pollev.com/kakiryan

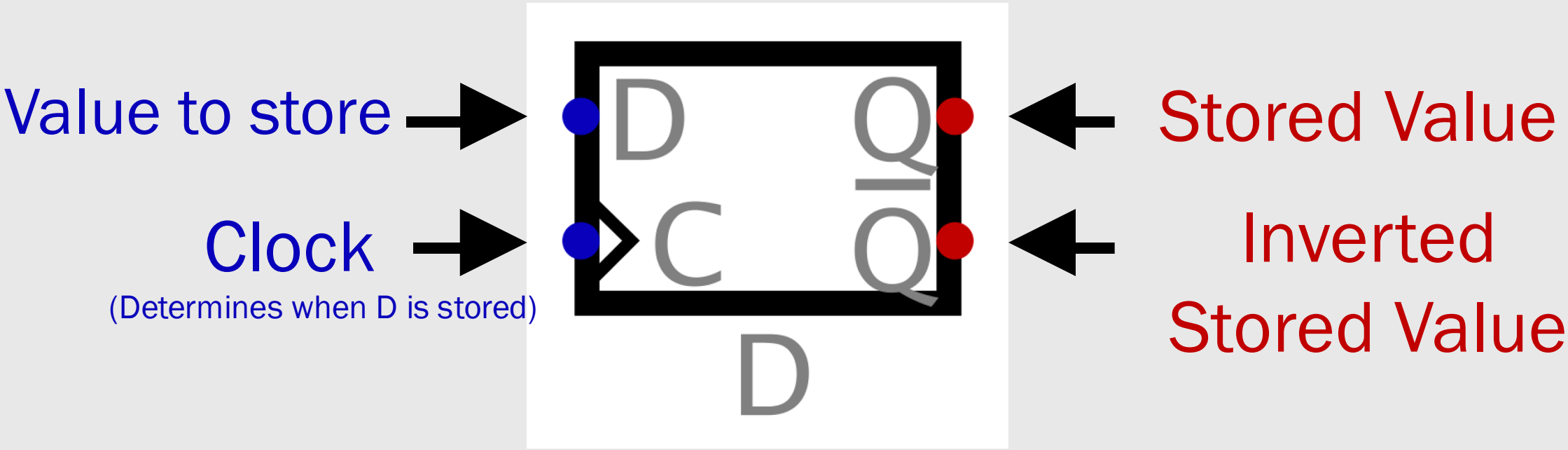
COMP311: *COMPUTER ORGANIZATION!*

Lecture 15: Registers, Timing Analysis cont.

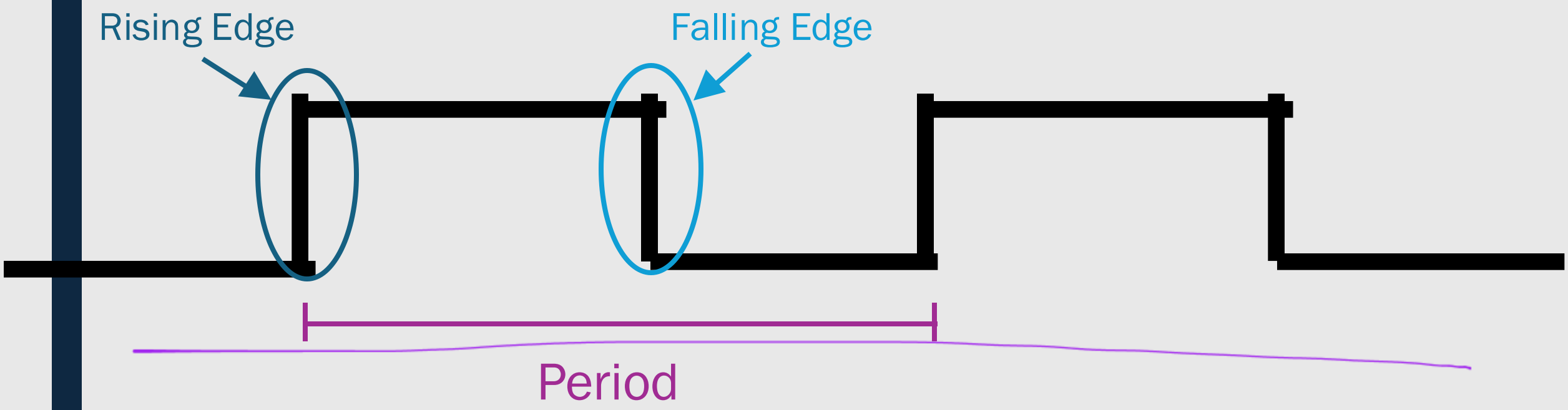
tinyurl.com/comp311-fa25

FLIP FLOPS AND REGISTERS

D Flip-Flop


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

Clock

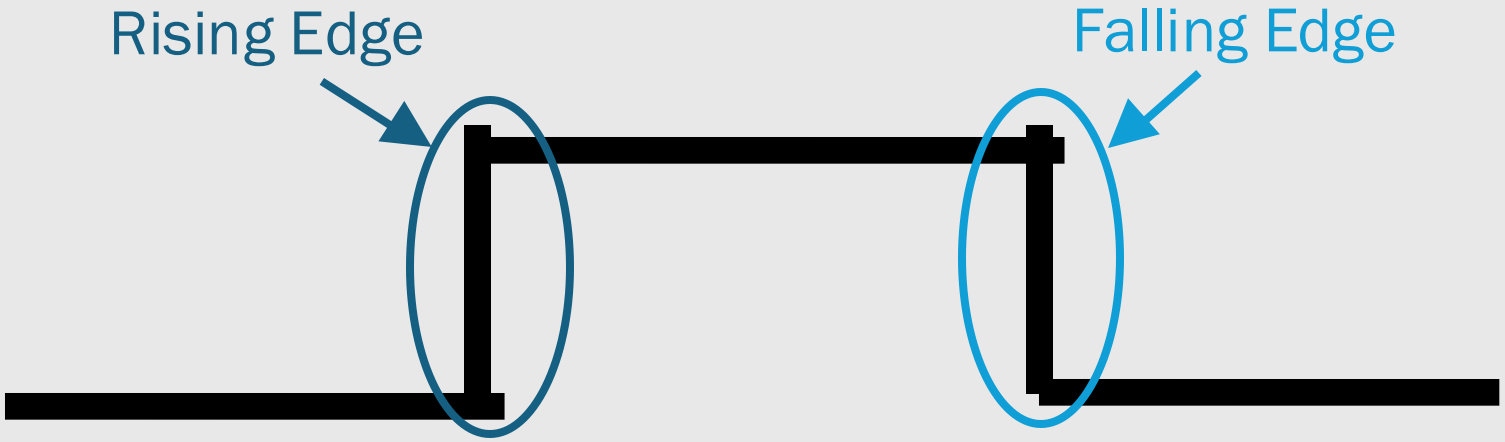


Clock

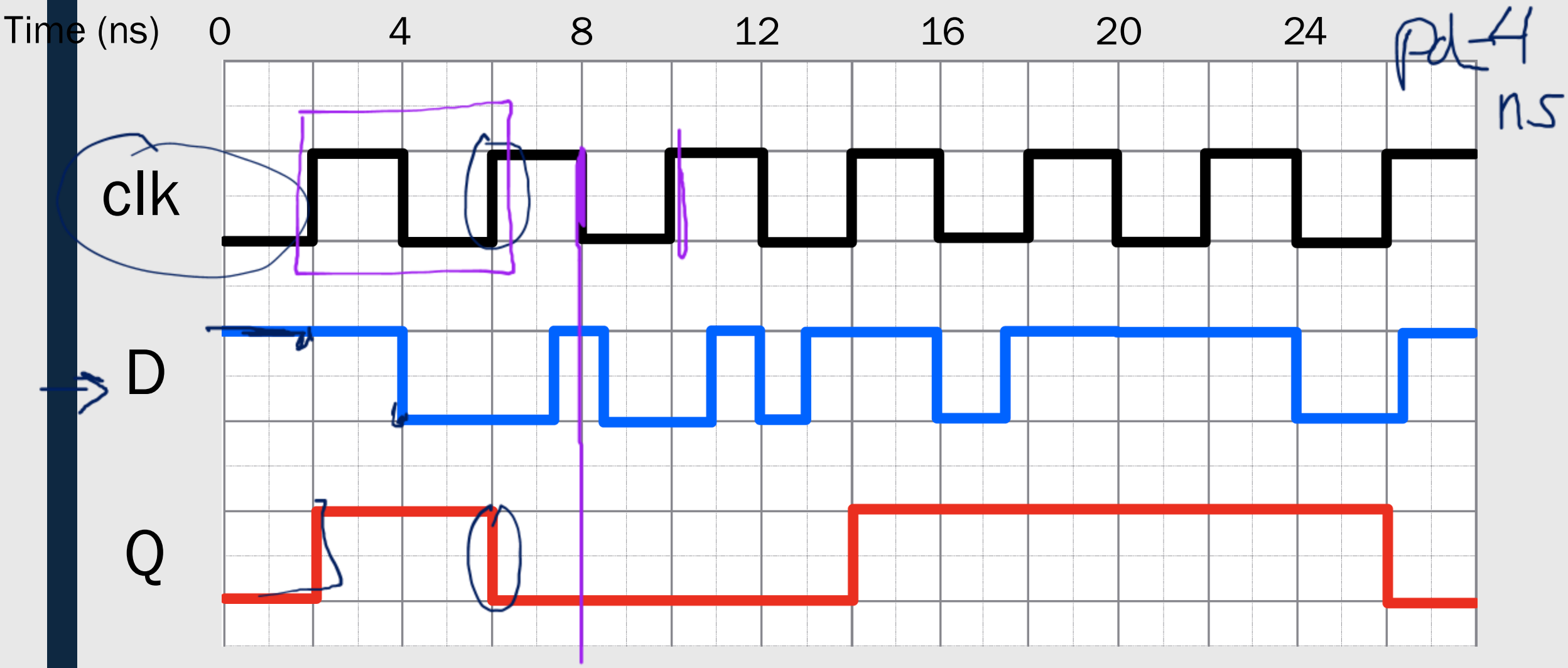
- The clock is a one-bit signal that oscillates (or toggles) back and forth between 0 and 1
- The purpose of the clock is to ensure that our circuits stay in sync
- The period of the clock is the time between one rising edge to the next. The clock spends an equal time in low and high states.
  - *The 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*

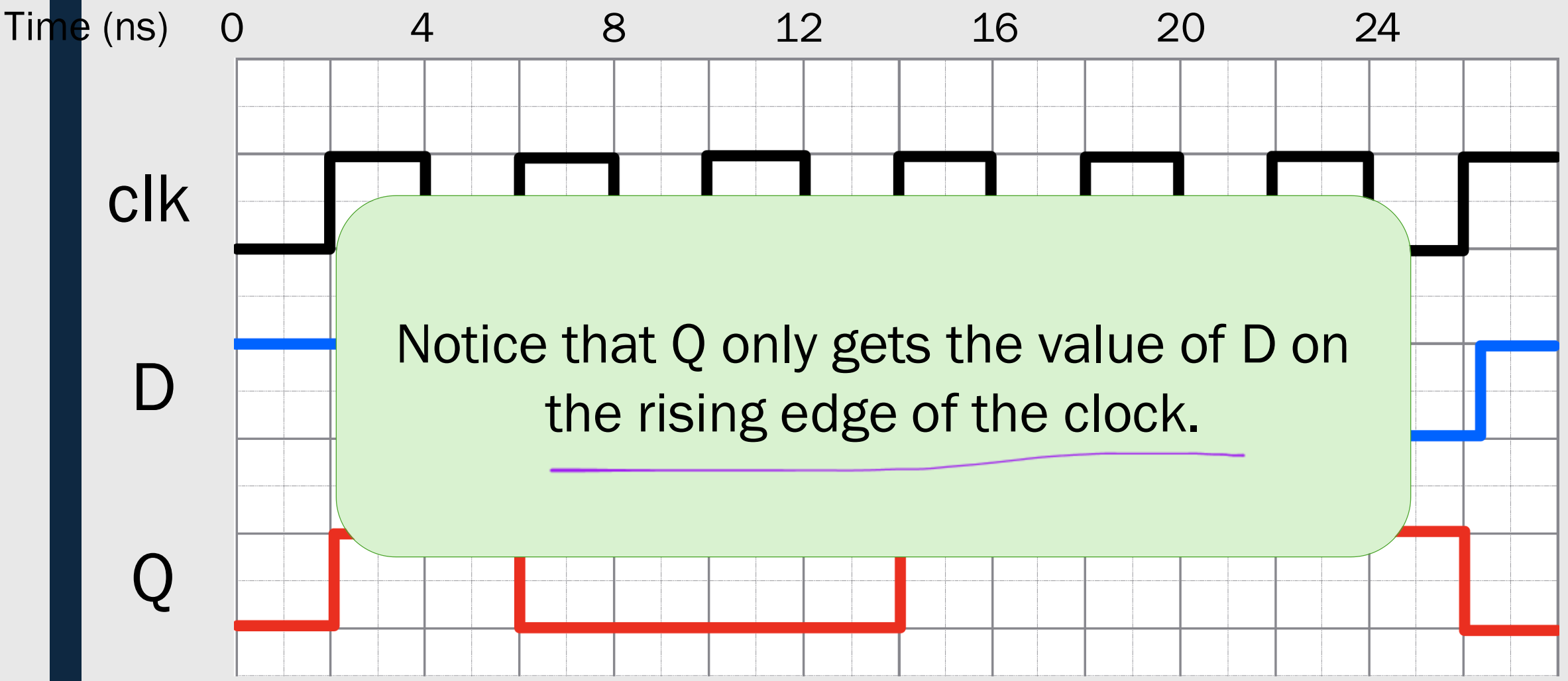


Positive Edge Triggered D Flip-Flop Waveform Diagram



assume Q starts at 0

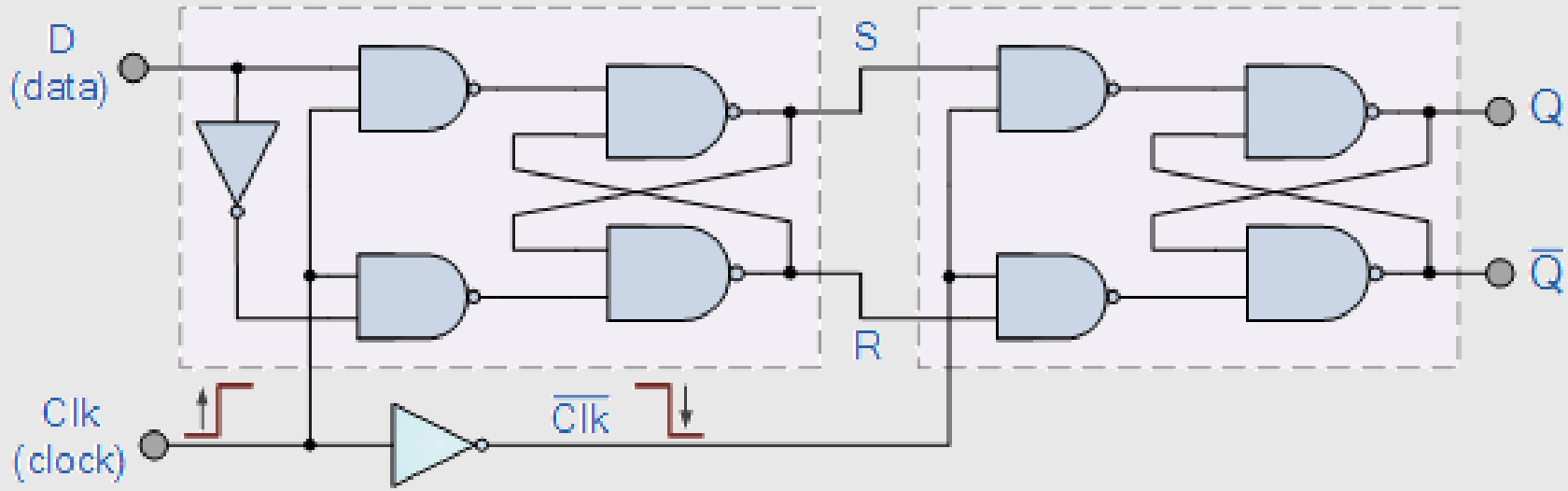
Positive Edge Triggered D Flip-Flop Waveform Diagram



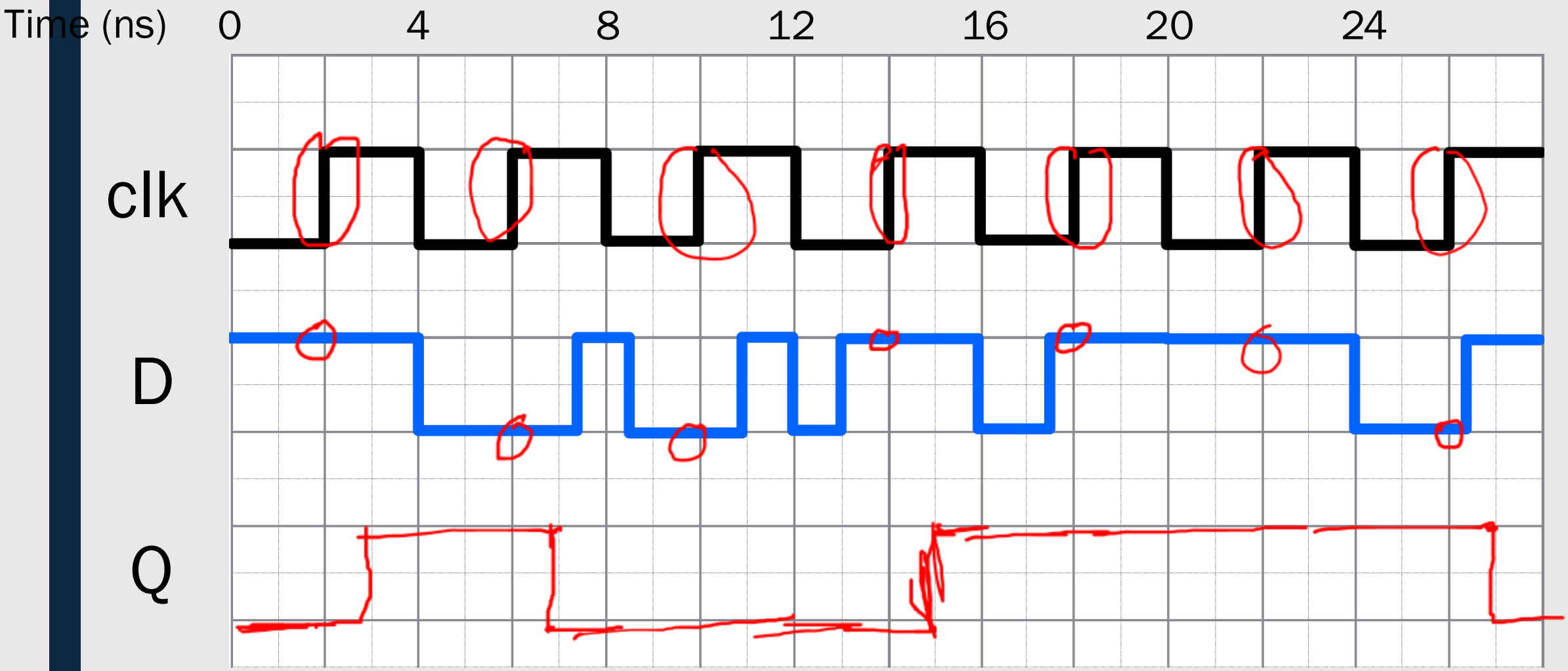
assume Q starts at 0

Clock to Q Delay

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



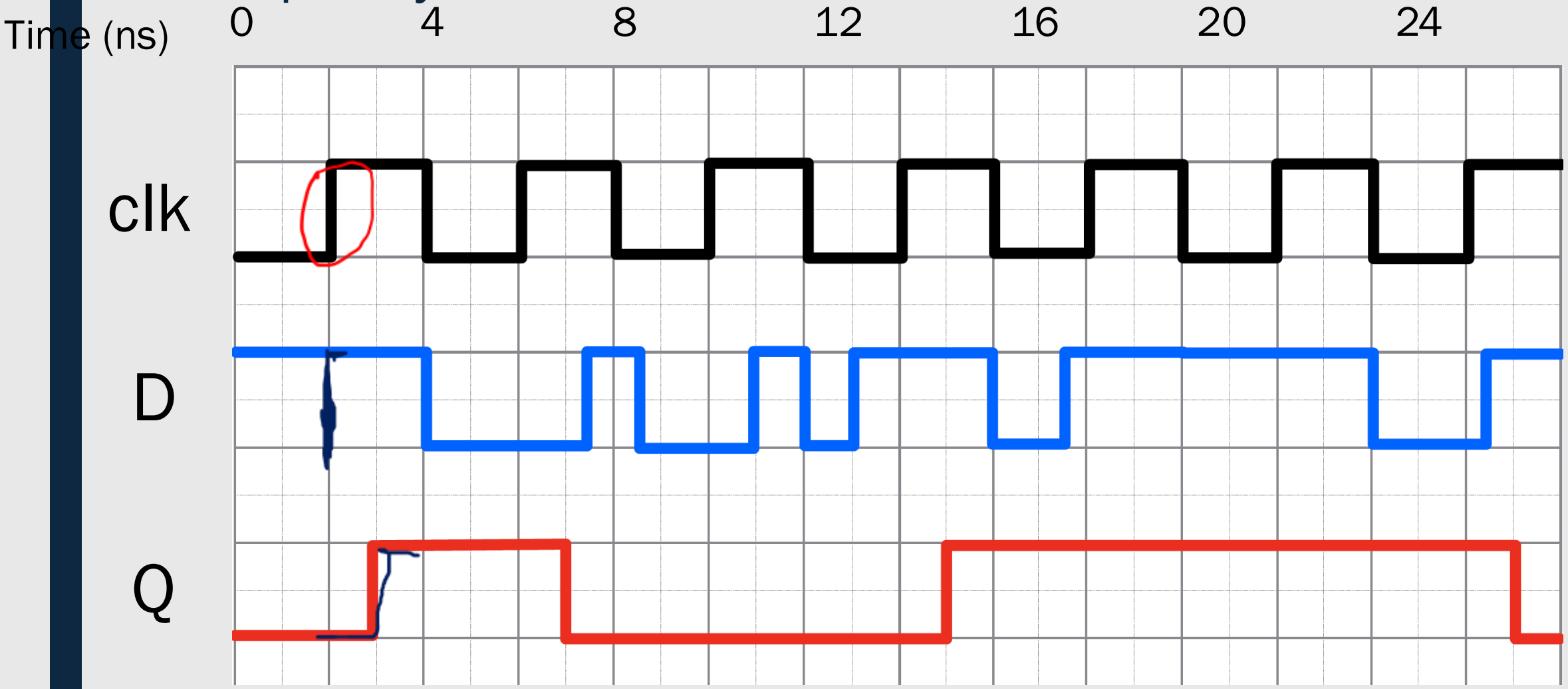
Positive Edge Triggered Flip-Flop Waveform Diagram with clk-to-q Delay



assume Q starts at 0

clk-to-q delay = 1 ns

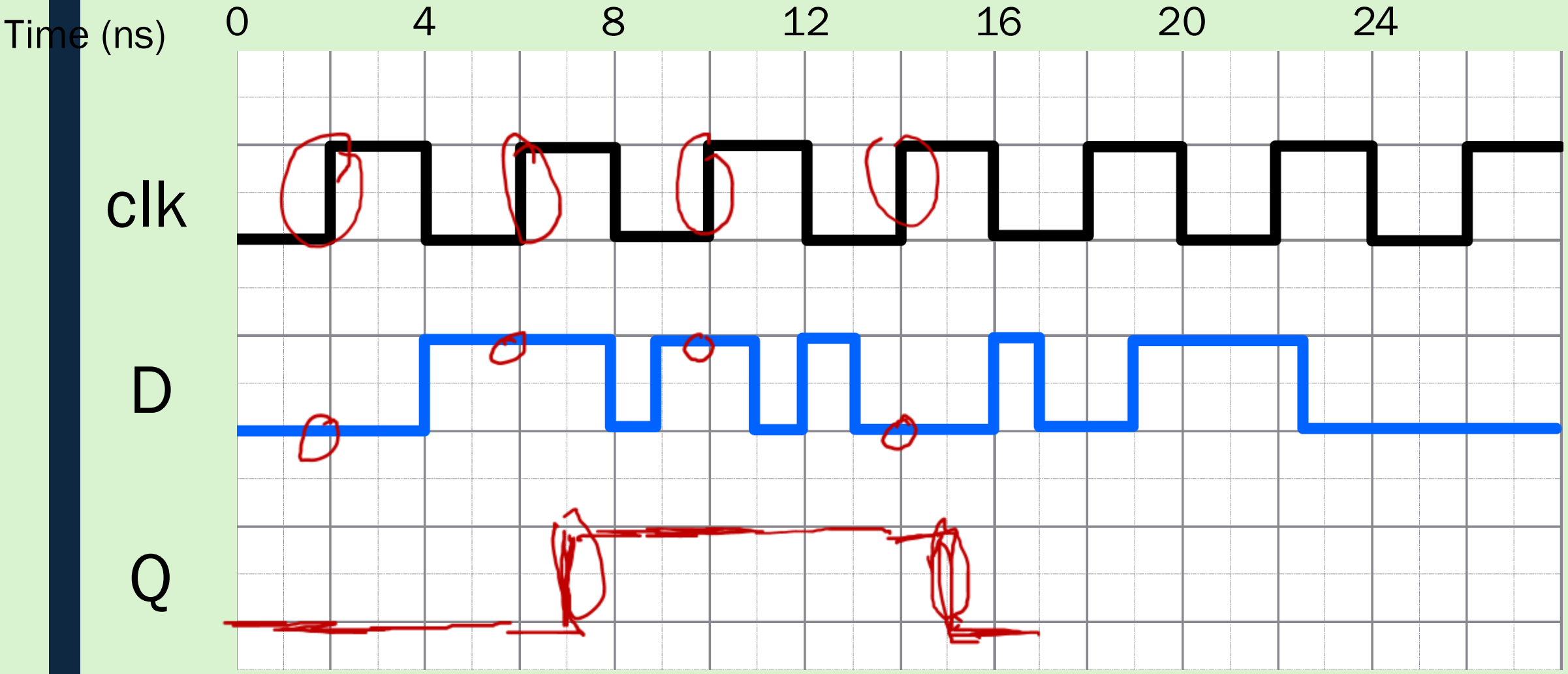
Positive Edge Triggered Flip-Flop Waveform Diagram with clk-to-q Delay



assume Q starts at 0

clk-to-q delay = 1 ns

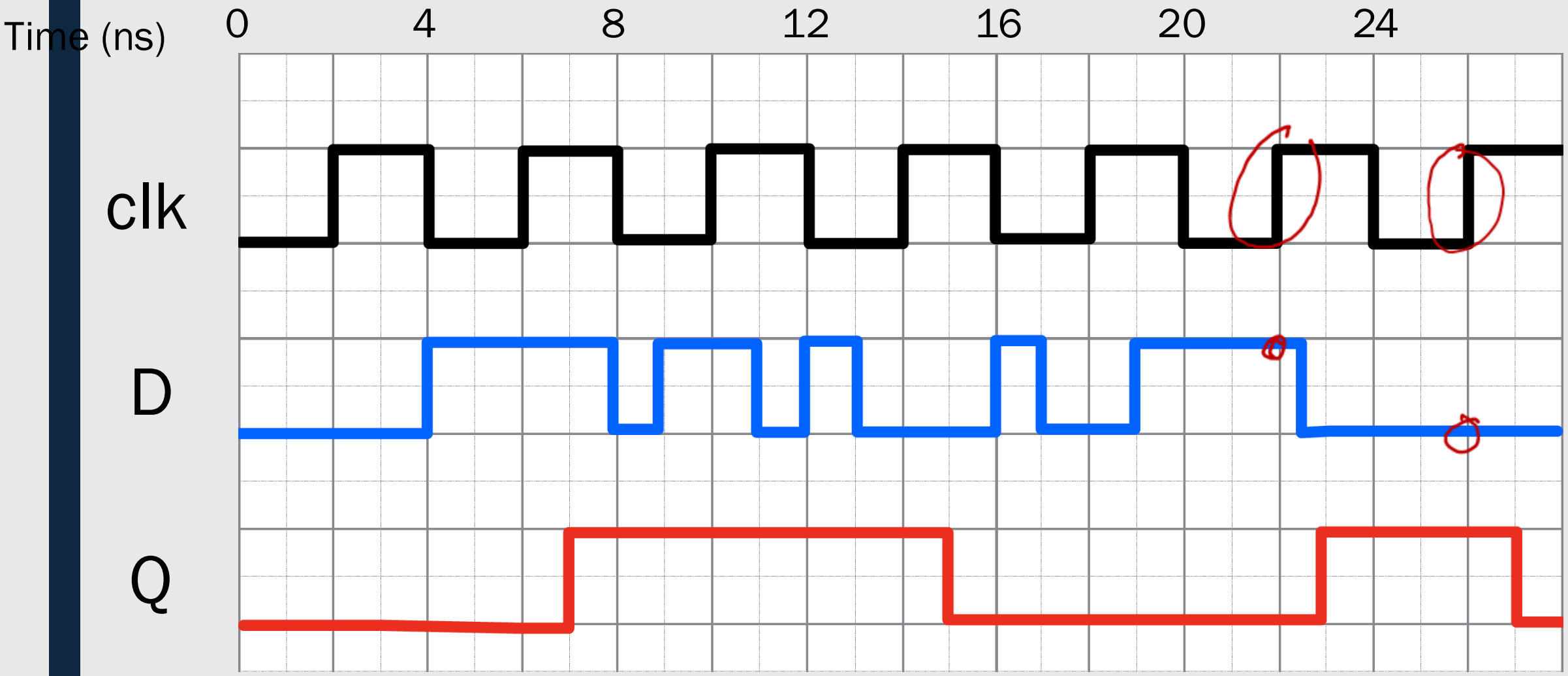
Positive Edge Triggered Flip-Flop Waveform Diagram with clk-to-q Delay



assume Q starts at 0

clk-to-q delay = 1 ns

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

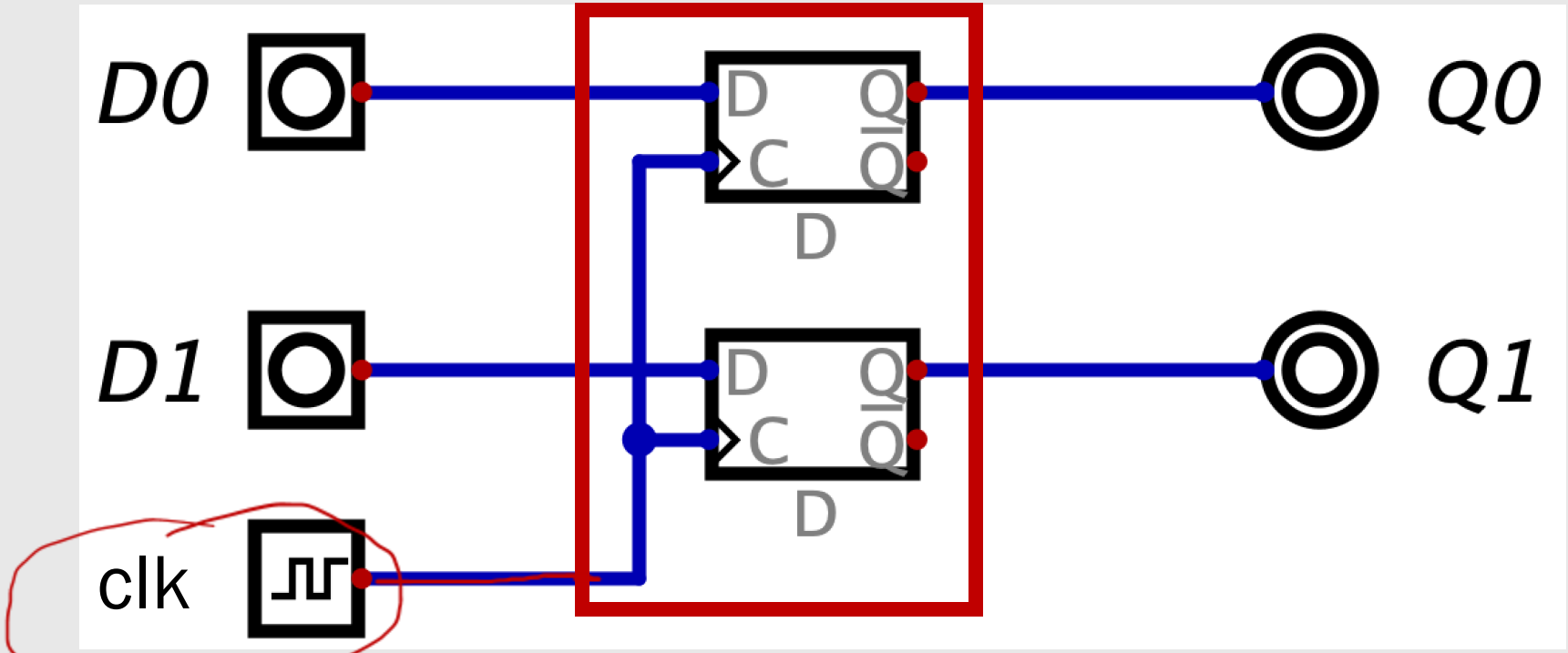


assume Q starts at 0

clk-to-q delay = 1 ns

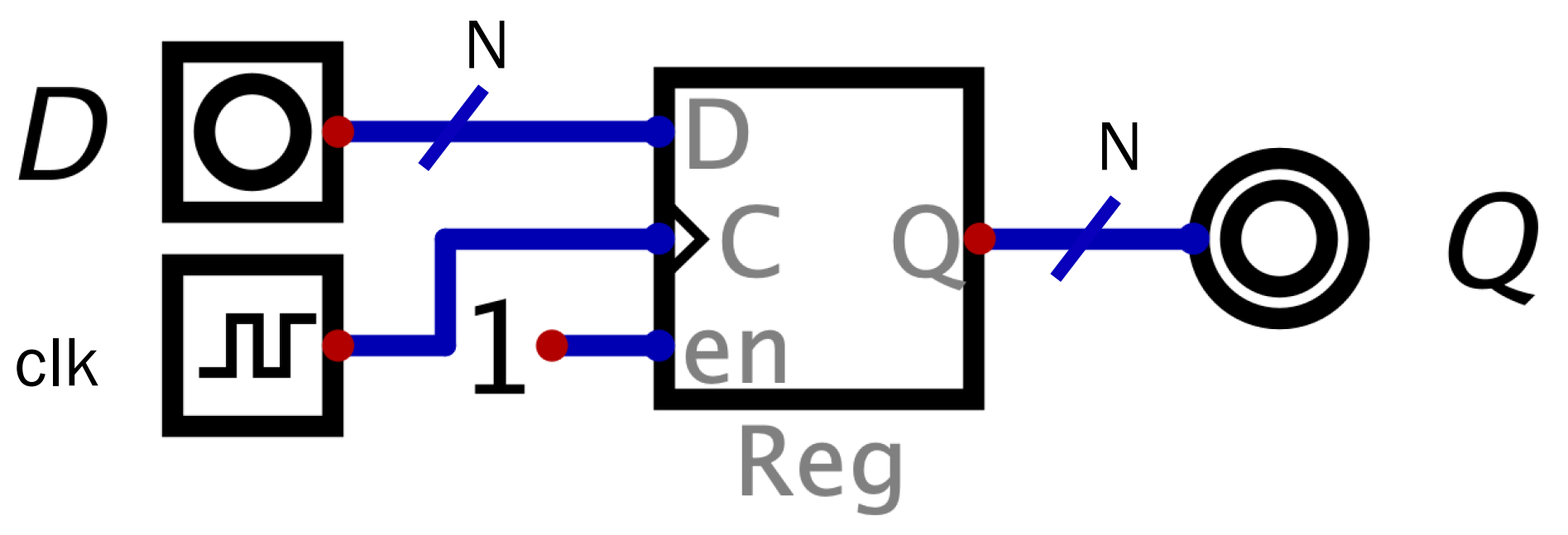
Registers

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

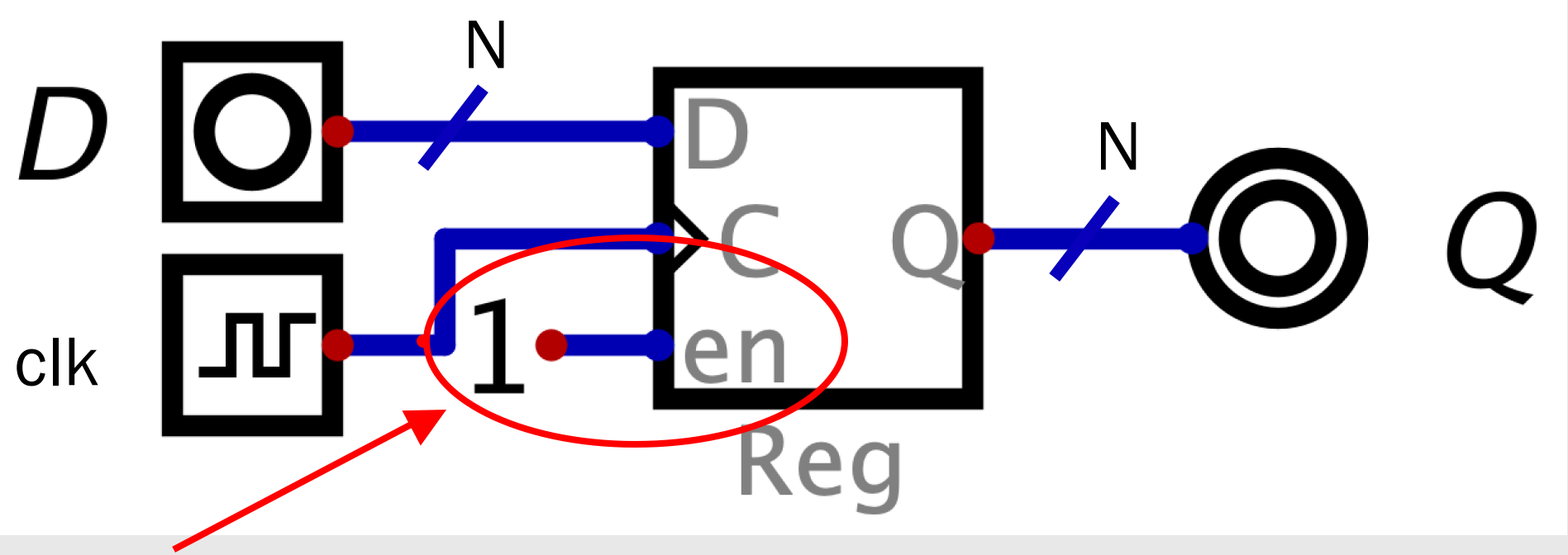


2-bit register

N-bit Positive Edge Triggered Register



N-bit Positive Edge Triggered Register



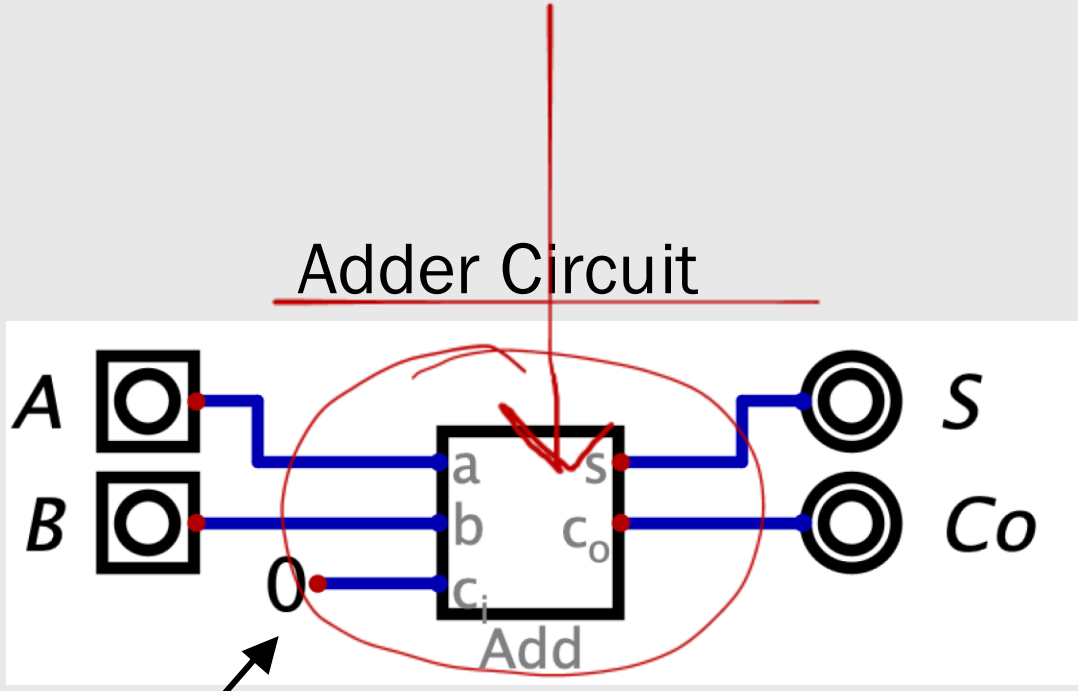
Enable

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

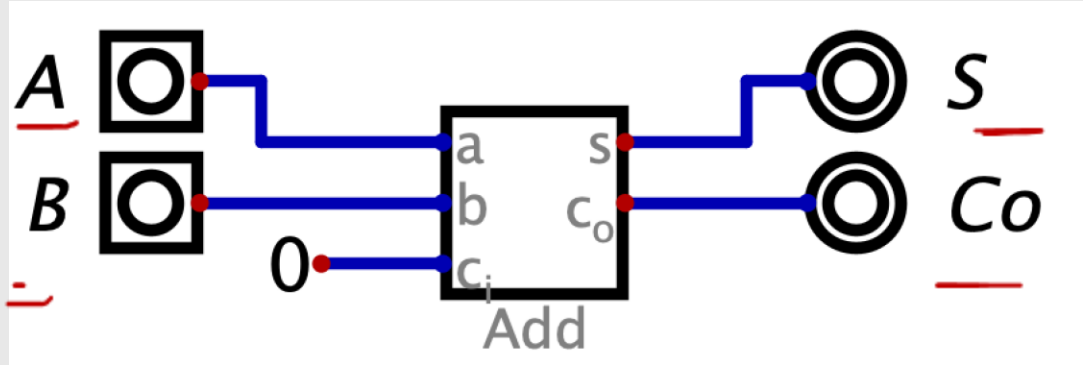
TIMING ANALYSIS

Back-to-Back Additions

- We want to perform the following additions
  - *7 + 7*
  - *18 + 7*
  - *30 + 8*
- This is a 10-bit adder, so none of these operations will overflow.
  - *C_o will be 0 for all of these operations*
- Let's say that the propagation delay of our adder circuit is 600ps



t = 0ps



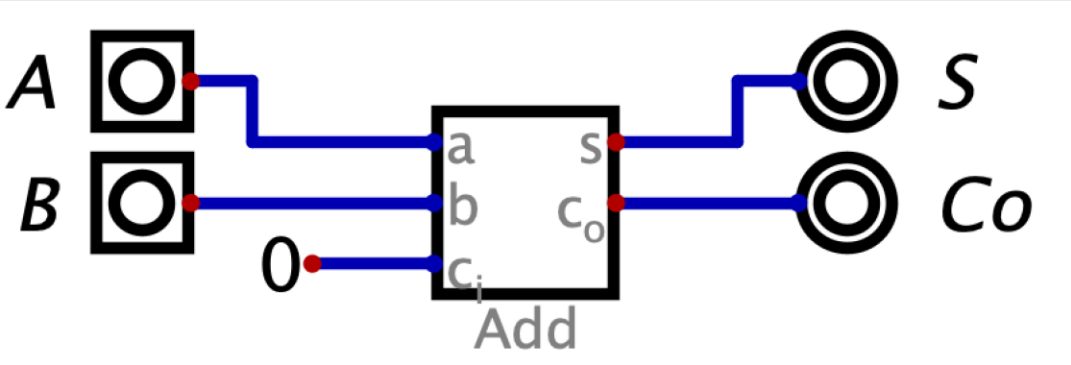
At time t = 0ps, we will
set the input A to 7 and the input B to 7.
S and C_o will start at 0.

t = 0ps

A = 7
B = 7

S = 0
Co = 0

t = 0ps



At time t = 600ps. The adder will finish its computation, so S = 14 and Co = 0.

We are now ready to perform the next addition. We will set A = 18 and B = 7.

t = 0ps

t = 600ps

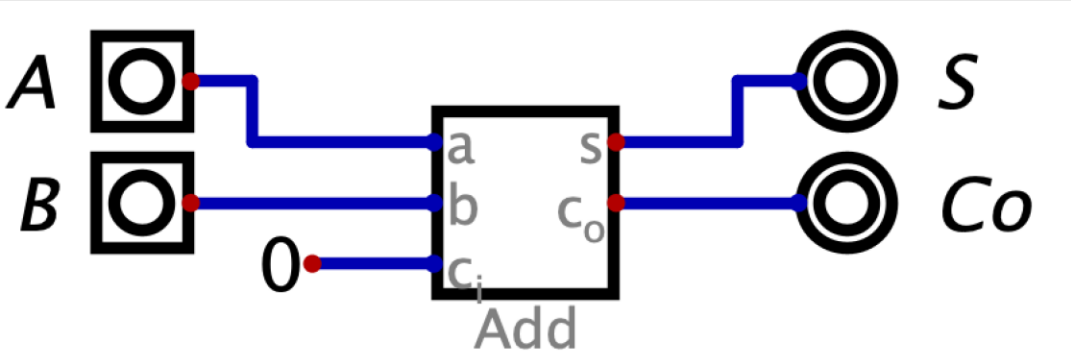
A = 7
B = 7

S = 0
Co = 0

A = 18
B = 7

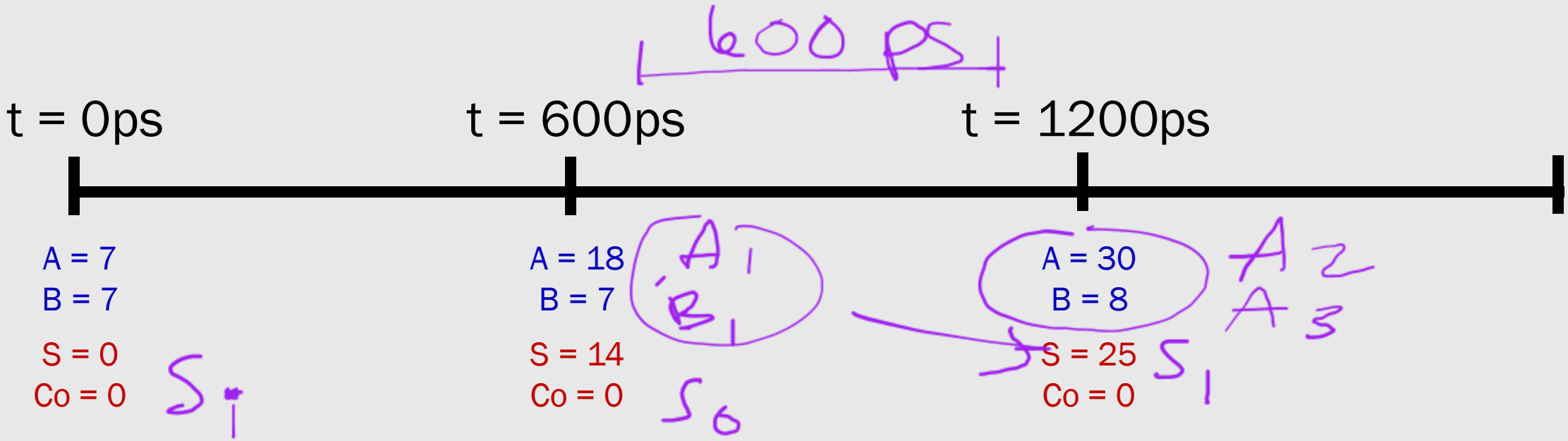
S = 14
Co = 0

t = 0ps

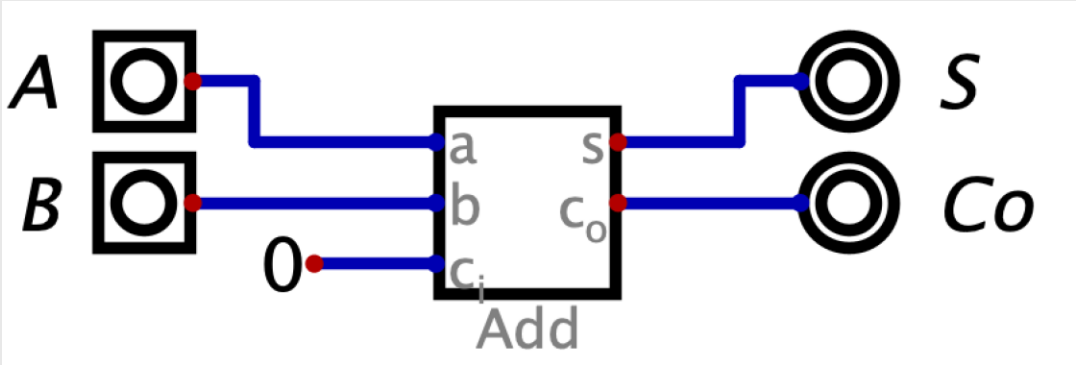


At time t = 1200ps. The adder will finish its computation, so S = 25 and Co = 0.

We are now ready to perform the next addition. We will set A = 30 and B = 8.



t = 0ps



At time t = 1800ps. The adder will finish its computation, so S = 38 and Co = 0.

We don't have any more additions to perform, so we'll set A and B to 0.

t = 0ps

A = 7
B = 7

S = 0
Co = 0

t = 600ps

A = 18
B = 7

S = 14
Co = 0

t = 1200ps

A = 30
B = 8

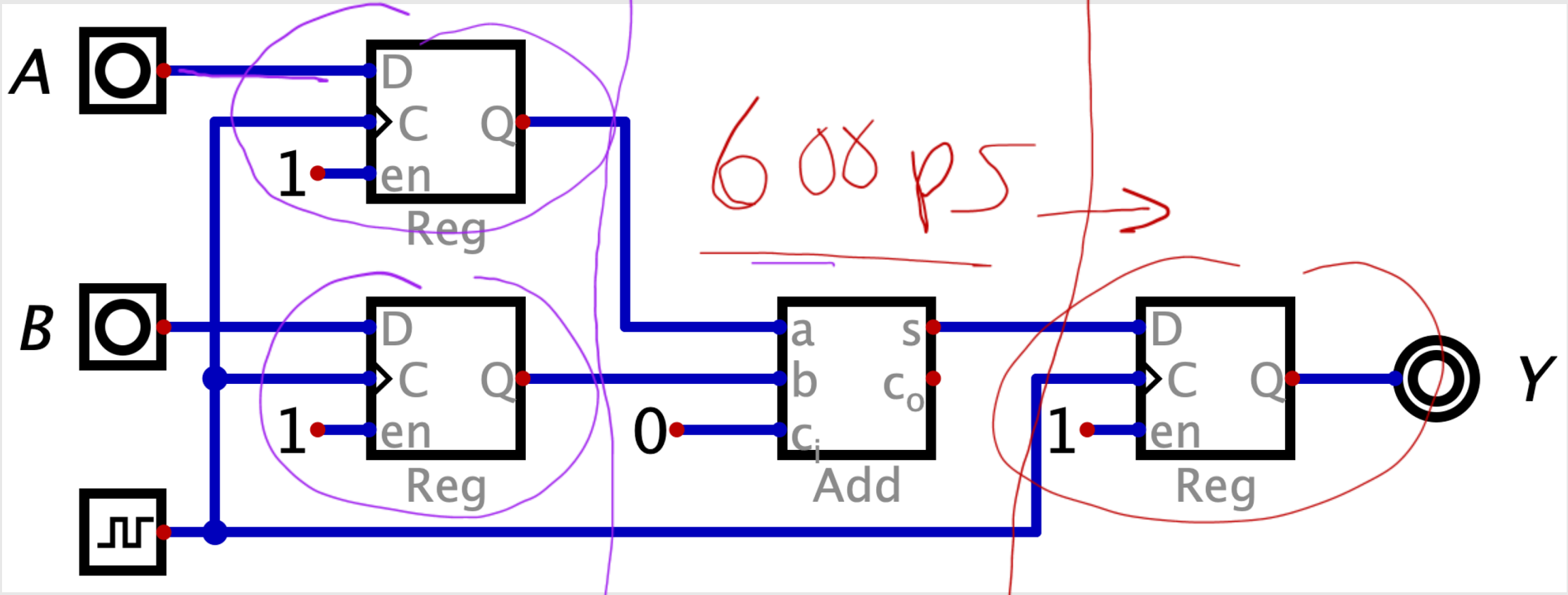
S = 25
Co = 0

t = 1800ps

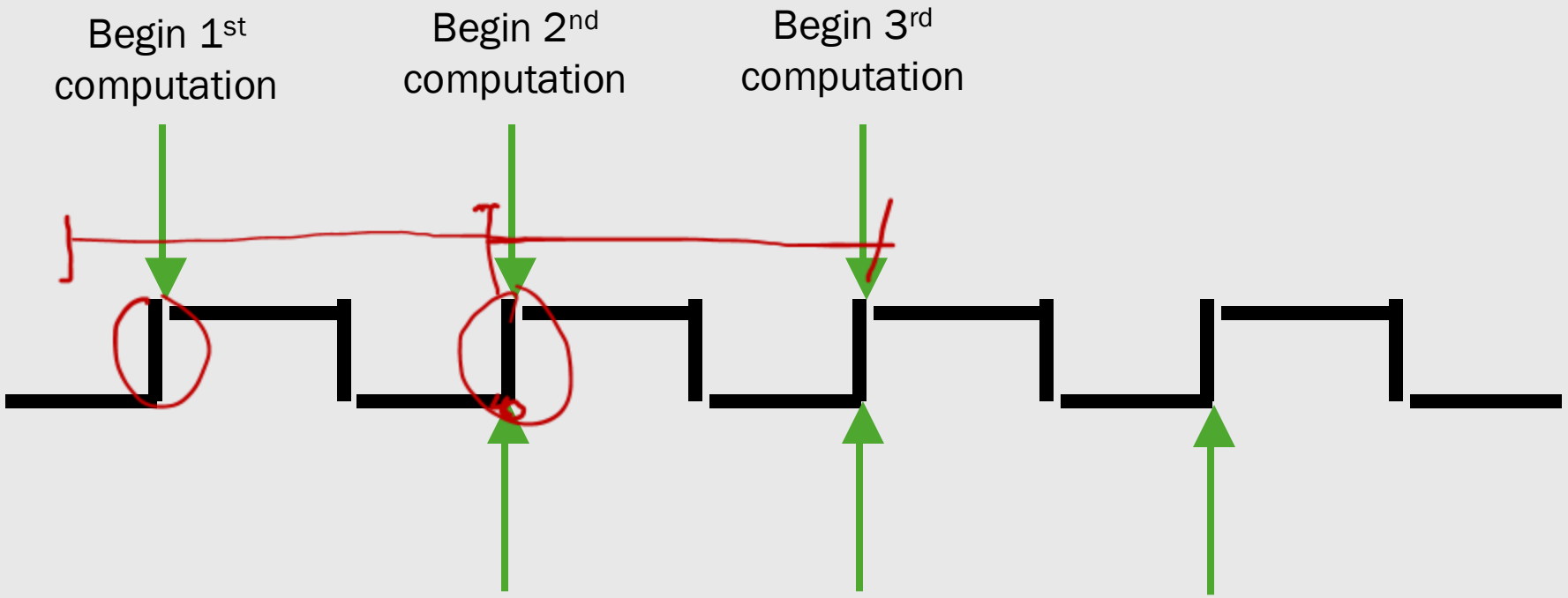
A = 0
B = 0

S = 38
Co = 0

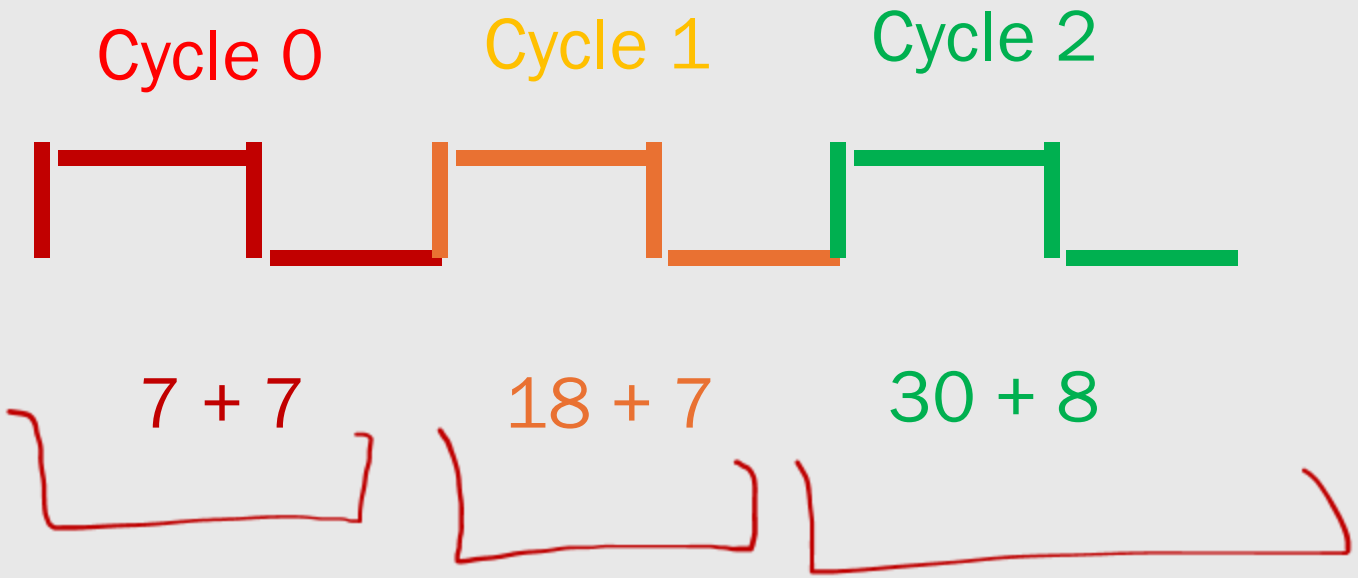
Using Registers to Set Inputs and Read Outputs at the Correct Times



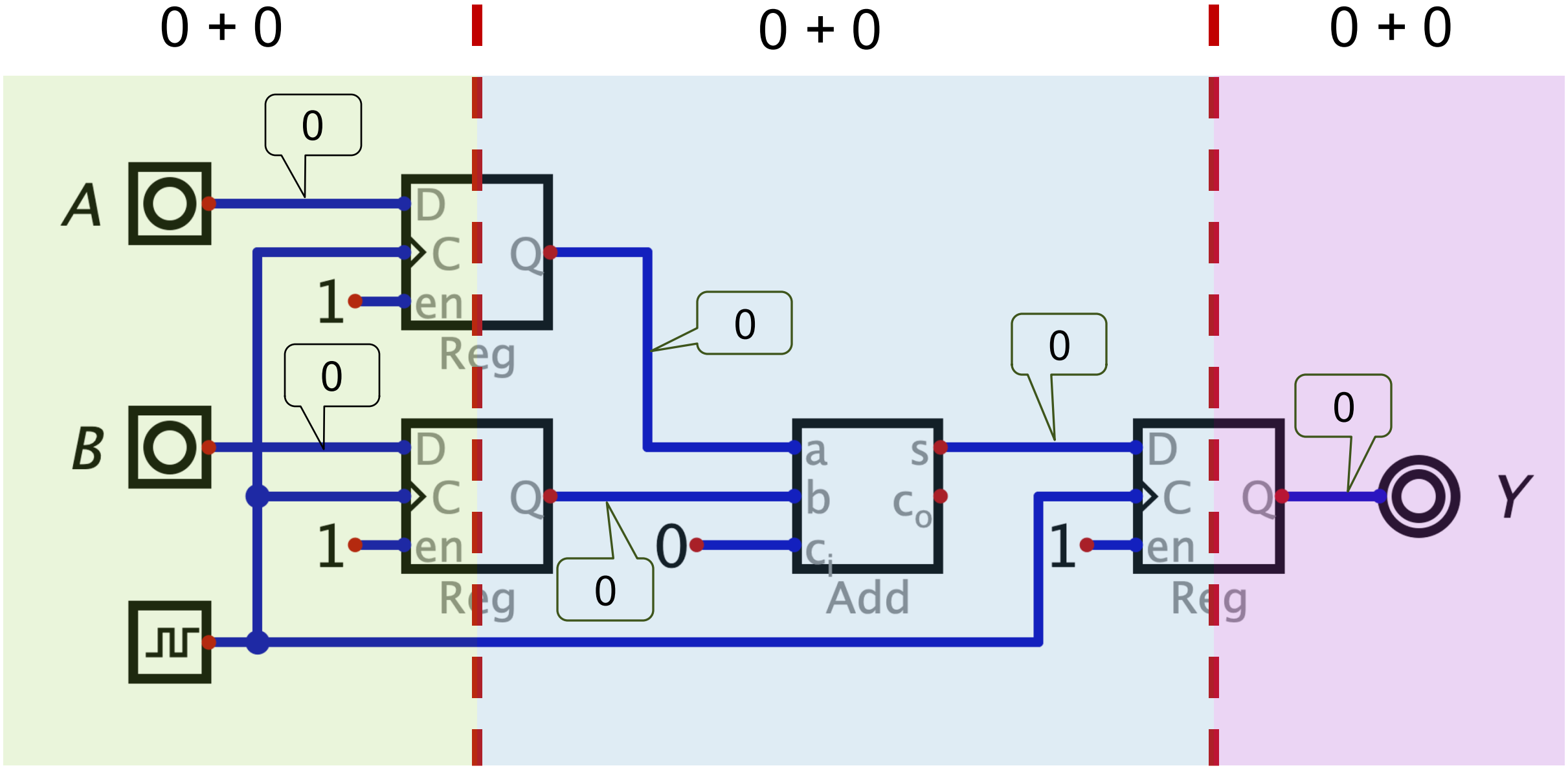
Timing


Begin 1st computation
Begin 2nd computation
Begin 3rd computation
1st computation should be completed by the next rising edge of the clock
2nd computation should be completed by the next rising edge of the clock
3rd computation should be completed by the next rising edge of the clock

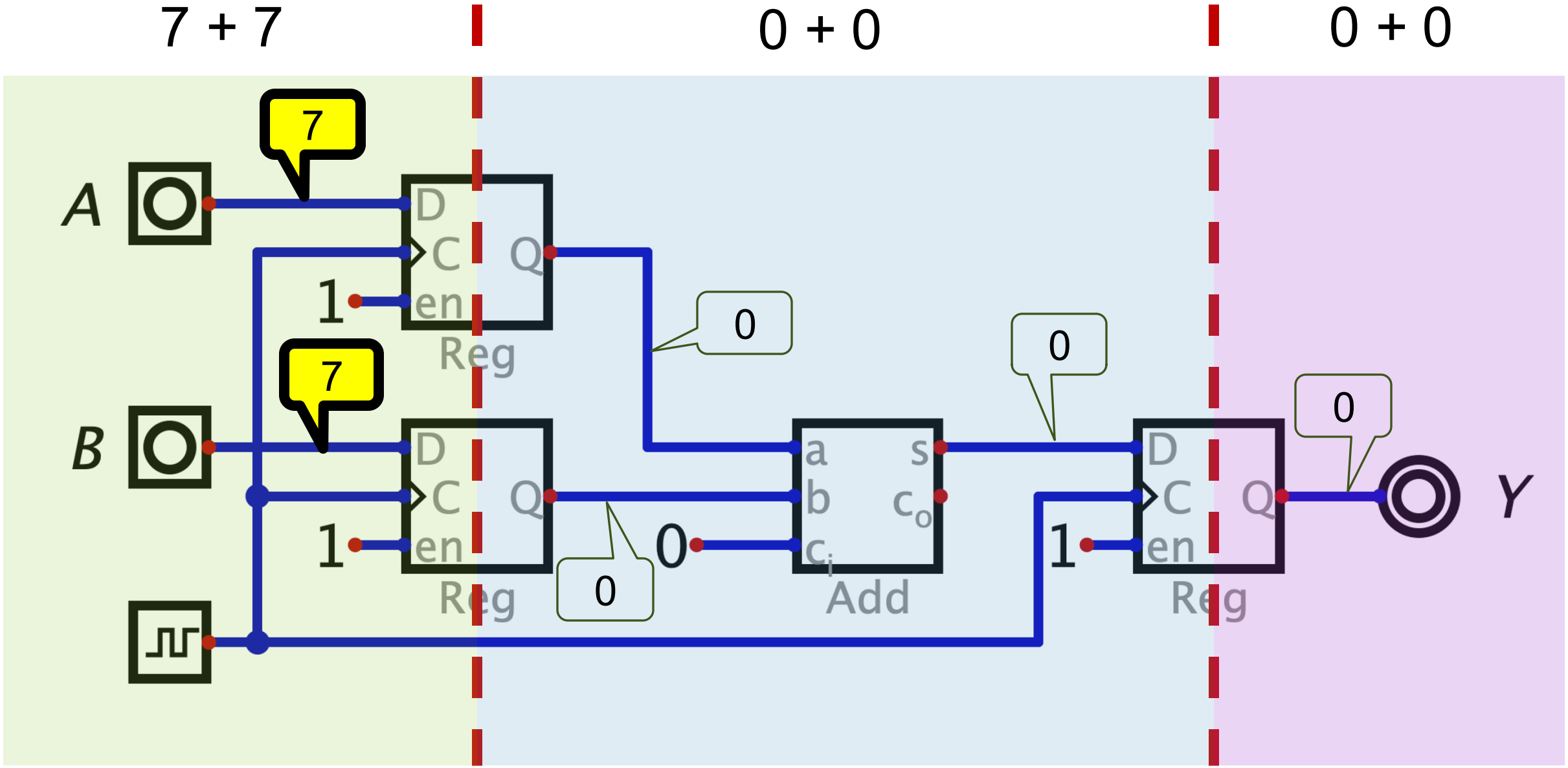
Timing



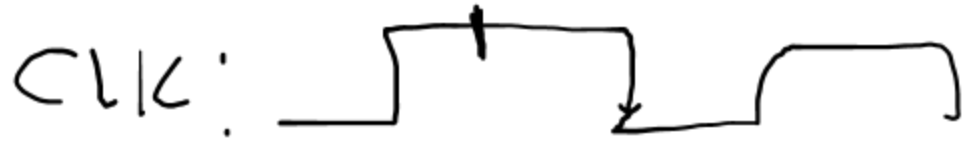
Starting State

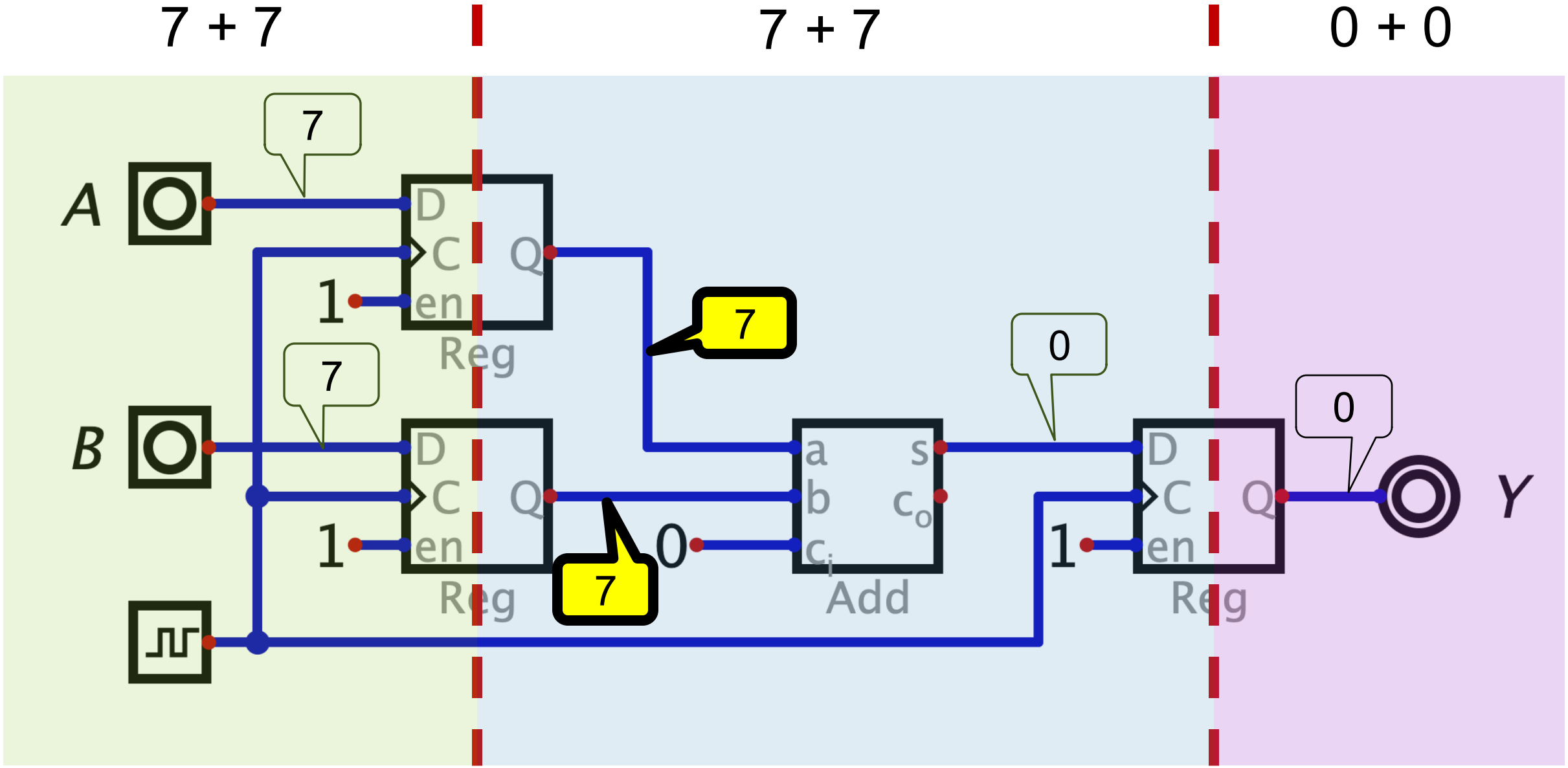


Cycle 0: On rising edge

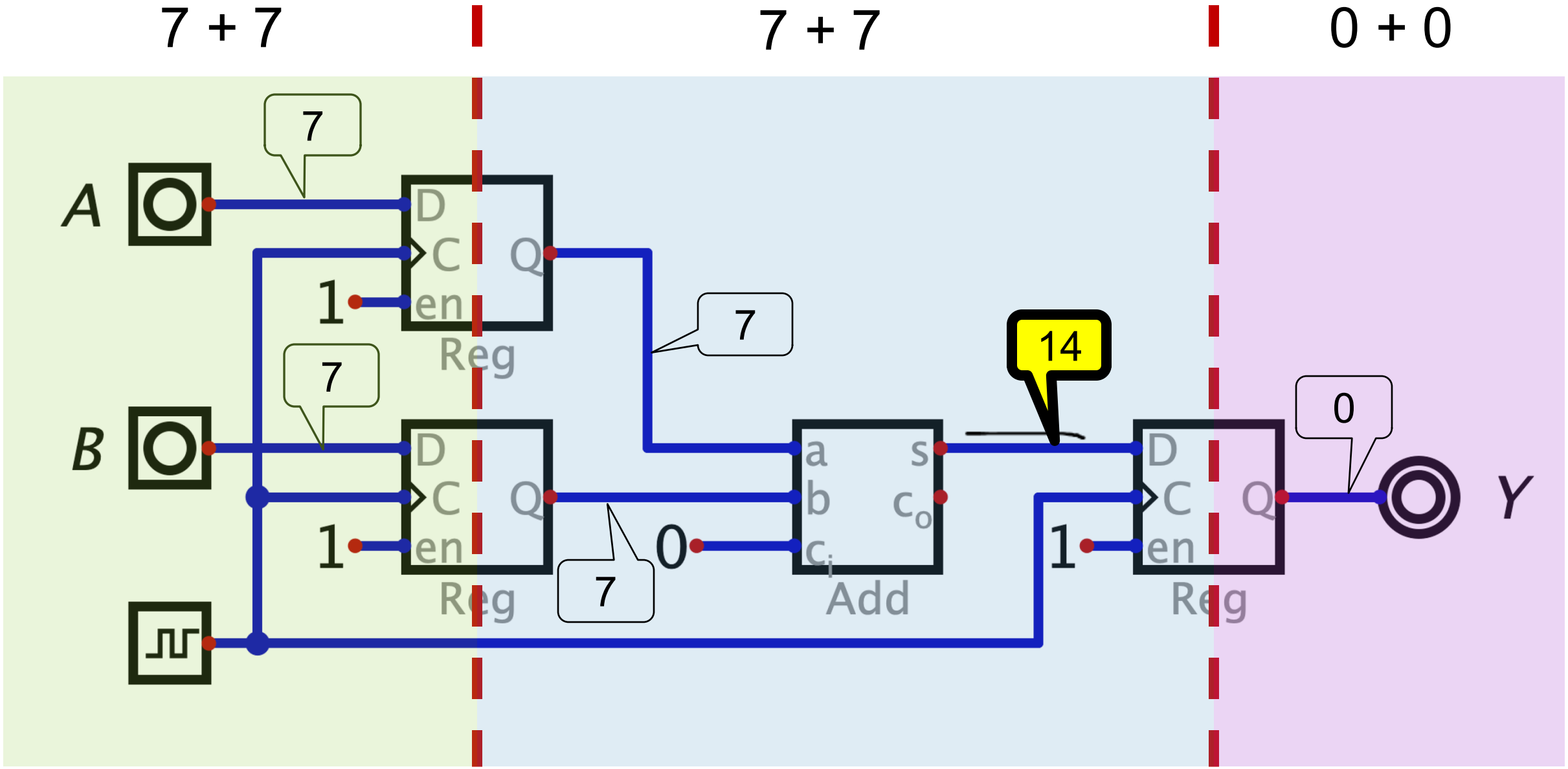


Cycle 0: After clk-to-q delay

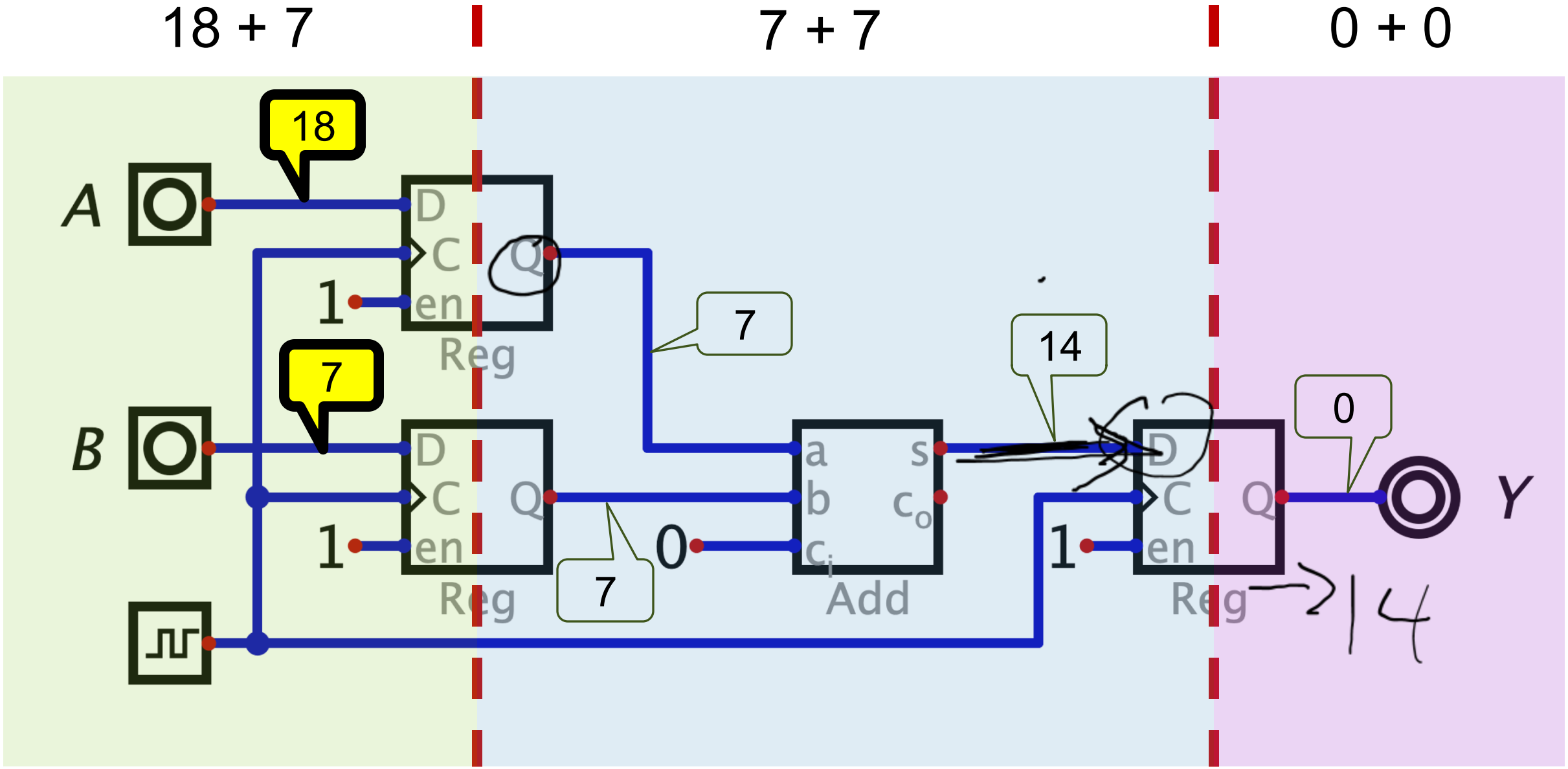




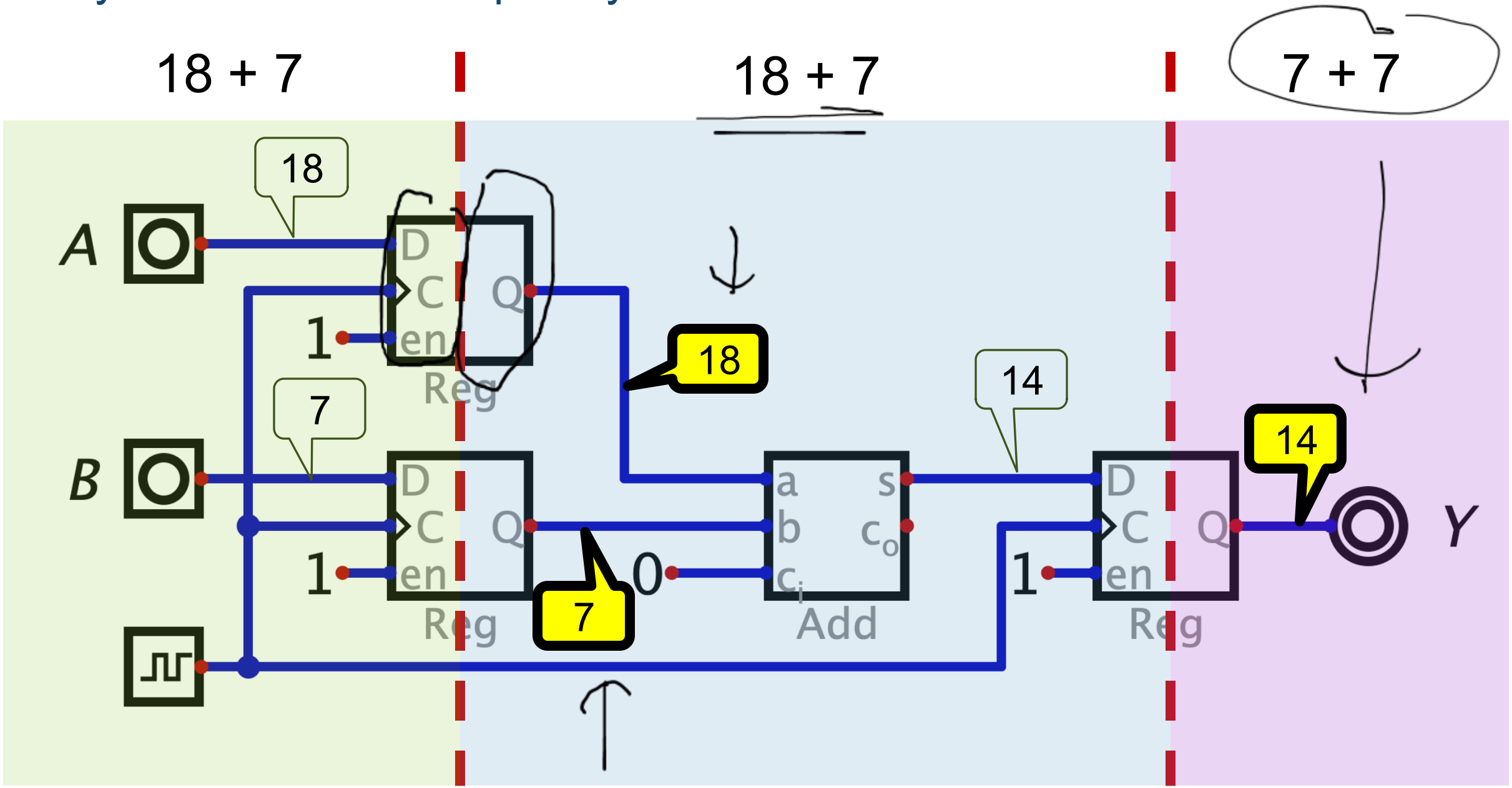
Cycle 0: After adder propagation delay



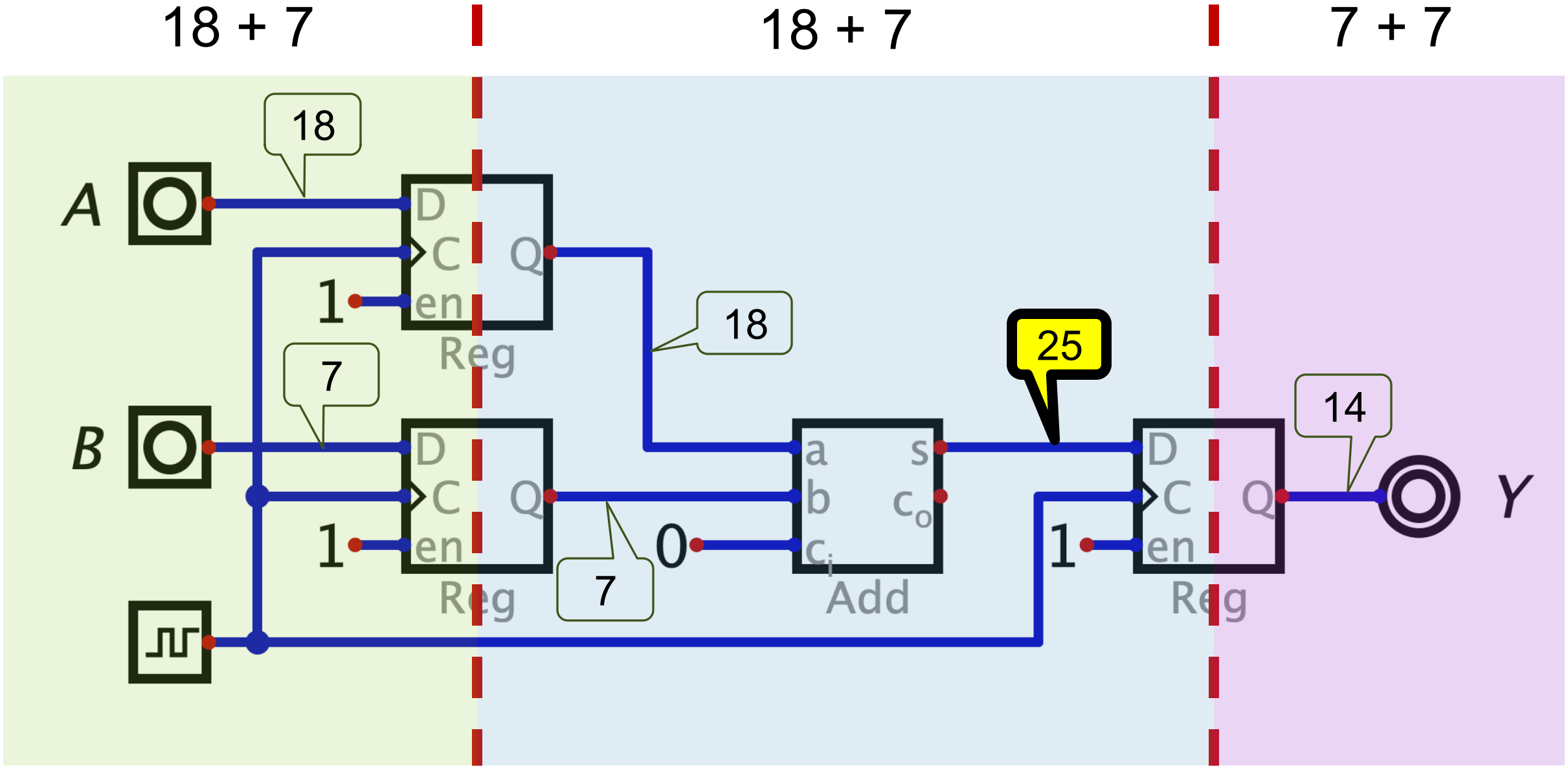
Cycle 1: On rising edge



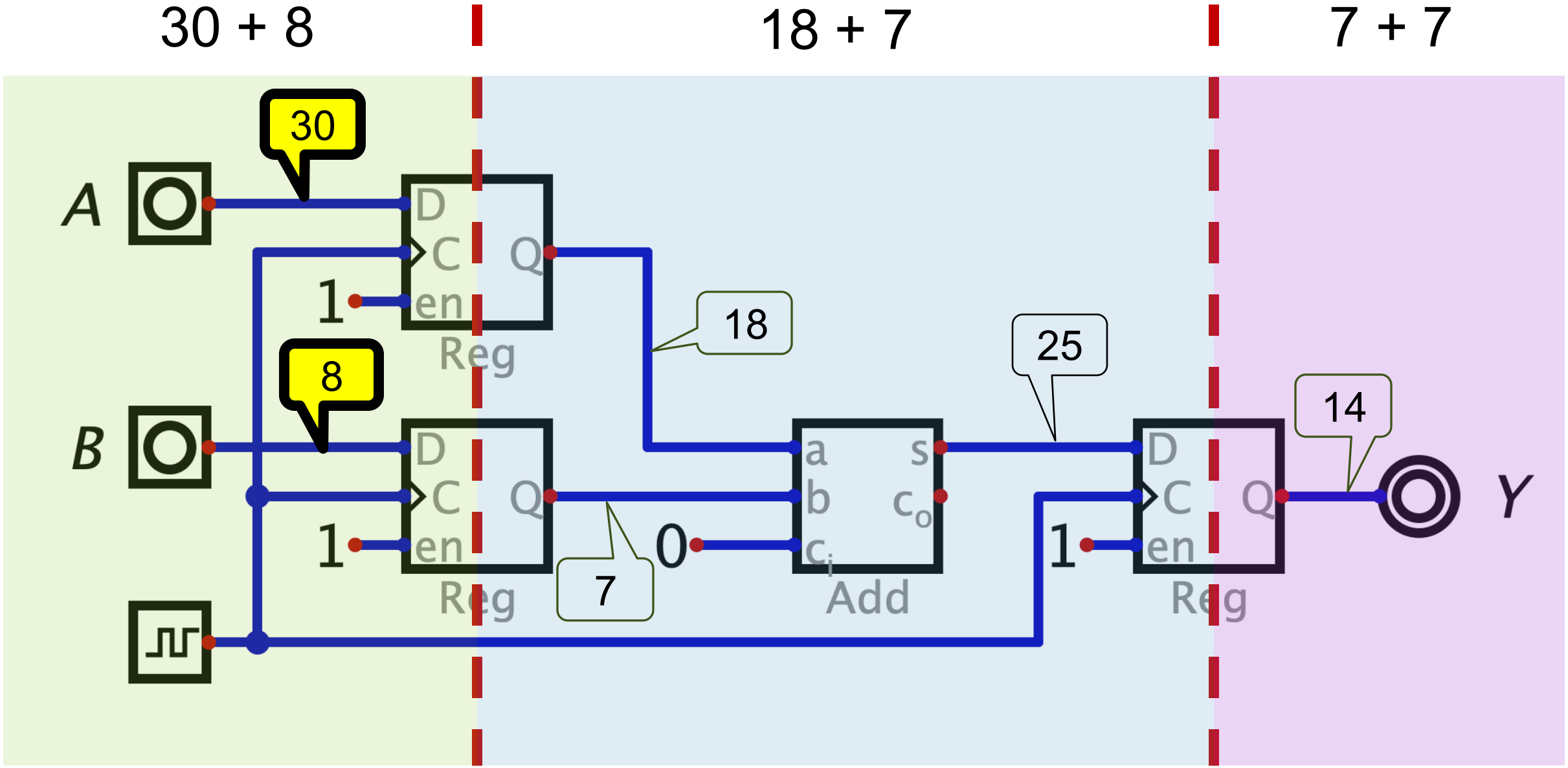
Cycle 1: After clk-to-q delay



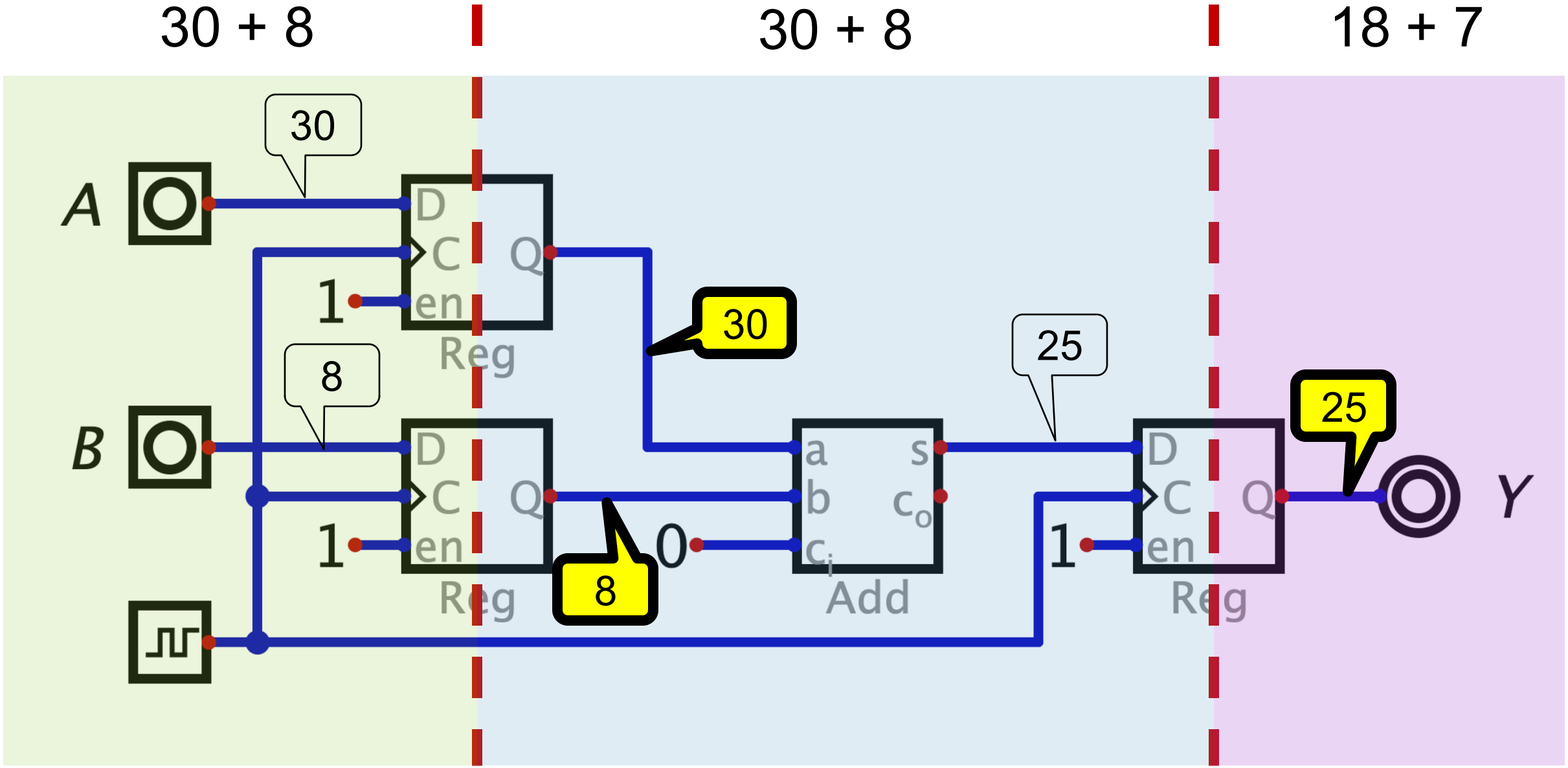
Cycle 1: After adder propagation delay



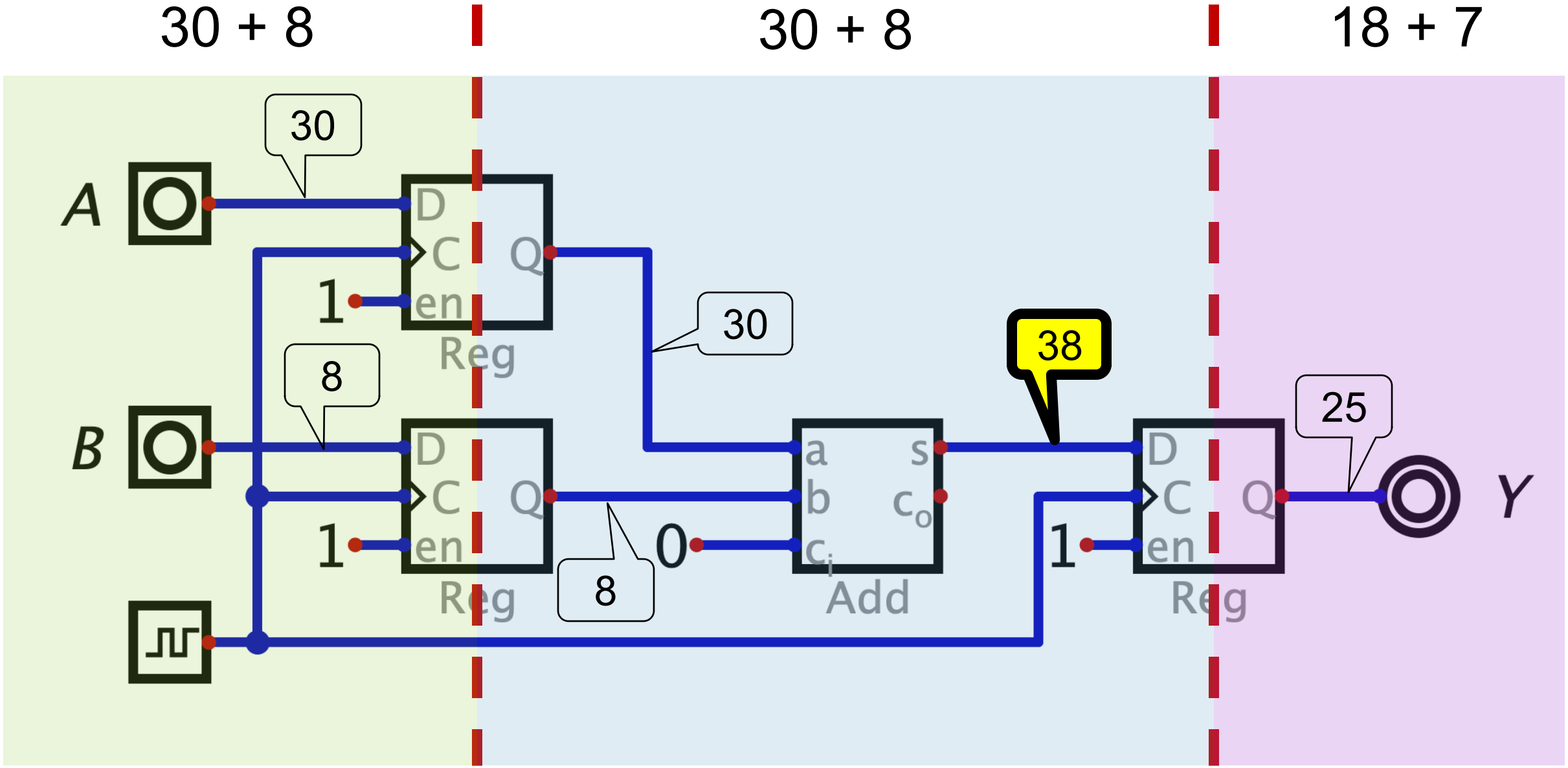
Cycle 2: On Rising Edge



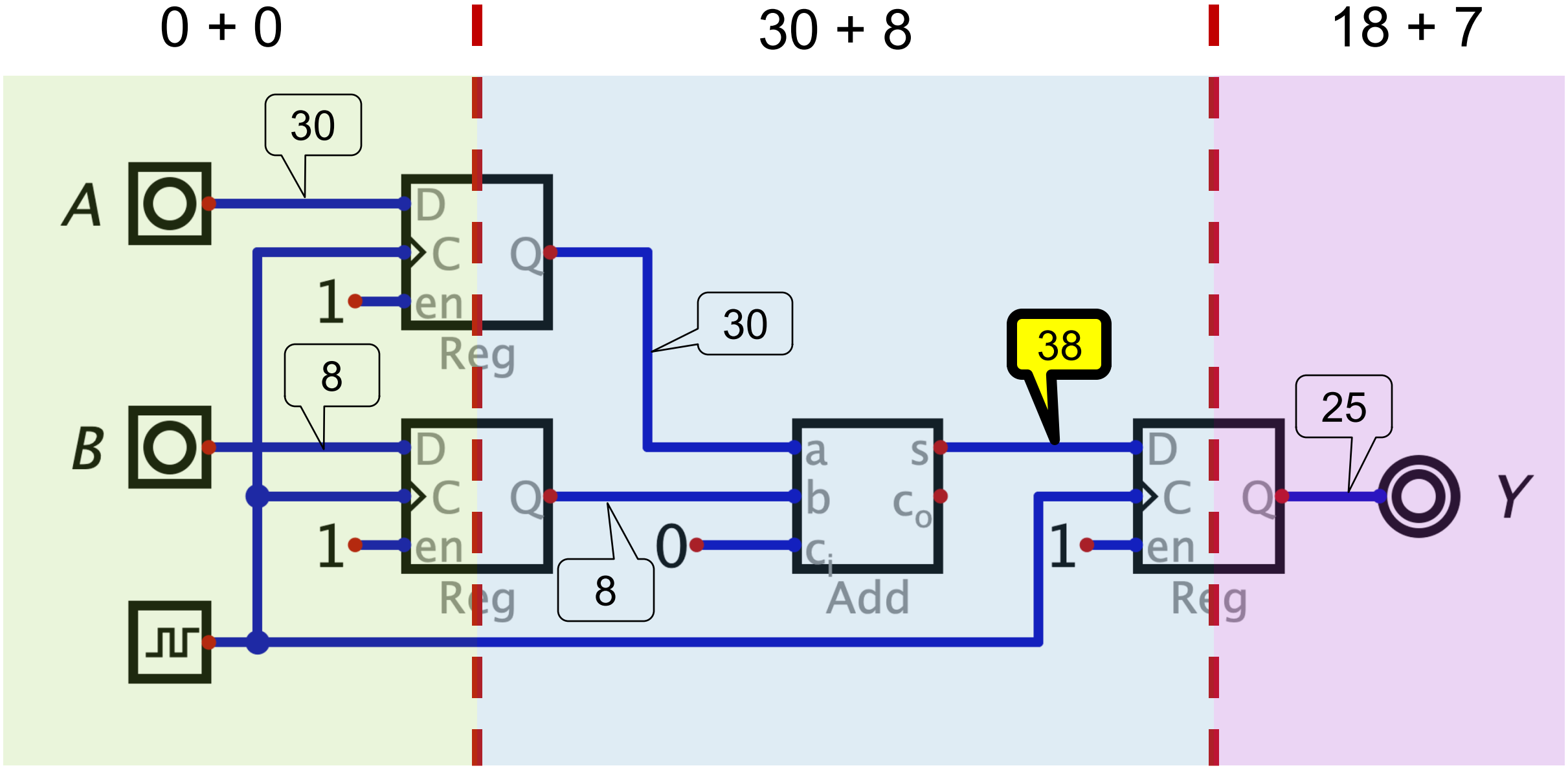
Cycle 2: After clk-to-q Delay



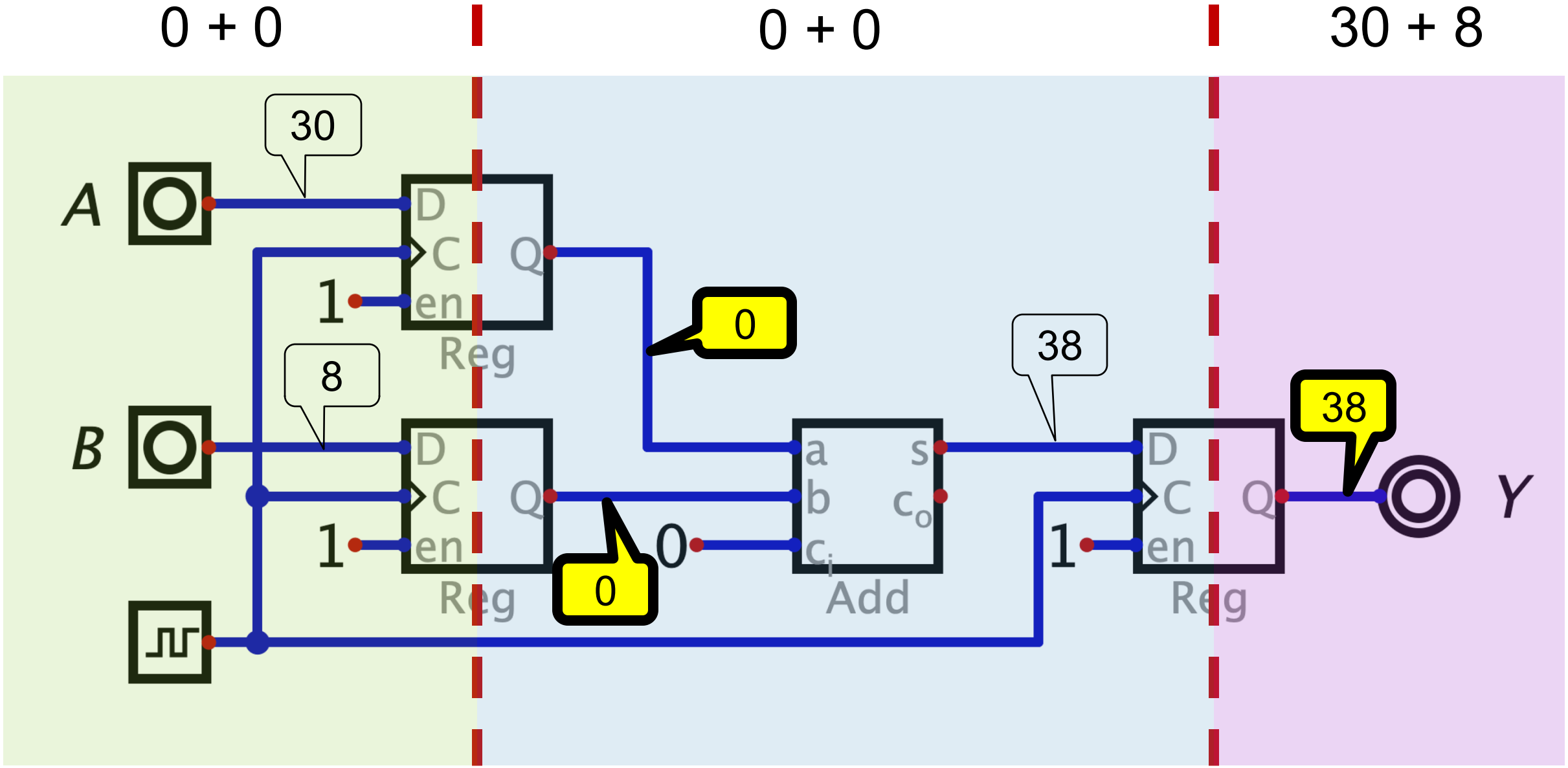
Cycle 2: After Adder Propagation Delay



Cycle 3: On Rising Edge



Cycle 3: After clk-to-q Delay



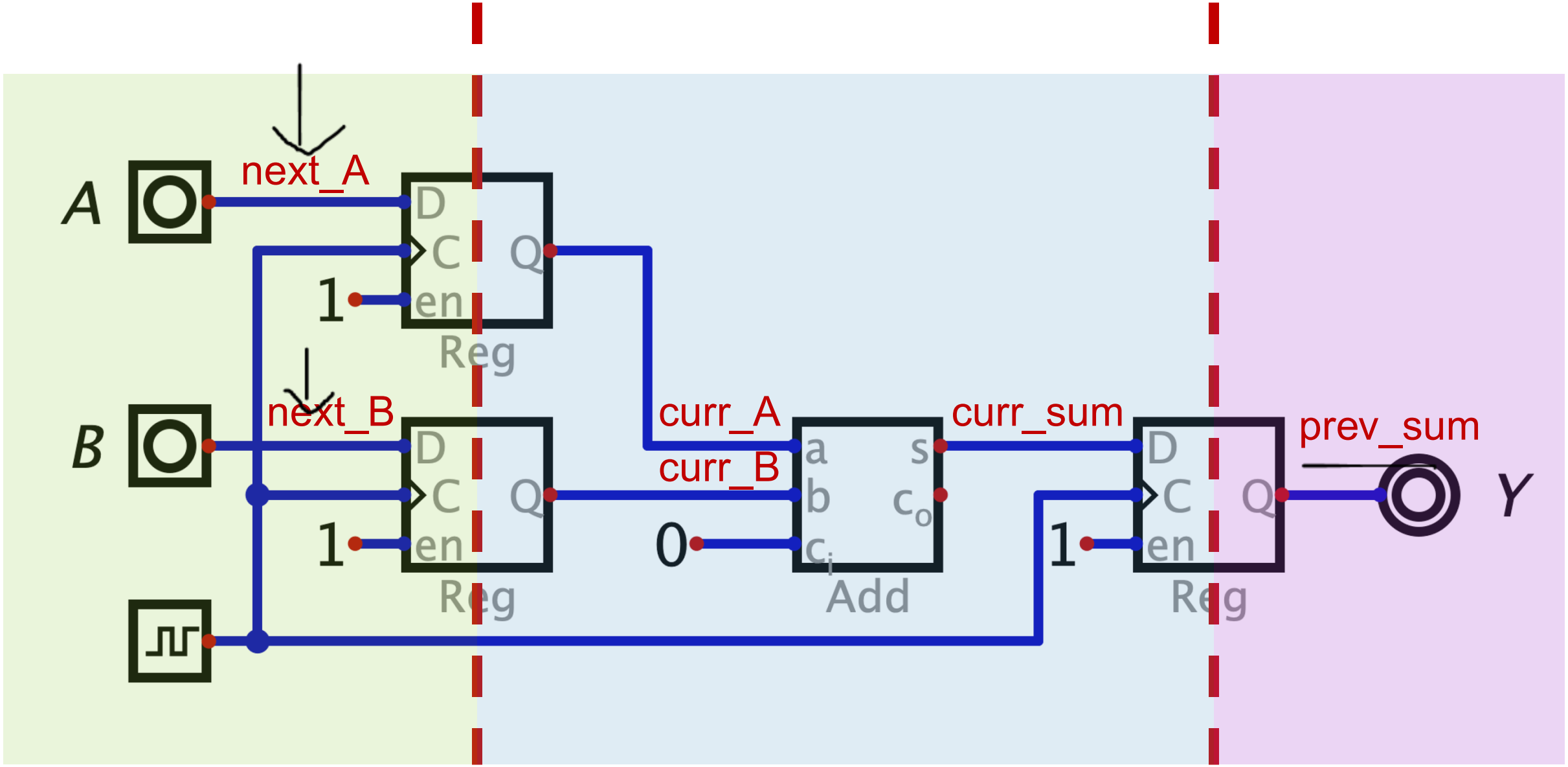
Delays

- Adder propagation delay = 600ps
- Register clk-to-q delay = 50ps
- These delays are innate properties of the components
- We can calculate these delays by finding the longest combinational delay
- We cannot change these values without changing the components

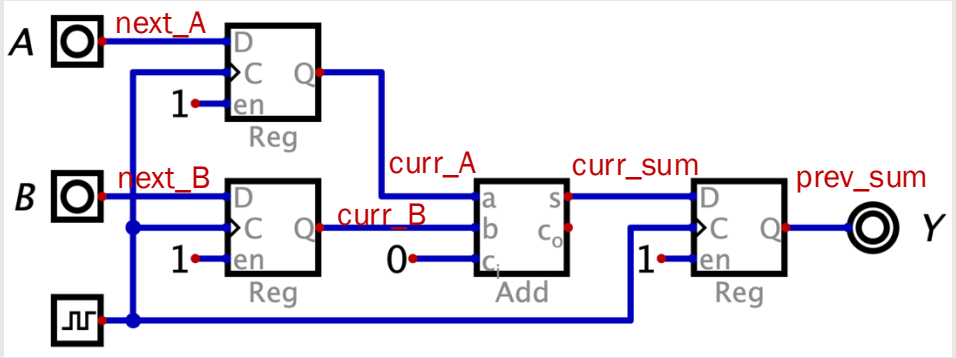
Clock Period

- We can adjust the clock period based on our needs
- There is a minimum clock period at which the circuit will work properly
- We can set our clock period to any value >= min clock period

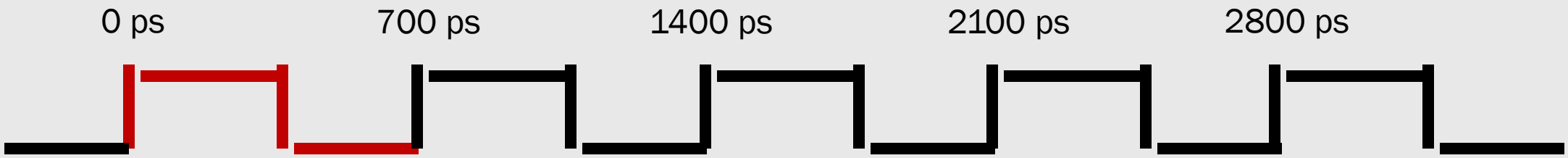
Ex1: Clock period = 700ps, clk-to-q delay = 50ps,
adder propagation delay = 600ps



Cycle 0: 7 + 7

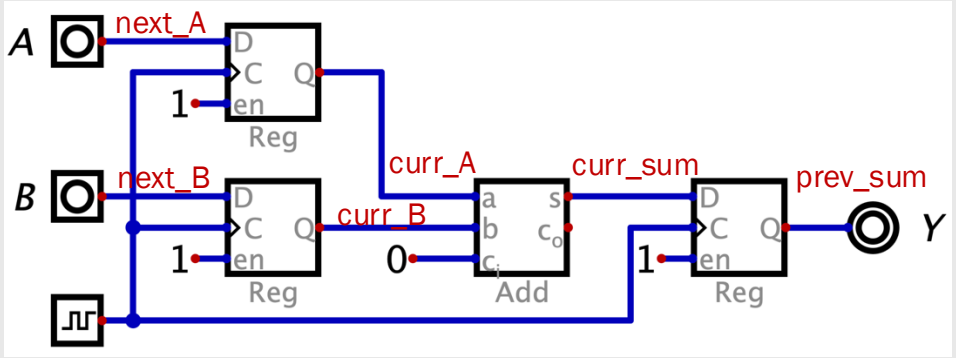


adder propagation delay = 600 ps
clk-to-q delay is 50ps
clock period = 700ps

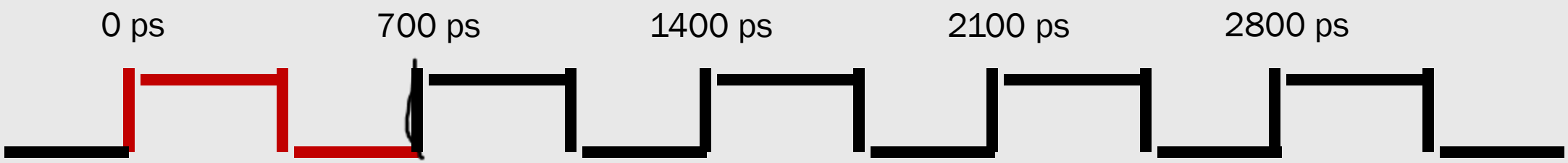


Signal	Starting State	t = 0 A = 7 B = 7	t = 50ps A = 7 B = 7	t = 650ps A = 7 B = 7	t = 700 ps A = 18 B = 7
next_A	0				
next_B	0				
curr_A	0				
curr_B	0				
curr_sum	0				
prev_sum	0				

Cycle 0: 7 + 7

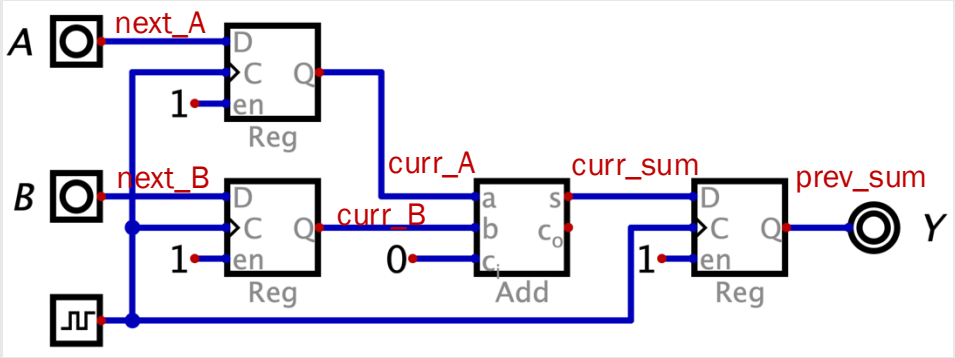


adder propagation delay = 600 ps
clk-to-q delay = 50ps
clock period = 700ps

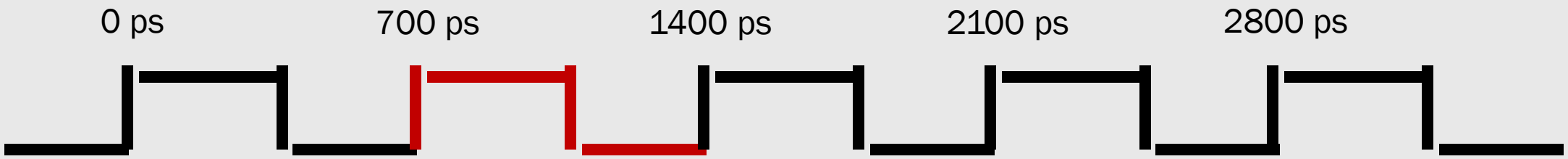


Signal	Starting State	t = 0 A = 7 B = 7	t = 50ps A = 7 B = 7	t = 650ps A = 7 B = 7	t = 700 ps A = 18 B = 7
next_A	0	7	7	7	18
next_B	0	7	7	7	7
curr_A	0	0	7	7	7
curr_B	0	0	7	7	7
curr_sum	0	0	0	14	14
prev_sum	0	0	0	0	0

Cycle 1: 18 + 7

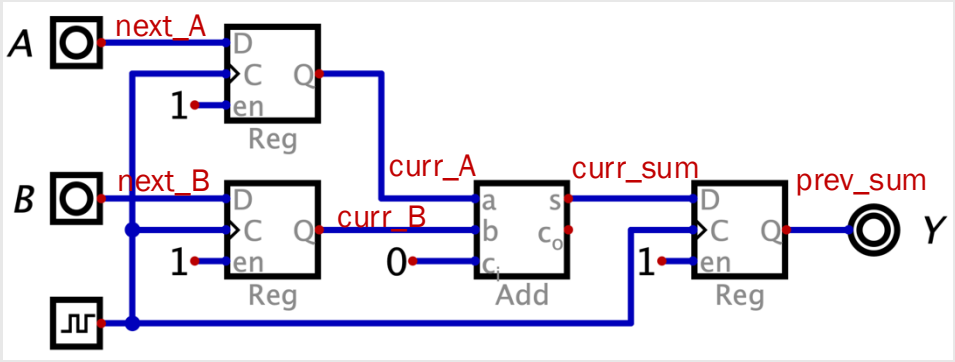


adder propagation delay = 600 ps
clk-to-q delay = 50ps
clock period = 700ps

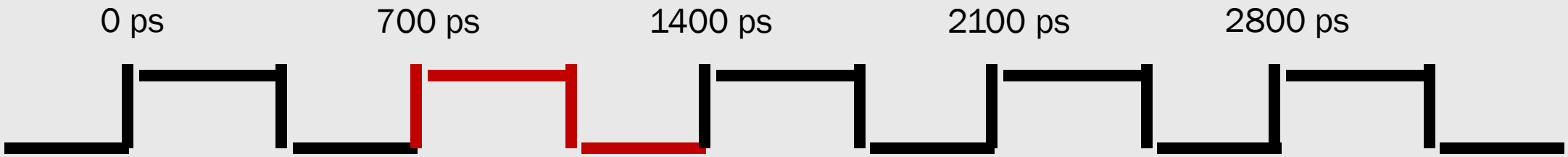


Signal	t = 700 ps A = 18 B = 7	t = 750 ps A = 18 B = 7	t = 1350 ps A = 18 B = 7	t = 1400 ps A = 30 B = 8
next_A	18			
next_B	7			
curr_A	7			
curr_B	7			
curr_sum	14			
prev_sum	0			

Cycle 1: 18 + 7

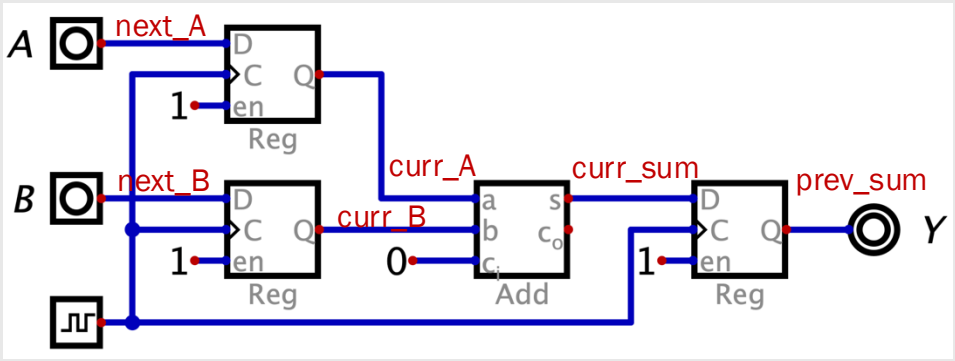


adder propagation delay = 600 ps
clk-to-q delay = 50ps
clock period = 700ps

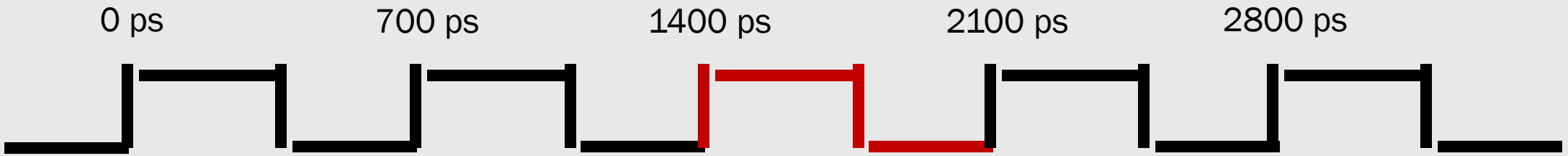


Signal	t = 700 ps A = 18 B = 7	t = 750 ps A = 18 B = 7	t = 1350 ps A = 18 B = 7	t = 1400 ps A = 30 B = 8
next_A	18	18	18	30
next_B	7	7	7	8
curr_A	7	18	18	18
curr_B	7	7	7	7
curr_sum	14	14	25	25
prev_sum	0	14	14	14

Cycle 2: 30 + 8

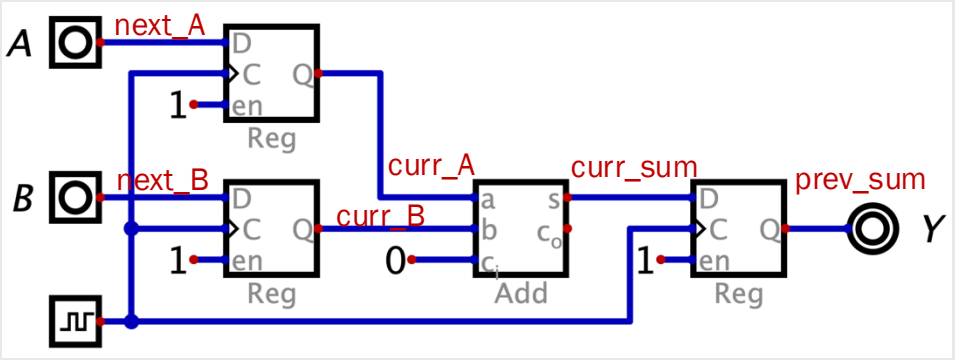


adder propagation delay = 600 ps
clk-to-q delay = 50ps
clock period = 700ps

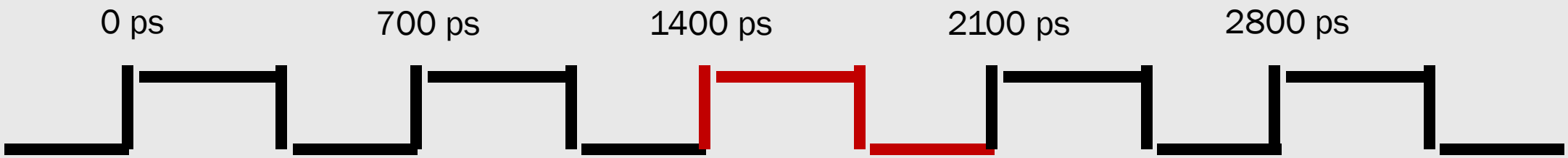


Signal	t = 1400 ps A = 30 B = 8	t = 1450 ps A = 30 B = 8	t = 2050 ps A = 30 B = 8	t = 2100 ps A = 15 B = 30
next_A	30			
next_B	8			
curr_A	18			
curr_B	7			
curr_sum	25			
prev_sum	14			

Cycle 2: 30 + 8

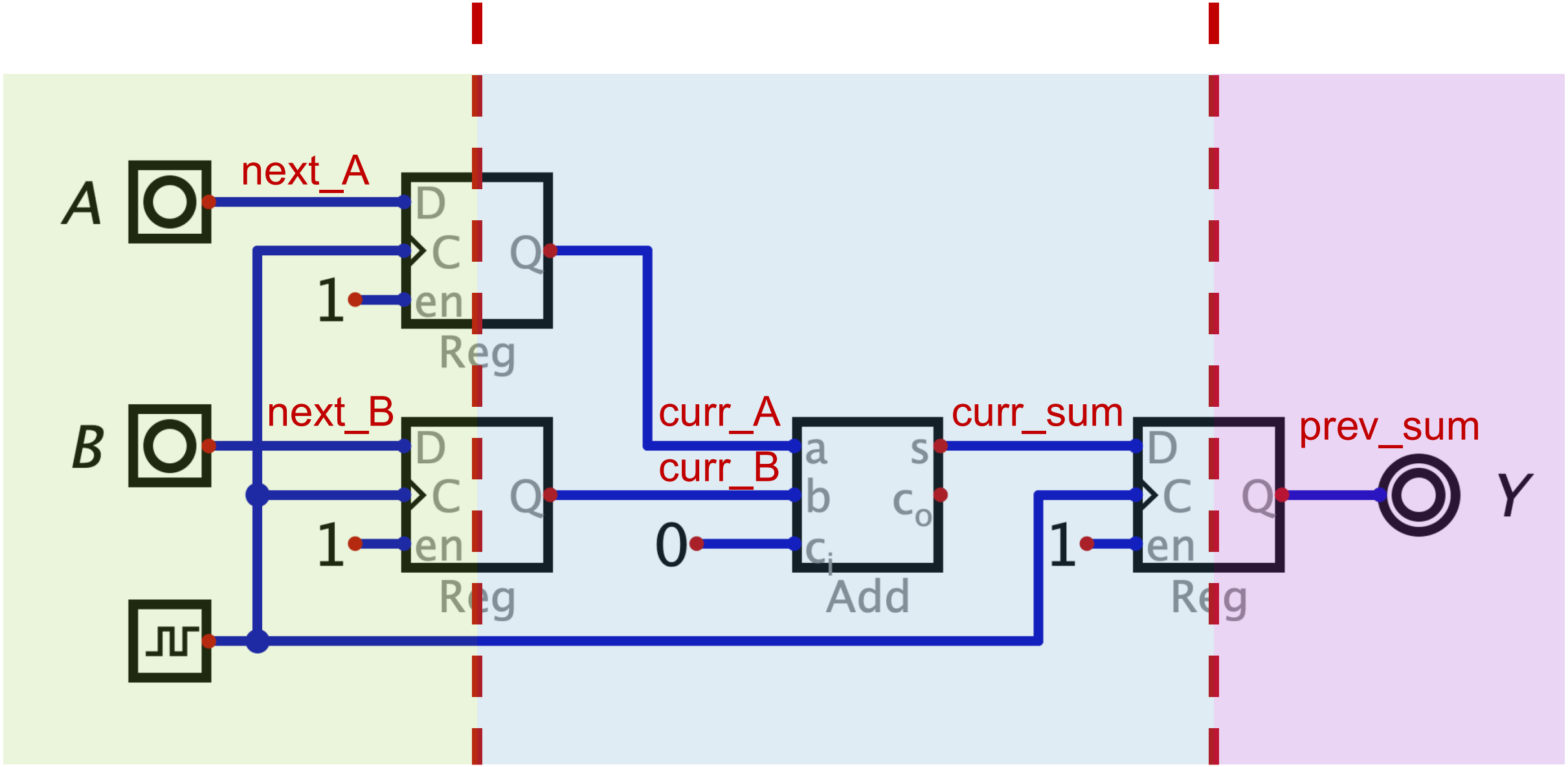


adder propagation delay = 600 ps
clk-to-q delay = 50ps
clock period = 700ps

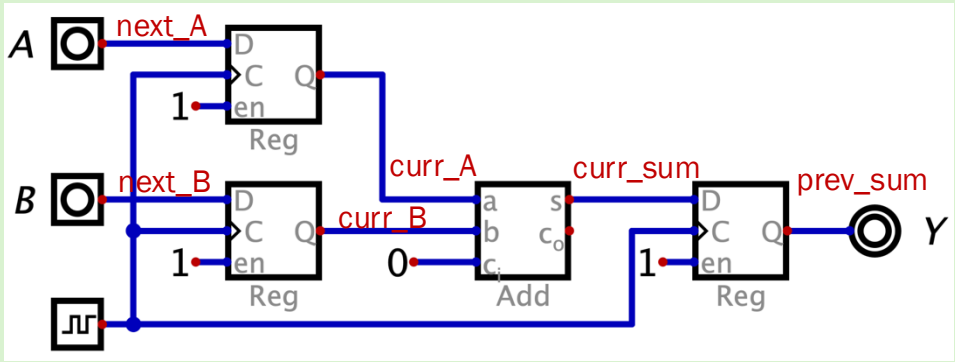


Signal	t = 1400 ps A = 30 B = 8	t = 1450 ps A = 30 B = 8	t = 2050 ps A = 30 B = 8	t = 2100 ps A = 15 B = 30
next_A	30	30	30	15
next_B	8	8	8	30
curr_A	18	30	30	30
curr_B	7	8	8	8
curr_sum	25	25	38	38
prev_sum	14	25	25	25

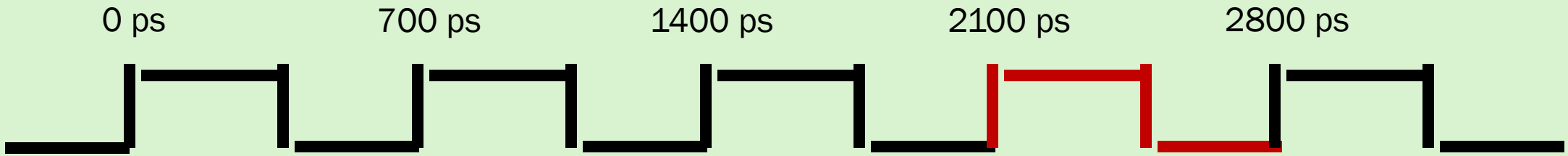
Ex1: Clock period = 700ps, clk-to-q delay = 50ps, adder propagation delay = 600ps



Cycle 3: 15 + 30



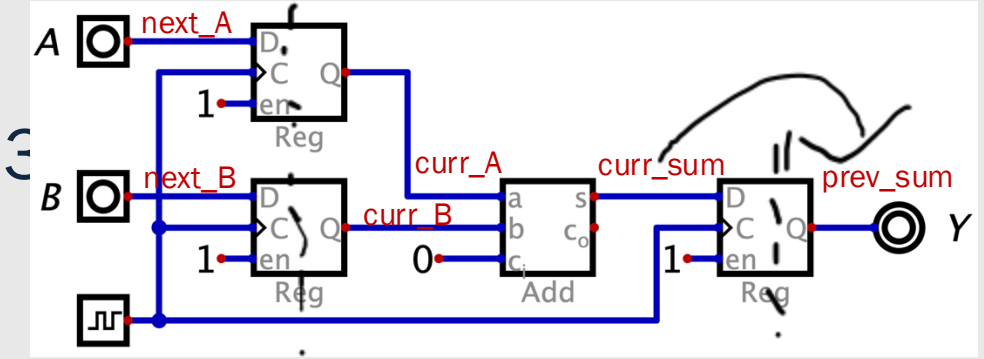
adder propagation delay = 600 ps
clk-to-q delay = 50ps
clock period = 700ps



Signal	t = 2100 ps A = 15 B = 30	t = 2150 ps A = 15 B = 30	t = 2750 ps A = 20 B = 30	t = 2800 ps A = 20 B = 30
next_A	15			
next_B	30			
curr_A	30			
curr_B	8			
curr_sum	38			
prev_sum	25			

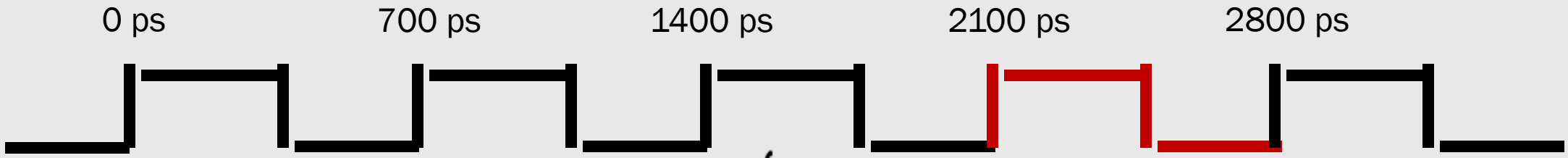
Not a typo! What will happen in this case?

Cycle 3: 15 + 3



adder propagation delay = 600 ps
clk-to-q delay = 50ps
clock period = 700ps

[2100, 2800)



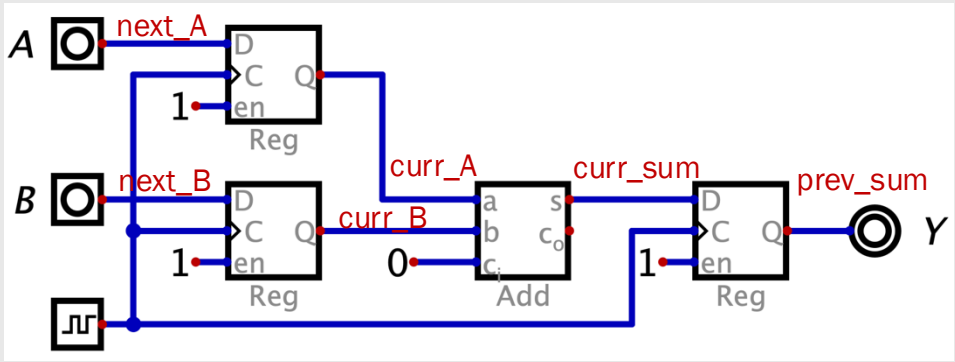
Signal	t = 2100 ps A = 15 B = 30	t = 2150 ps A = 15 B = 30	t = 2750 ps A = 20 B = 30	t = 2800 ps A = 20 B = 30
next_A	15	15	20	20
next_B	30	30	30	30
curr_A	30	15	15	15
curr_B	8	30	30	30
curr_sum	38	38	45	45
prev_sum	25	38	38	38

Not a typo! What will happen in this case?

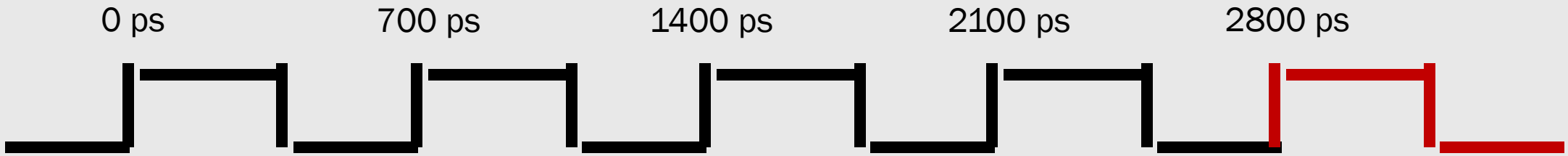
Input A updating at t = 2750ps

- Our addition will work the same as before!
- The register prevents the value of A from moving forward until the rising edge of the clock
- We can update the inputs anytime from t = 2150ps to t = 2800ps
  - *(aka until the next rising edge)*
- The value will simply just stay at the input of the register until the rising edge of the clock

Cycle 4: 20 + 30

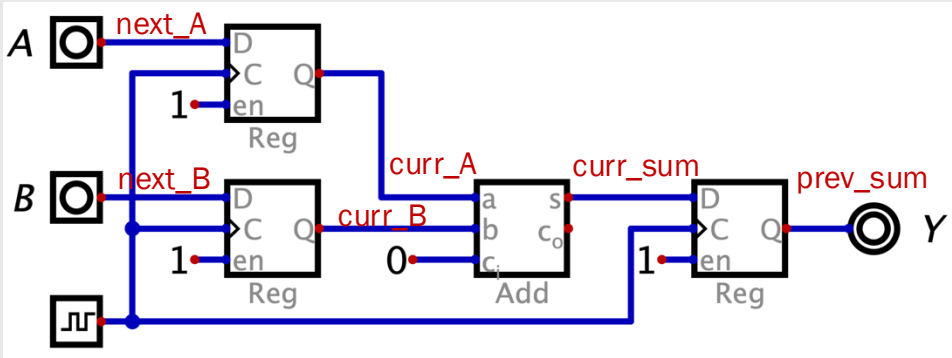


adder propagation delay = 600 ps
clk-to-q delay = 50ps
clock period = 700ps

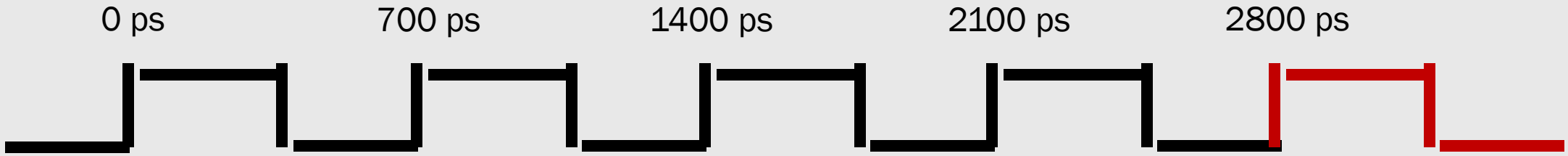


Signal	t = 2800 ps A = 20 B = 30	t = 2850 ps A = 20 B = 30	t = 3450 ps A = 20 B = 30	t = 3500 ps A = 0 B = 0
next_A	20			
next_B	30			
curr_A	15			
curr_B	30			
curr_sum	45			
prev_sum	38			

Cycle 4: 20 + 30

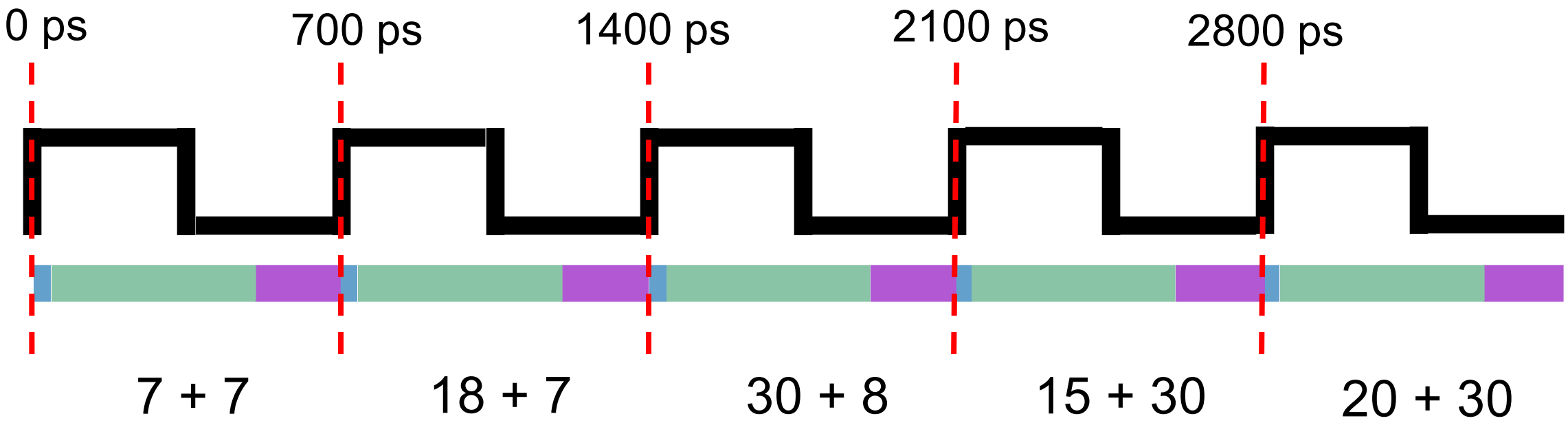


adder propagation delay = 600 ps
clk-to-q delay = 50ps
clock period = 700ps



Signal	t = 2800 ps A = 20 B = 30	t = 2850 ps A = 20 B = 30	t = 3450 ps A = 20 B = 30	t = 3500 ps A = 0 B = 0
next_A	20	20	20	0
next_B	30	30	30	0
curr_A	15	20	20	20
curr_B	30	30	30	30
curr_sum	45	45	50	50
prev_sum	38	45	45	45

Clock Period = 700ps



clk-to-q delay = 50ps

adder propagation delay = 600ps

idle time = 50ps