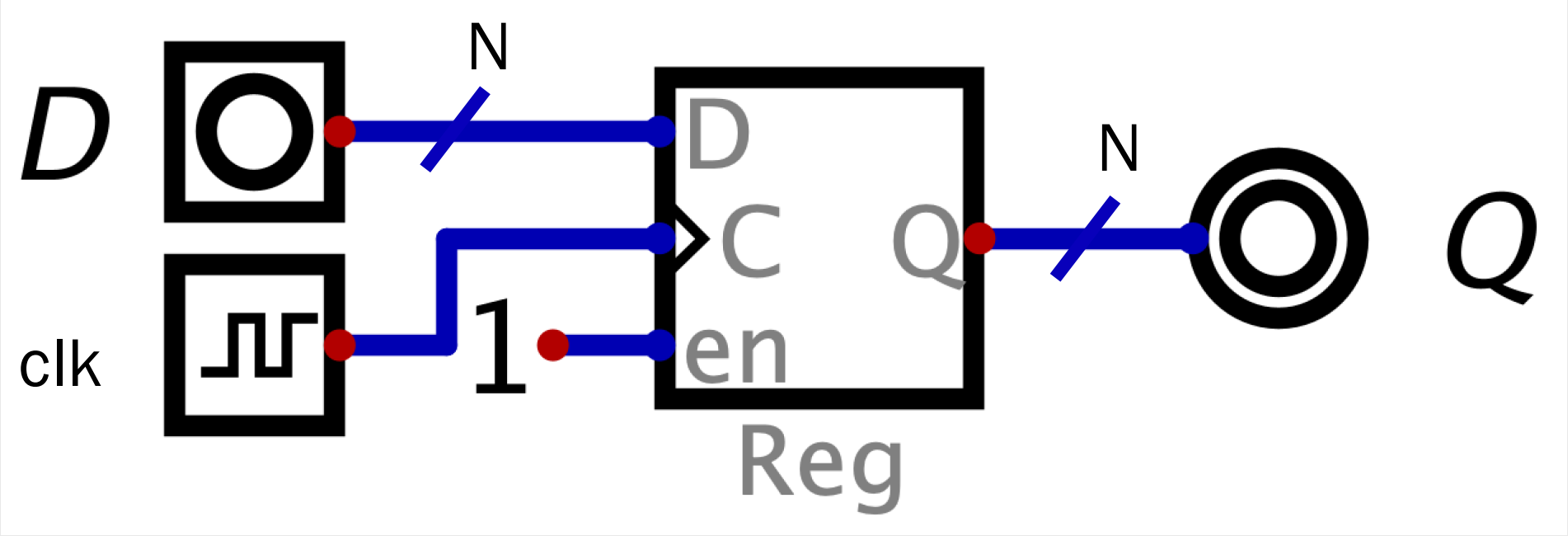
Registers

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

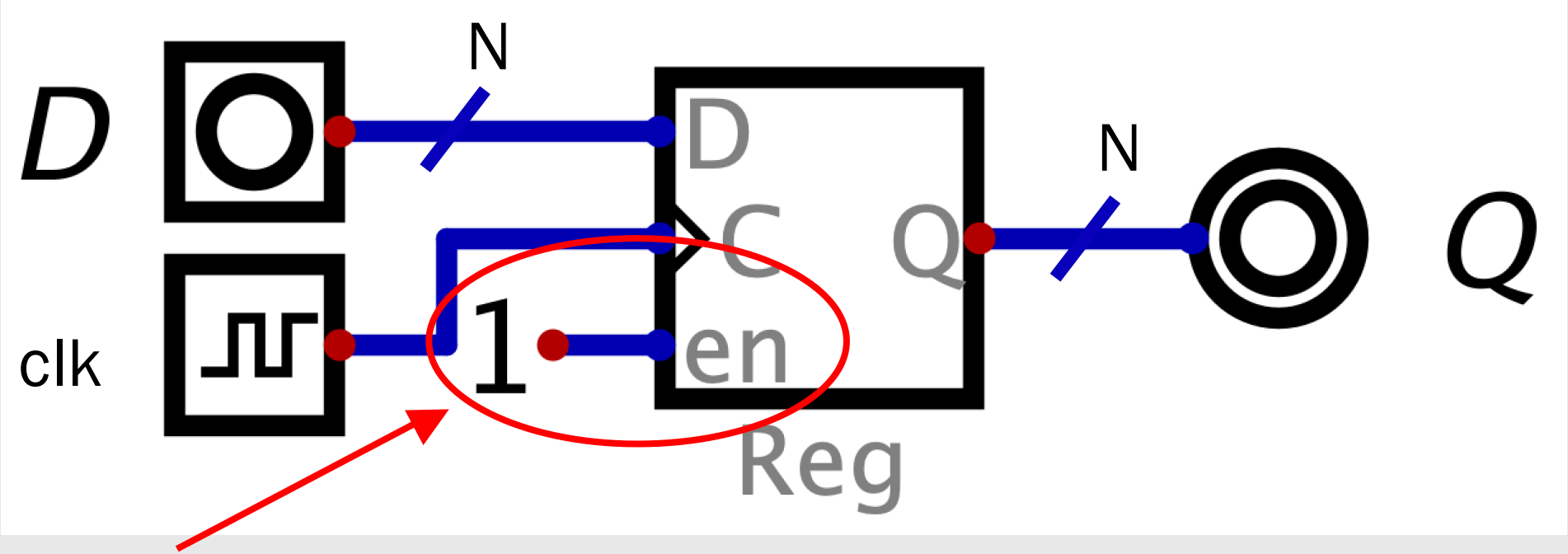


2-bit register

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

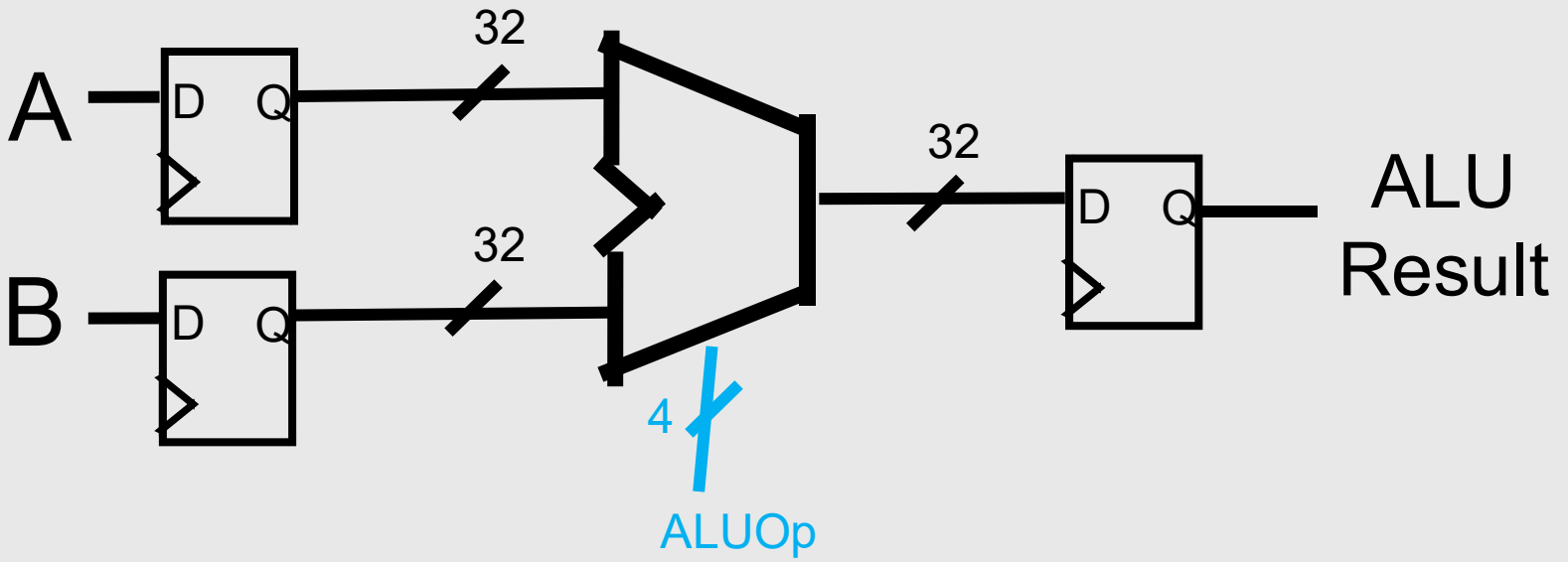
3 + 4

5 – 2

9 OR 4

6 AND 8

5 + 9



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

ALU

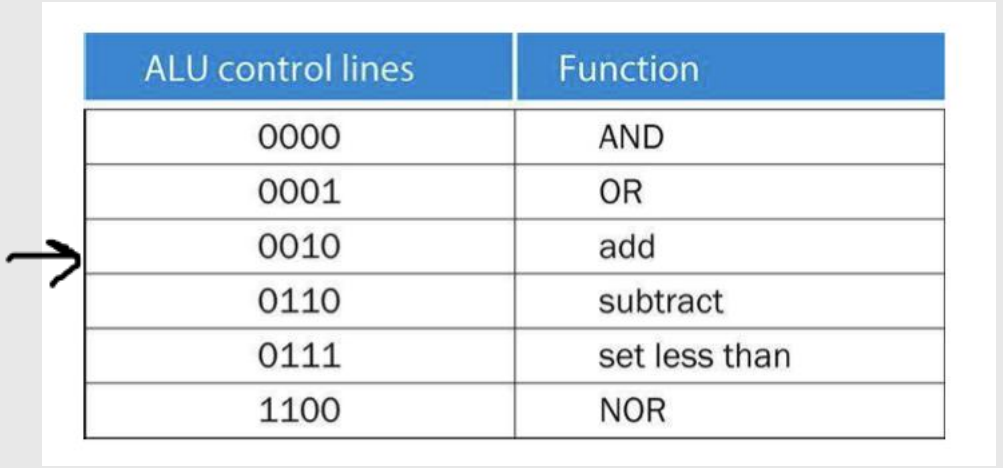
- The ALU will perform one operation per clock cycle
- To perform an operation, we have three inputs to set: A, B, and ALUOp
  - *(ALU Op (ALU Control line from the textbook) is also synchronized with a register, but that is abstracted away for now)*

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 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

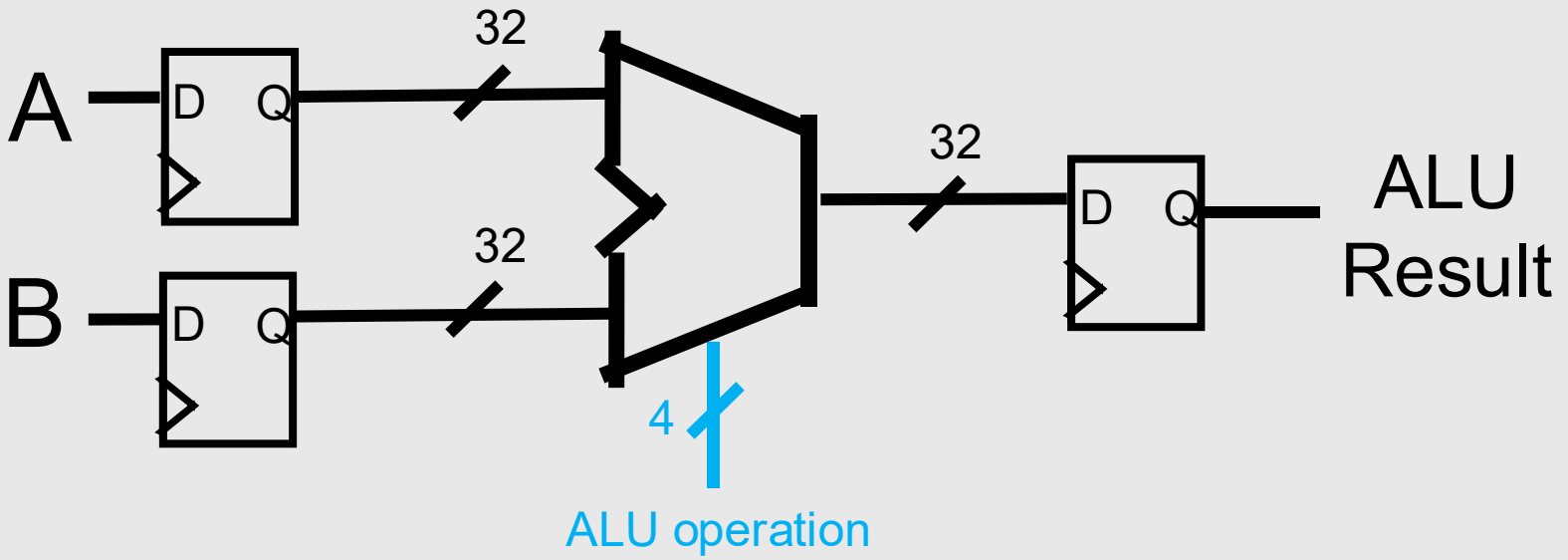
Dependent Operations

Operations to Perform

a = 5 + 4

b = 7 + 8

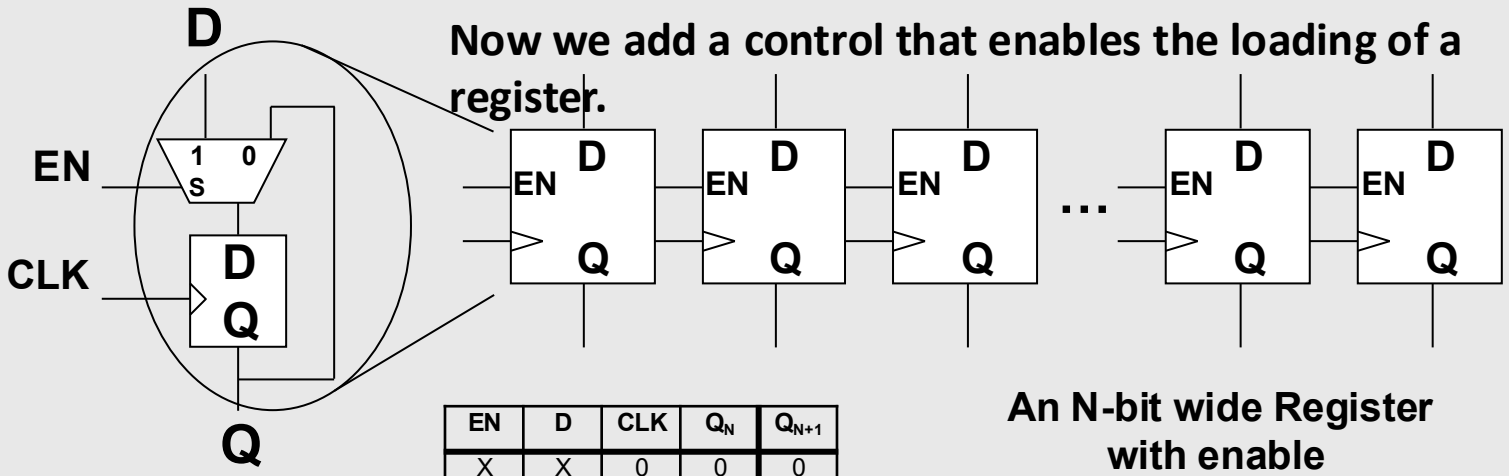
c = a + b



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

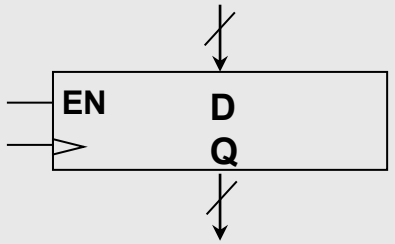
Implementation Details on the Register File!

Thus far, our building blocks units have focused on logical and arithmetic functions. We'll also need functional units for storing intermediate results. By now, we are used to the notion of building wide registers.



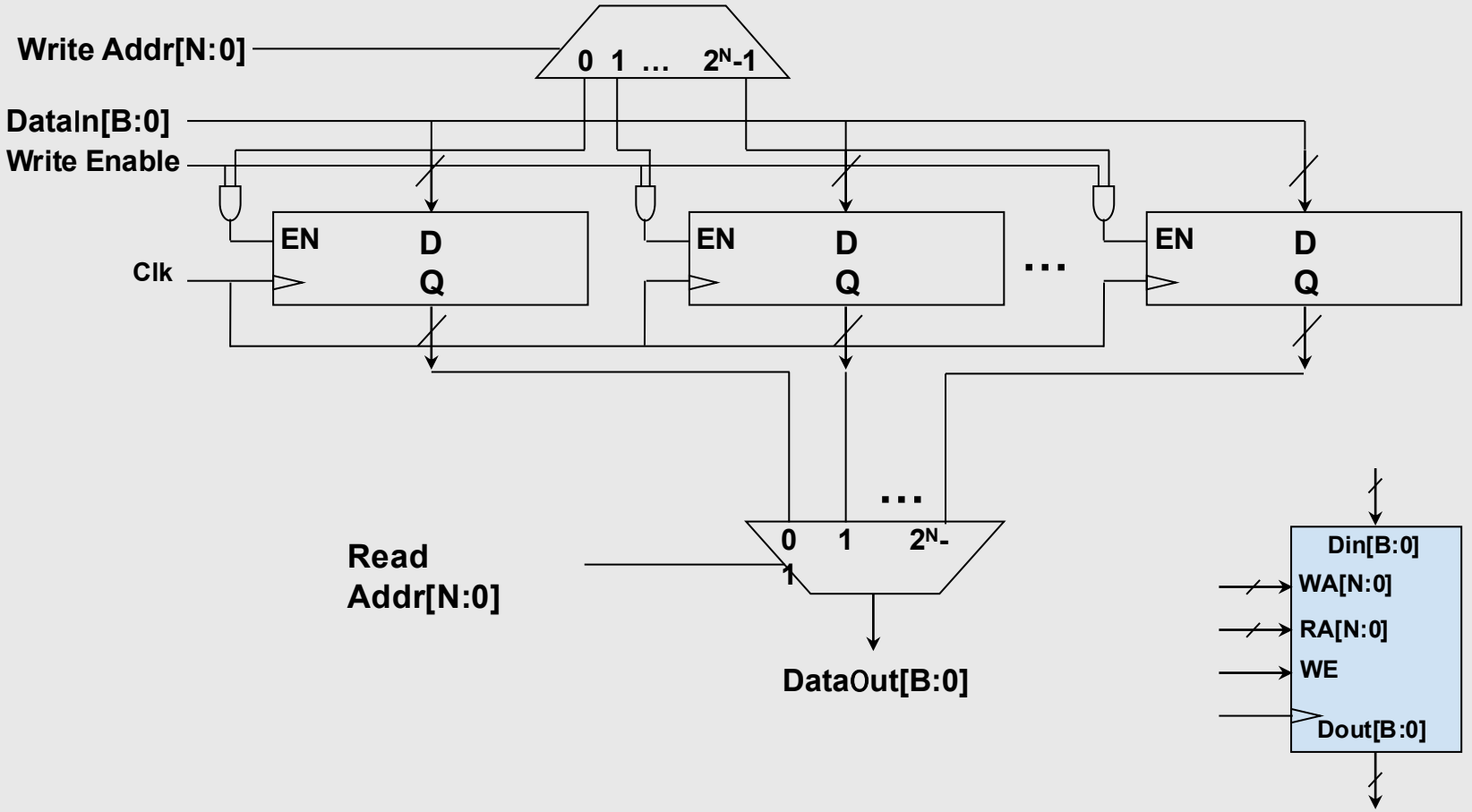
EN	D	CLK	Q_N	Q_{N+1}
X	X	0	0	0
X	X	0	1	1
X	X	1	0	0
X	X	1	1	1
0	X	↑	0	0
0	X	↑	1	1
1	0	↑	X	0
1	1	↑	X	1

An N-bit wide Register with enable



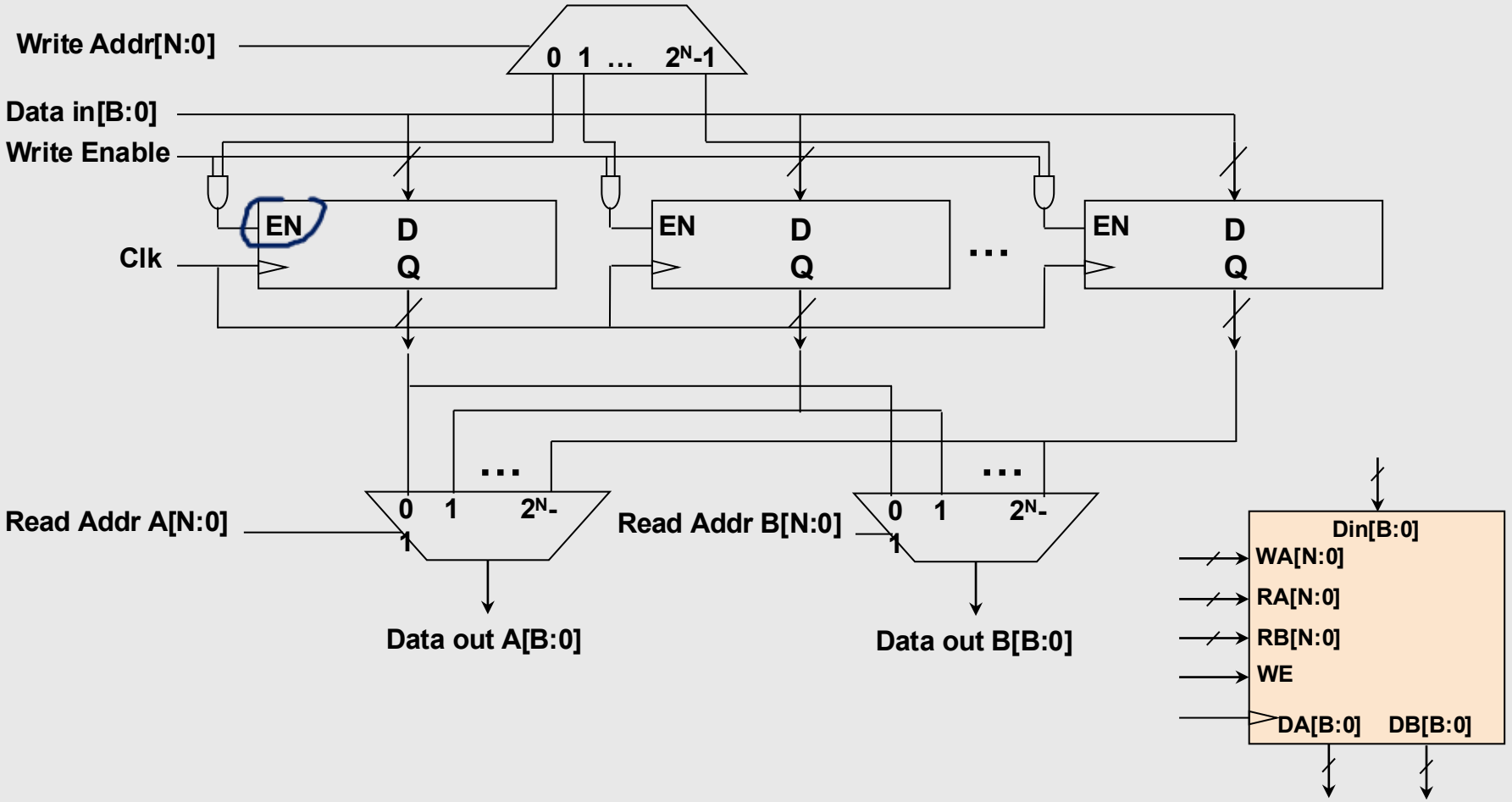
A Register File

We can also construct an addressable array of registers

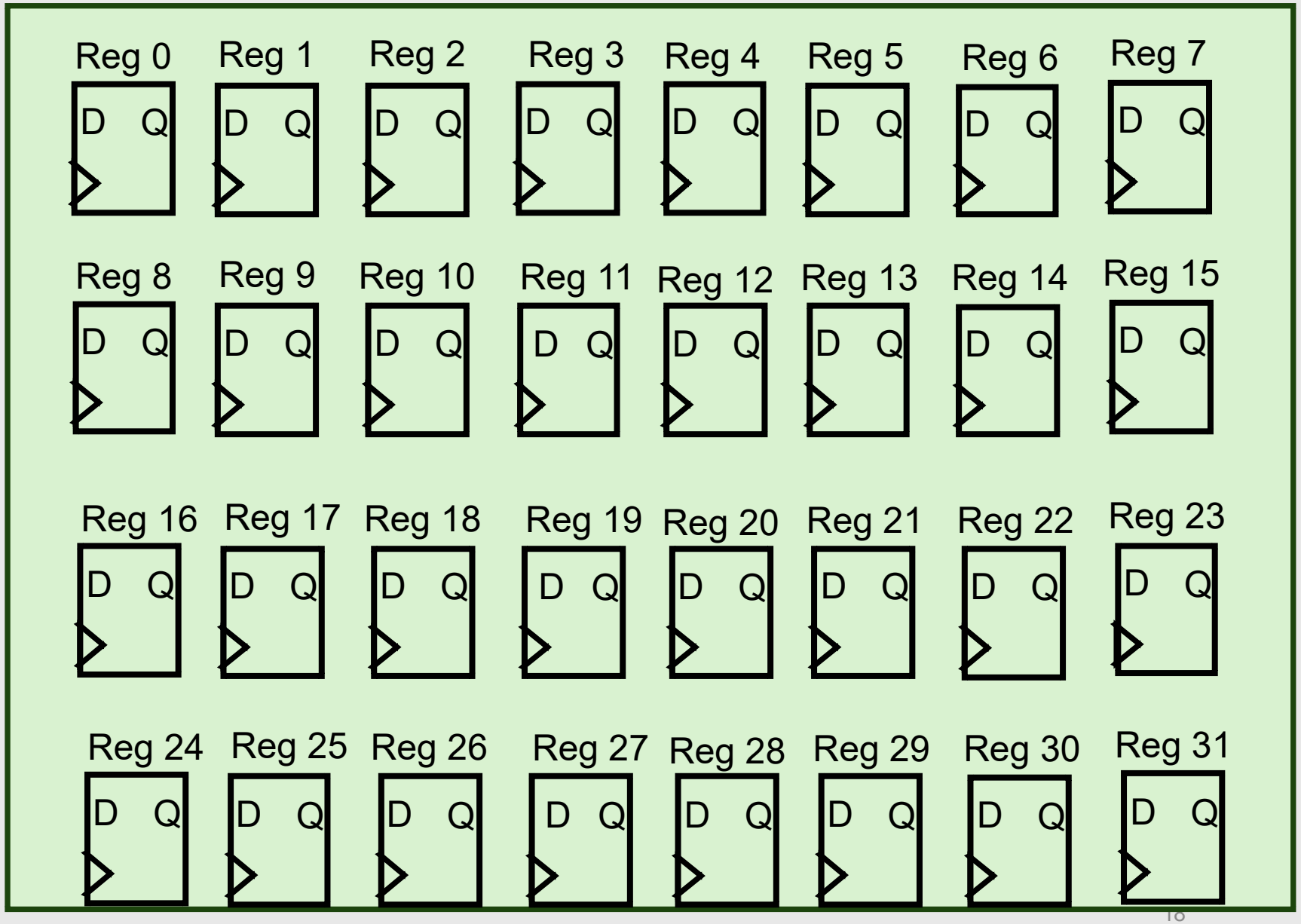


A Multi-Ported Register File

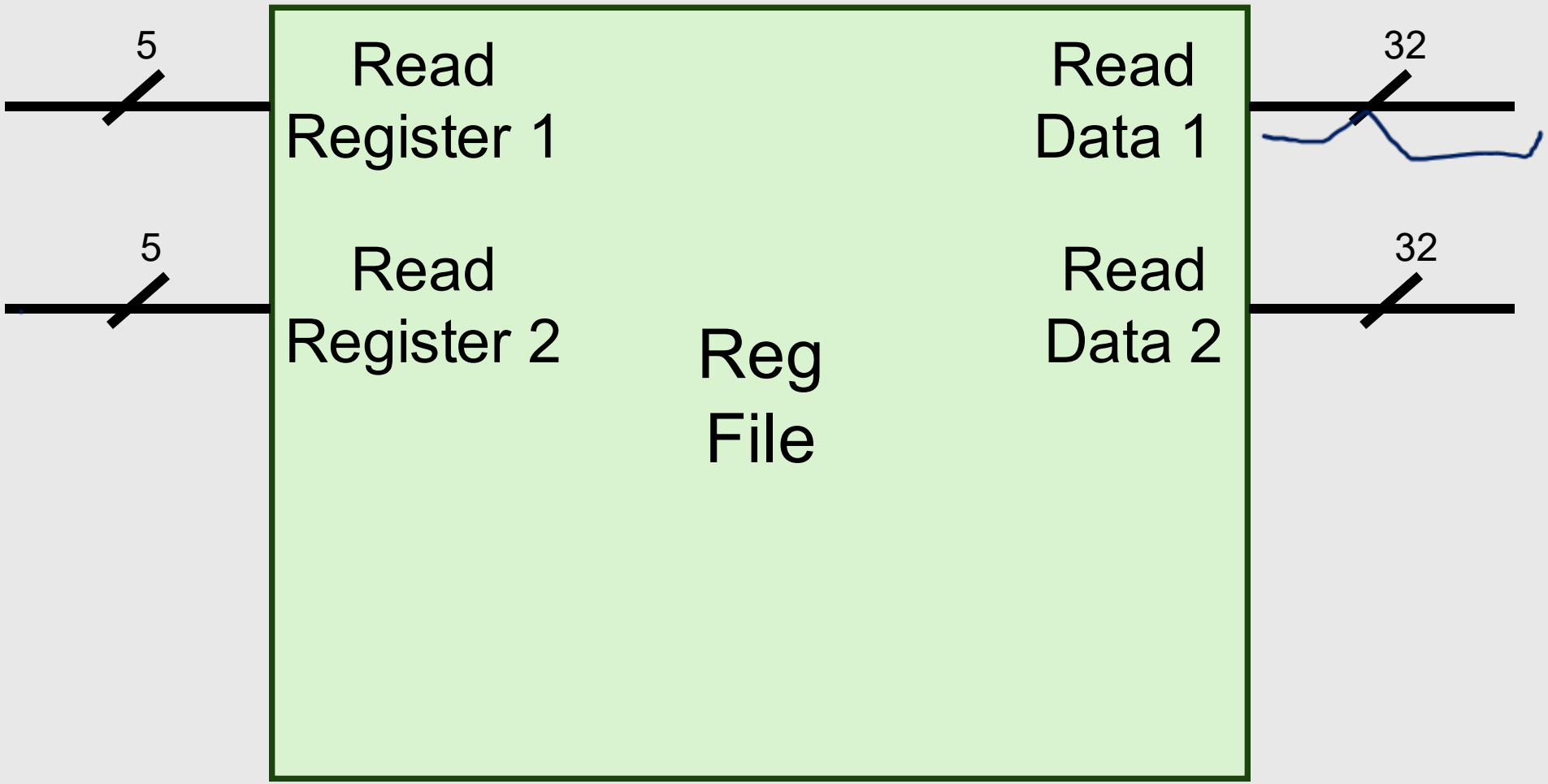
Multiple read ports by simply adding more output MUXs



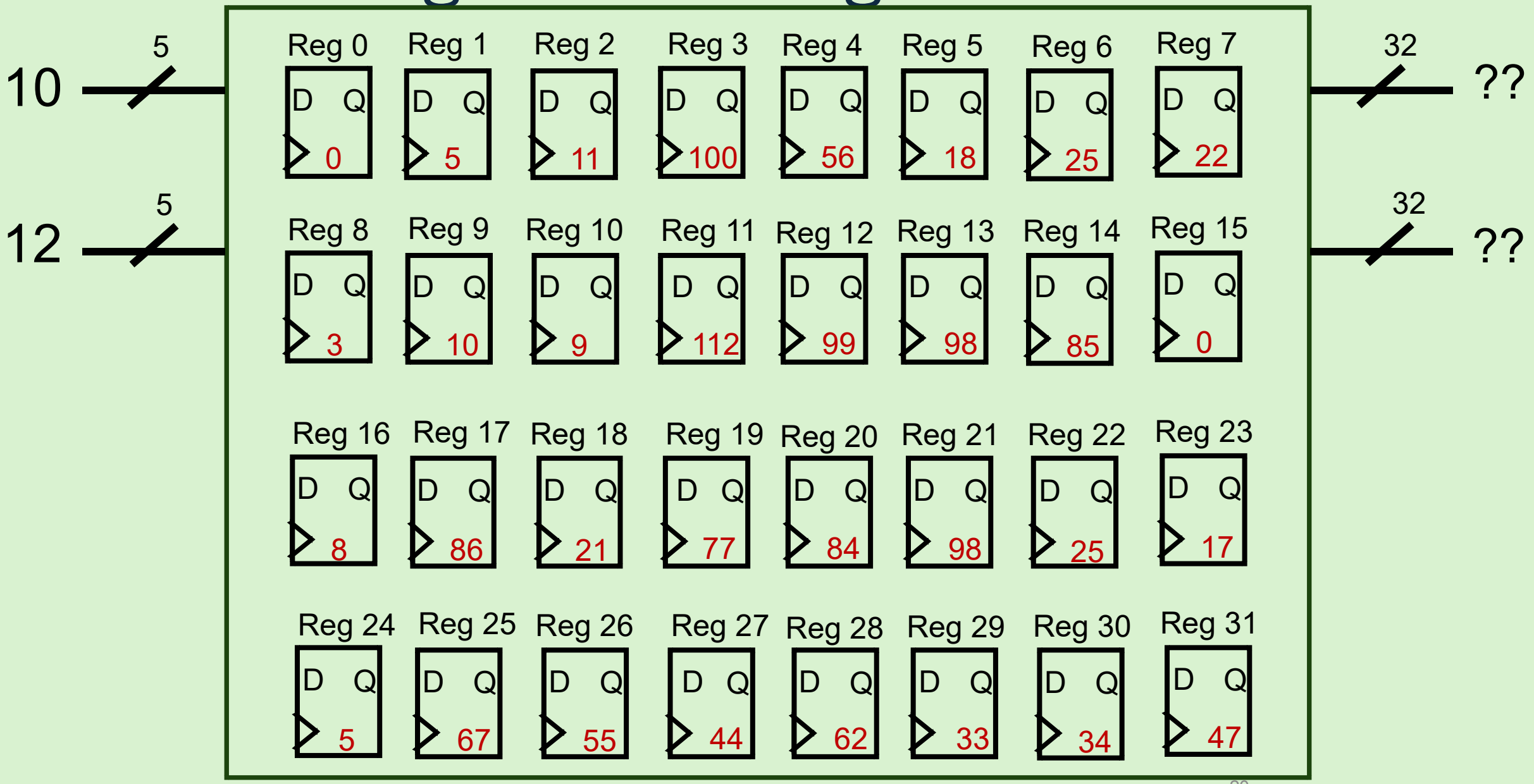
Register File



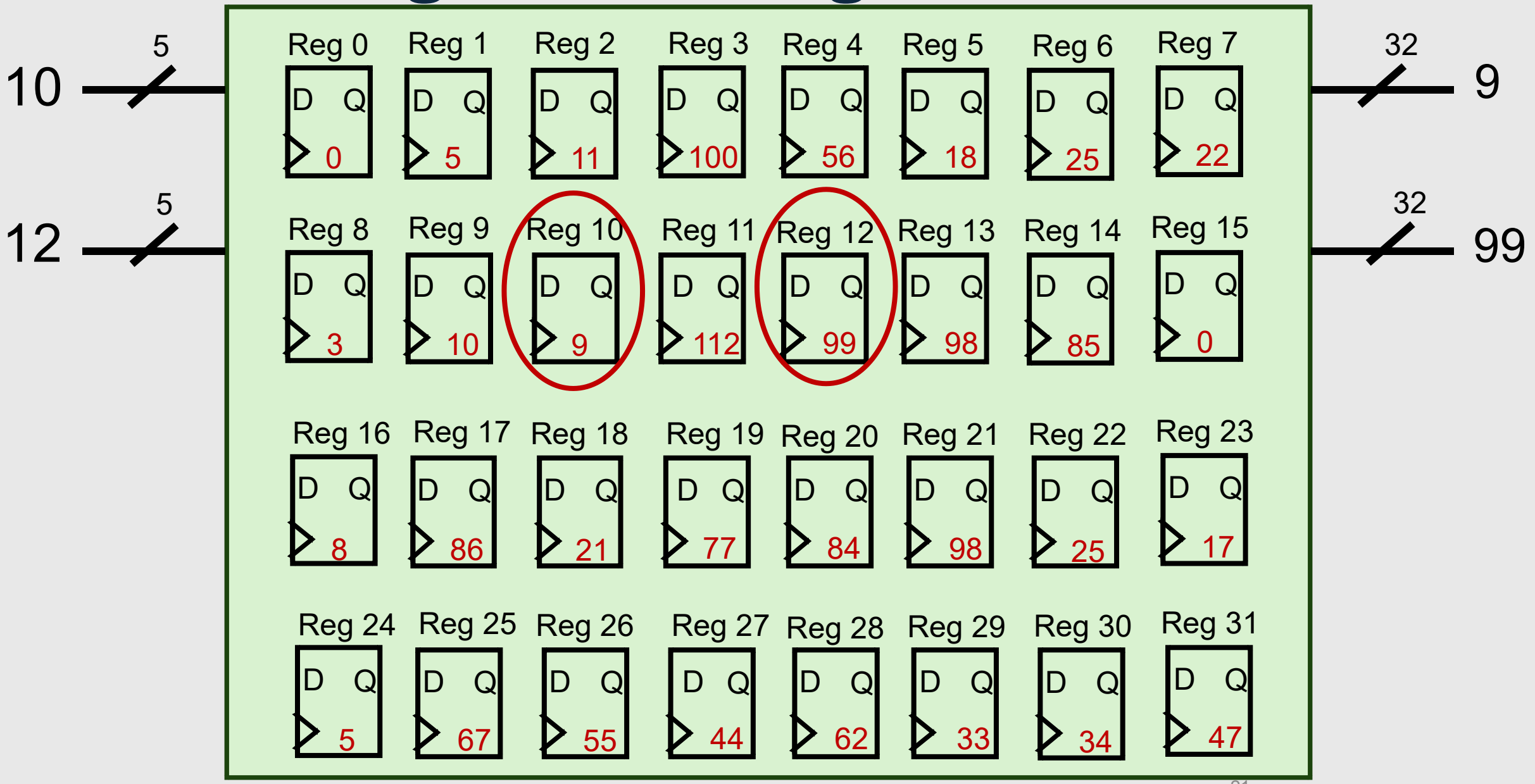
Reading from the Register File



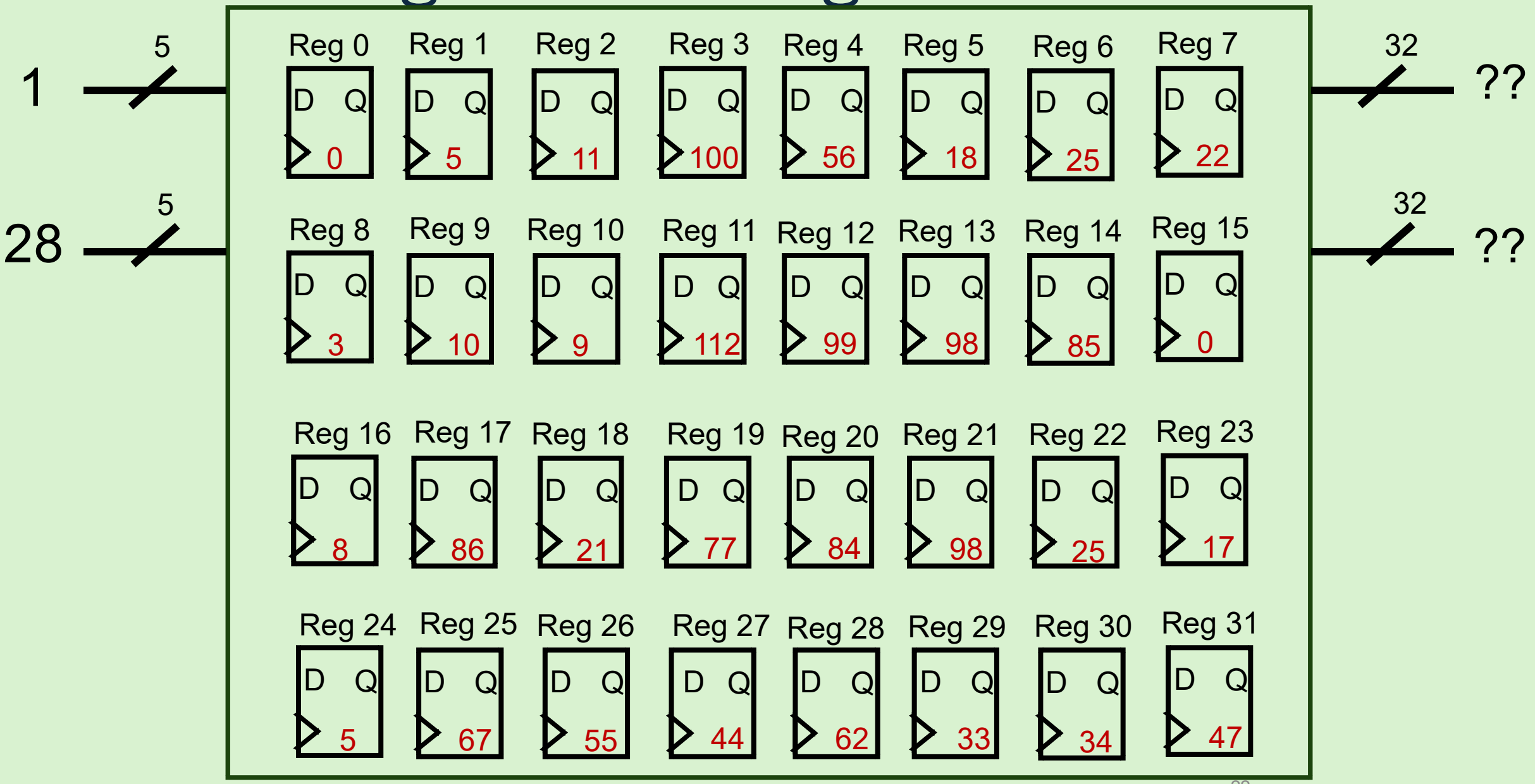
Ex1: Reading from the RegFile



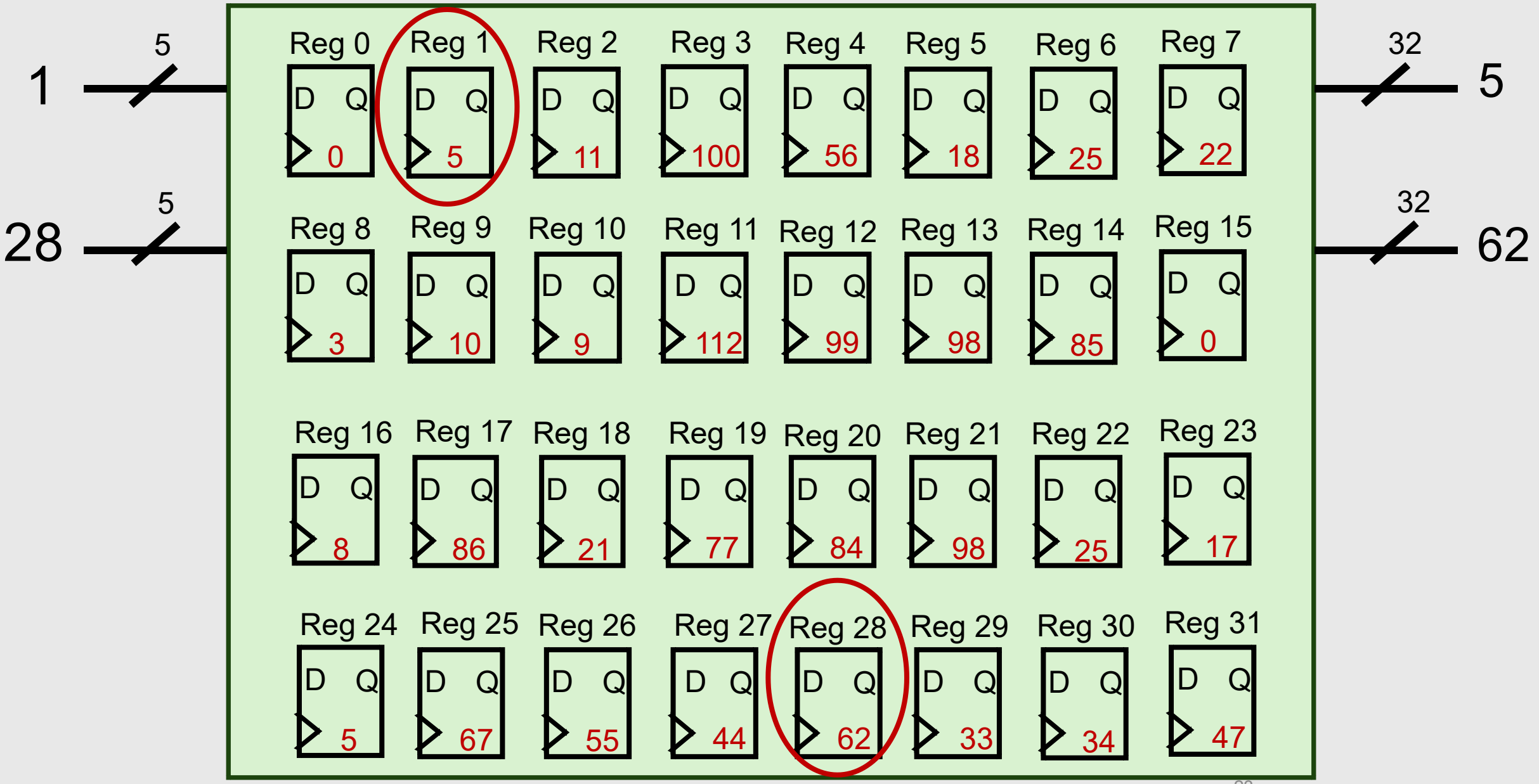
Ex1: Reading from the RegFile



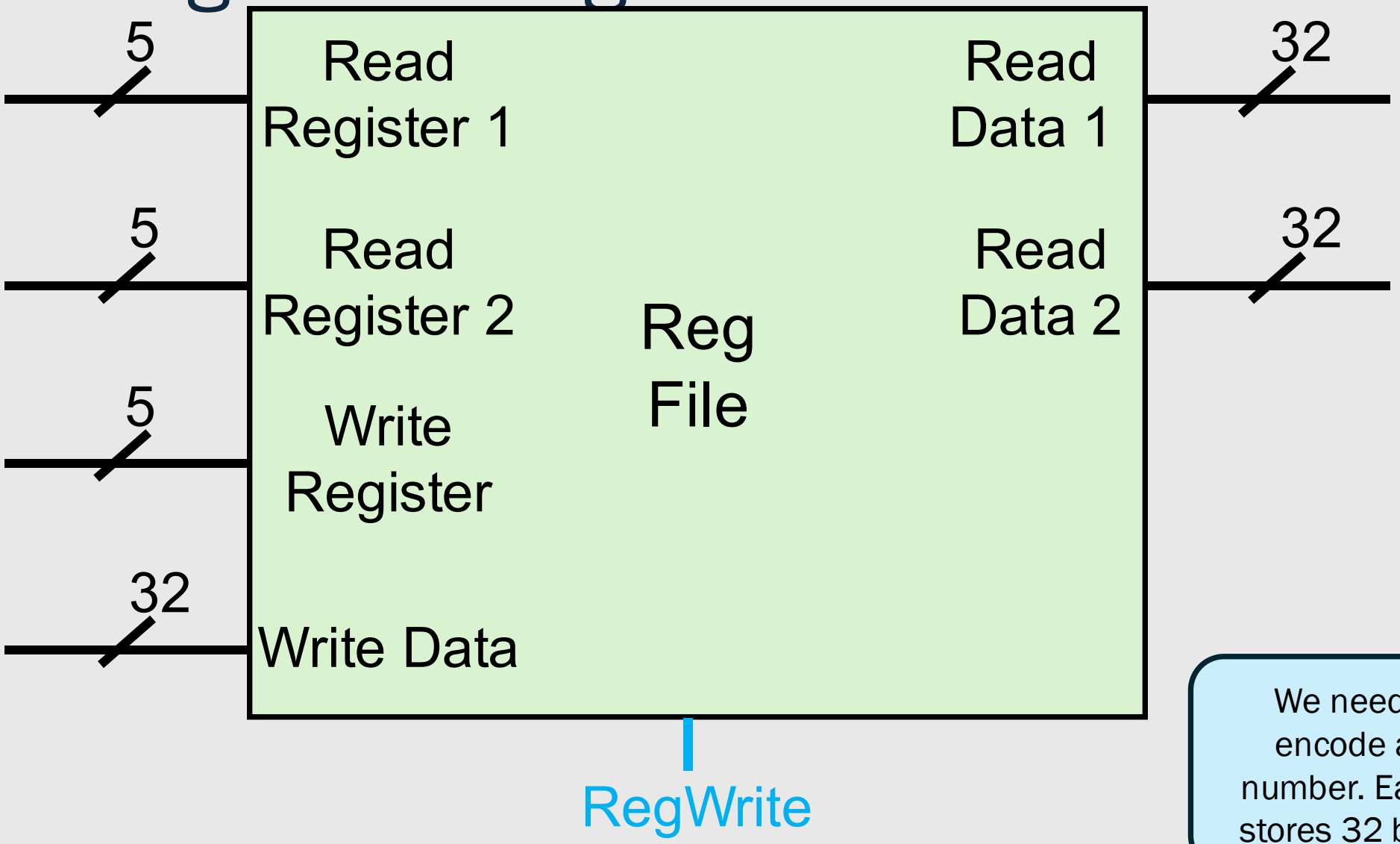
Ex2: Reading from the RegFile



Ex2: Reading from the RegFile



Writing to the Register File



We need 5 bits to encode a register number. Each register stores 32 bits of data!

Writing to the Register File

- To write to a register inside of the register file, set the following signals
  - *Write register: The number of the register you are writing to*
  - *Write data: The data you want to store inside the register*
  - *RegWrite: This is a control signal whose value needs to be 1 to write to a register*
    - When we are not writing to a register, this value should be 0

TIMING ANALYSIS

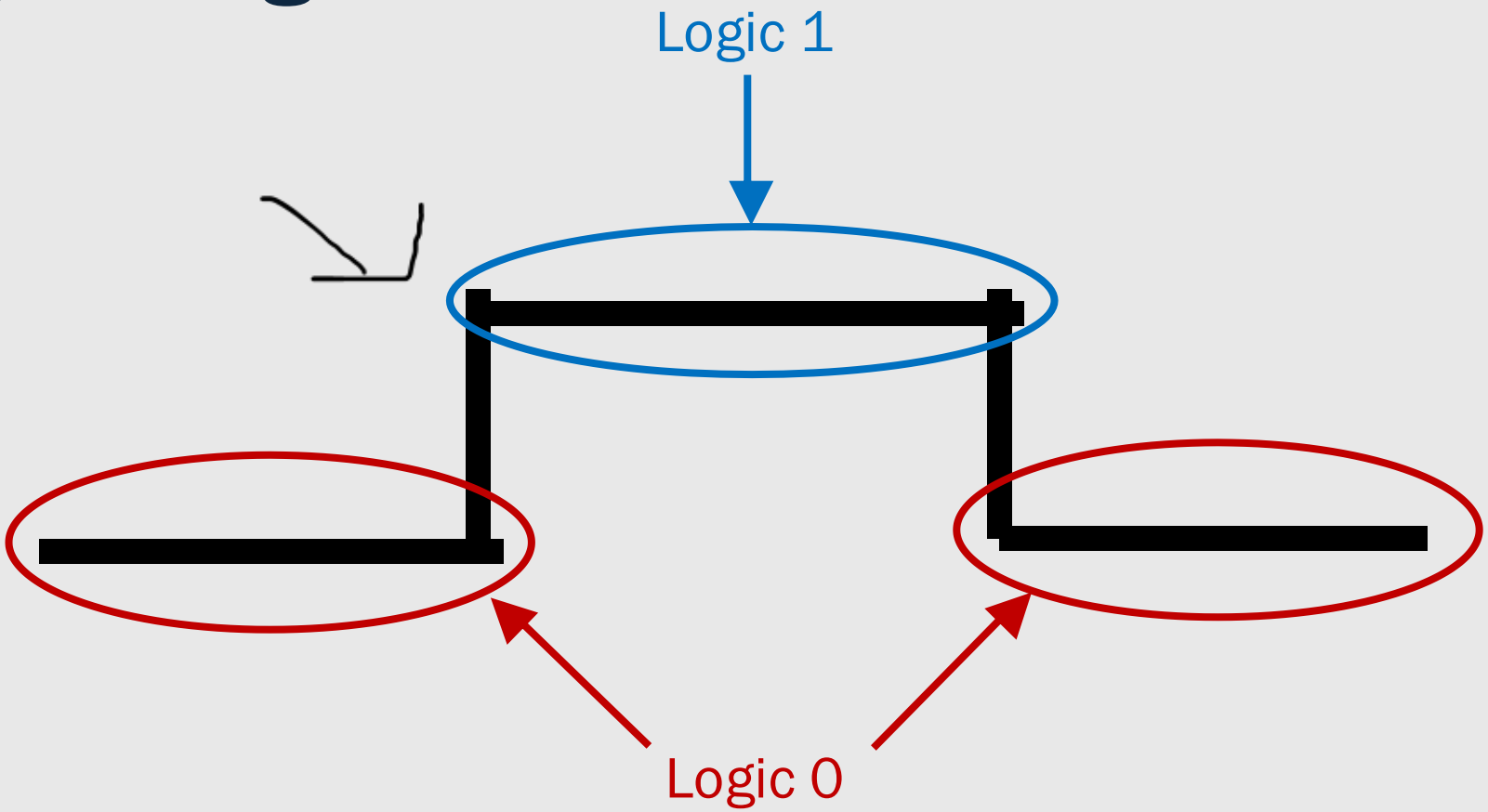
Plan looking ahead: We are going to zoom in for a moment to see how to analyze timing at the gate level, before popping back up to analyze instruction-level & CPU performance!

Waveform Diagrams

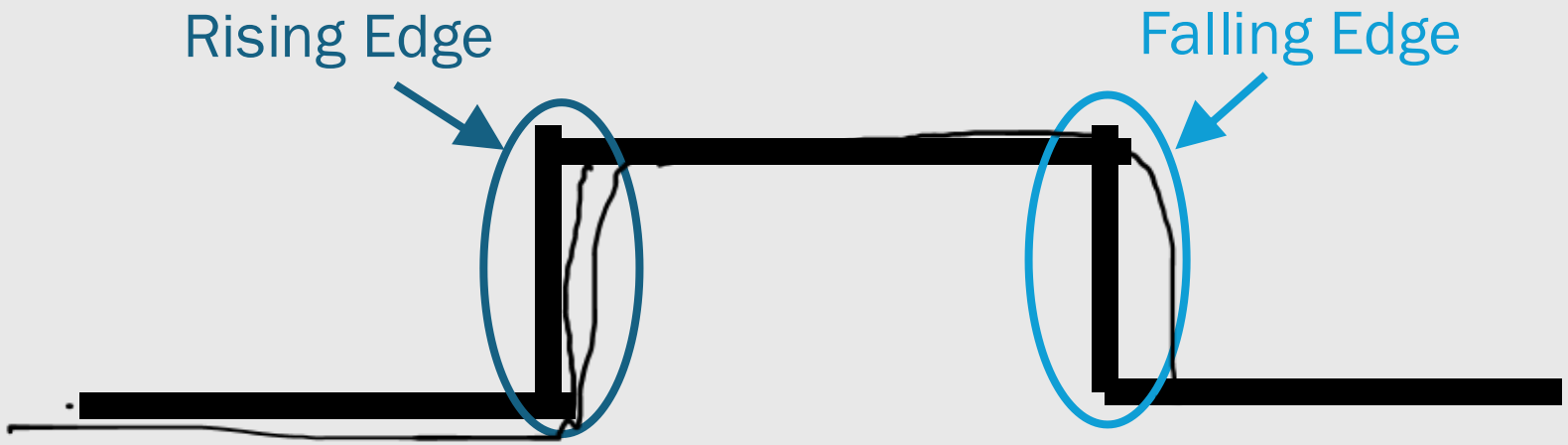
- We can represent the values of our signals over time using a waveform diagram
- Each one-bit signal will have its own waveform
- The waveforms are square waves where
  - *a low value represents logic 0*
  - *a high value represents logic 1*

Digital Signals

Digital Signals



Digital Signals

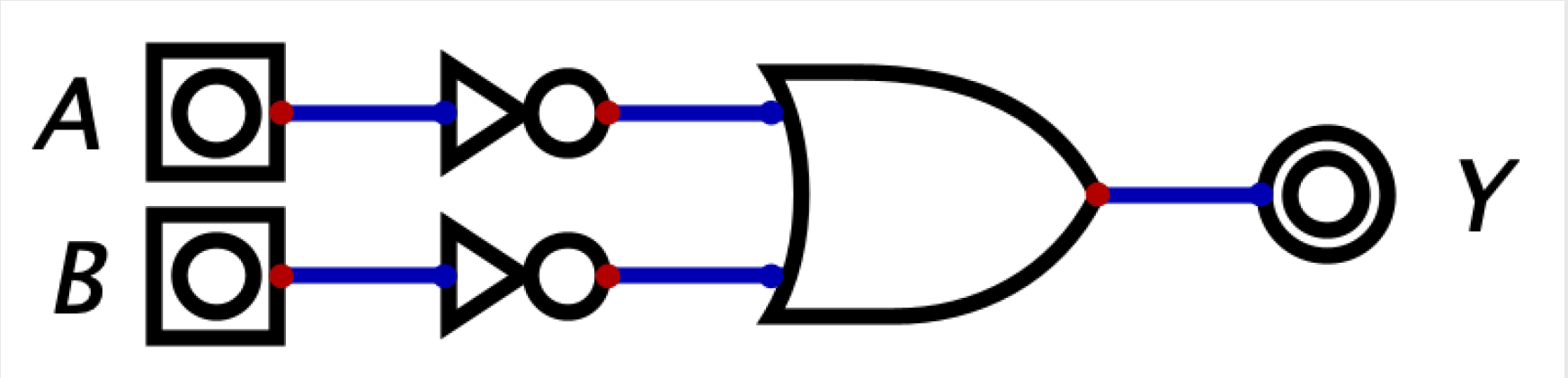


Digital Signals

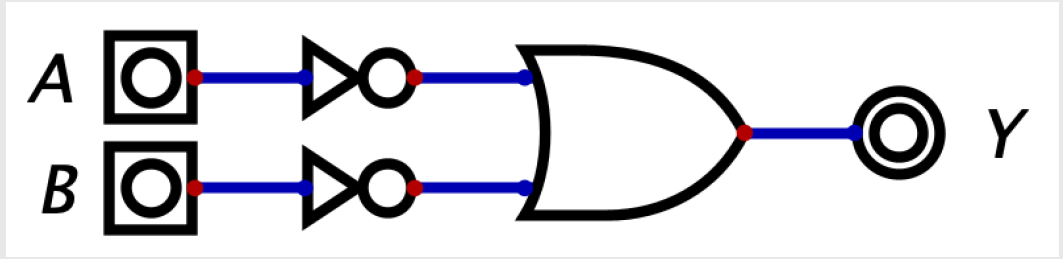
- Rising Edge
  - *When a signal changes from logic low to logic high*
  - *You can think of the signal as rising from 0 to 1*
- Falling Edge
  - *When a signal changes from logic high to logic low*
  - *You can think of the signal as falling from 1 to 0*

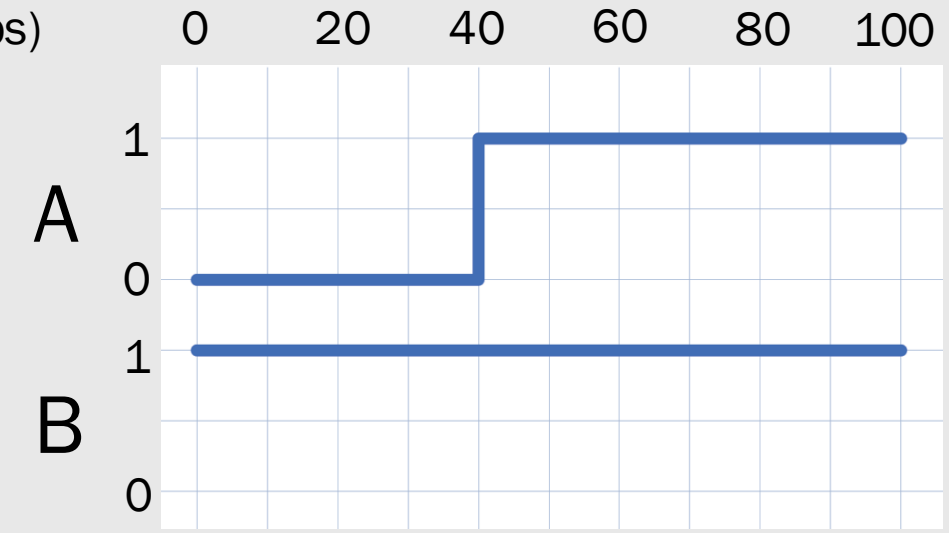
Ex1: Waveform Diagram

- 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

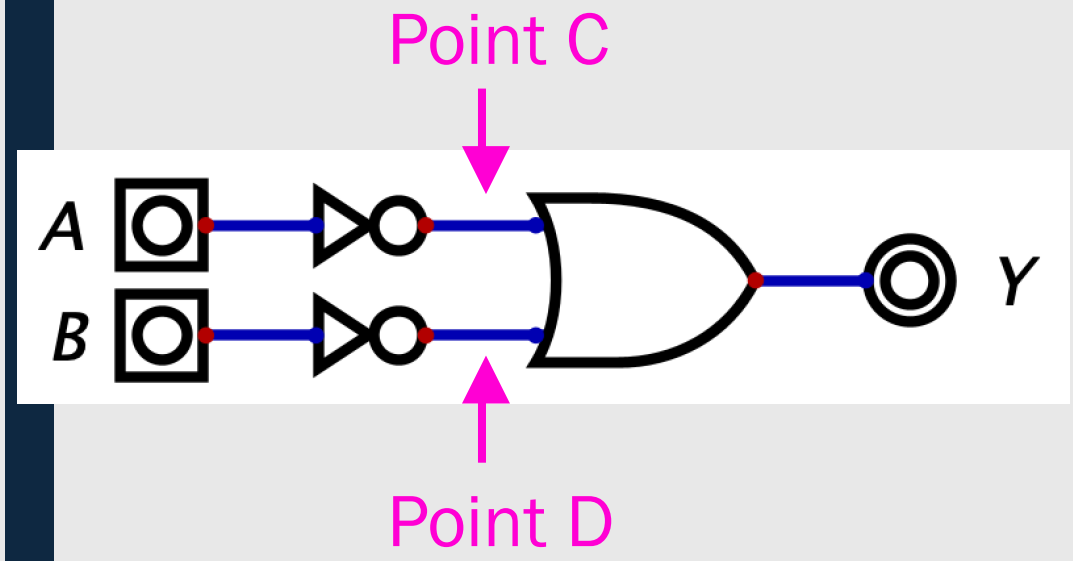


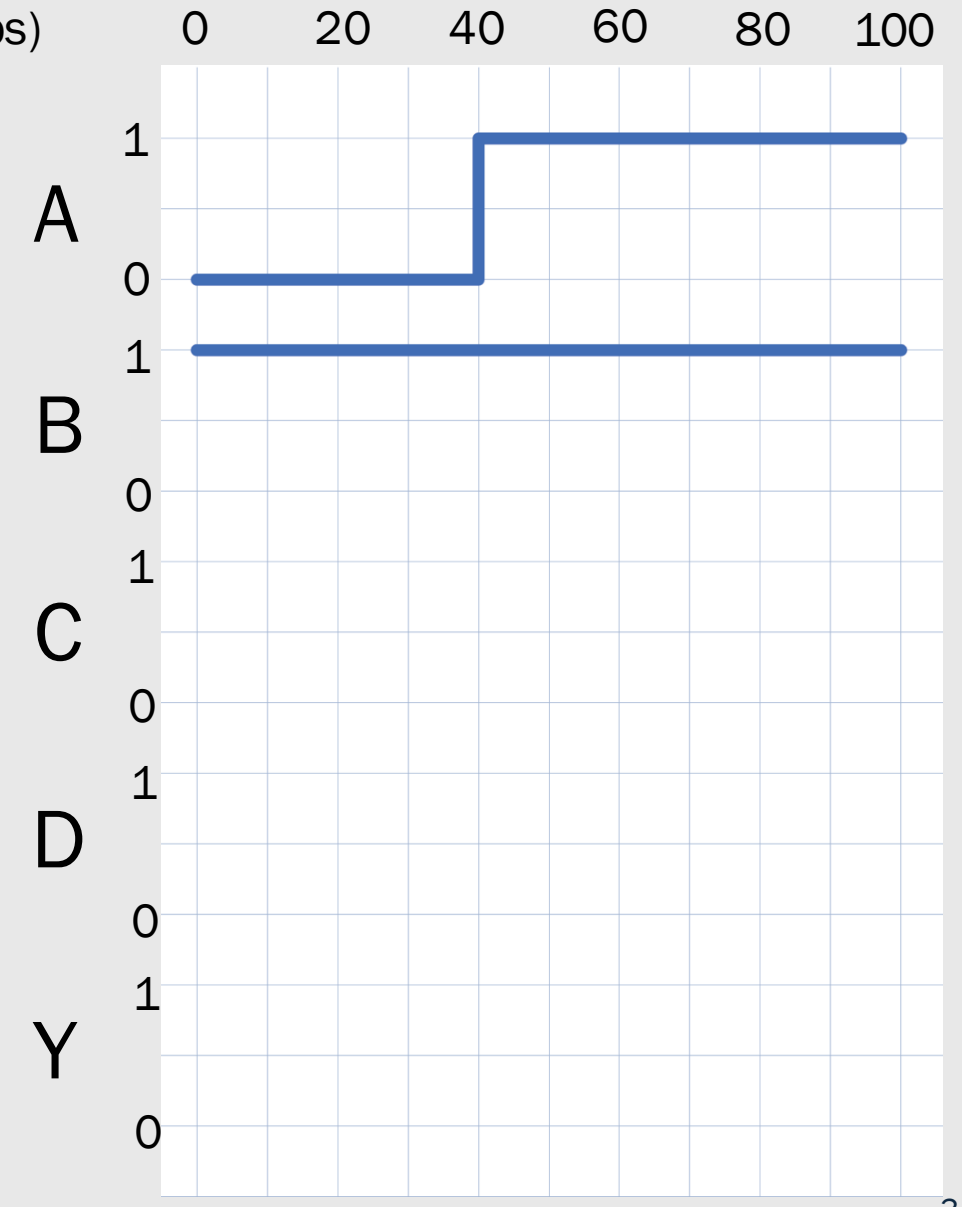


The waveform diagram above shows how we change the input values over time

- 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

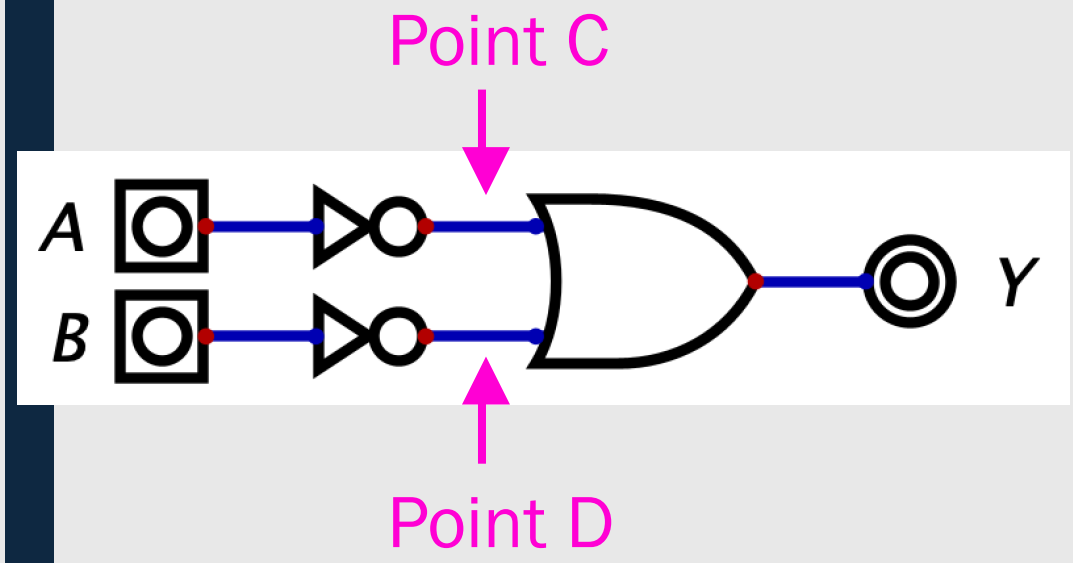
Ex1: Waveform Diagram

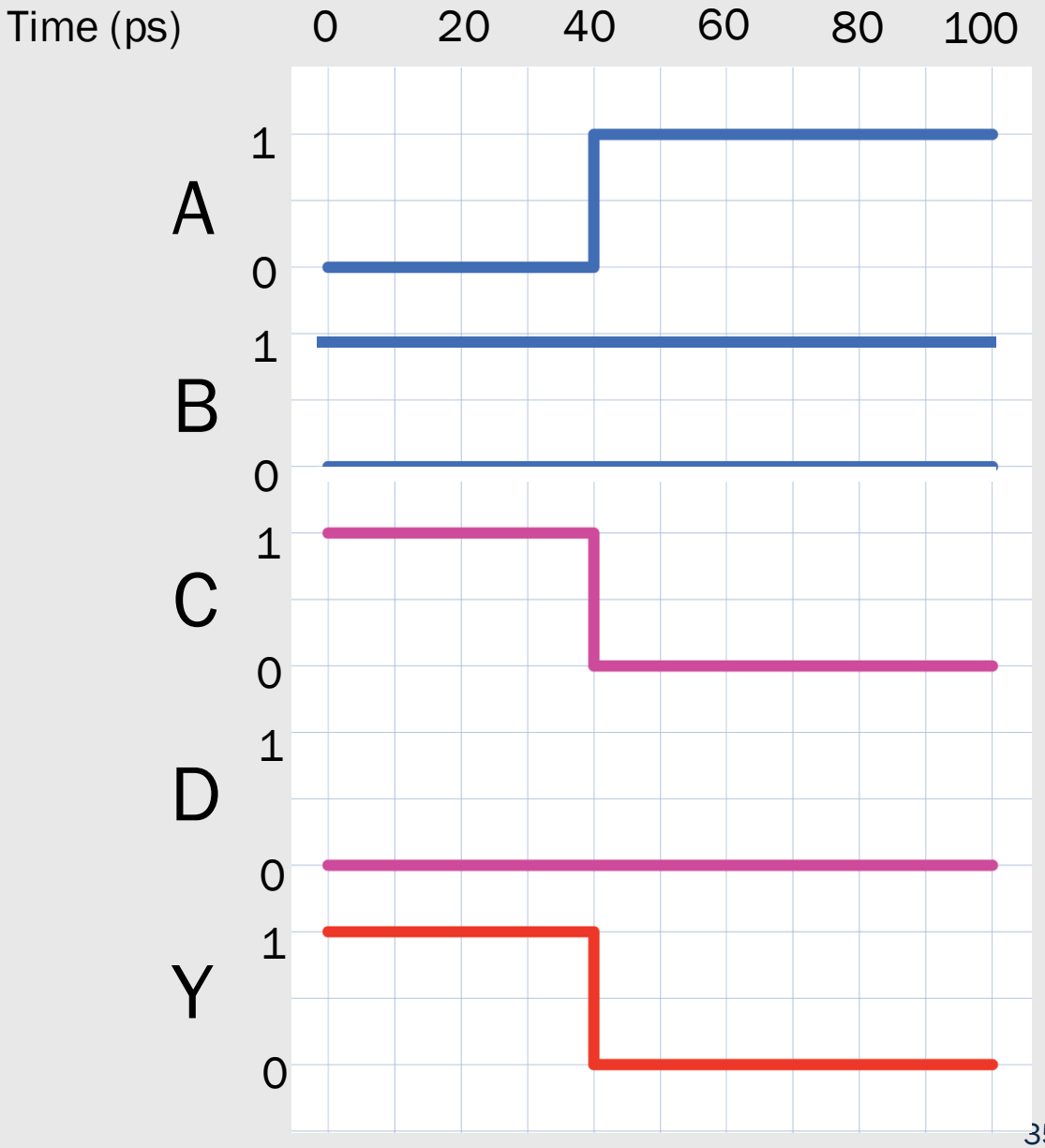




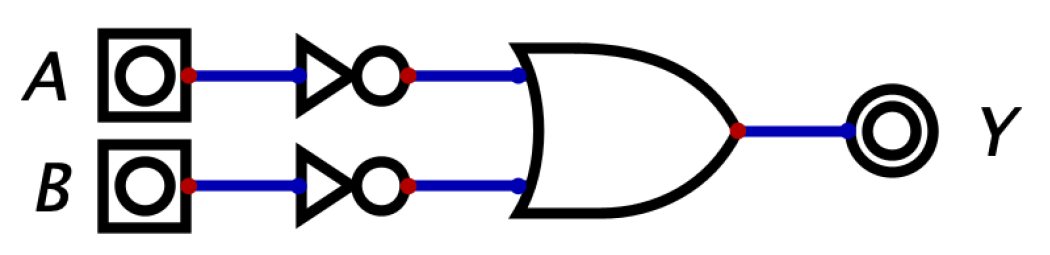
Now, we will analyze how the intermediate signals (C and D) and the output signal (Y) change as the inputs change.

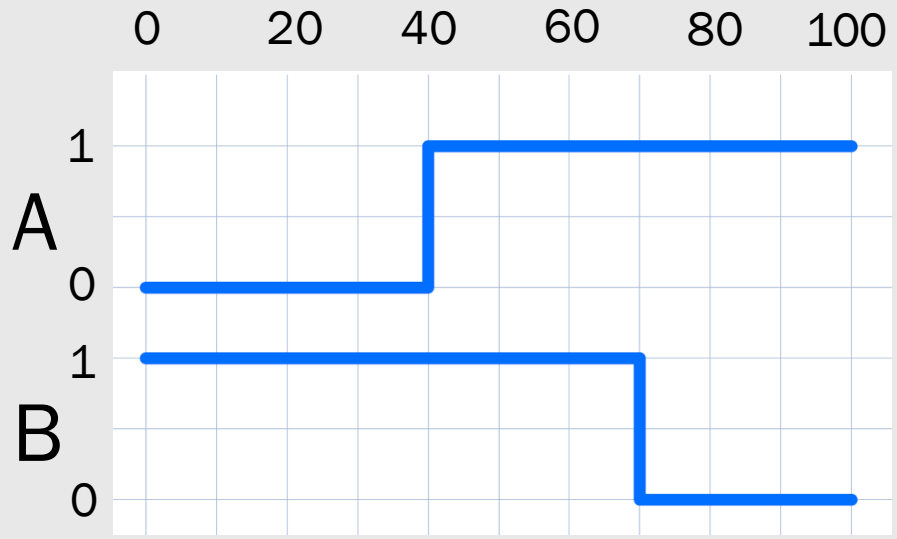
Ex1: Waveform Diagram





Ex2: Waveform Diagram

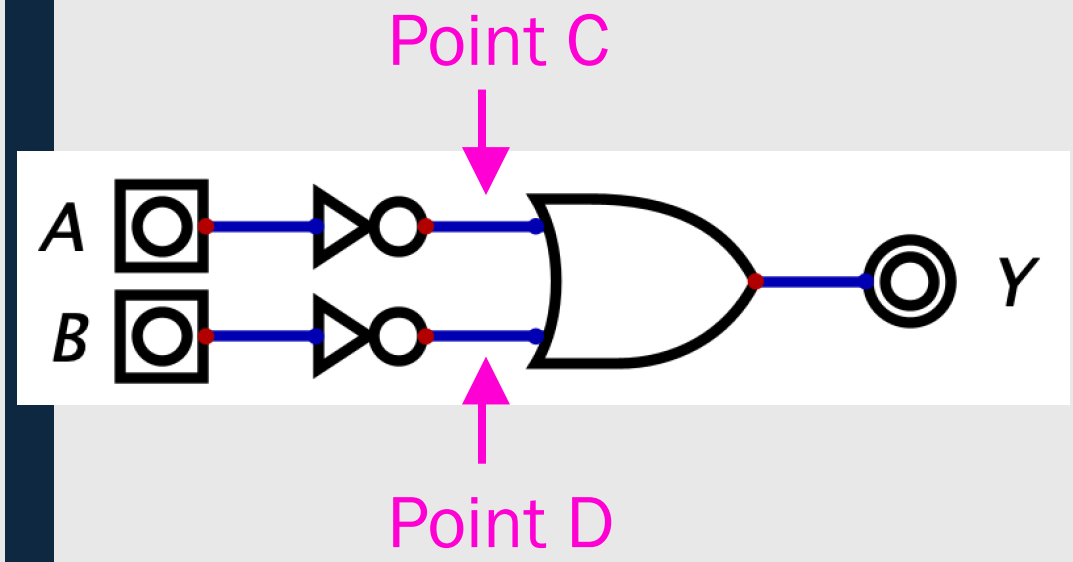


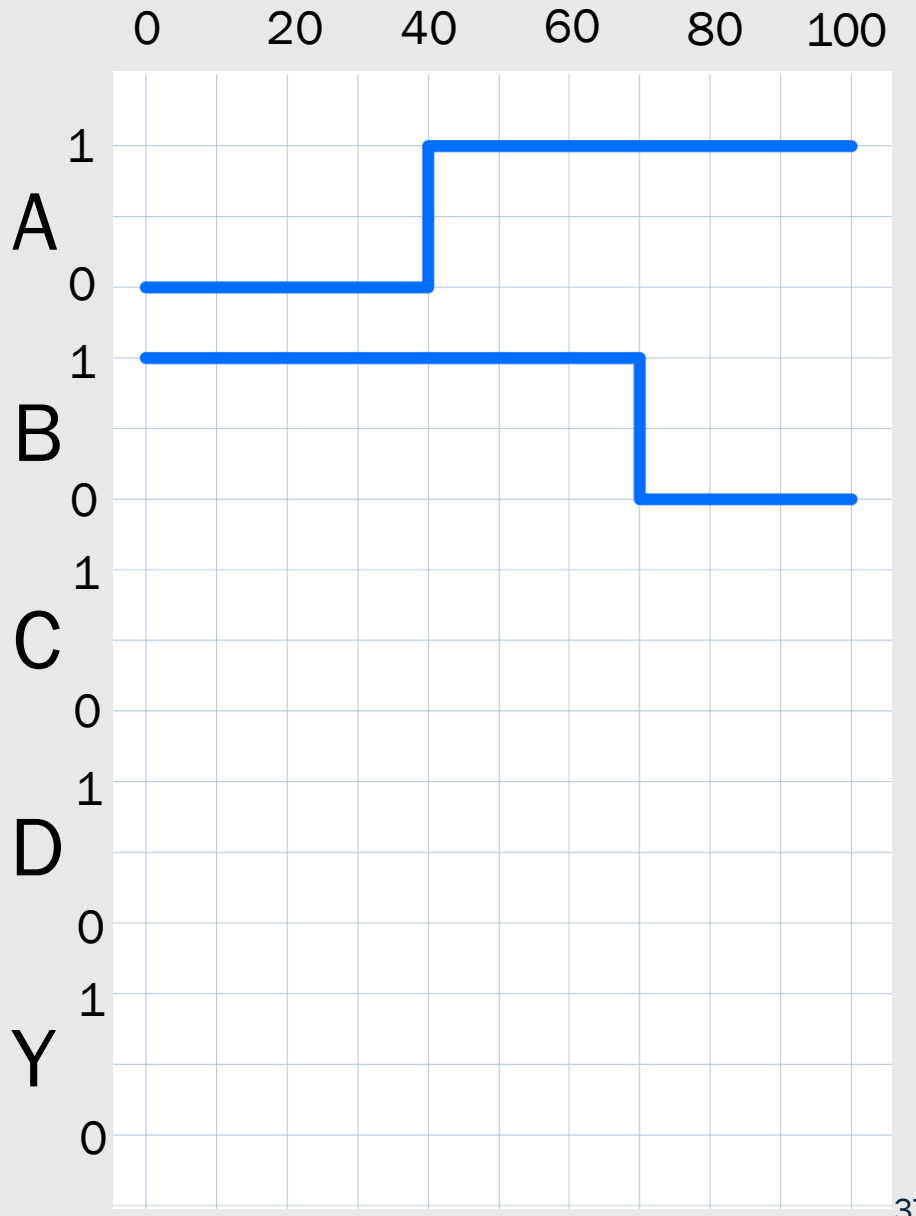


Let's look at another example with the same circuit

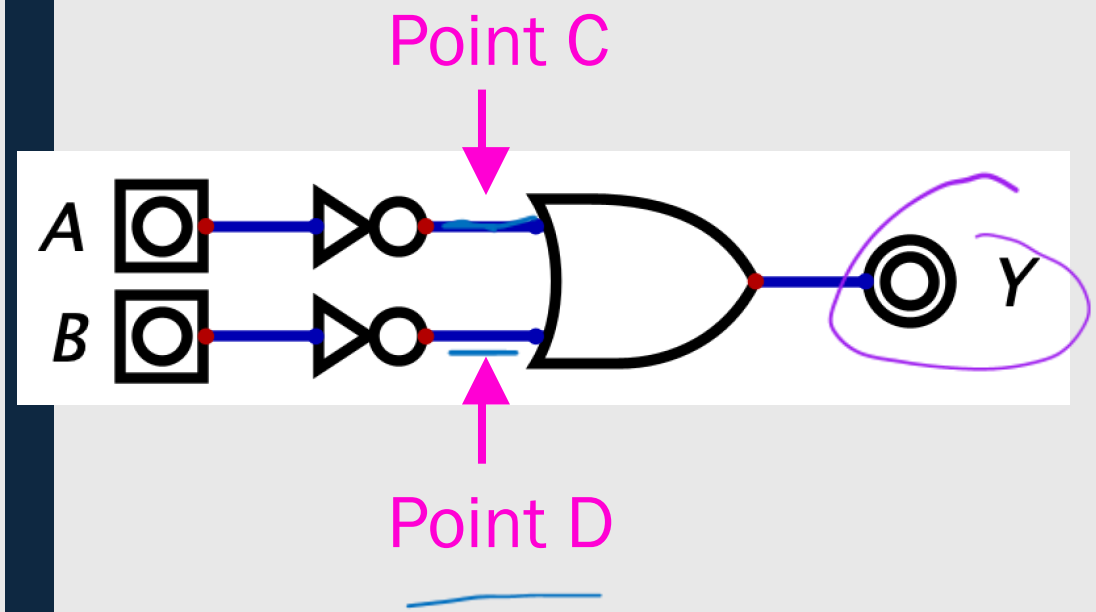
- At time t = 0ps, we will set the input A to 0 and the input B to 1
- At time t = 40ps, we will change the input A to 1
- At time t = 70ps, we will change the input B to 0

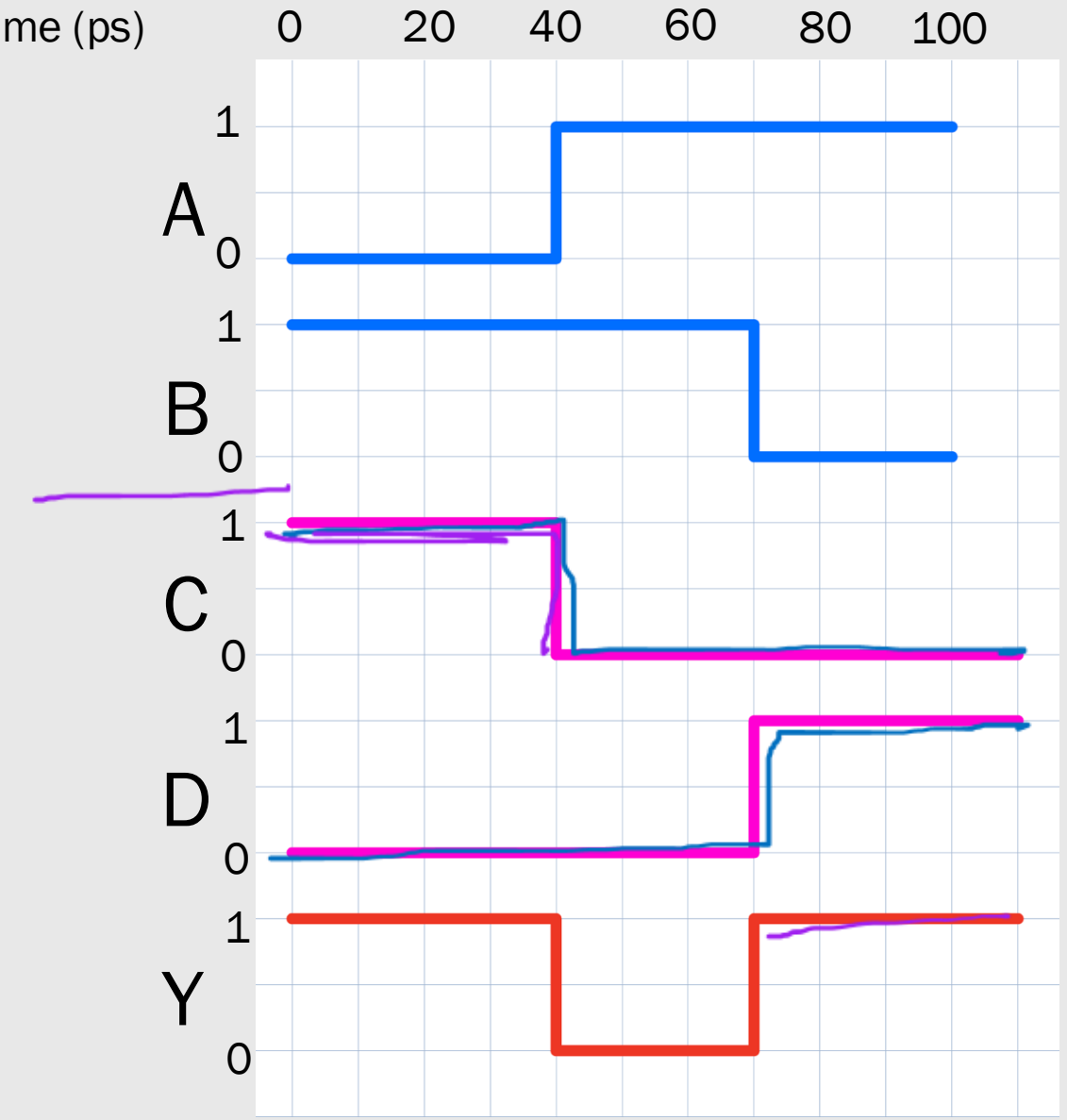
Ex2: Waveform Diagram



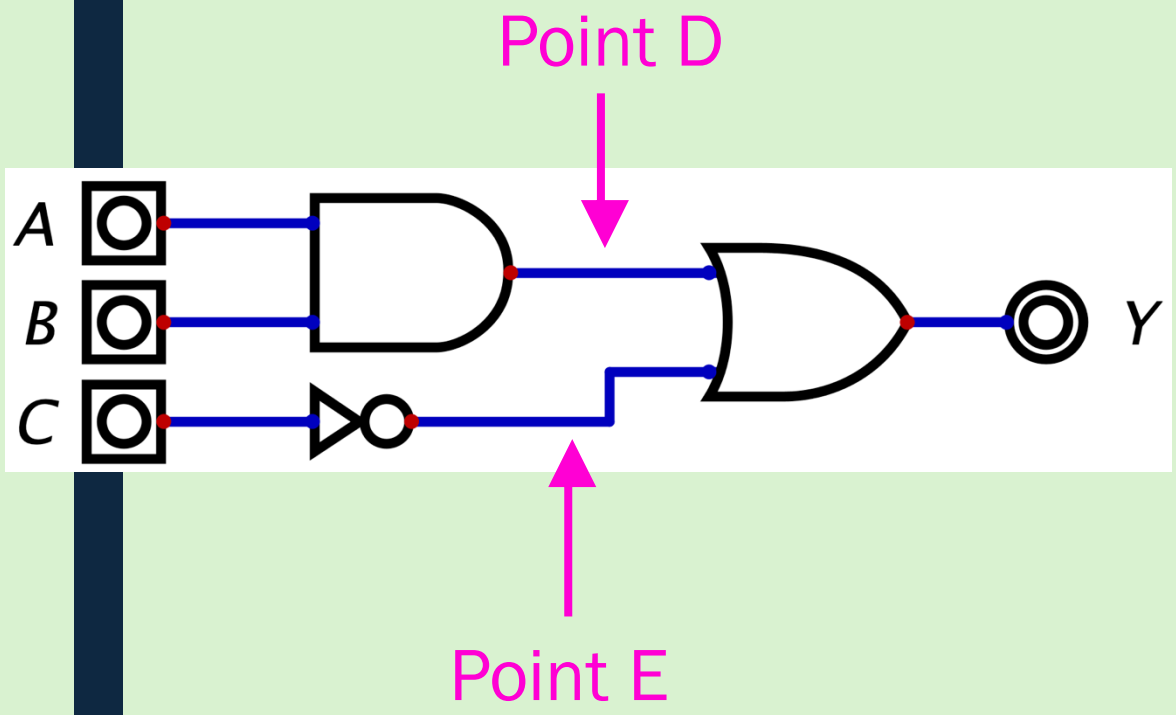


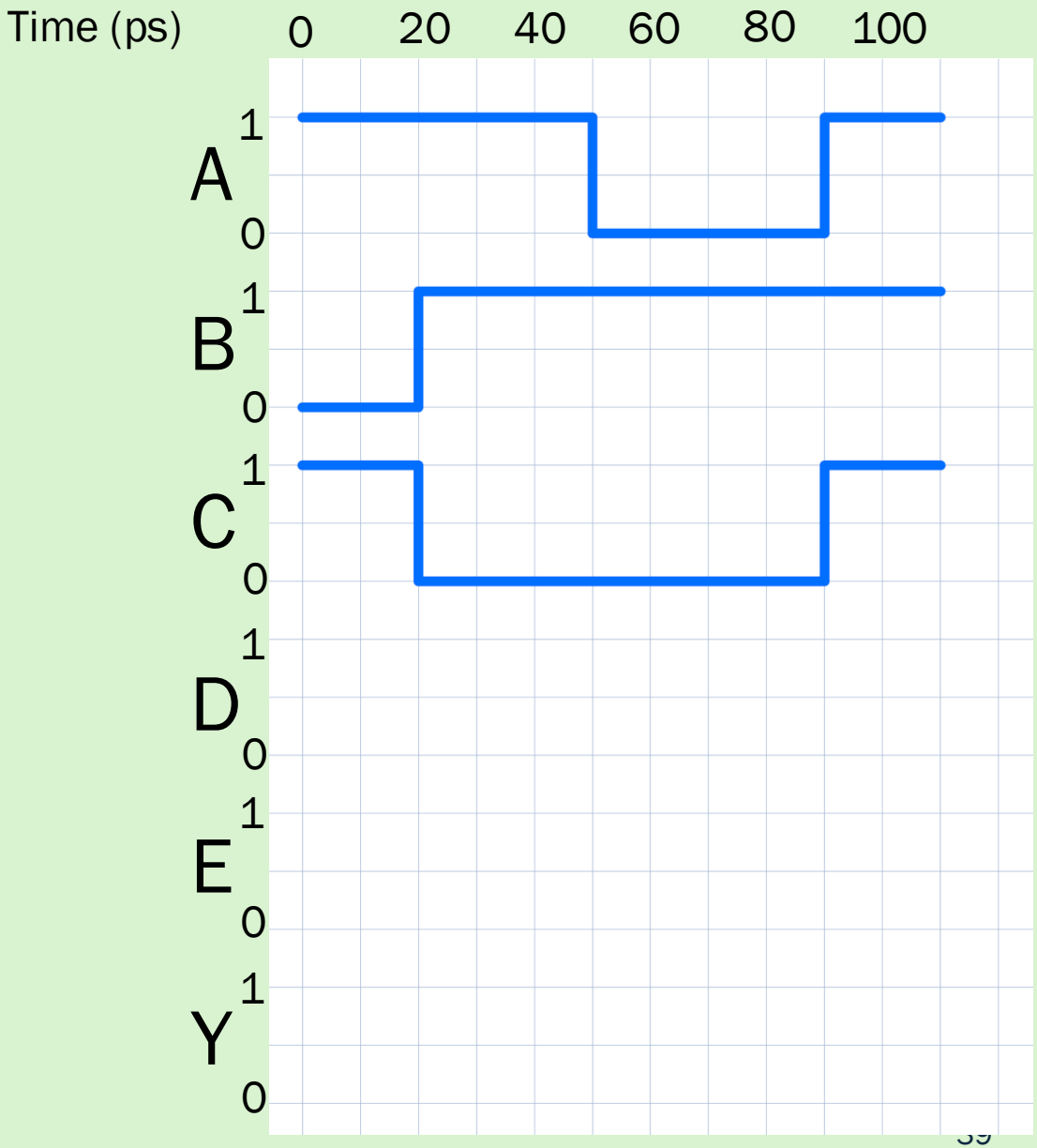
Ex2: Waveform Diagram



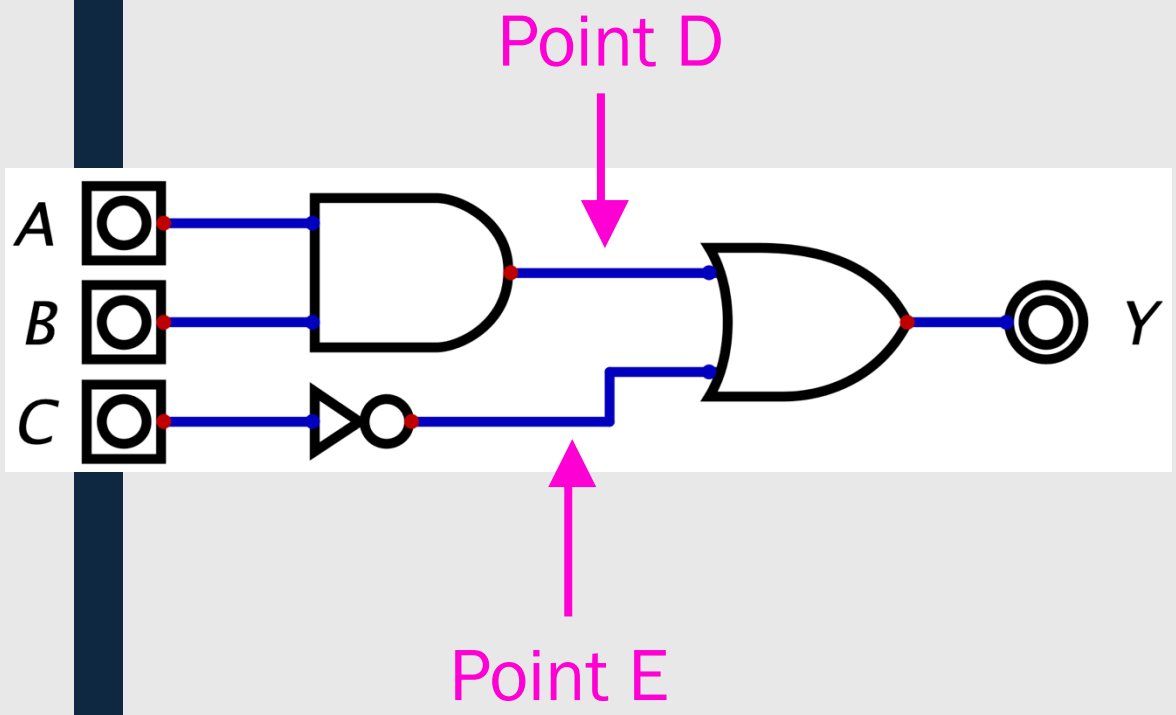


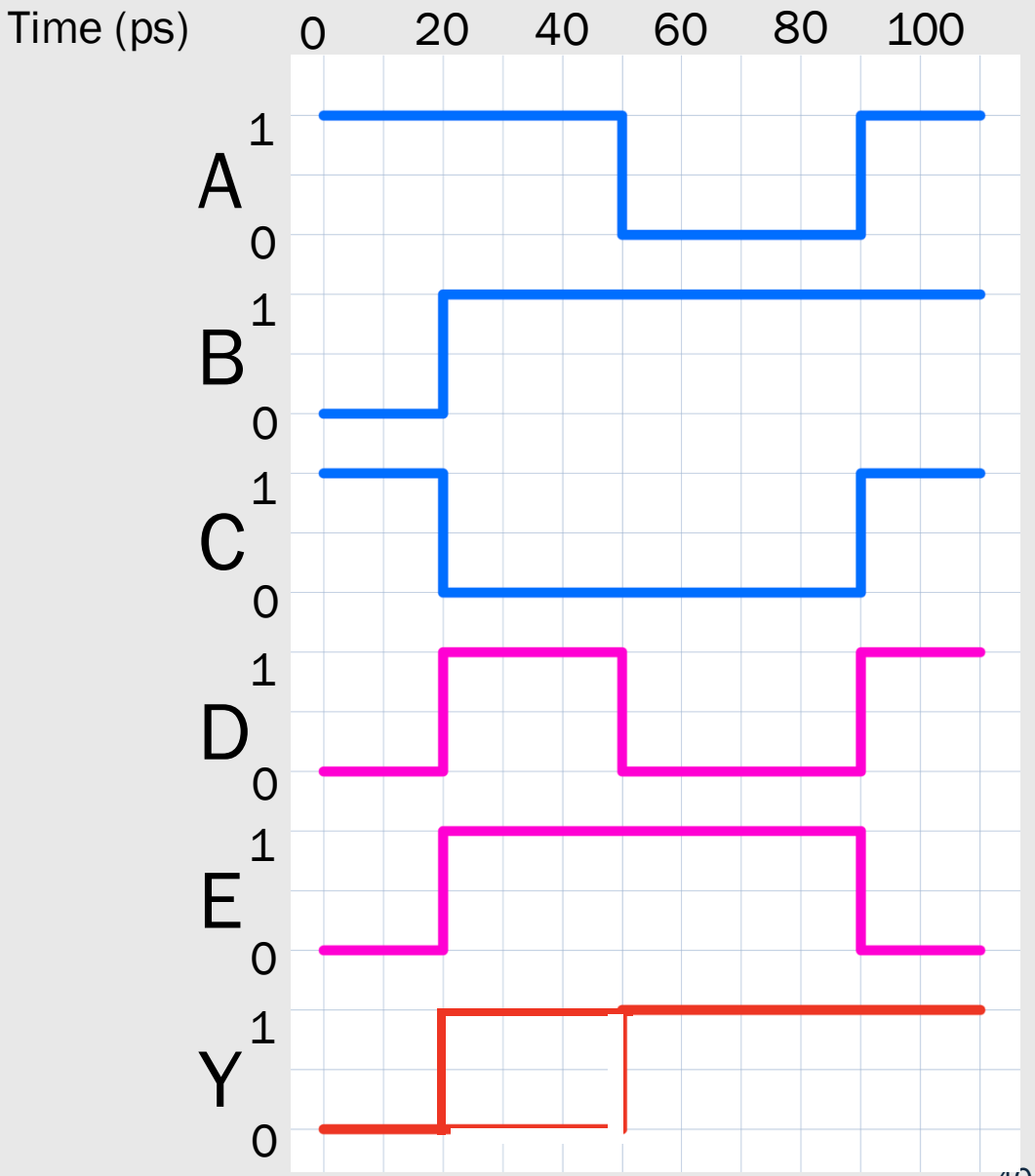
Q1: Waveform Diagram





Q1: Waveform Diagram





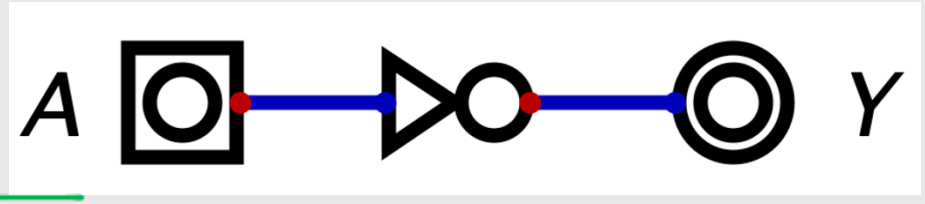
DELAYS!

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)

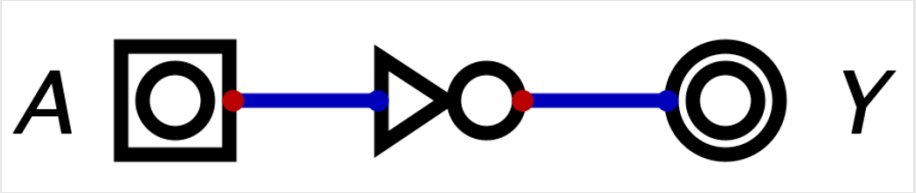
Propagation delay (t_{pd})

- 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})

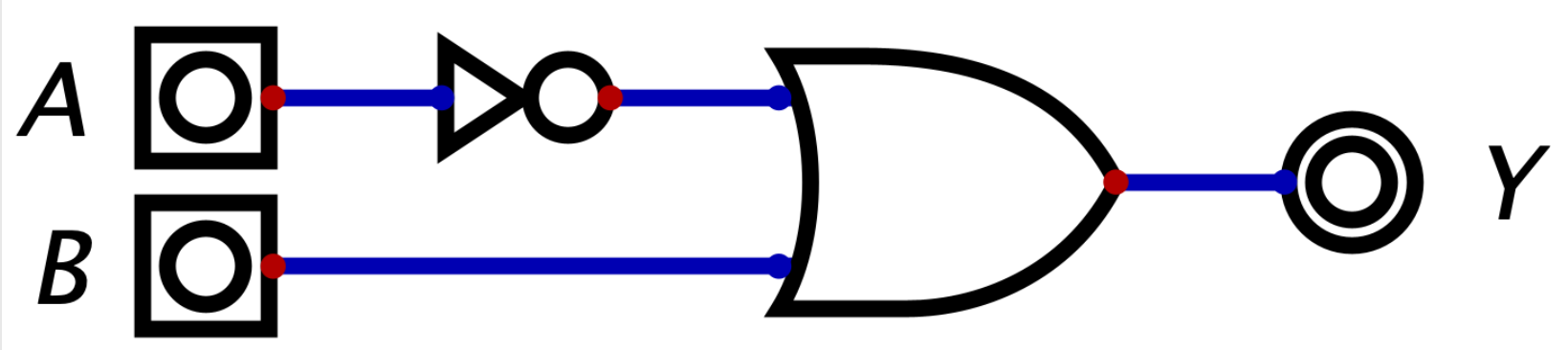
- Let's say the input to this inverter starts at 0.
- When we change the input to 1, the output will not become 0 immediately. There will be a delay.



A: 0 → 1

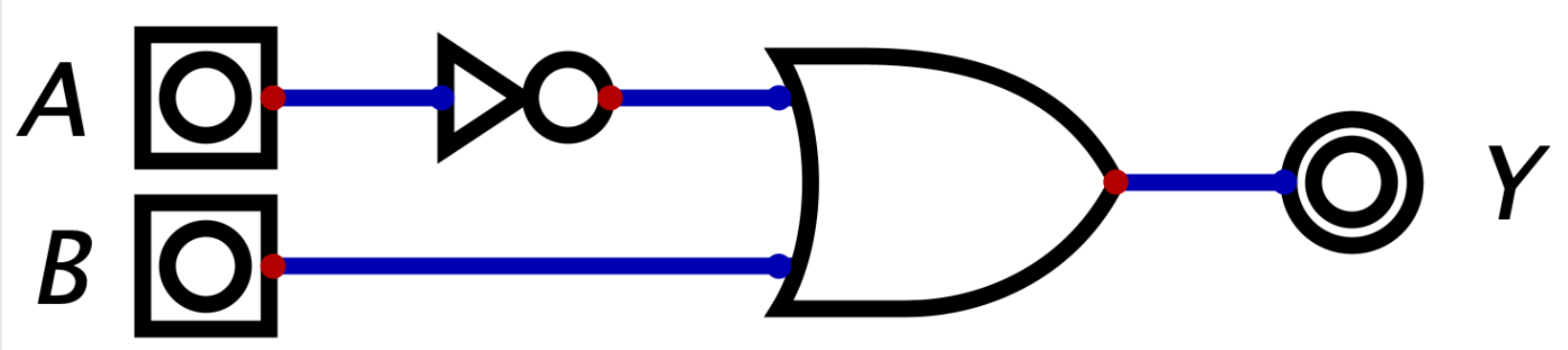
Y: 1 → 0

Ex1: Computing Total Propagation Delay



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

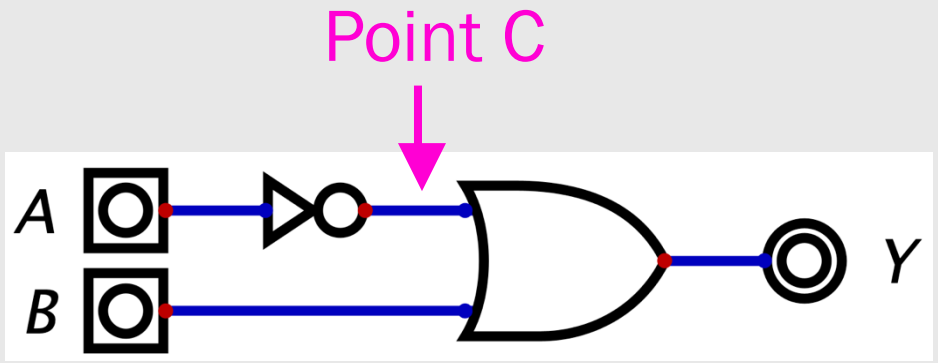
Ex1: Computing Total Propagation Delay



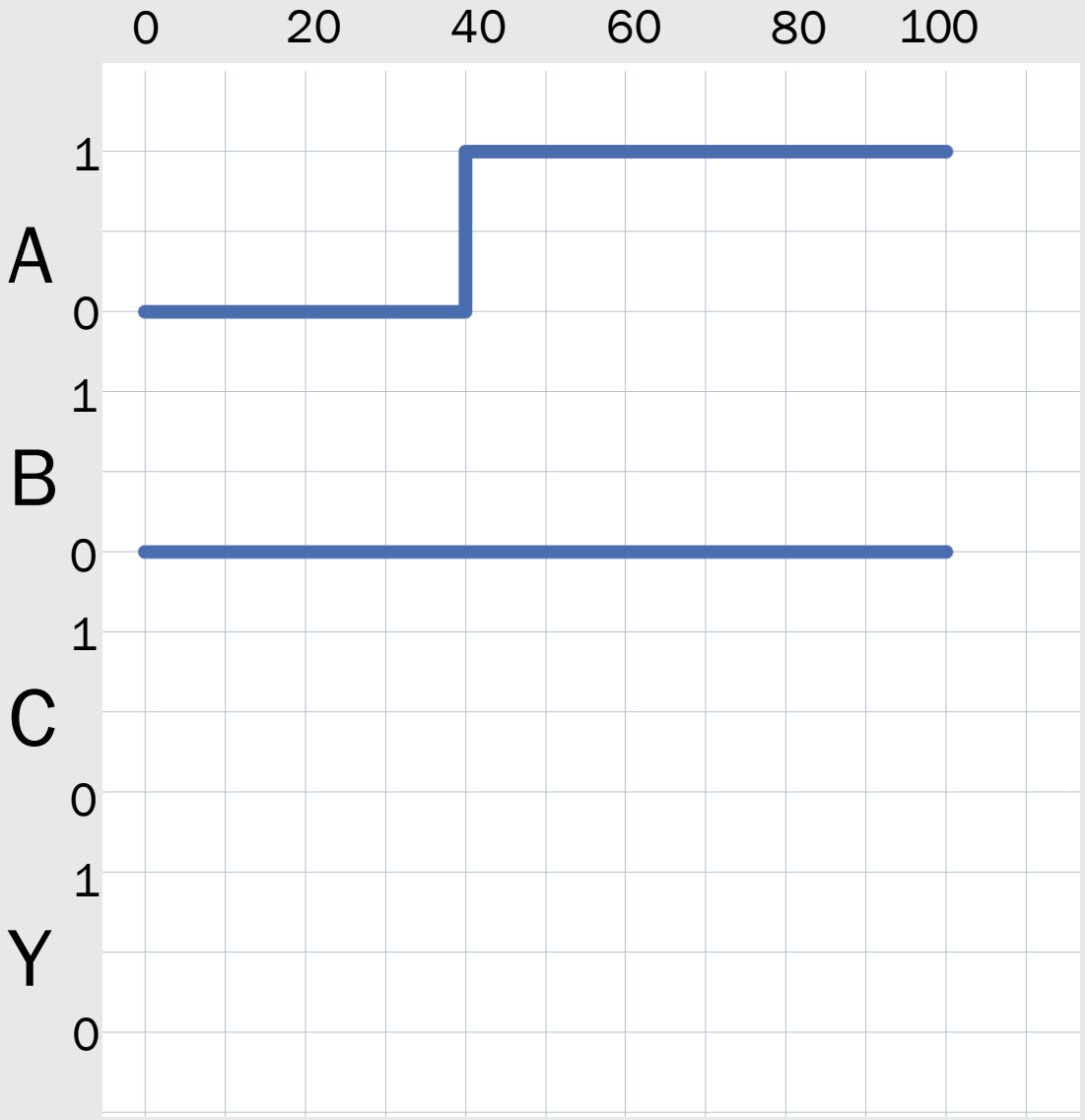
Gate	t_{pd} (ps)
NOT	10
OR	20

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

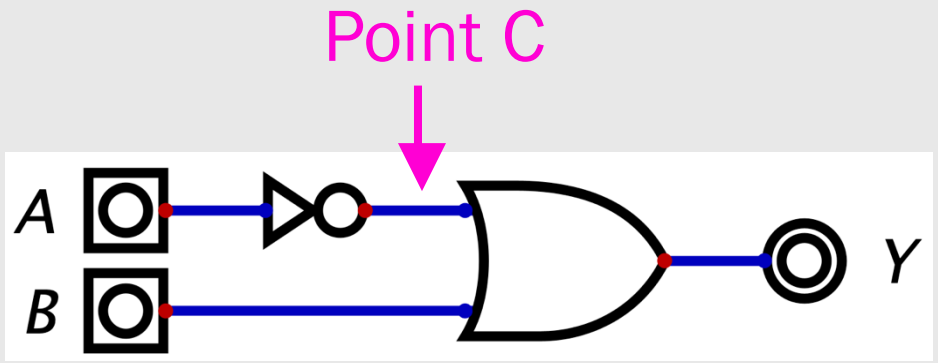
Ex1: Waveform Diagram with Propagation Delay



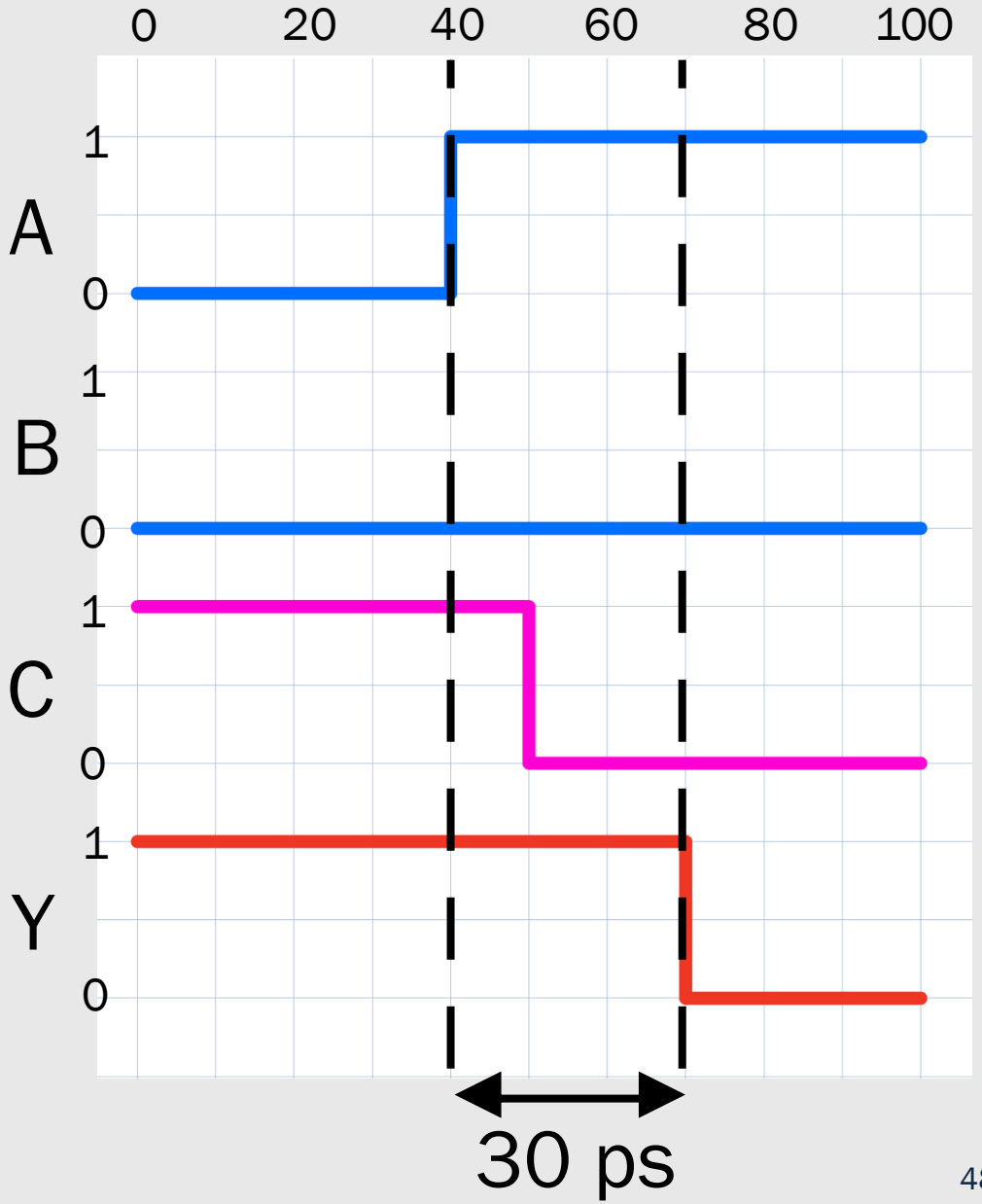
Gate	t_{pd} (ps)
NOT	10
OR	20



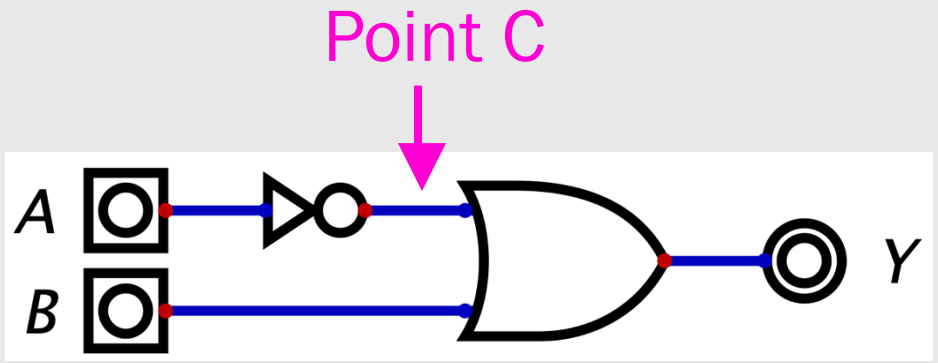
Ex1: Waveform Diagram with Propagation Delay



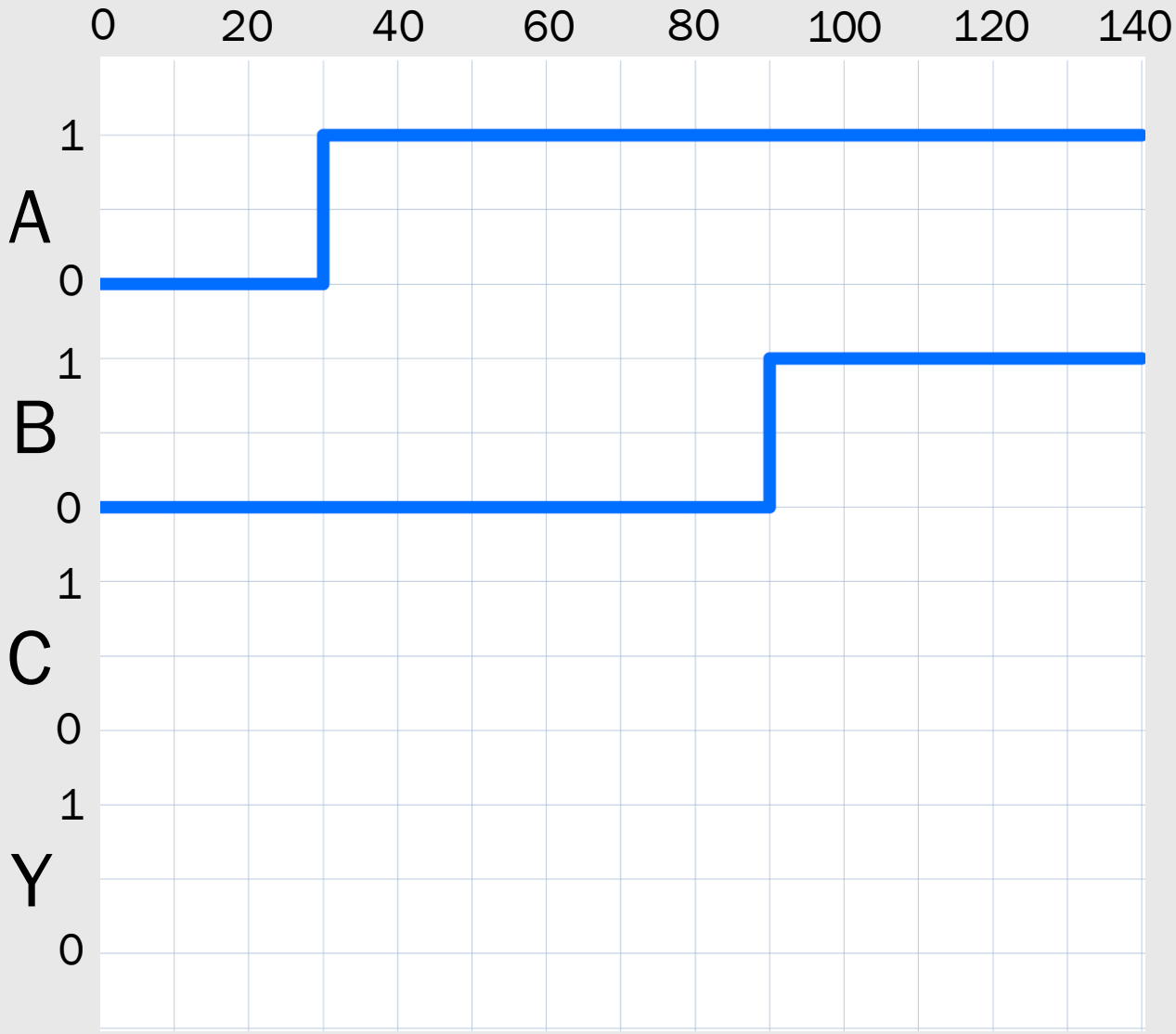
Gate	t_{pd} (ps)
NOT	10
OR	20



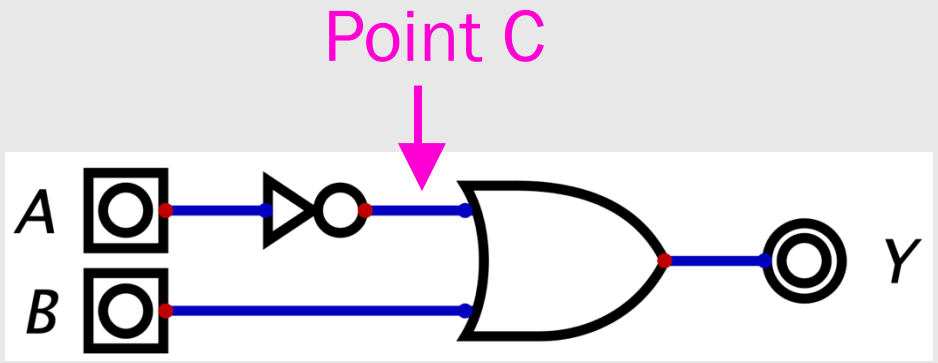
Ex2: Waveform Diagram with Propagation Delay



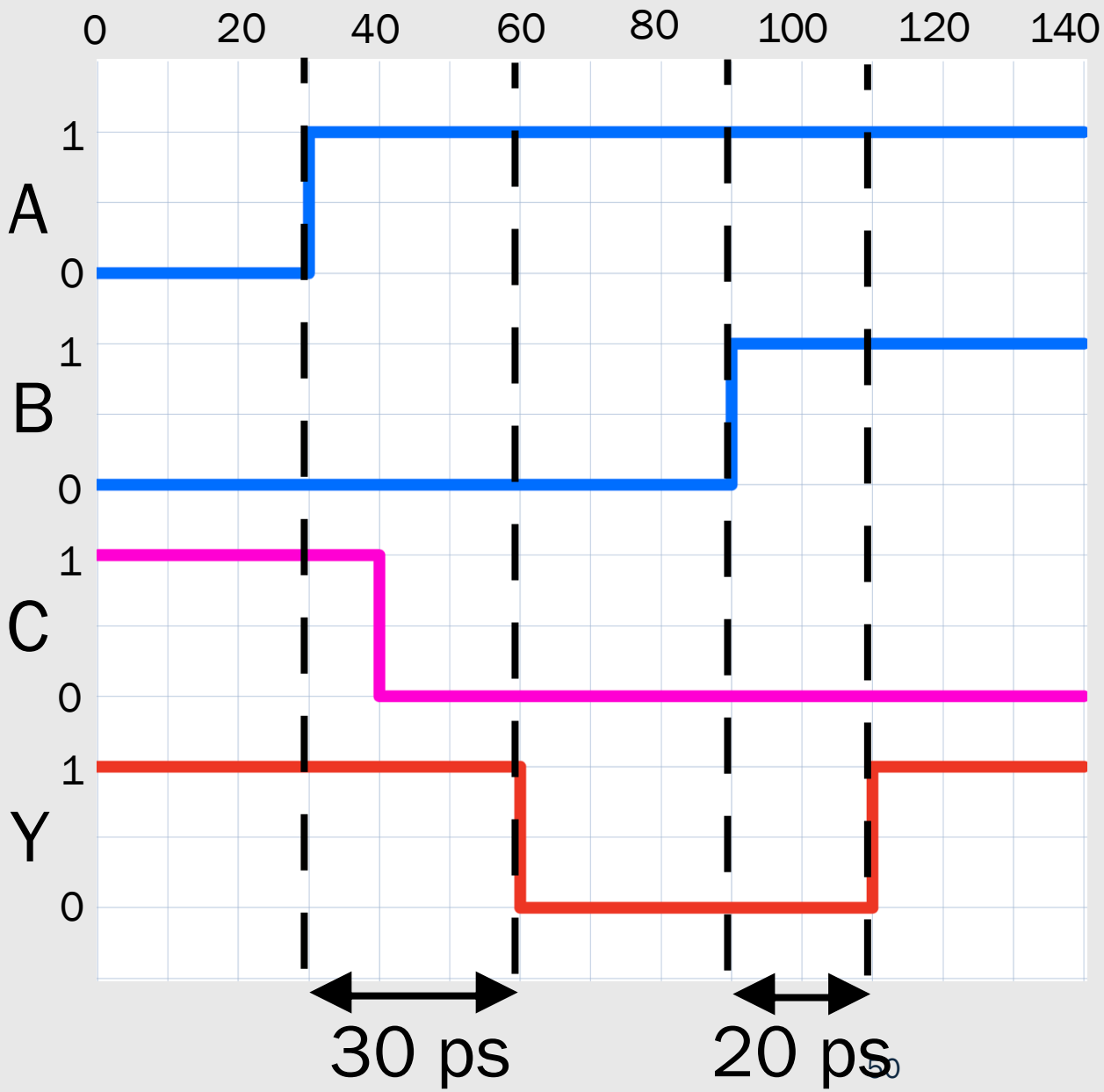
Gate	t_{pd} (ps)
NOT	10
OR	20



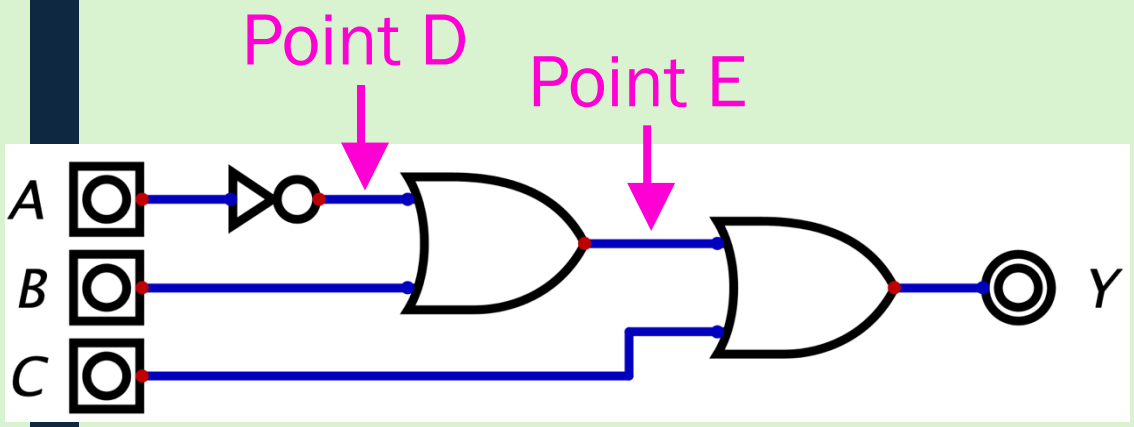
Ex2: Waveform Diagram with Propagation Delay



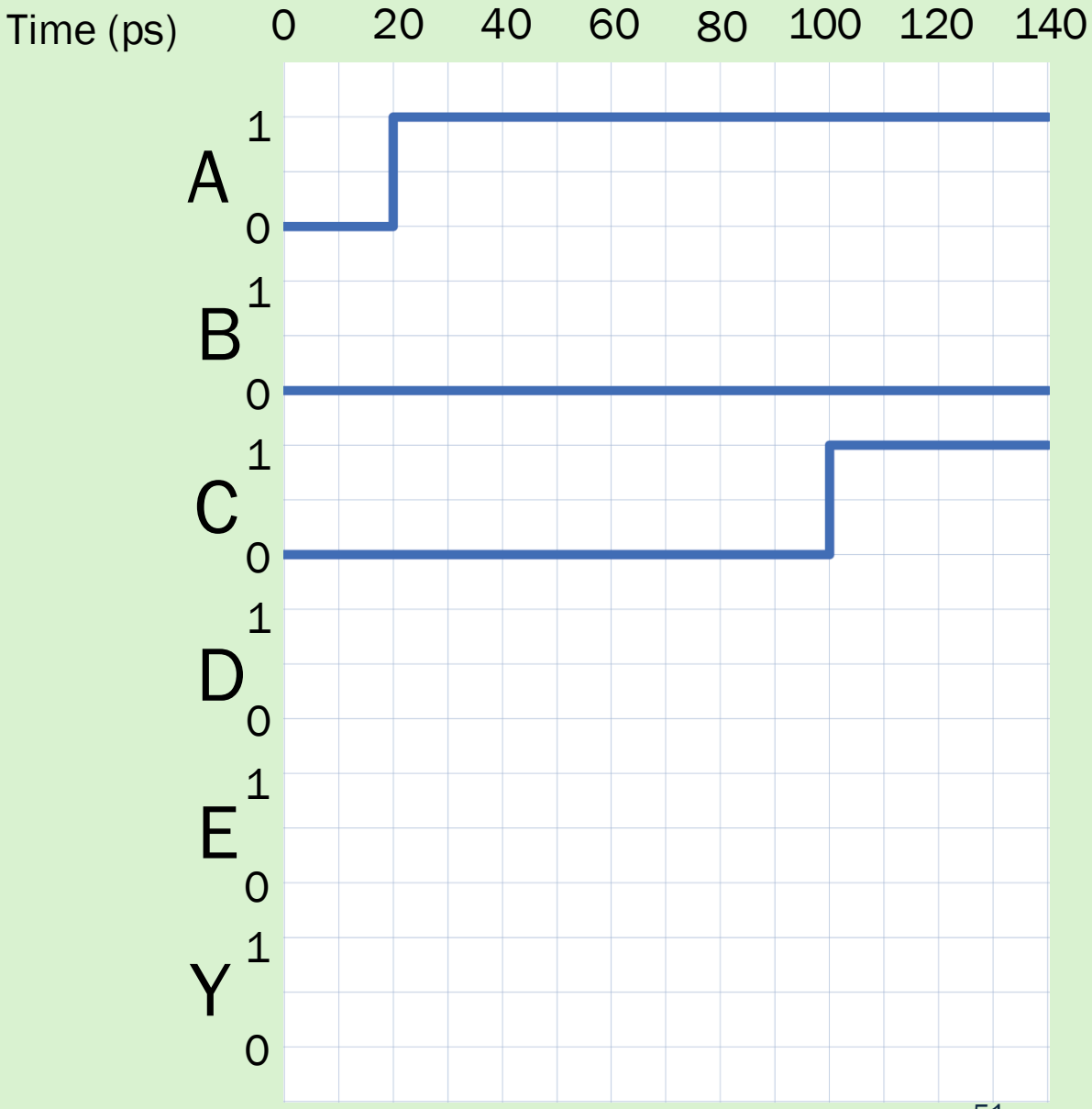
Gate	t_{pd} (ps)
NOT	10
OR	20



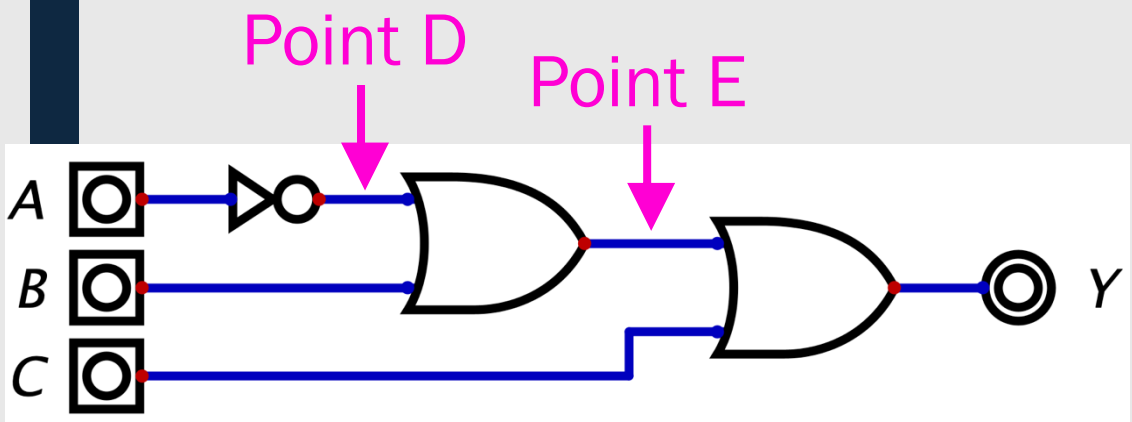
Waveform Diagram with Propagation Delay



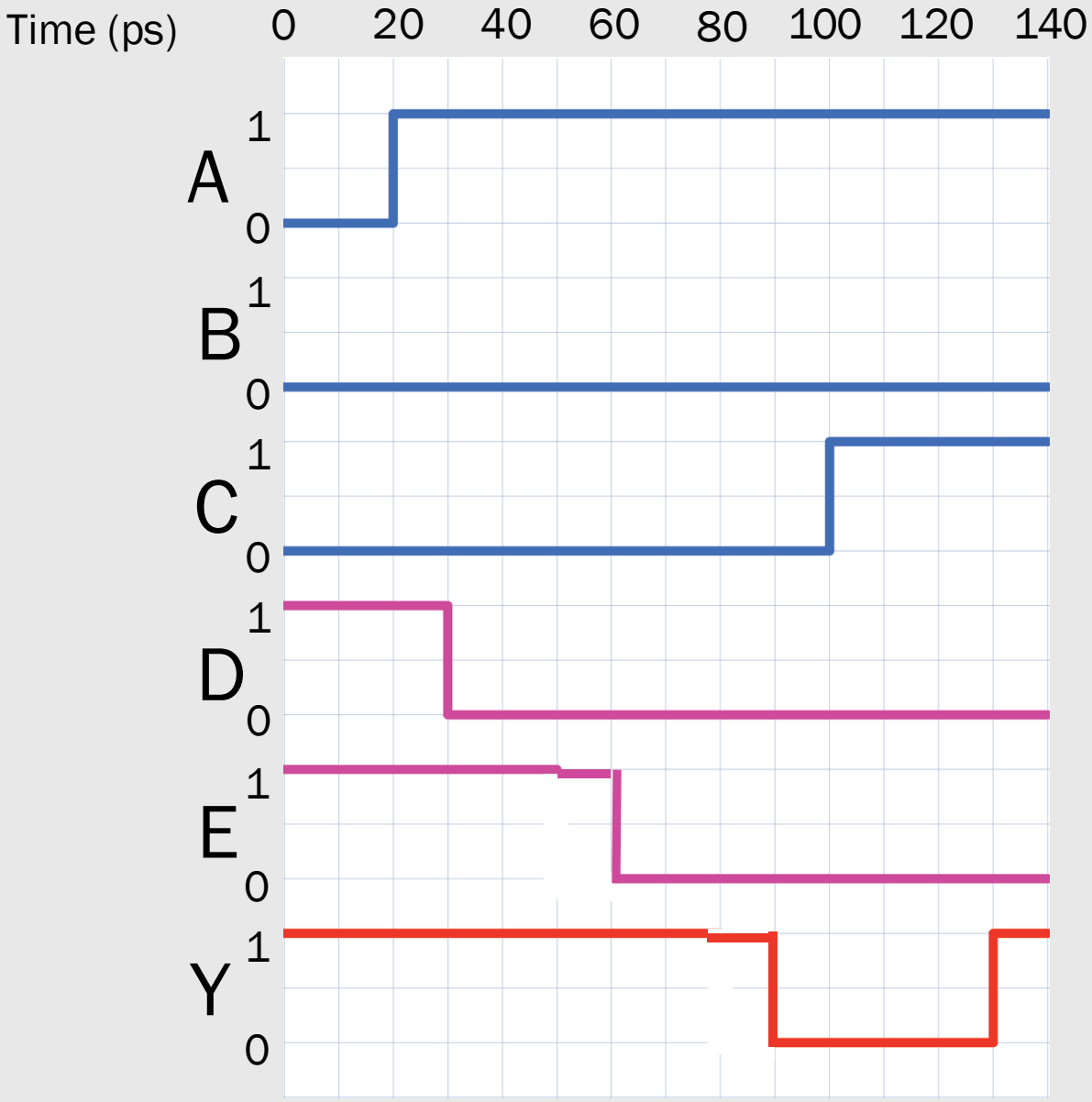
Gate	t_{pd} (ps)
NOT	10
OR	30



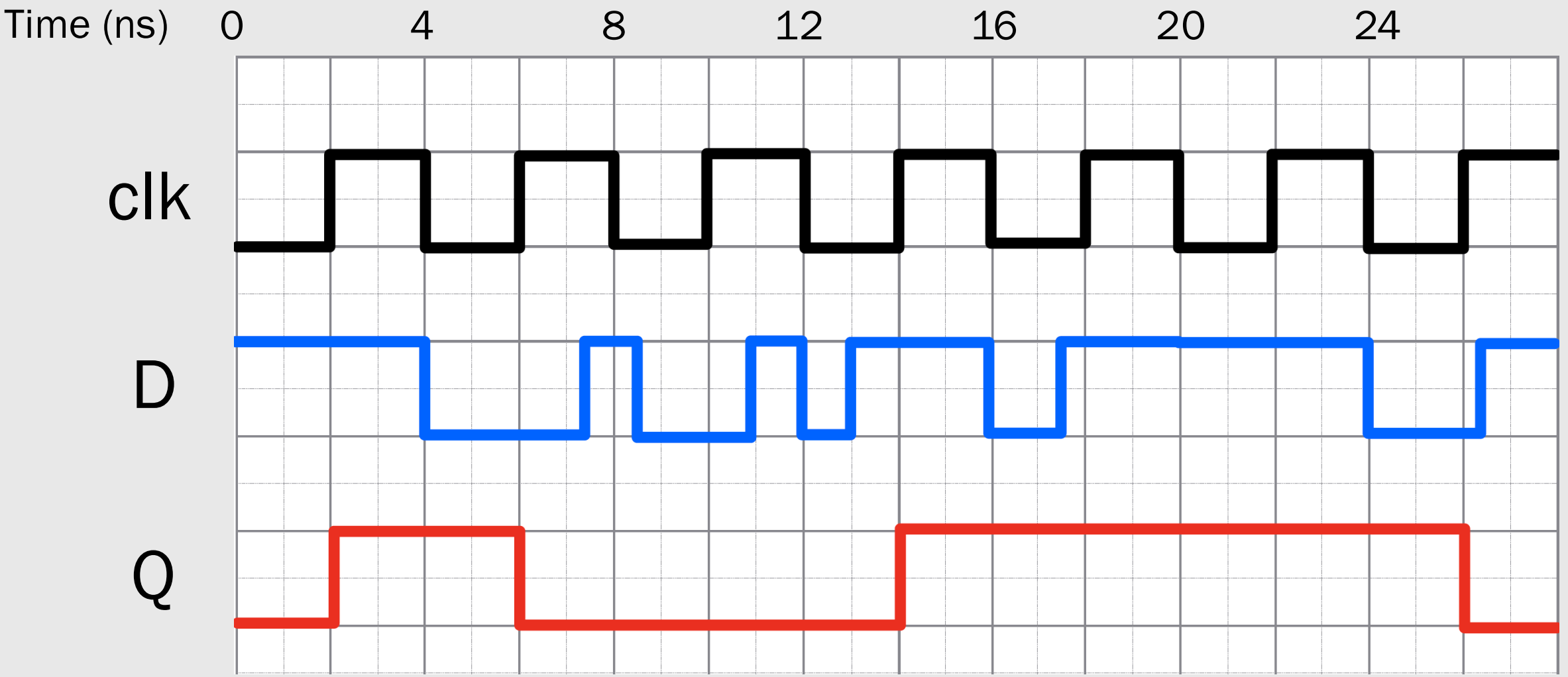
Waveform Diagram with Propagation Delay



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

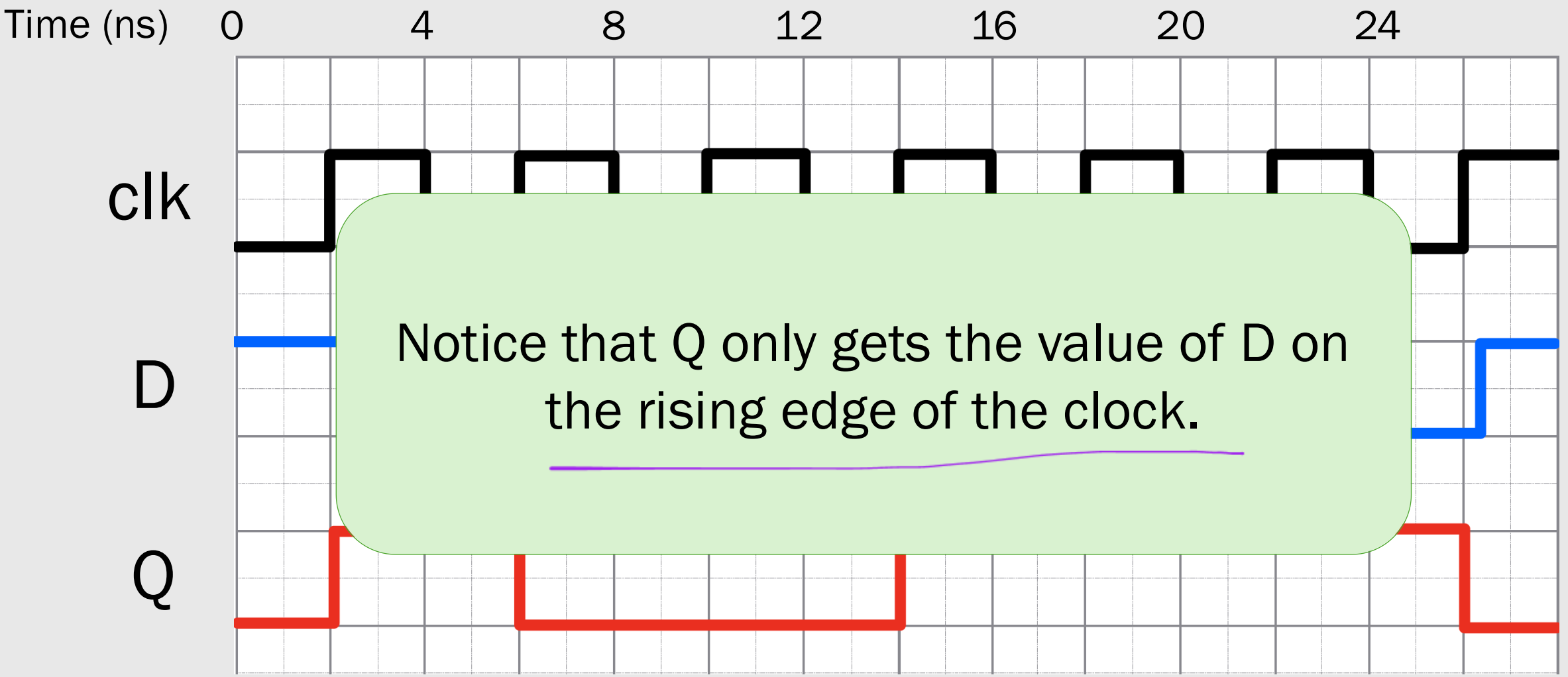


Positive Edge Triggered D Flip-Flop Waveform Diagram



assume Q starts at 0

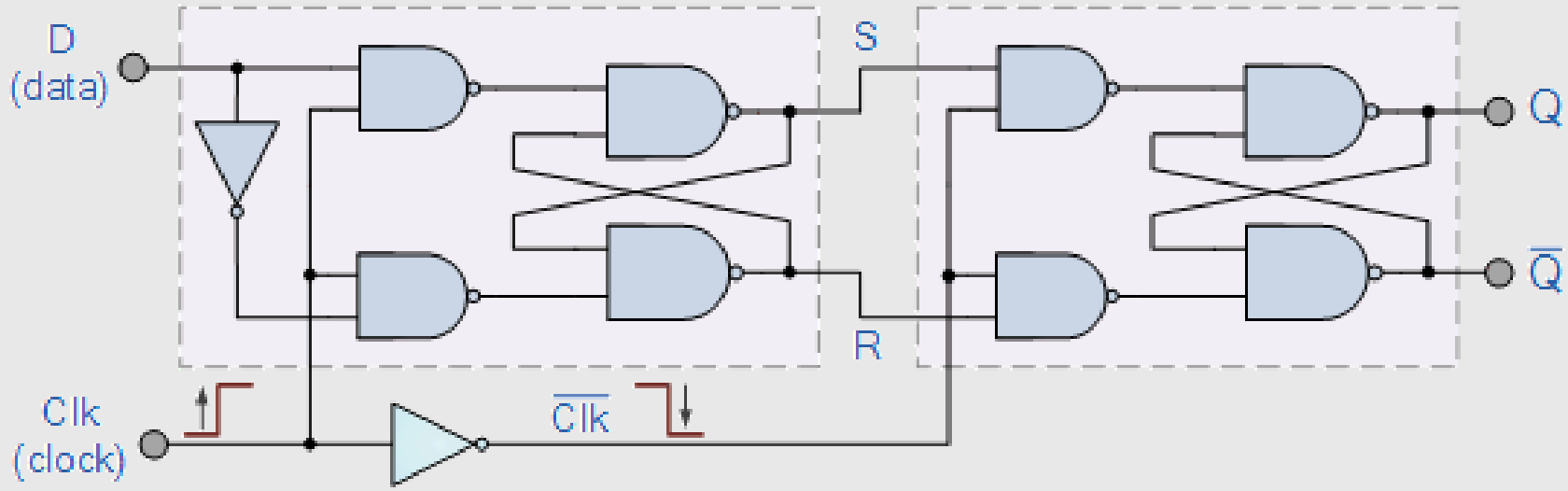
Positive Edge Triggered D Flip-Flop Waveform Diagram



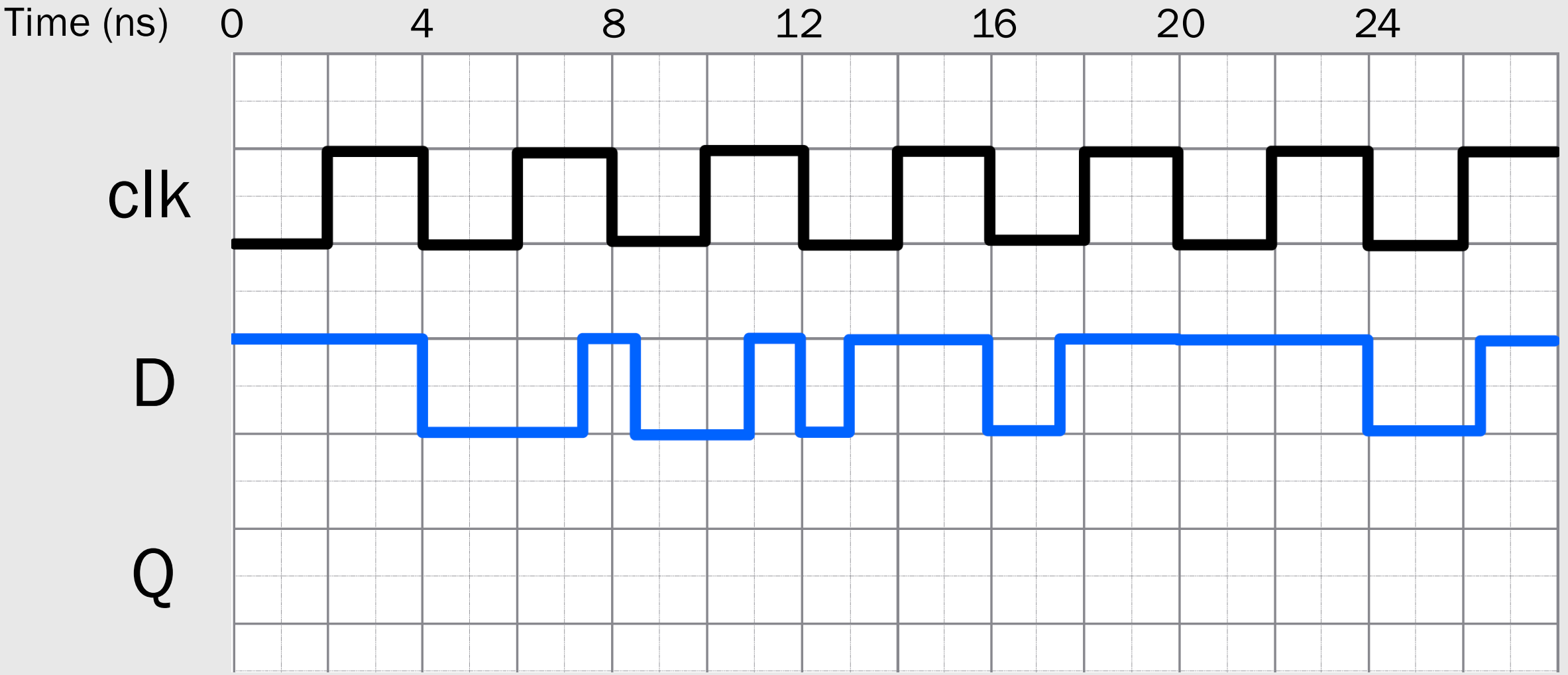
assume Q starts at 0

Clock to Q Delay

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



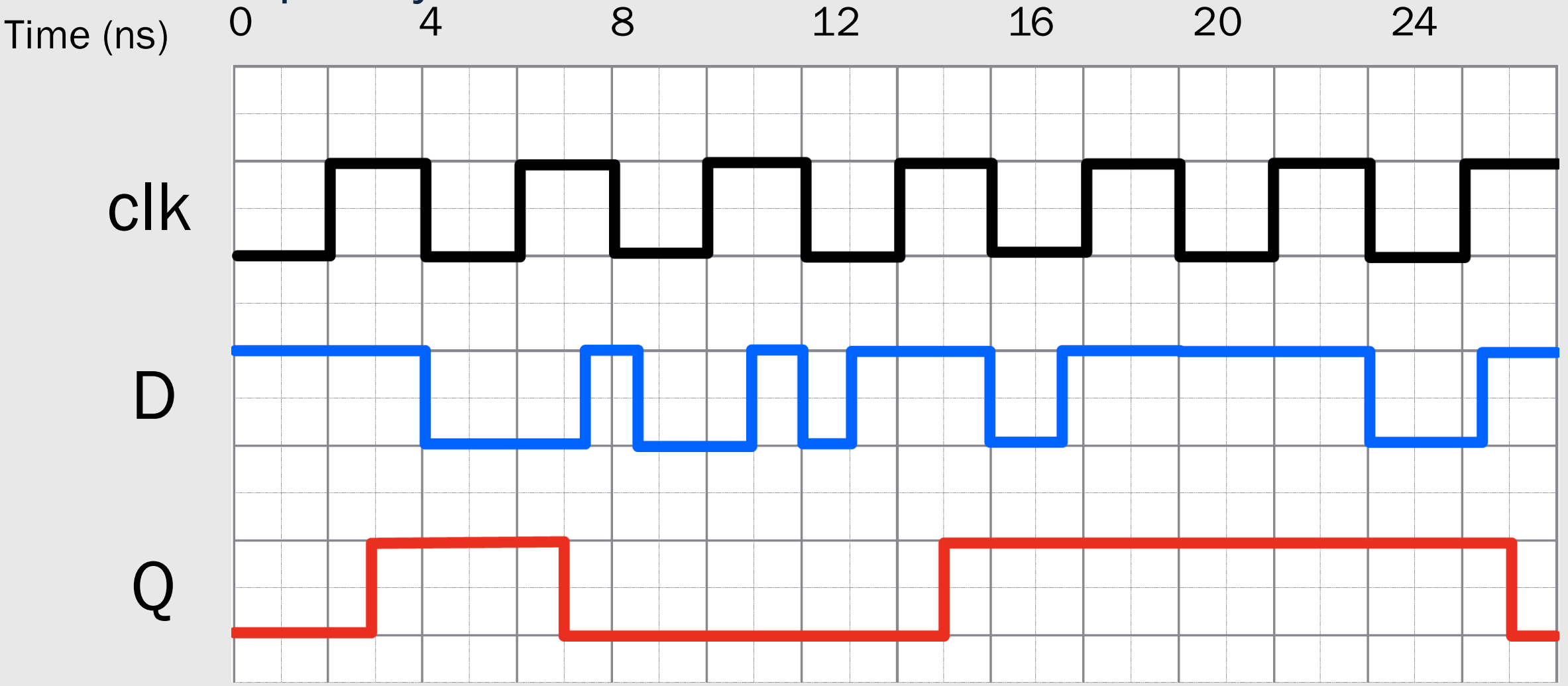
Positive Edge Triggered Flip-Flop Waveform Diagram with clk-to-q Delay



assume Q starts at 0

clk-to-q delay = 1 ns

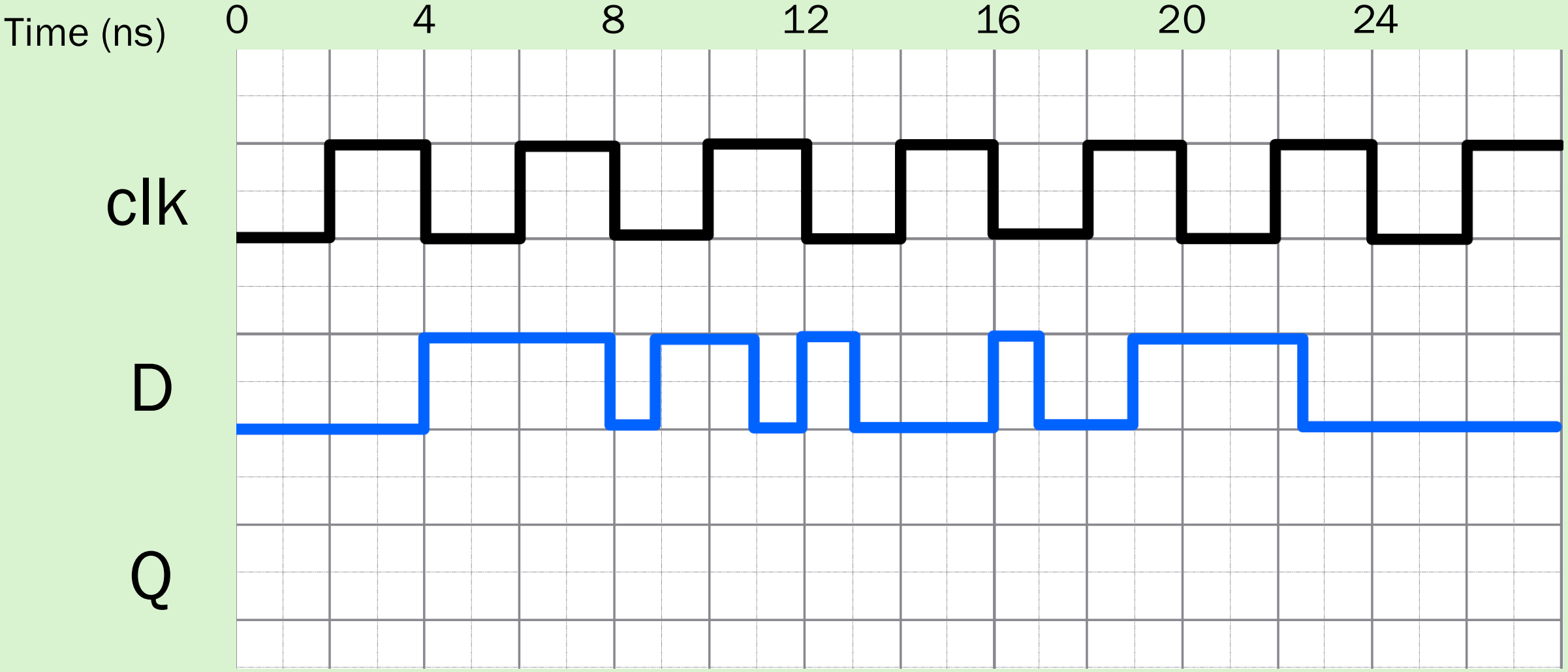
Positive Edge Triggered Flip-Flop Waveform Diagram with clk-to-q Delay



assume Q starts at 0

clk-to-q delay = 1 ns

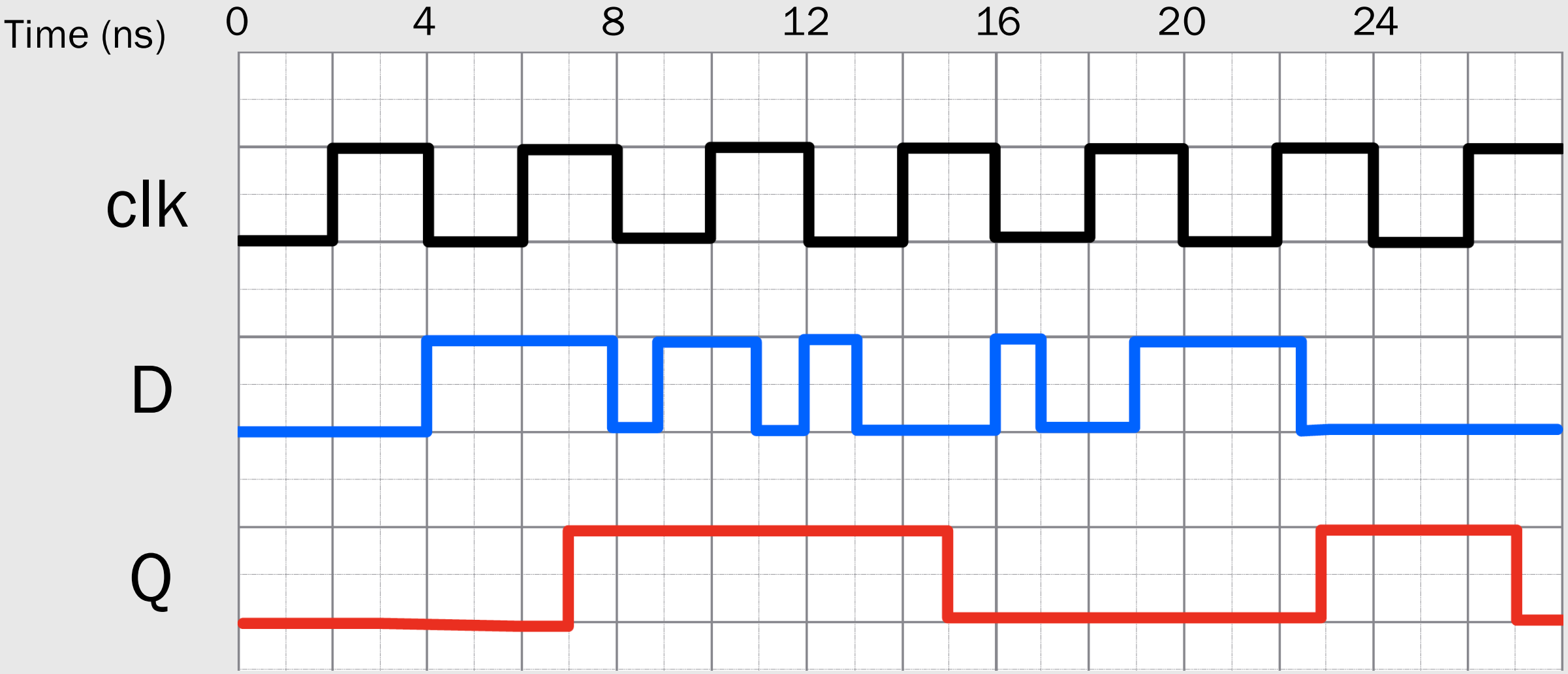
Positive Edge Triggered Flip-Flop Waveform Diagram with clk-to-q Delay



assume Q starts at 0

clk-to-q delay = 1 ns

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



assume Q starts at 0

clk-to-q delay = 1 ns