11/2/26

COMP 311: *Computer Organization!*

Lecture 20: Intro to Pipelining



The University
of North Carolina
at Chapel Hill

https://tinyurl.com/comp311-fa26

Logistics Update

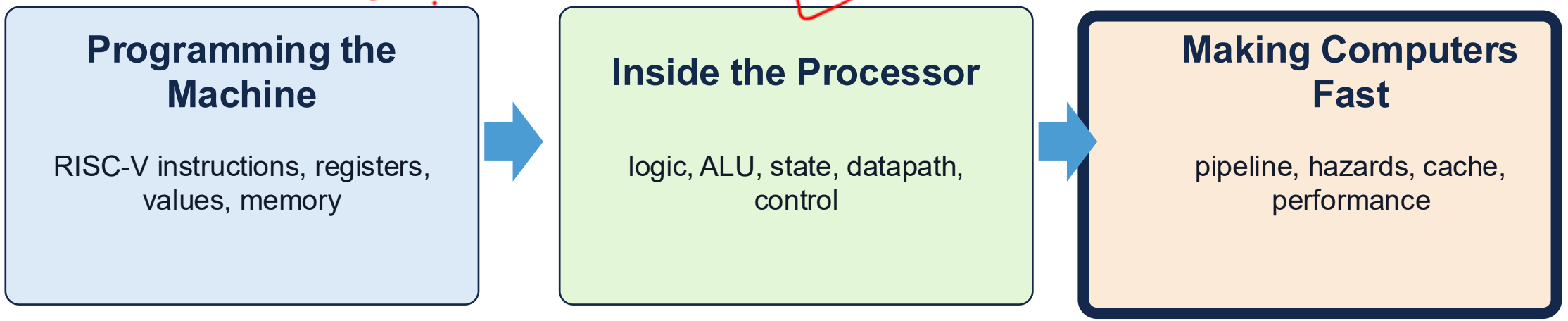
- Lab 3 is out
  - Due 11/6
- WA4
  - Due today!
- Quiz Corrections!

Quiz Topics



- RISC-V ALU and register-file
- D flip-flops and clocked circuit behavior
- Waveform tracing and propagation delay
- Timing analysis: critical paths, minimum clock period, and maximum clock rate
- Performance metrics: execution time, clock cycles, clock rate, and CPI

Where are we?



- **Today:** How can we organize a circuit to complete more work in less time?
- **Big idea:** Pipelining overlaps operations by dividing work into stages. It can improve **throughput** without reducing the **latency** of one operation.

Plan for Class Today

- 3:35-4:00: Computer Performance Terms
- 4:00-4:15: Timing Analysis with Multiple Computations
- 4:15-4:20: BREAK
- 4:20-4:50: Practice

Guiding Question ☺

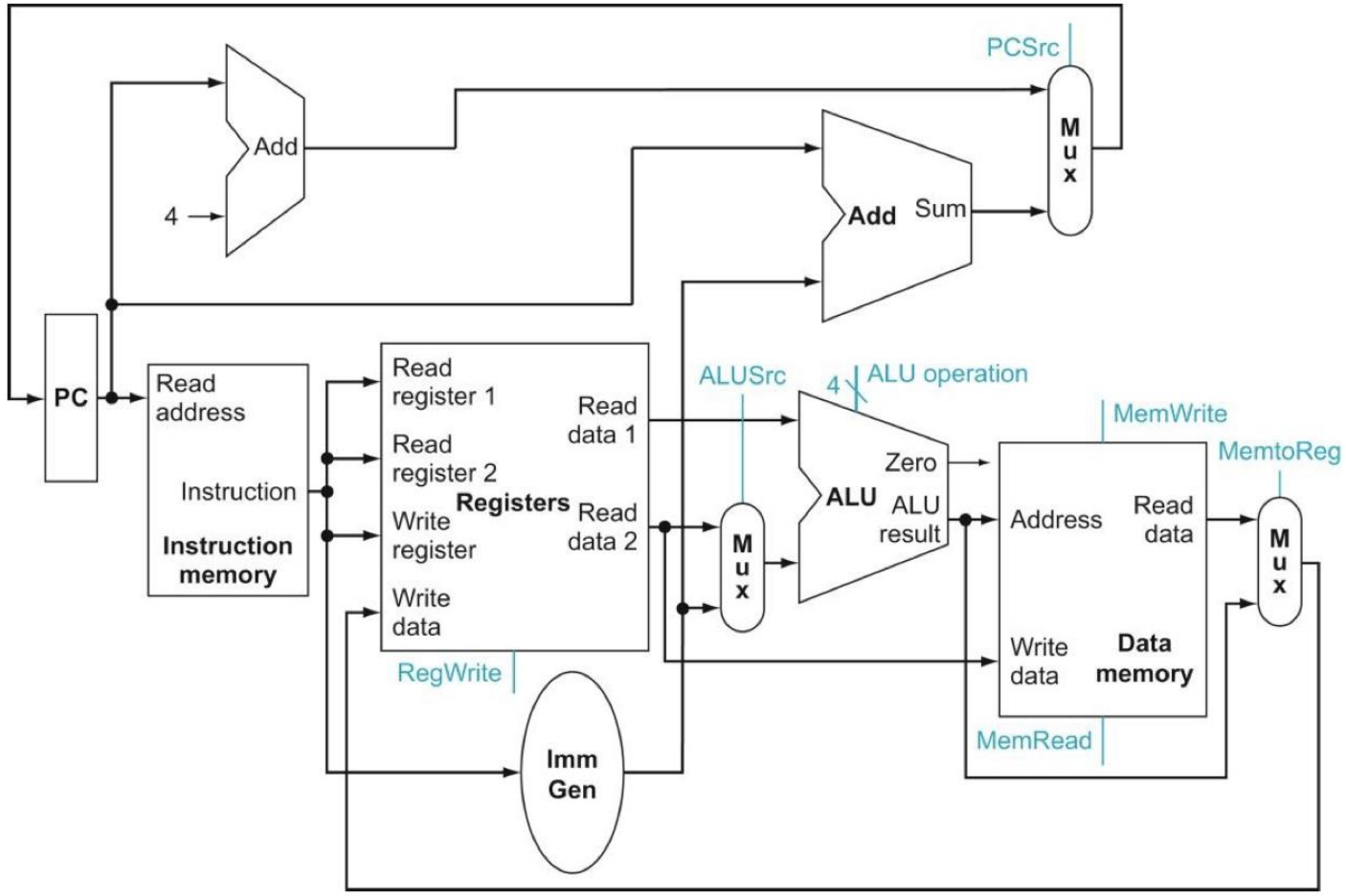
How does hardware remember information, and how can we predict when its outputs change?



More Timing Analysis

Where are we headed?

Figure 4.11 The simple datapath for the core RISC-V architecture

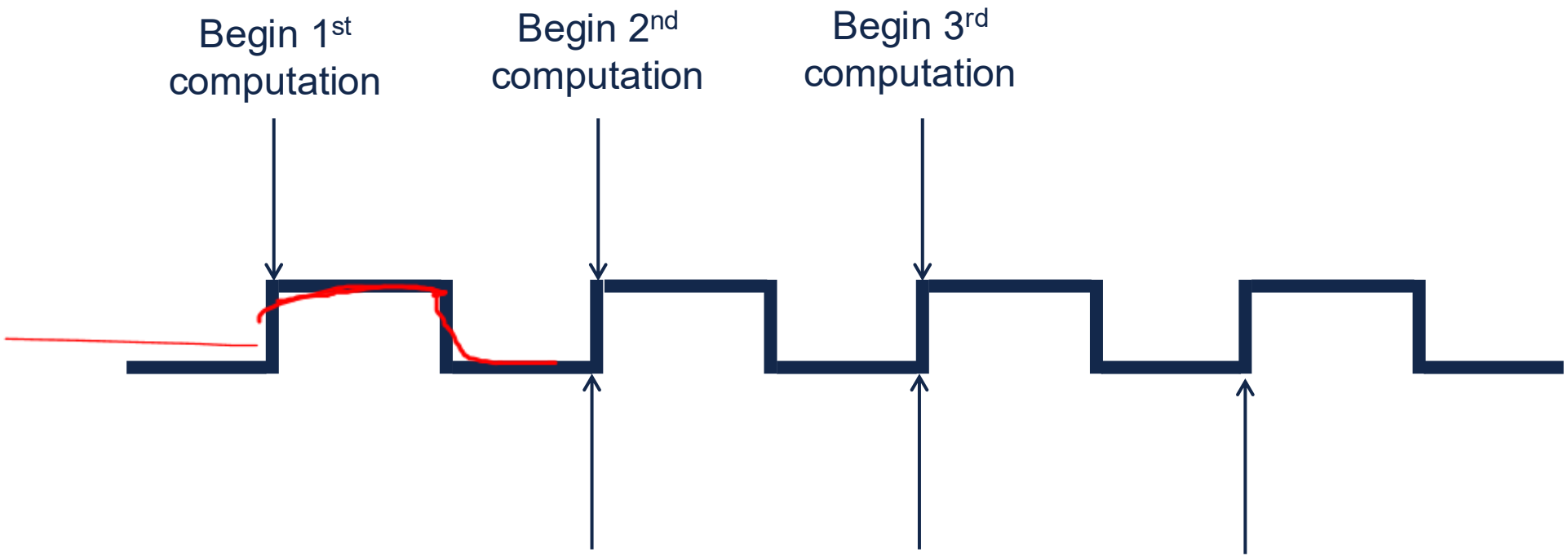


Our ultimate goal: a full CPU

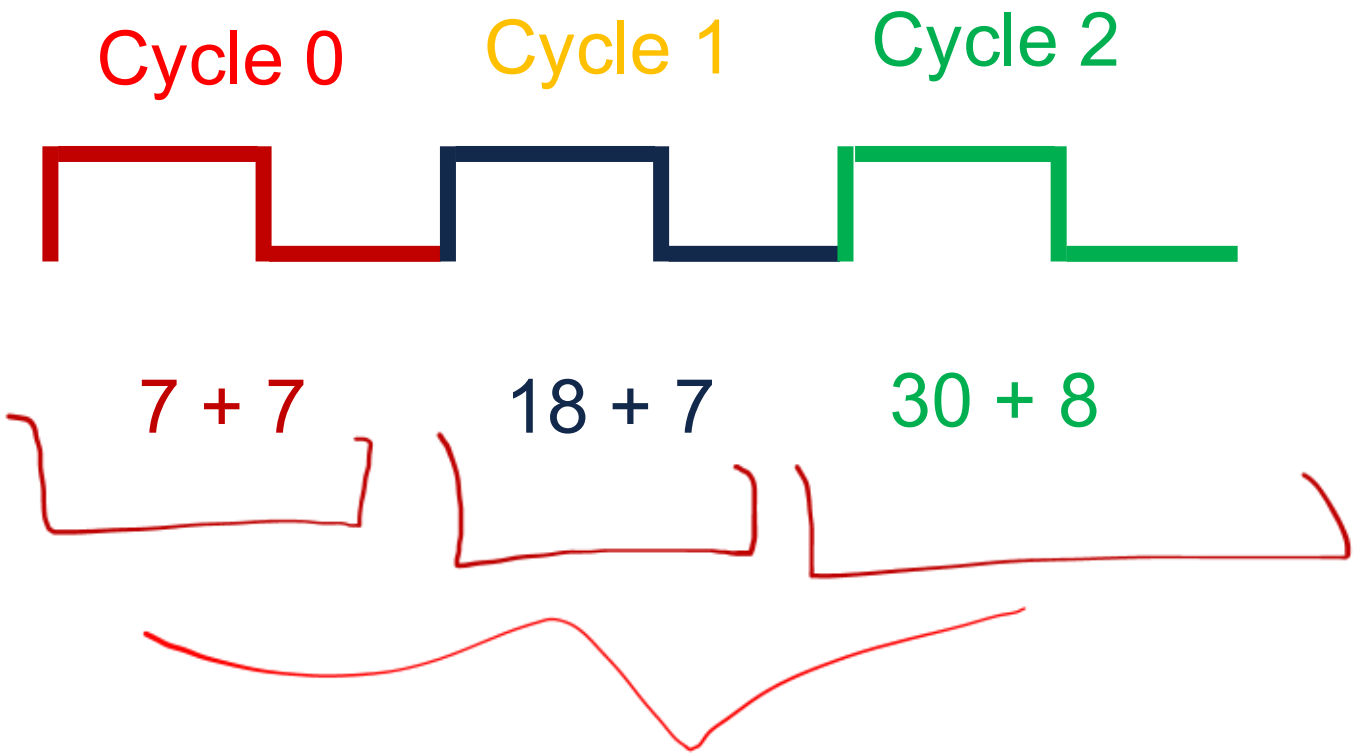


- Our goal: understand how fast we can safely clock a CPU
- That depends on how long signals take to travel through its circuits. We started small. Last time:
  - how individual gate delays determine when a circuit's output is ready
- Today: how registers and computation work together:
  - When inputs change, when does the circuit compute the new result, and on which clock edge is that result stored?

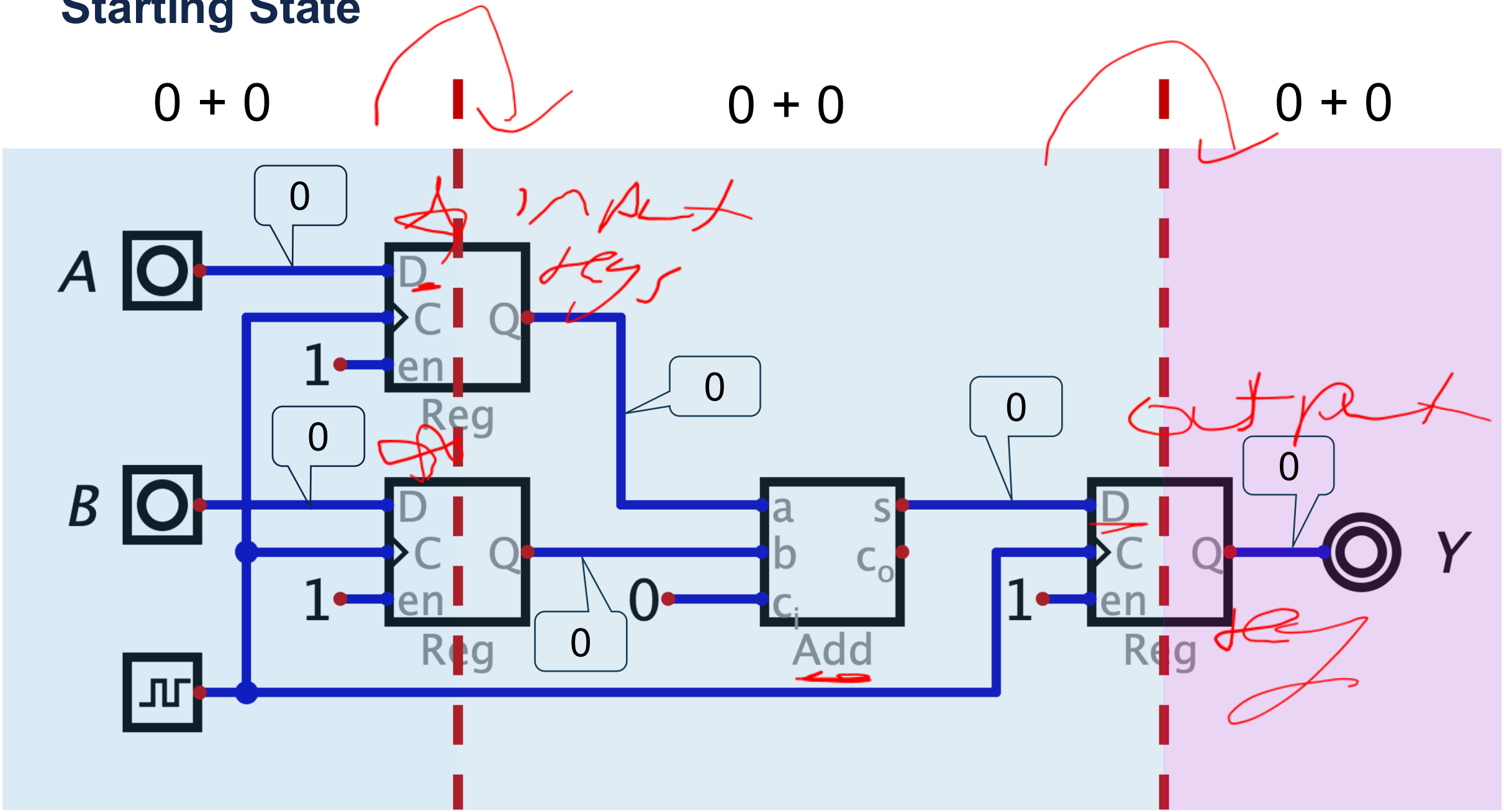
Timing


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

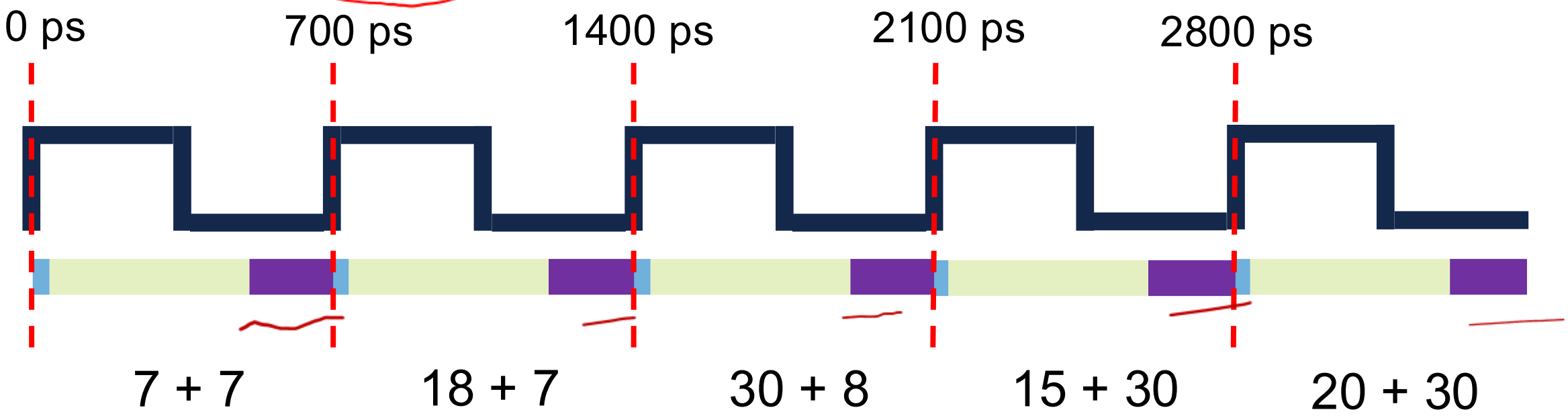
Timing



Starting State



Clock Period = 700ps

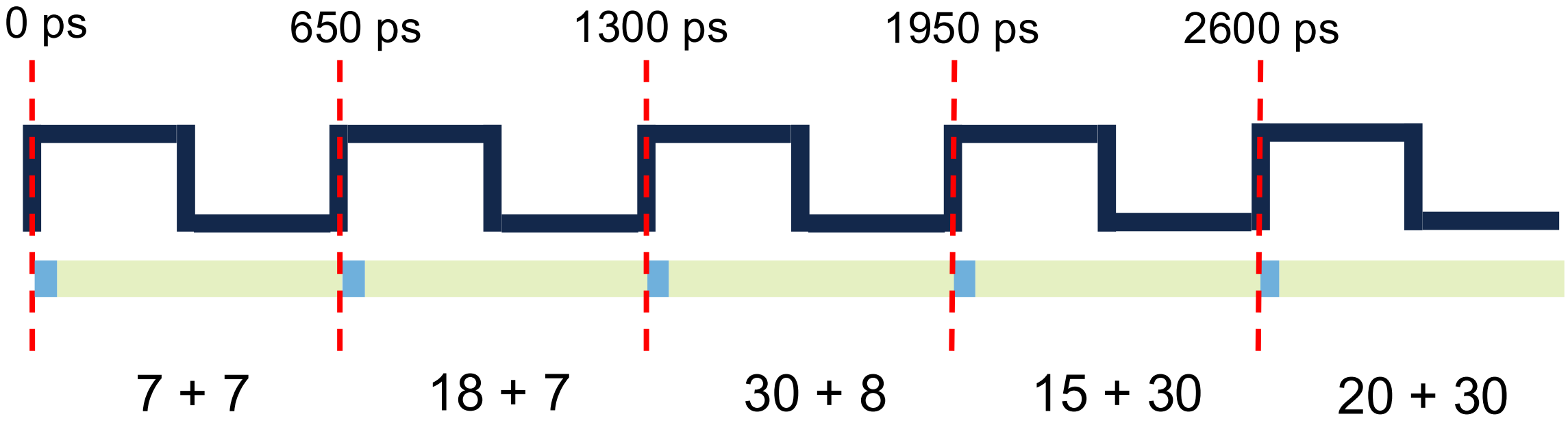


clk-to-q delay = 50ps

adder propagation delay = 600ps

idle time = 50ps

Clock Period = 650ps



clk-to-q delay = 50ps

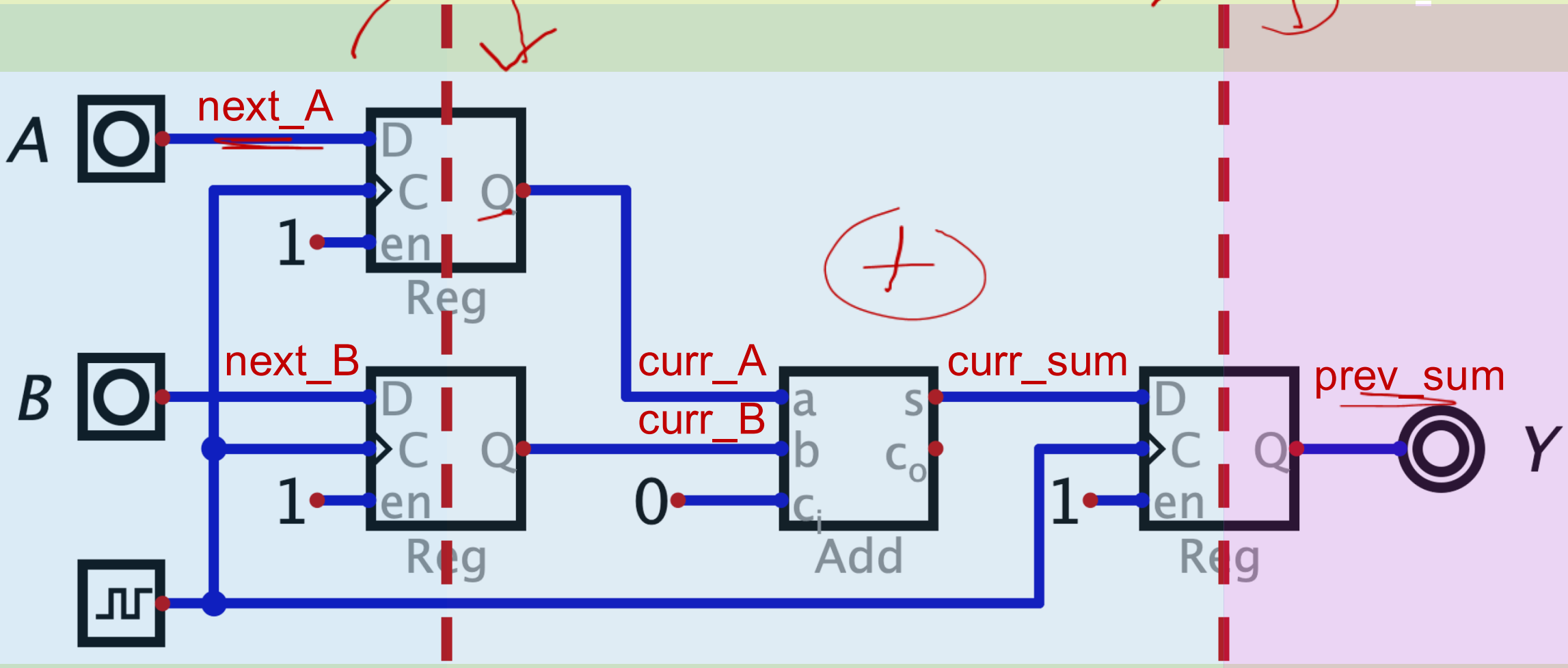
adder propagation delay = 600ps

Comparing Clock Periods

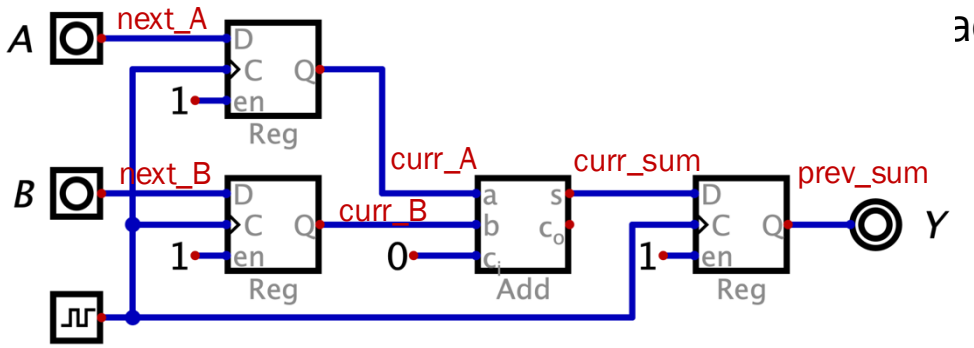


- Our circuit functioned correctly at both of the clock periods we tried
  - 650ps and 700ps
- Our five addition operations finished faster when we set our clock period to 650ps compared to 700ps

Ex3: Clock period = 500ps, clk-to-q delay = 50ps, adder propagation delay = 600ps. On WS, do cycle 0 and 1



Cycle 0 & 1: 7 + 7

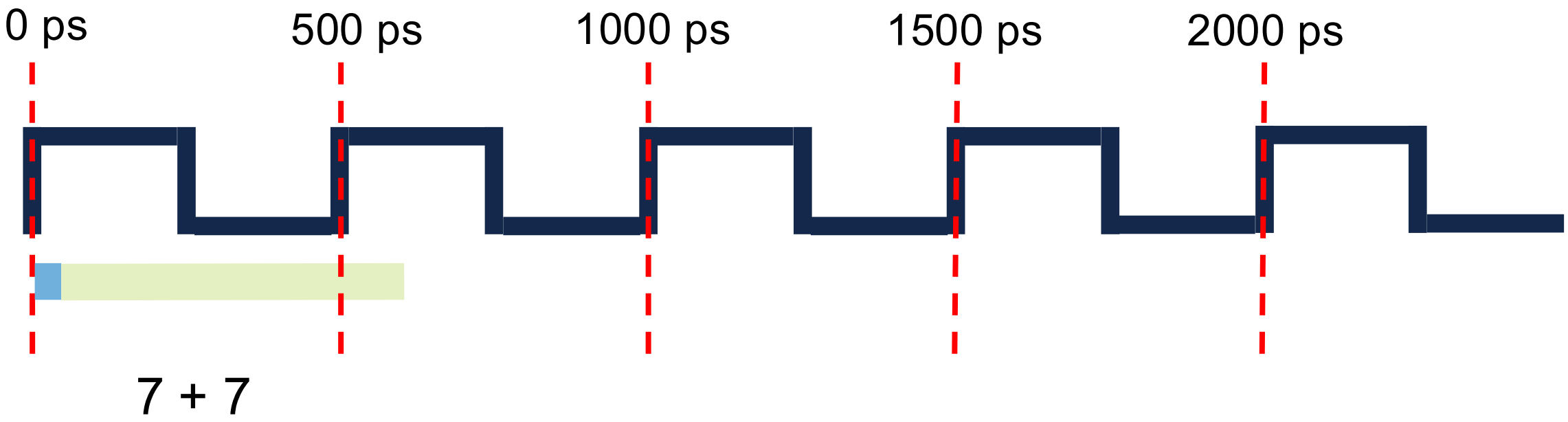


adder propagation delay = 600 ps
clk-to-q delay is 50ps
clock period = 500ps



Signal	Starting State	t = 0 A = 7 B = 7	t = 50ps A = 7 B = 7	t = 500ps A = 18 B = 7	t = 550ps A = 18 B = 7
next_A	0	7	7	18	18
next_B	0	7	7	7	7
curr_A	0	0	7	7	18
curr_B	0	0	7	7	7
curr_sum	0	0	0	??	??
prev_sum	0	0	0	0	??

Clock Period = 500ps



clk-to-q delay = 50ps

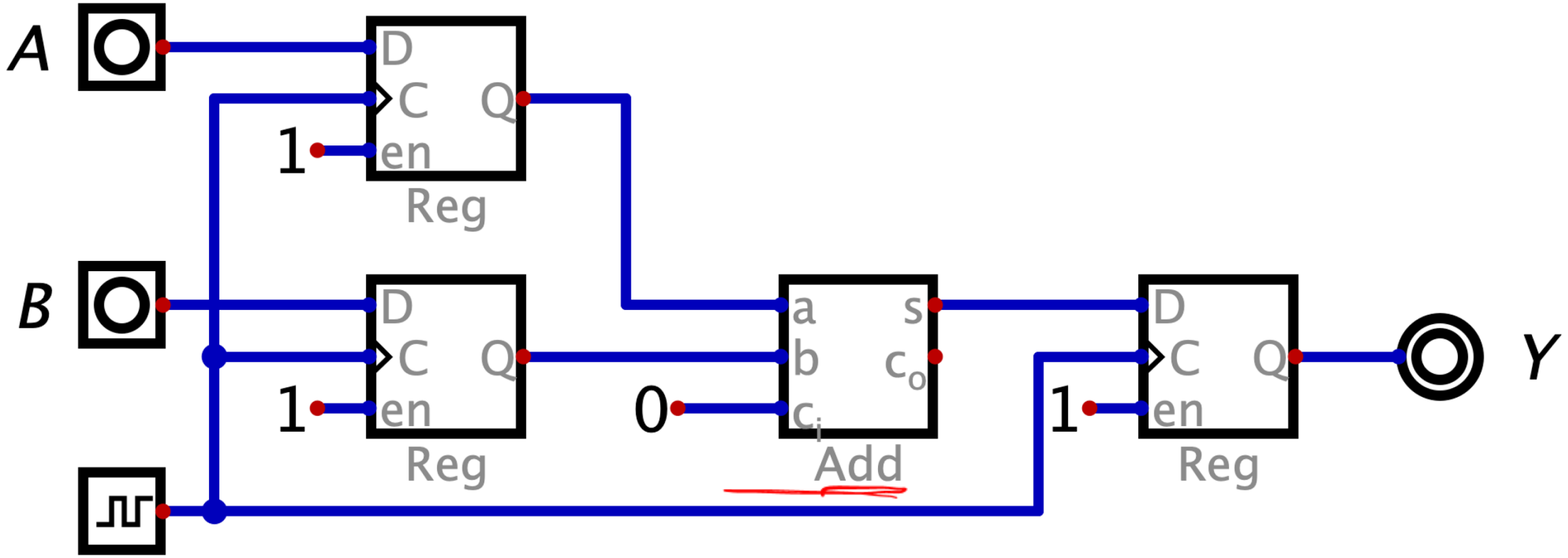
adder propagation delay = 600ps

Clock Period = 500ps



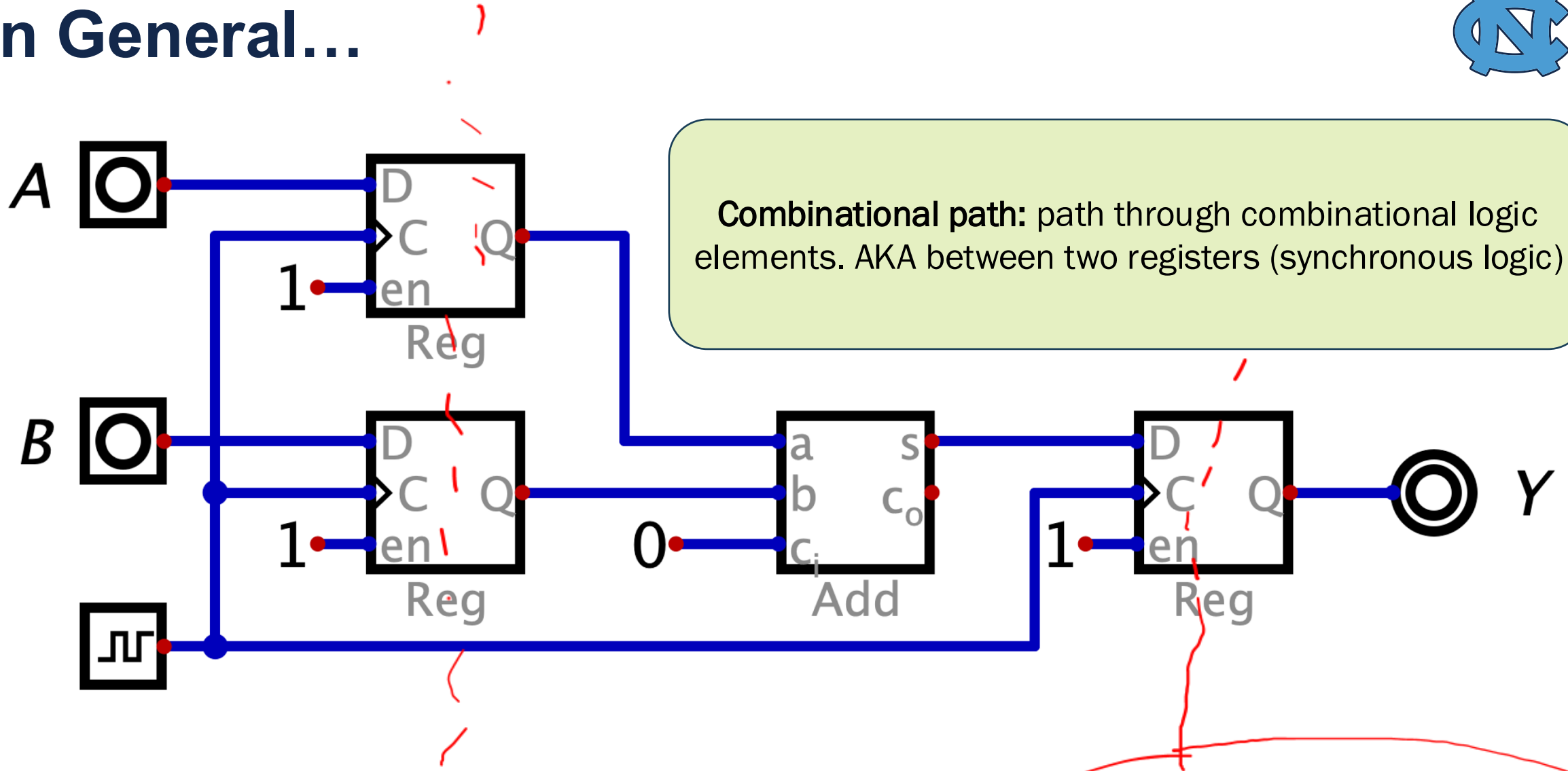
- Our circuit does not operate correctly with a period of 500ps
- Why?
  - The adder cannot finish its operation before the next rising edge of the clock
  - The output register ends up reading garbage

Take Away: How to determine the minimum clock period?



min clock period = clk-to-q delay of the input registers + adder propagation delay

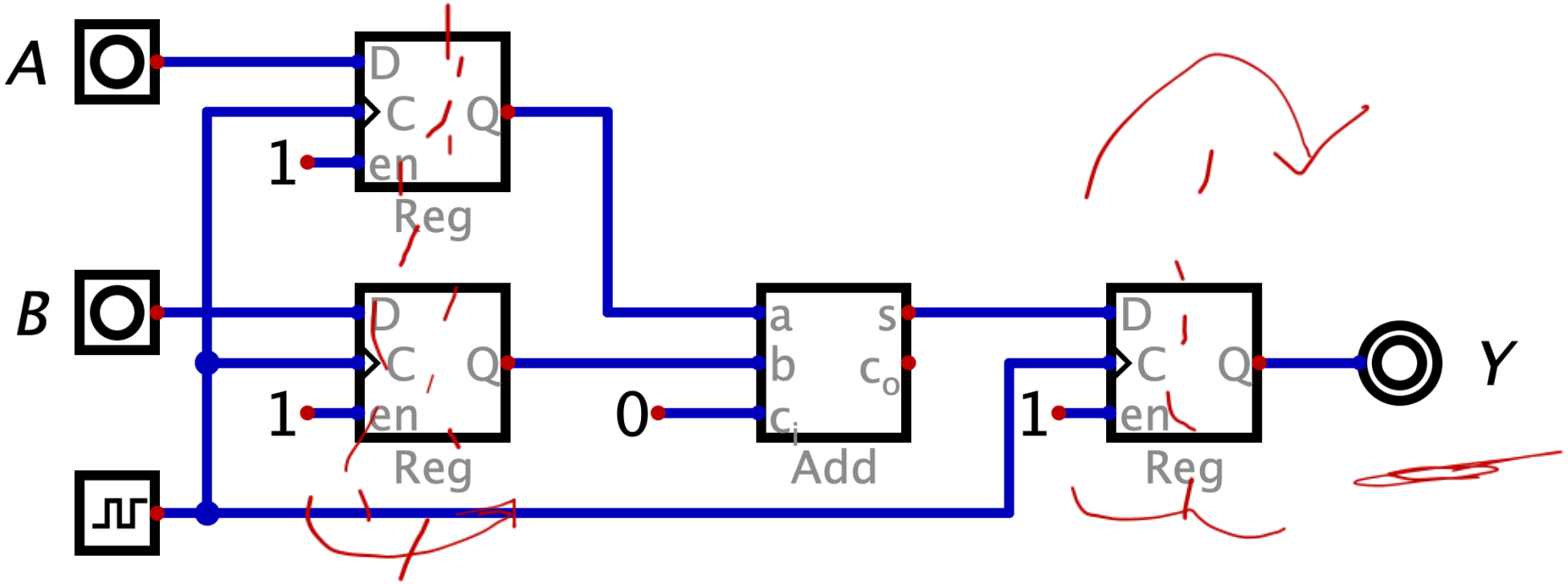
In General…



Combinational path: path through combinational logic elements. AKA between two registers (synchronous logic)

min clock period = clk-to-q delay of the input registers + ***longest combinational path***

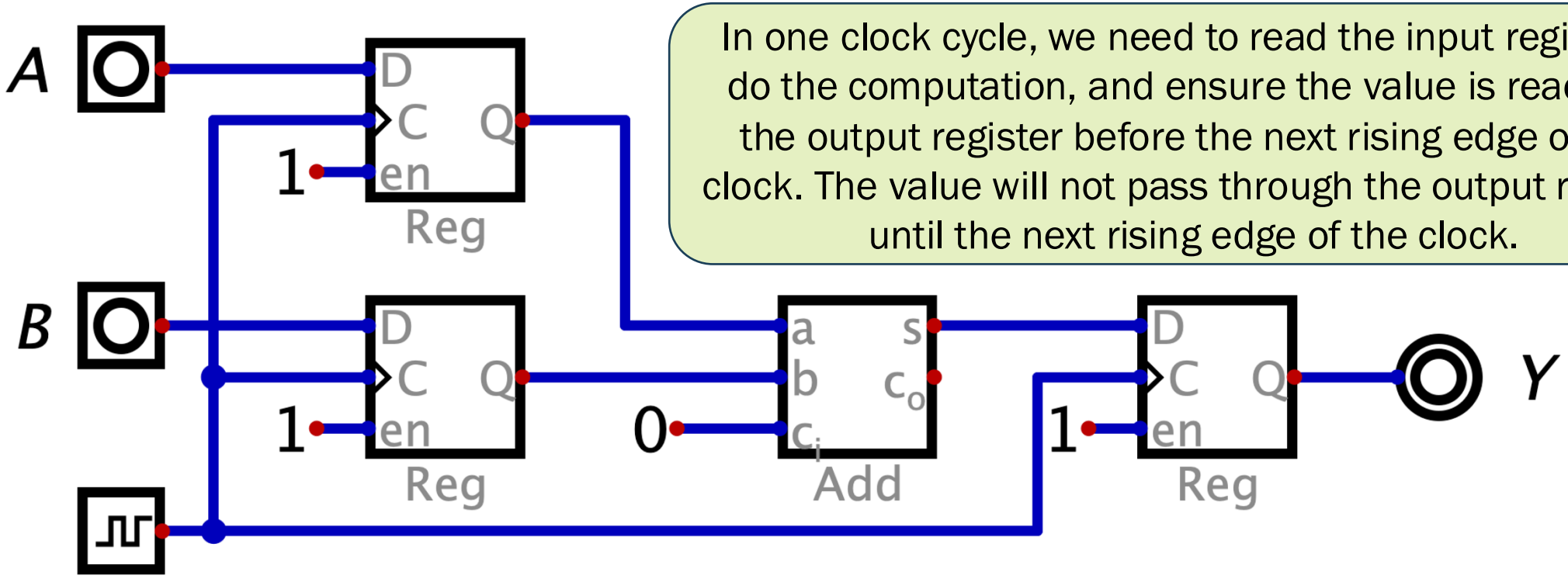
Min Clock Period



- Why do we not include the clk-to-q delay of the output register when determining the min clock period?

Min Clock Period

- Why do we not include the clk-to-q delay of the output register when determining the min clock period?



In one clock cycle, we need to read the input registers, do the computation, and ensure the value is ready for the output register before the next rising edge of the clock. The value will not pass through the output register until the next rising edge of the clock.



Pipelining!

Min Clock Pd vs. Max Frequency



Min Clock Period

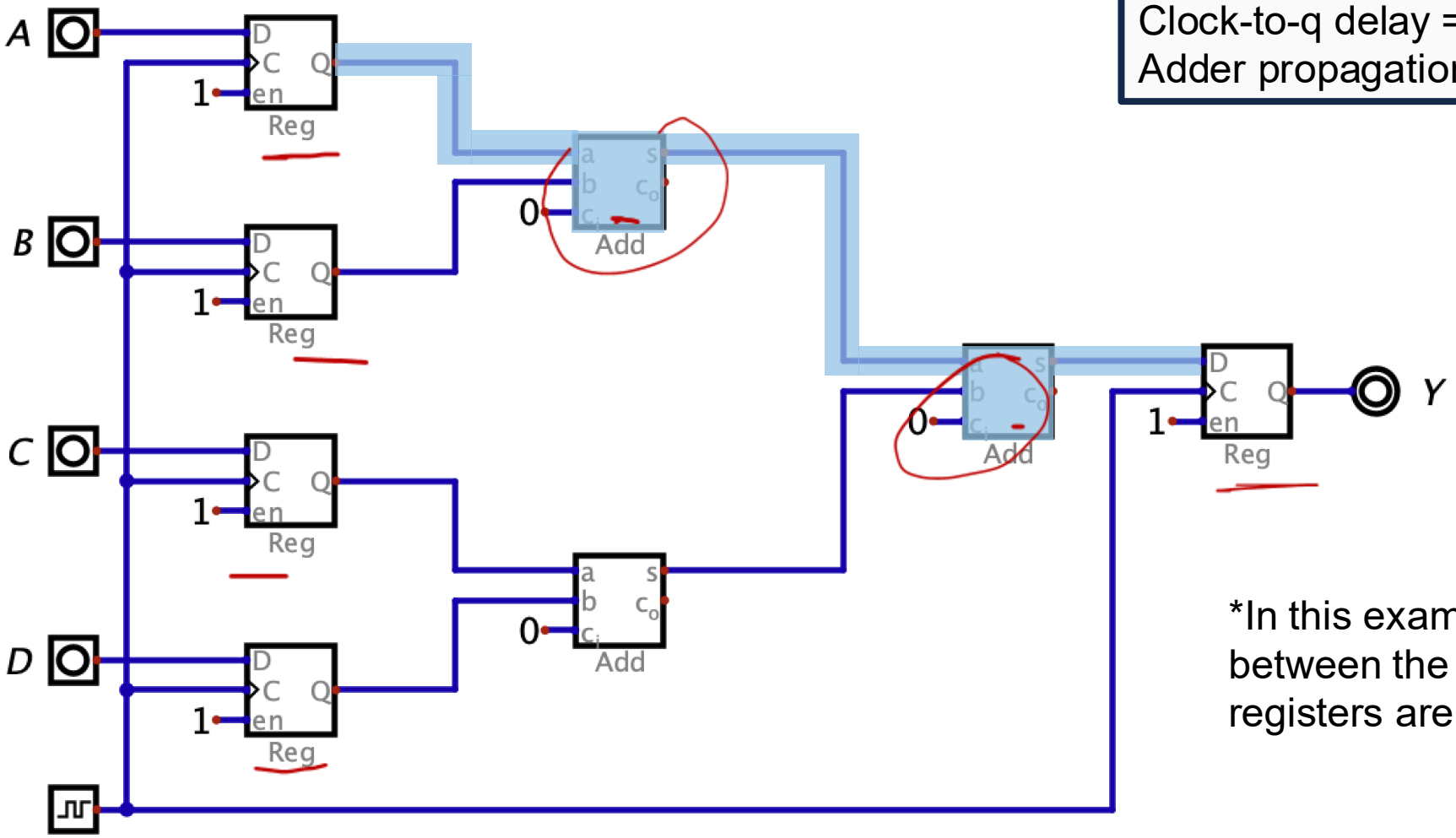
clk-to-q delay + longest combinational delay

Max Clock Frequency

1 / min period

The longest combinational path is the **critical path** or sequence of gates between two registers that has the **maximum propagation delay**

Single-Cycle: Longest Combinational Path

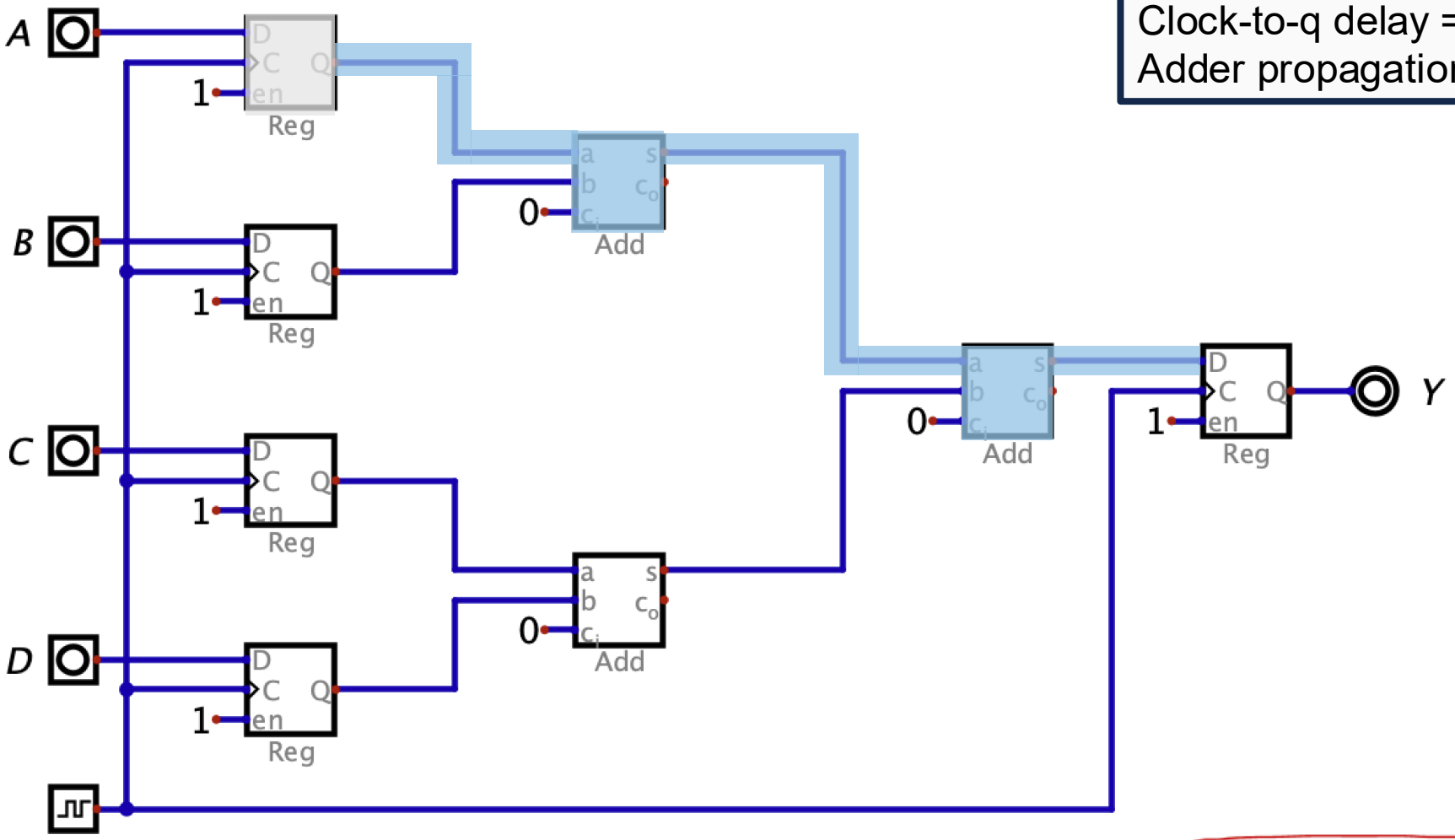


Clock-to-q delay = 20ps
Adder propagation delay = 480ps

*In this example, all paths between the two sets of registers are the same length

Longest comb path = 480ps + 480ps = 960ps

Single-Cycle: Min Clock Period = Critical Path



Clock-to-q delay = 20ps
Adder propagation delay = 480ps

Min clock period = clk-to-q delay + longest comb path = 20ps + 960ps = 980ps

Clock Period

- At which of the following clock periods will the previous circuit work properly?
  - 1000ps
  - 960ps
  - 2000ps
  - 4000ps

min clk pd: 980

Clock Period



- At which of the following clock periods will the previous circuit work properly?
  - 1000ps
  - 960ps
  - 2000ps
  - 4000ps

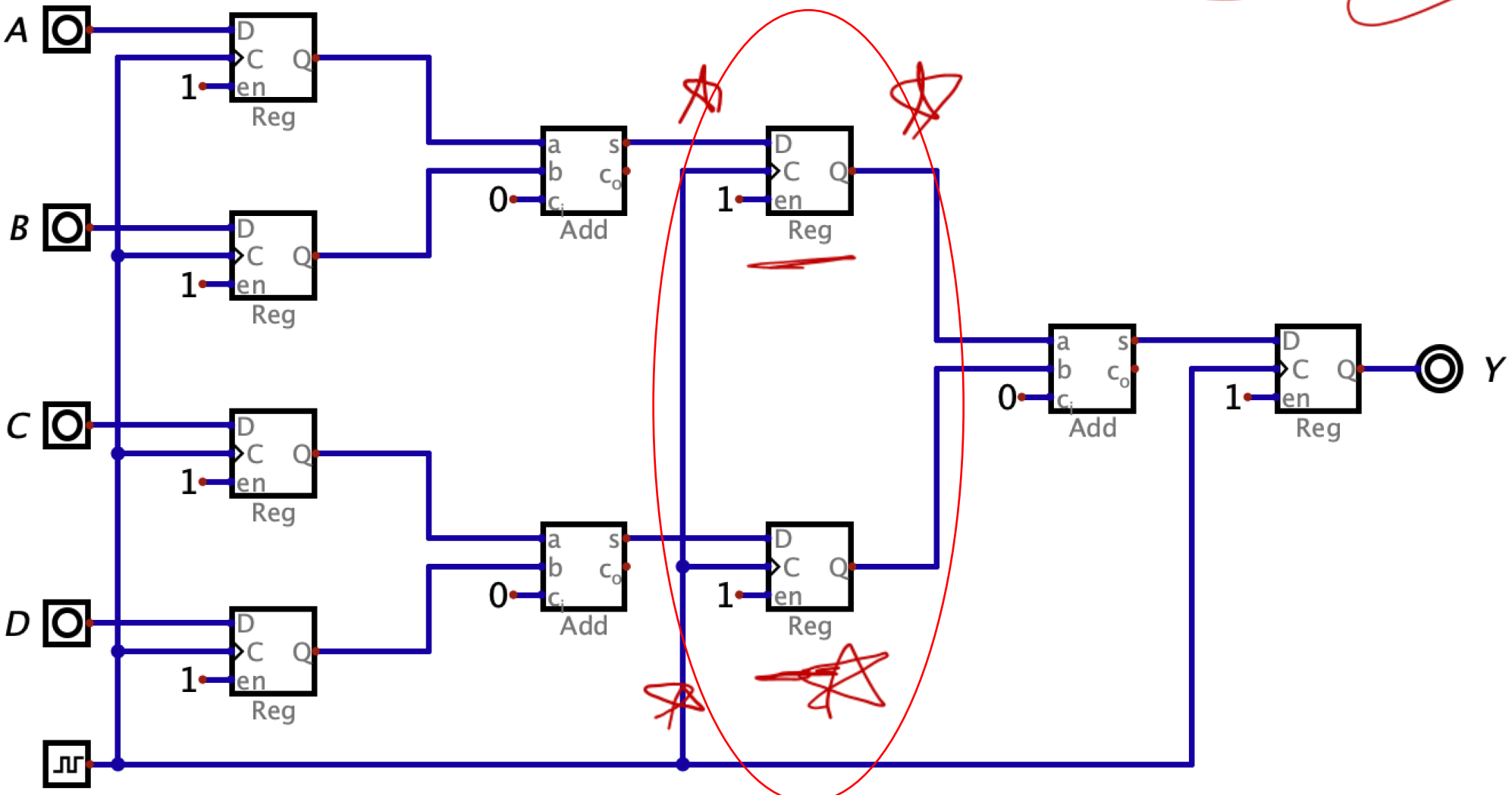
Any clock period greater than or equal to the min clock period

Pipelining



- The previous circuit is referred to as a single-cycle implementation because it completes one operation within a single clock cycle.
- We'll improve the performance of our implementation by using a technique called pipelining
- We'll break down our circuit into **two stages** that can **operate simultaneously**
  - This is more efficient, it keeps more of our hardware busy / doing work

Pipelining

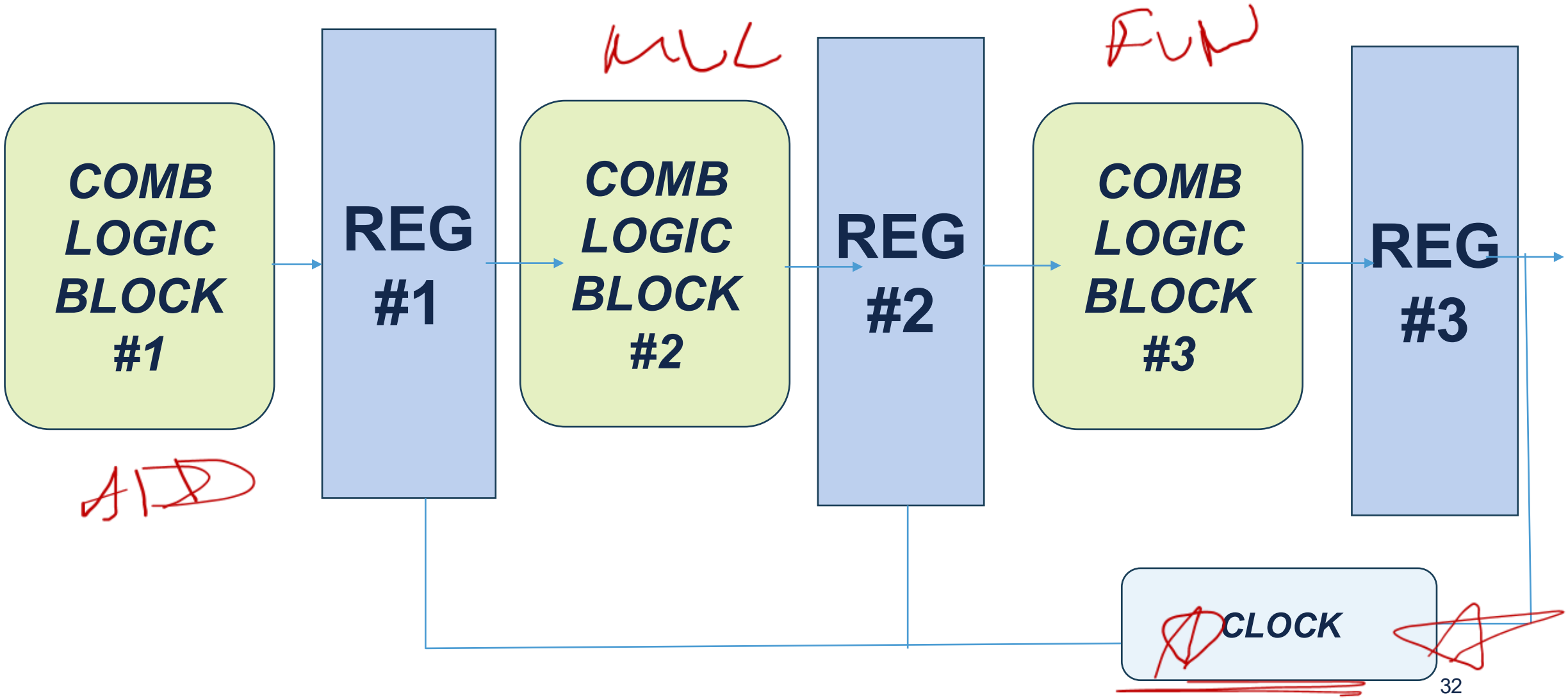


Pipeline Registers



- Inserted between stages of execution to hold intermediate values.
  - Ex: ALU operands that were decoded!
- These registers are not visible / programmable by the user
  - Not part of the 32 general purpose registers
- Allow the **synchronization** of execution of multiple operations (AKA: **instructions**)
  - Different stages operate on different instructions in each clock cycle

Pipeline Registers



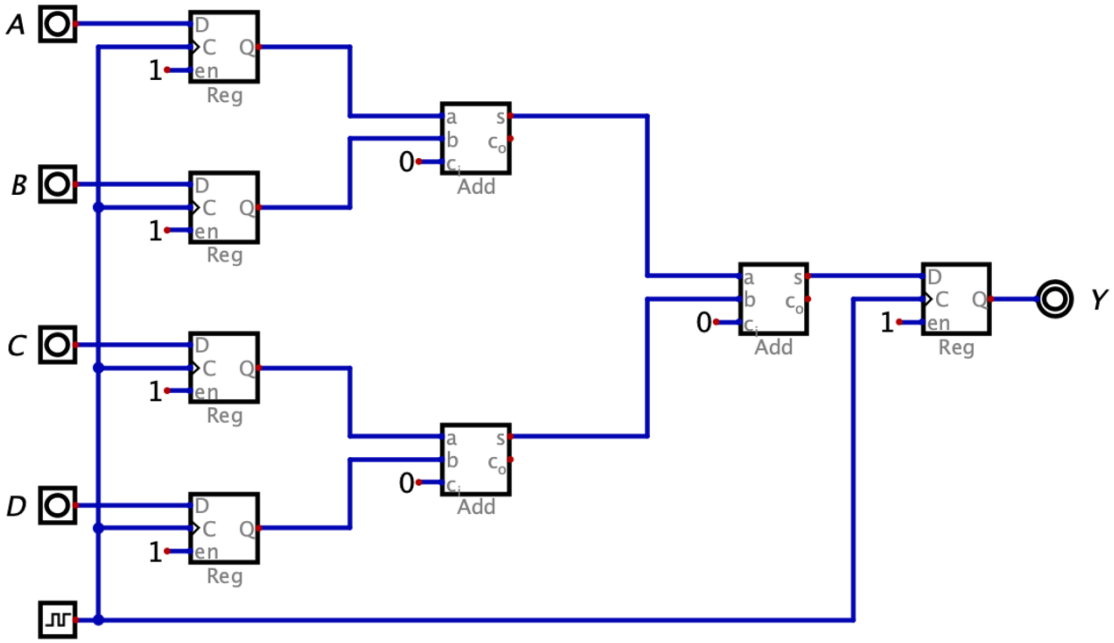
Single Cycle vs Pipelined

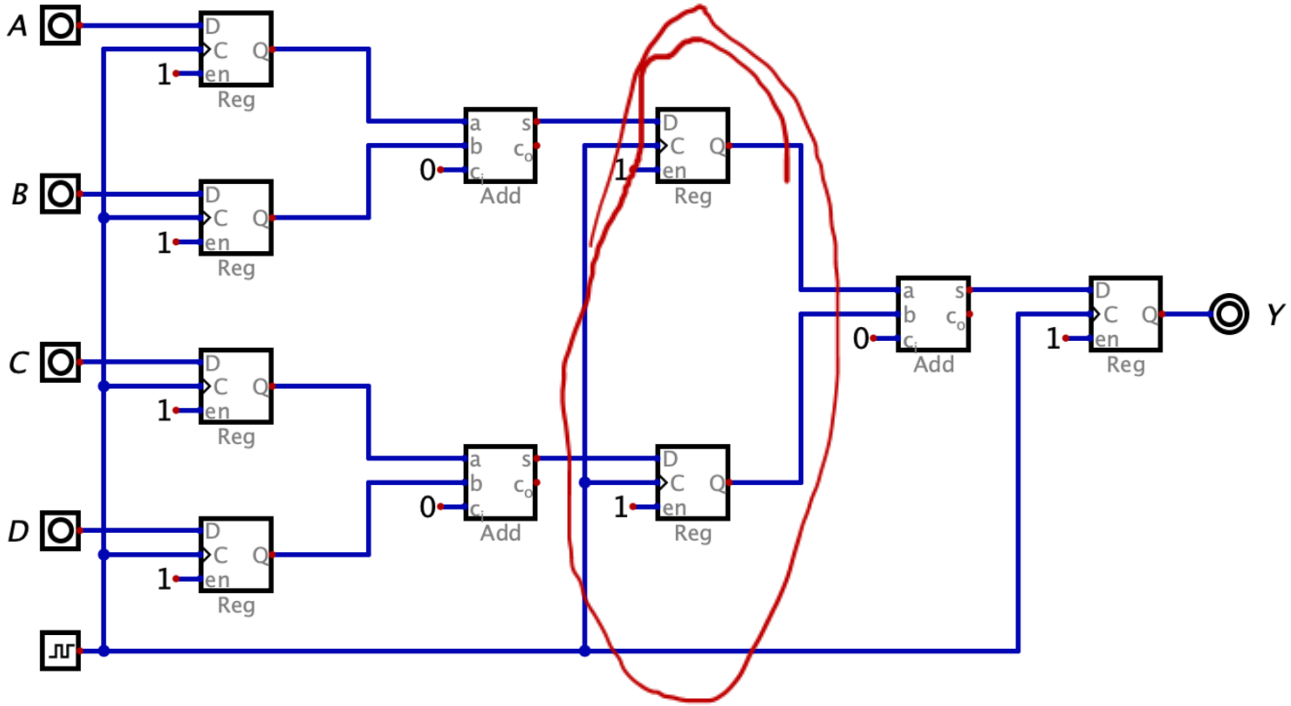


Single-Cycle Implementation

Pipelined Implementation

- Below are two circuits that perform the operation Y = A + B + C + D





Laundry Analogy: Single-Cycle



1. Put the first load of clothes in the washer.
2. When the washer is done, put the clothes in the dryer
3. When the dryer is done, take the clothes out. Then place the next load of laundry in the washer....

Let's say you have five loads of laundry and the washer and dryer each take one hour.

Time (h)	0	1	2	3	4	5	6	7	8	9	10
Load 0	W	D									
Load 1			W	D							
Load 2					W	D					
Load 3											
Load 4											

Laundry Analogy: Single-Cycle

It takes **2n** hours to complete n loads of laundry



1. Put the first load of clothes in the washer.
2. When the washer is done, put the clothes in the dryer
3. When the dryer is done, take the clothes out. Then place the next load of laundry in the washer....

Let's say you have five loads of laundry and the washer and dryer each take one hour.

Time (h)	0	1	2	3	4	5	6	7	8	9	10
Load 0	W	D									
Load 1			W	D							
Load 2					W	D					
Load 3							W	D			
Load 4									W	D	

Laundry Analogy: Pipelined



1. Put the first load of clothes in the washer.
2. When the washer is done, put the clothes in the dryer. Place the next load in the washer.
3. When the dryer is done, take the clothes out. Then move the clothes from the washer into the dryer. Repeat.

Let's say you have five loads of laundry and the washer and dryer each take one hour.

Time (h)	0	1	2	3	4	5	6
Load 0	W	D					
Load 1		W					
Load 2							
Load 3							
Load 4							

Laundry Analogy: Pipelined

It takes **n + 1** hours to complete n loads of laundry



1. Put the first load of clothes in the washer.
2. When the washer is done, put the clothes in the dryer. Place the next load in the washer.
3. When the dryer is done, take the clothes out. Then move the clothes from the washer into the dryer. Repeat.

Let's say you have five loads of laundry and the washer and dryer each take one hour.

Time (h)	0	1	2	3	4	5	6
Load 0	W	D					
Load 1		W	D				
Load 2			W	D			
Load 3				W	D		
Load 4					W	D	

Laundry Analogy: Pipelined

It takes **n + 1** hours to complete n loads of laundry



1. Put the first load of clothes in the washer.
2. When the washer is done, put the clothes in the dryer. Place the next load in the washer.
3. When the d… …ove the clothes from…

Let's say you … take one hour.

Tim…

L…						
Load 3				W	D	
Load 4					W	D

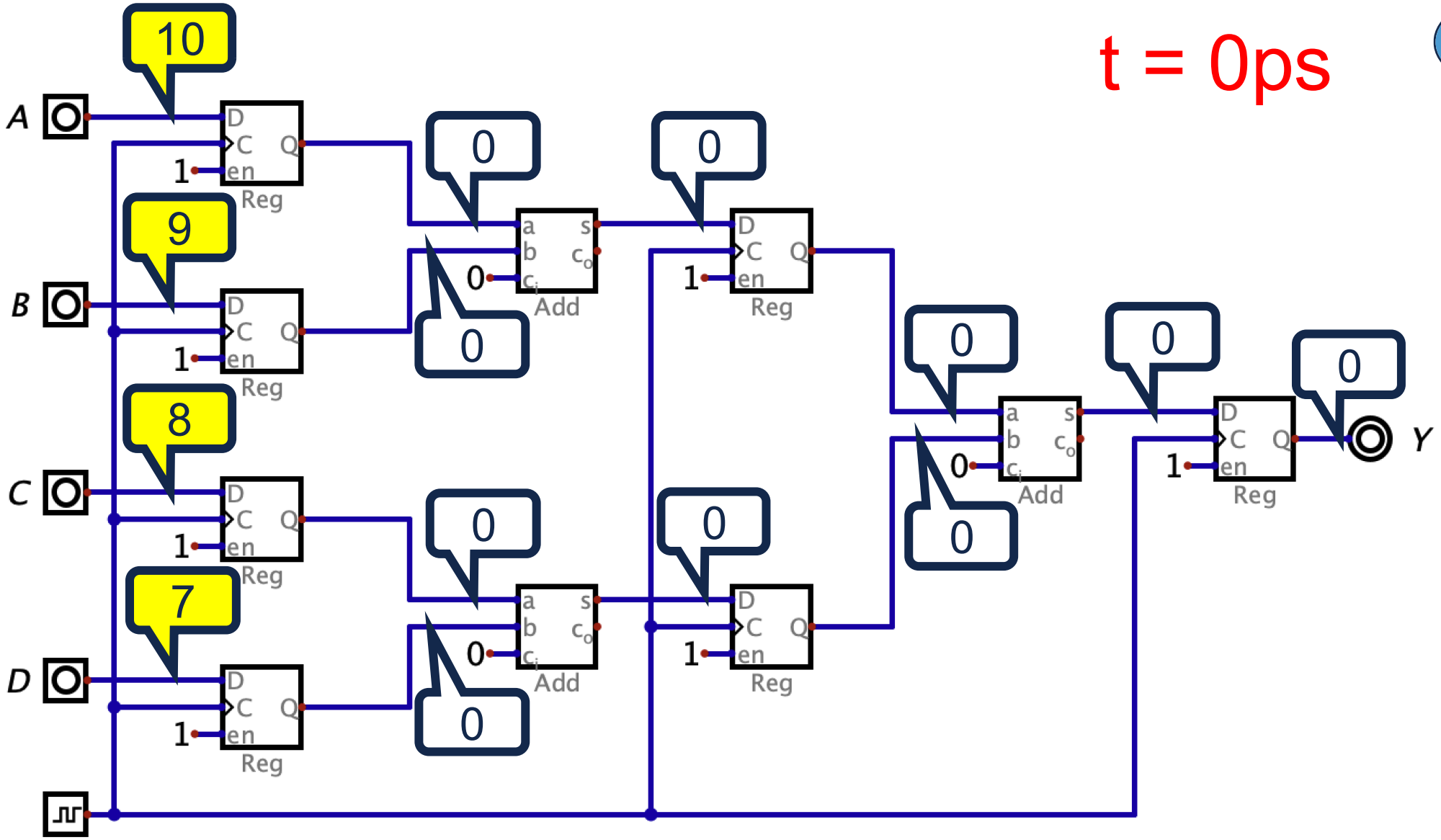
With pipelining, we can overlap the execution of multiple instructions by breaking them into stages, similar to an assembly line. ☺

Pipelining



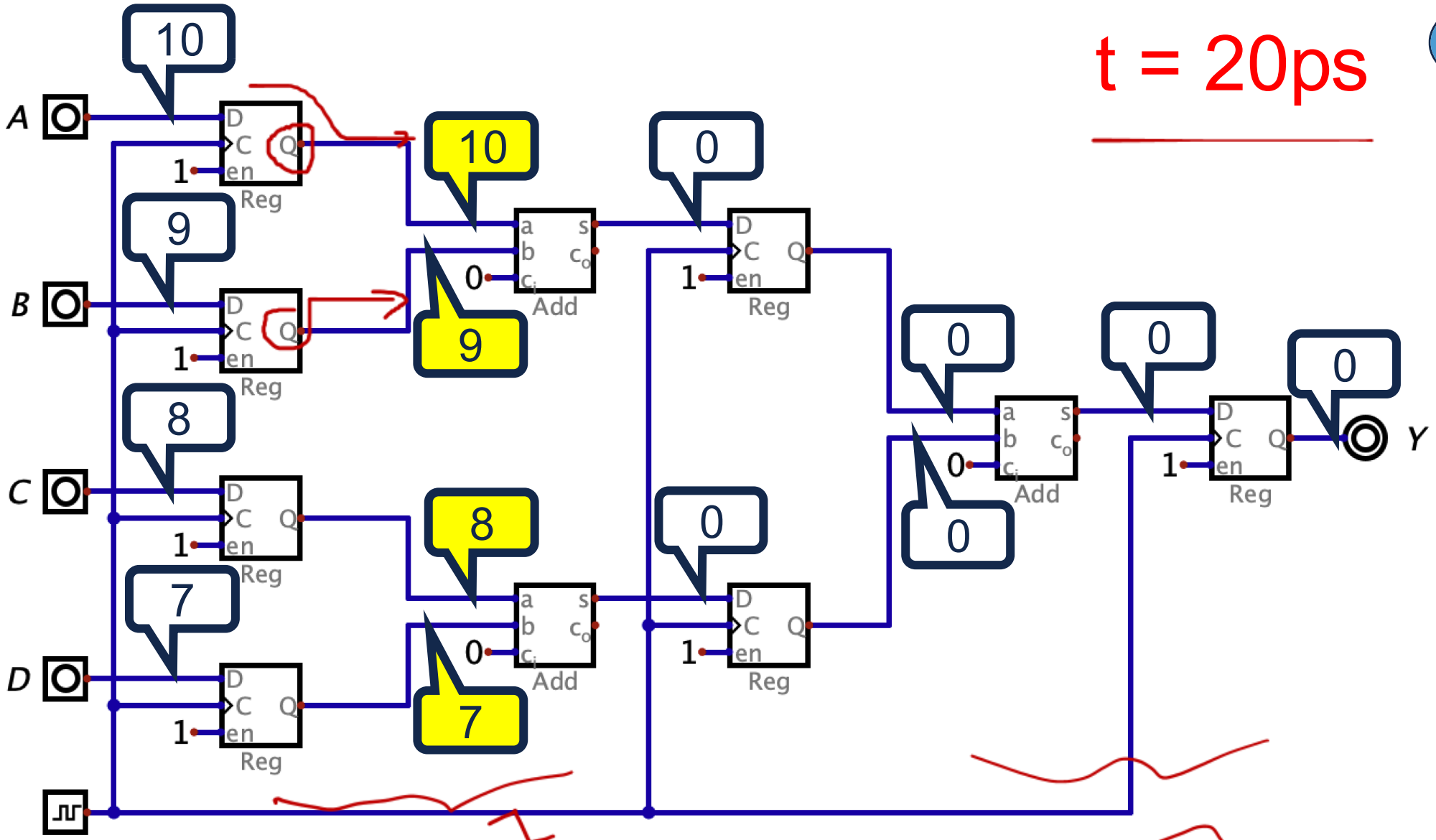
- Let's say that we want to perform the following operations
  - 10 + 9 + 8 + 7
  - 5 + 4 + 3 + 2
  - 10 + 10 + 20 + 20
  - 3 + 3 + 4 + 4
- We'll start at time t = 0
- We'll set our clock period to 500ps

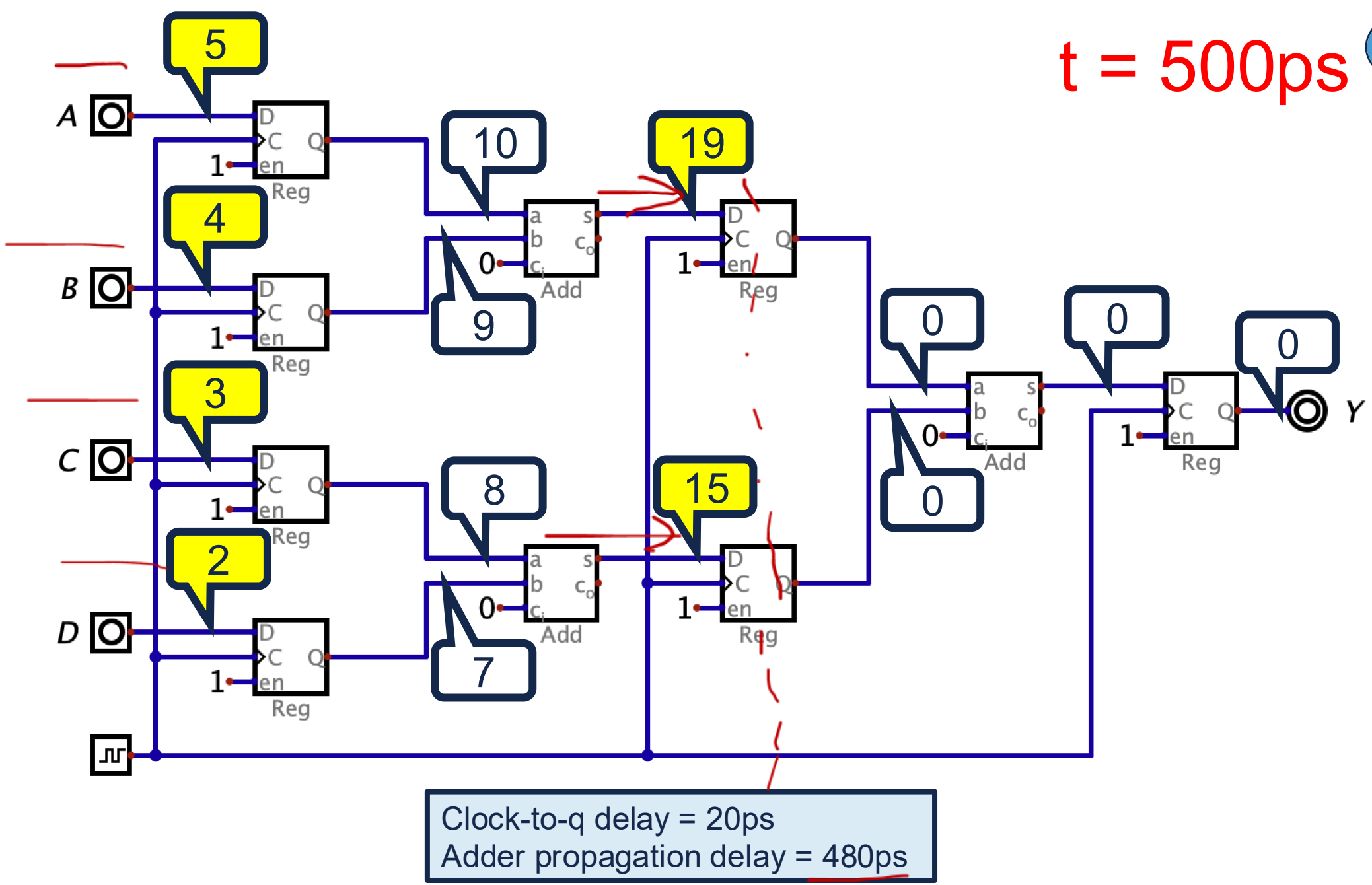
Clock-to-q delay = 20ps
Adder propagation delay = 480ps



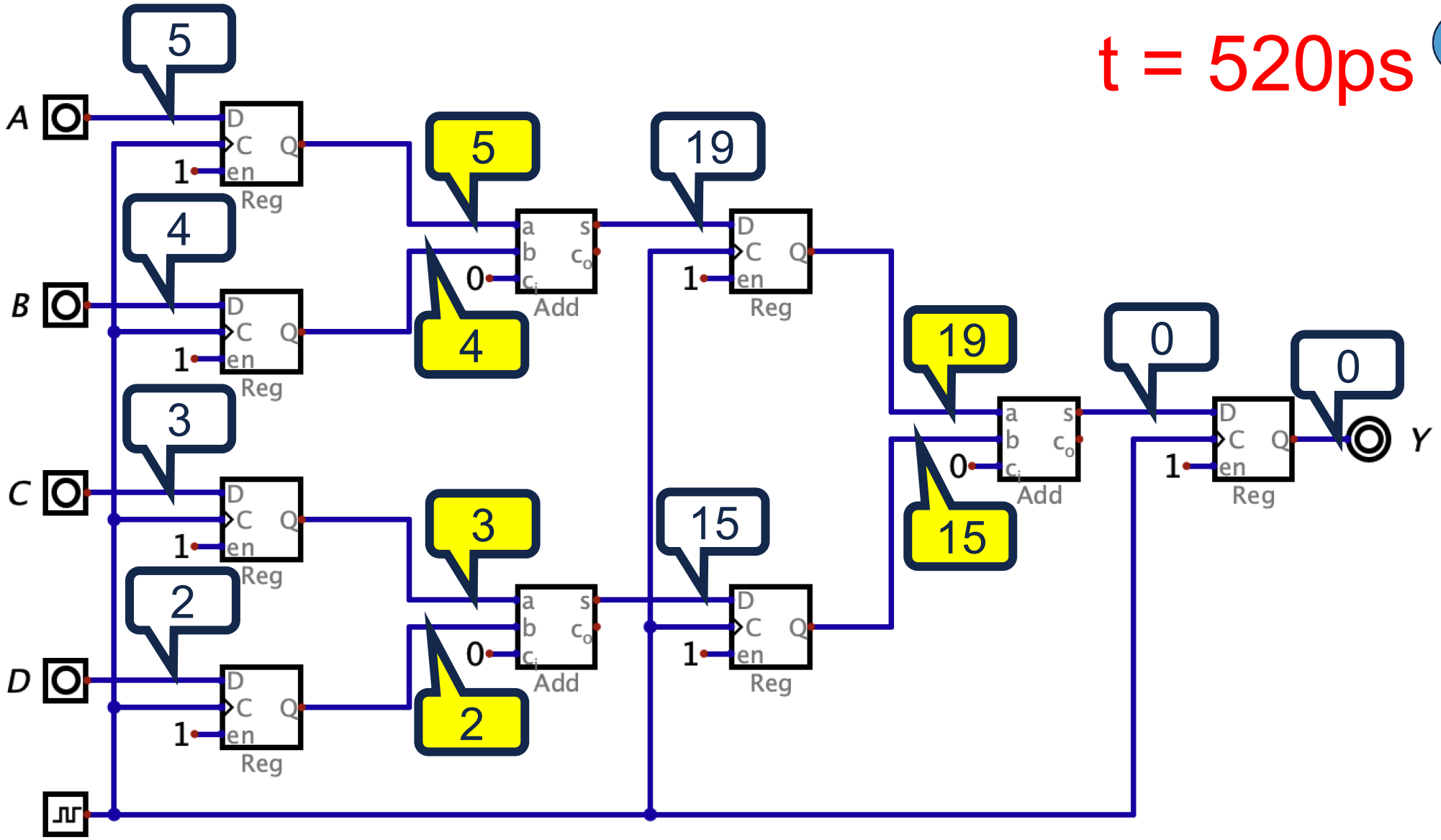
Clock-to-q delay = 20ps
Adder propagation delay = 480ps

t = 20ps


A
B
C
D
10
9
8
7
1
Reg
10
9
8
7
0
Add
0
1
Reg
0
0
0
0
Y
Clock-to-q delay = 20ps
Adder propagation delay = 480ps

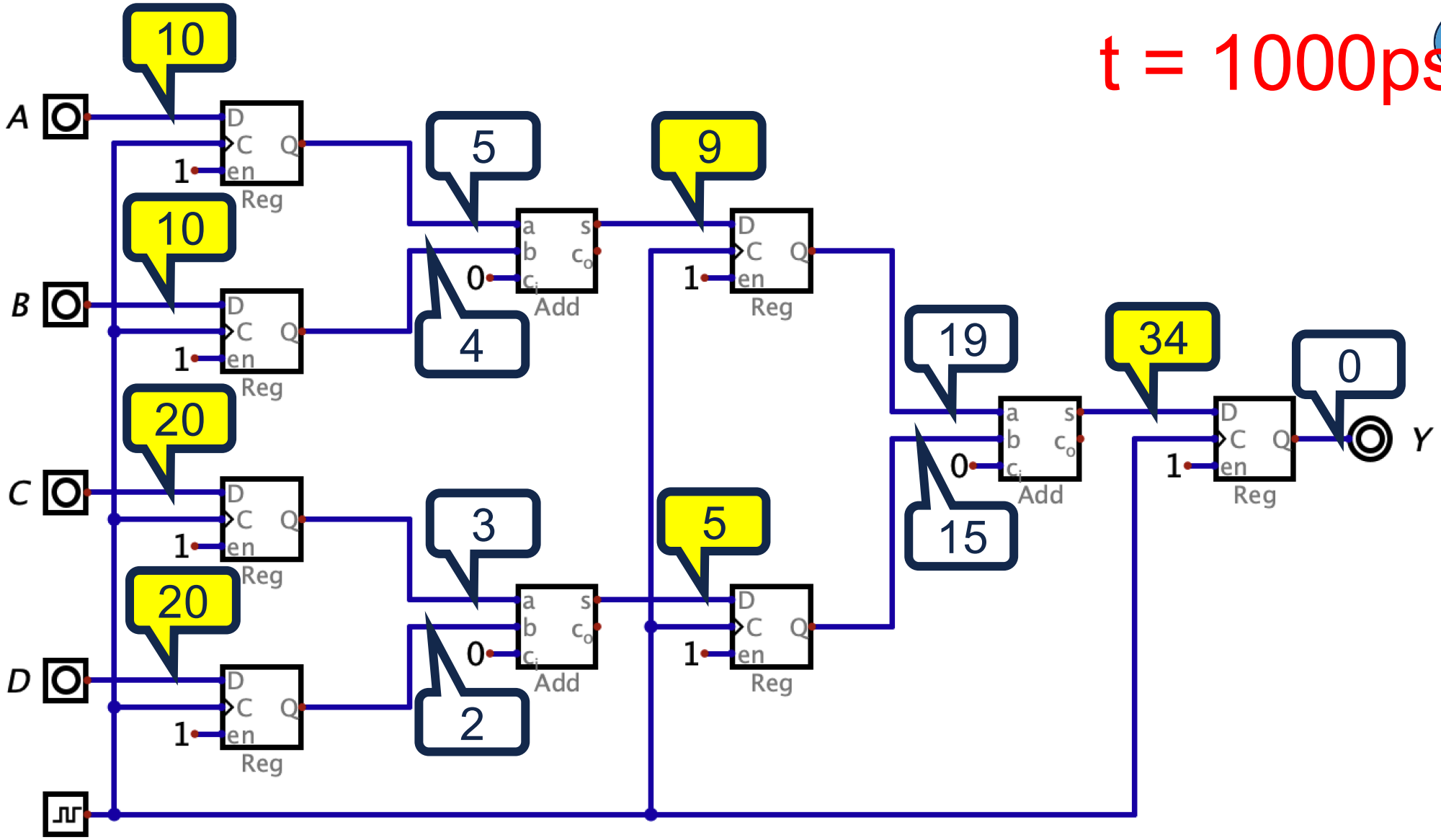

t = 500ps
A
B
C
D
5
4
3
2
10
9
8
7
19
15
0
0
0
0
0
0
Y
Add
Reg
Clock-to-q delay = 20ps
Adder propagation delay = 480ps

t = 520ps

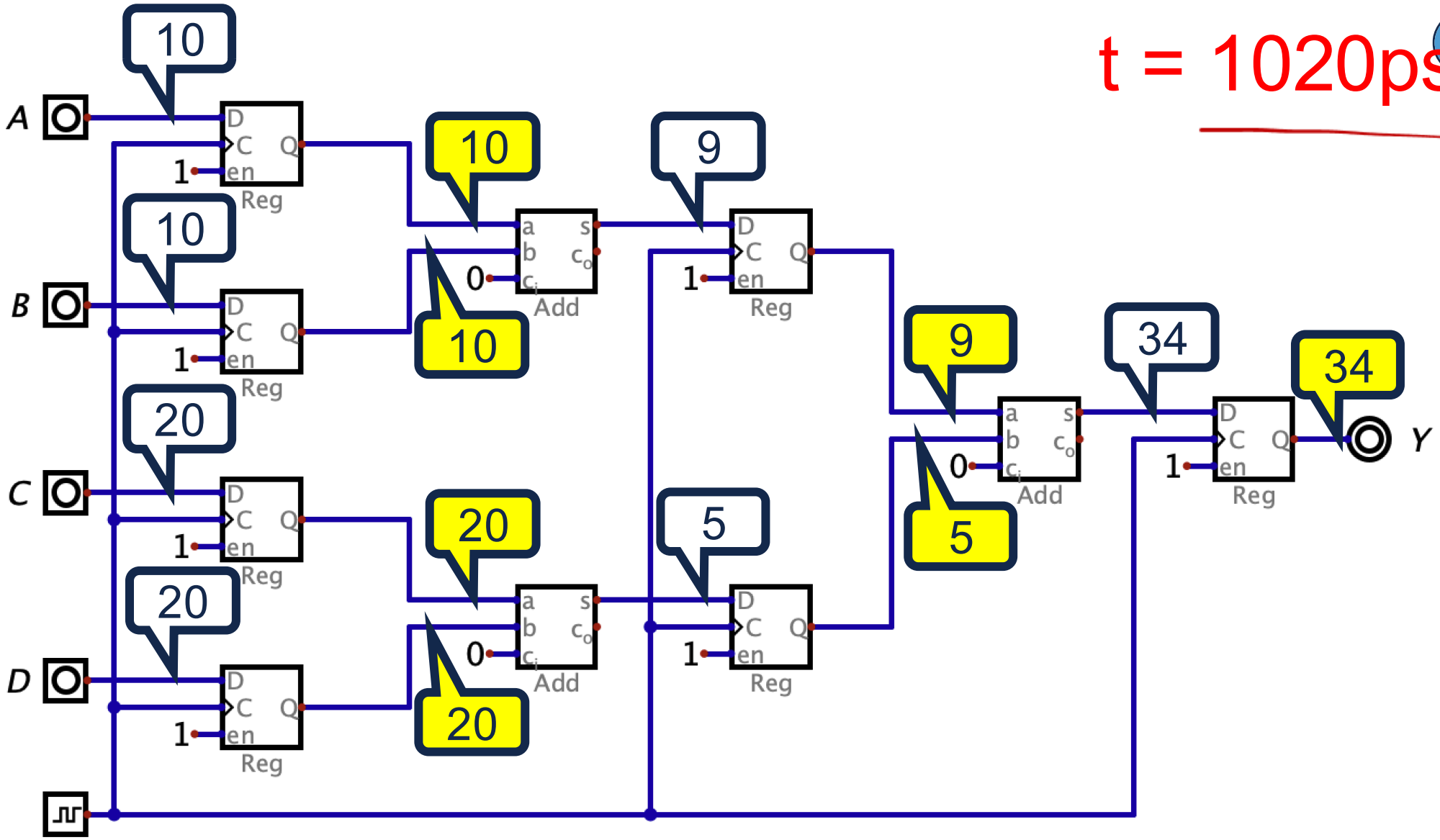


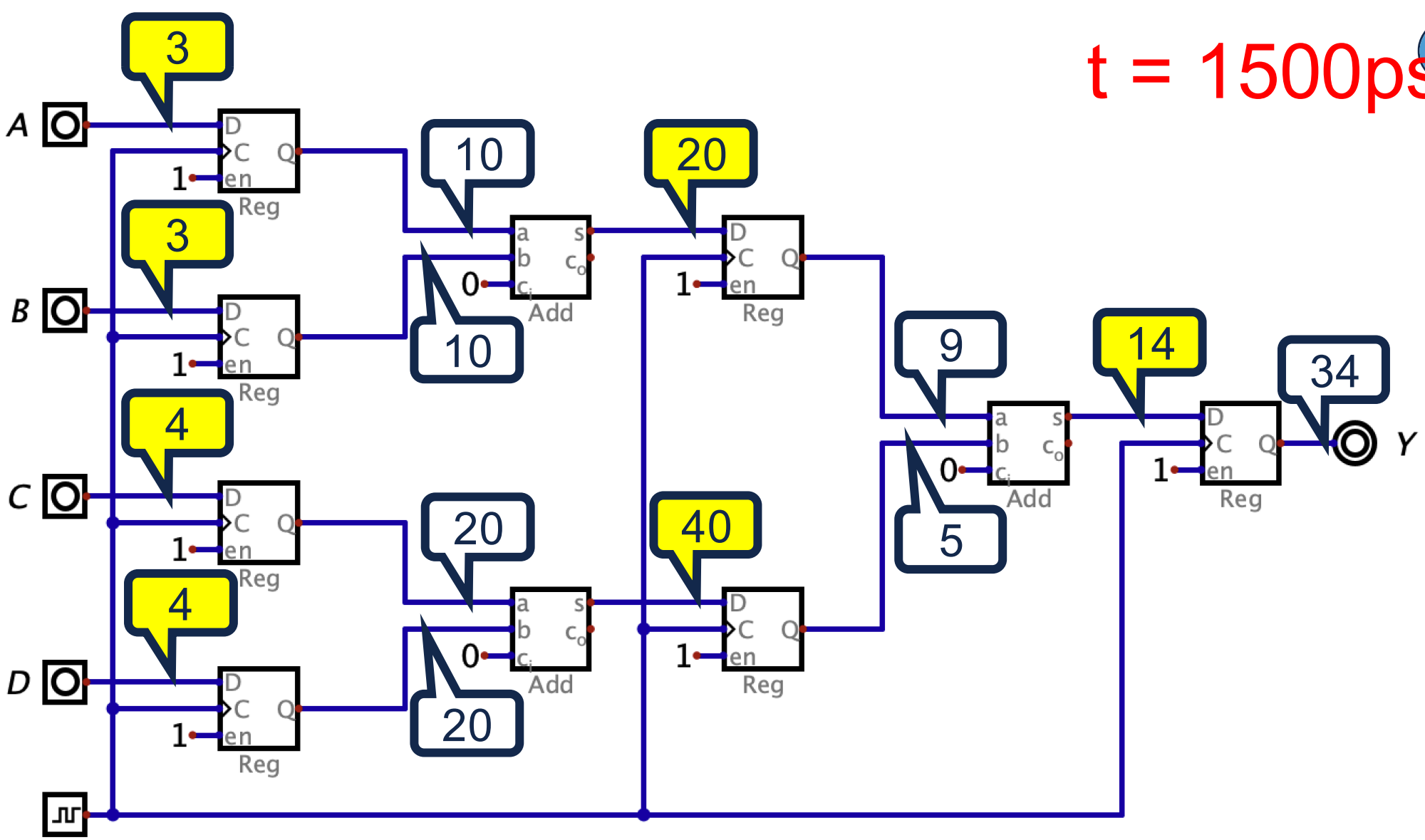
Clock-to-q delay = 20ps
Adder propagation delay = 480ps

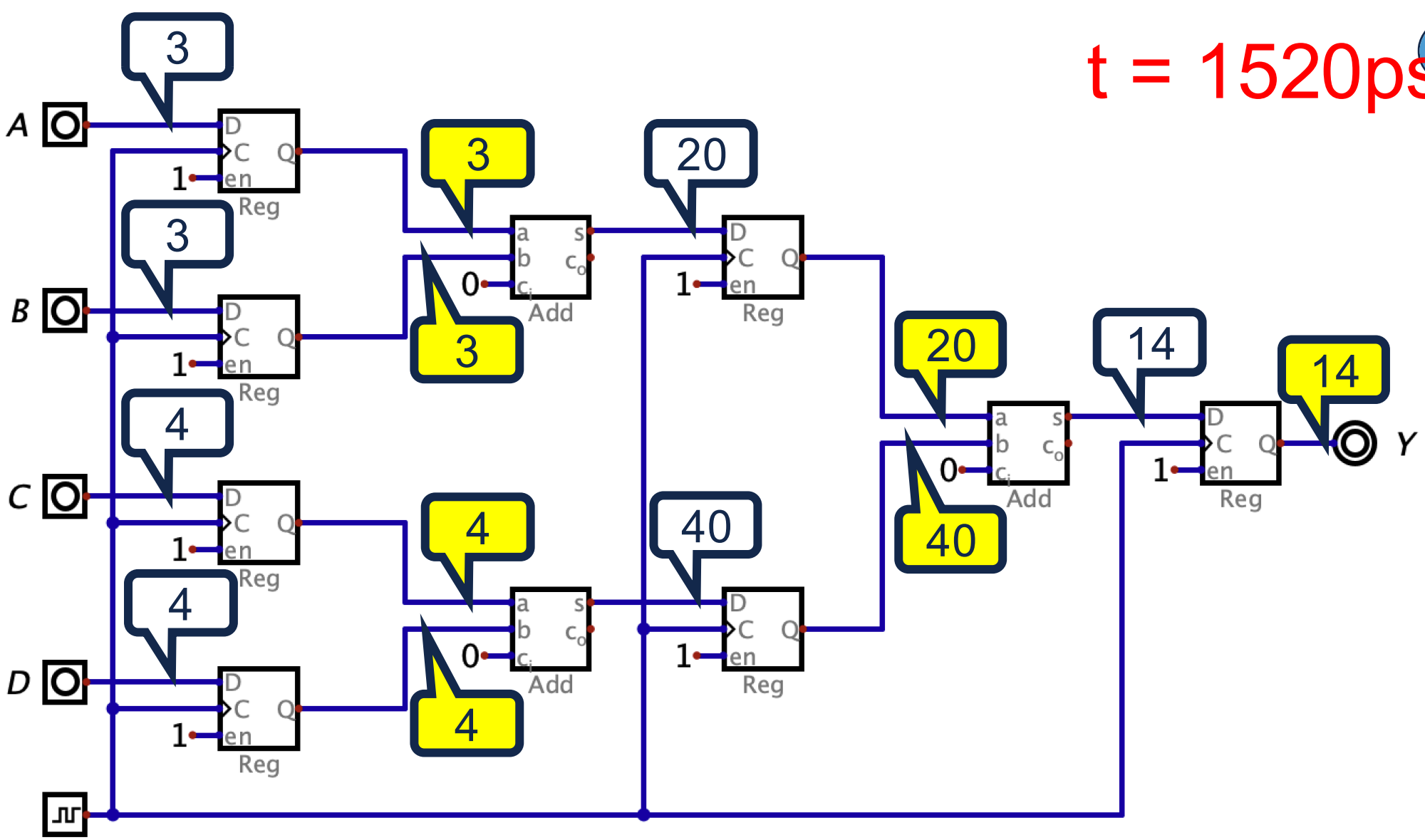
t = 1000ps



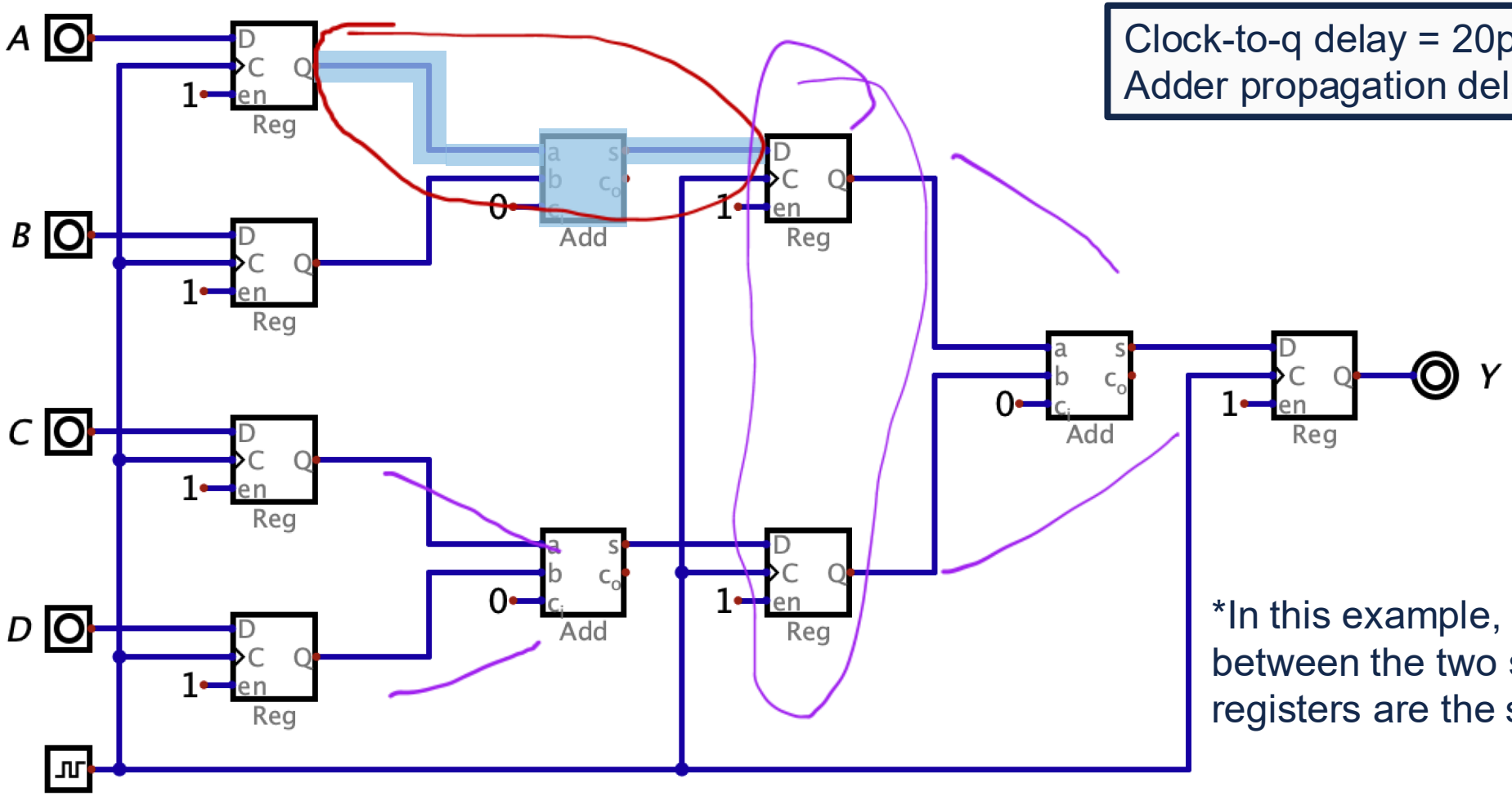
Clock-to-q delay = 20ps
Adder propagation delay = 480ps


t = 1020ps
A
B
C
D
Y
10
10
20
20
10
10
20
20
9
5
9
5
34
34
D
C
Q
en
Reg
a
b
s
c_o
c_i
Add
1
0
Clock-to-q delay = 20ps
Adder propagation delay = 480ps


t = 1500ps
A
B
C
D
3
3
4
4
10
10
20
20
40
9
5
14
34
Y
D
C
Q
en
Reg
a
b
s
c_o
c_i
Add
1
0
Clock-to-q delay = 20ps
Adder propagation delay = 480ps


t = 1520ps
A
B
C
D
Y
3
3
4
4
3
4
20
40
20
40
14
14
D
C
en
Q
Reg
a
b
s
c_o
c_i
Add
1
0
Clock-to-q delay = 20ps
Adder propagation delay = 480ps

Pipelined: Longest Combinational Path

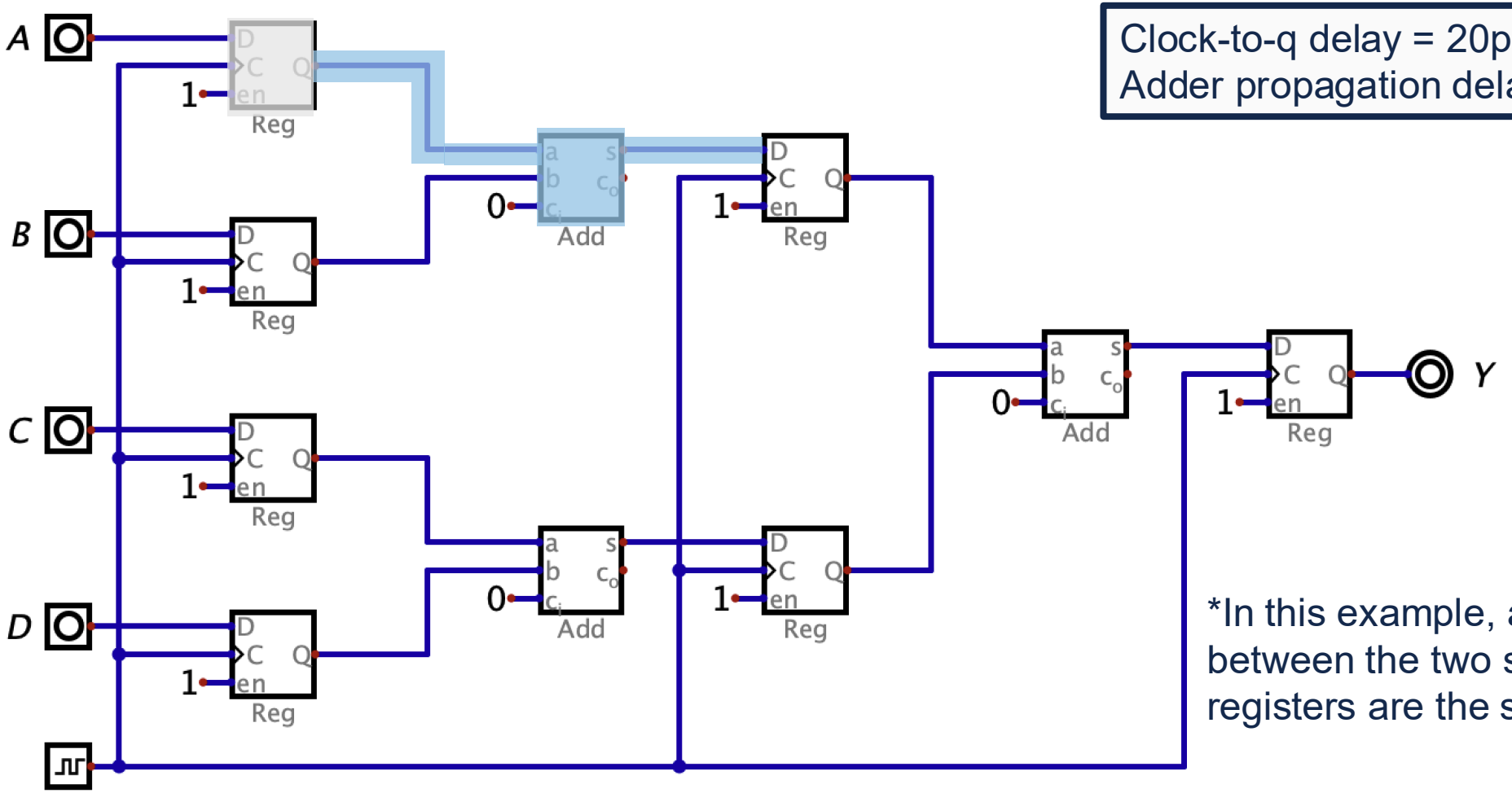


Clock-to-q delay = 20ps
Adder propagation delay = 480ps

*In this example, all paths between the two sets of registers are the same length

Longest comb path = 480ps

Pipelined: Min Clock Period = Critical Path



Clock-to-q delay = 20ps
Adder propagation delay = 480ps

*In this example, all paths between the two sets of registers are the same length

Min clock period = clk-to-q delay + longest comb path = 20ps + 480ps = 500ps

Clock Period



- At which of the following clock periods will the pipelined implementation work properly?
  - 480ps
  - 580ps
  - 1000ps
  - 4000ps

min clk pd: 500 ps

Clock Period



- At which of the following clock periods will the pipelined implementation work properly?
  - 480ps
  - 580ps
  - 1000ps
  - 4000ps

Any clock period greater than or equal to the min clock period

Single-Cycle vs Pipelined Execution Times



- To perform n operations
  - Single-cycle: (n)(clock period)
  - Pipelined (n + no. of pipeline regs)(clock period)
    - In this case (n + 1)(clk period)

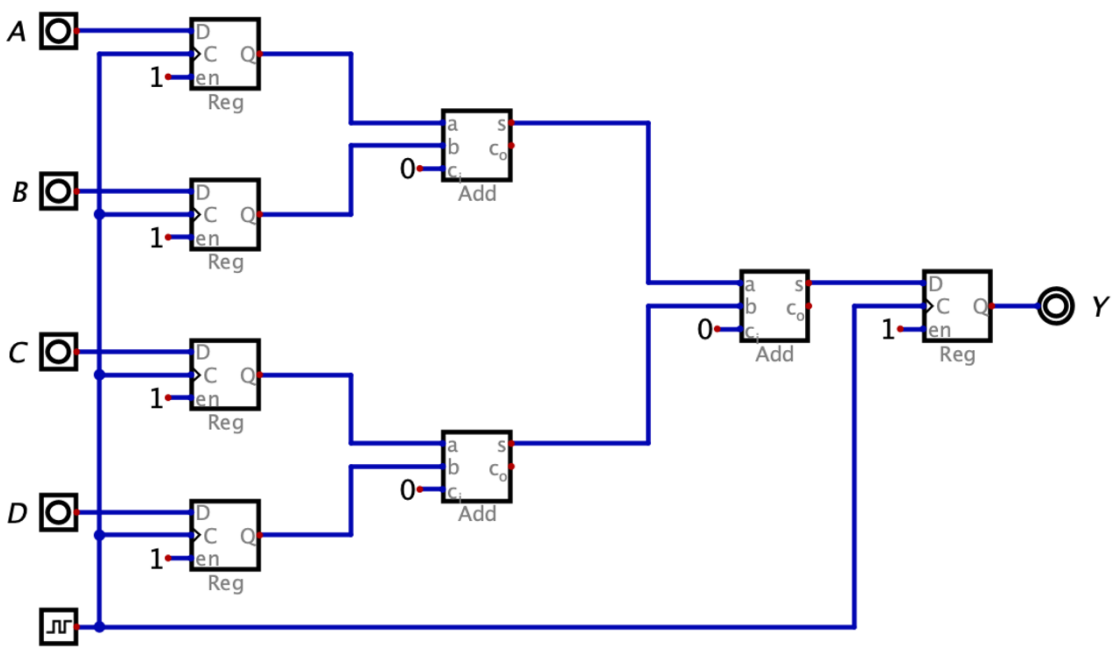
$(n+1)(500)$

$(10+1)(500)$

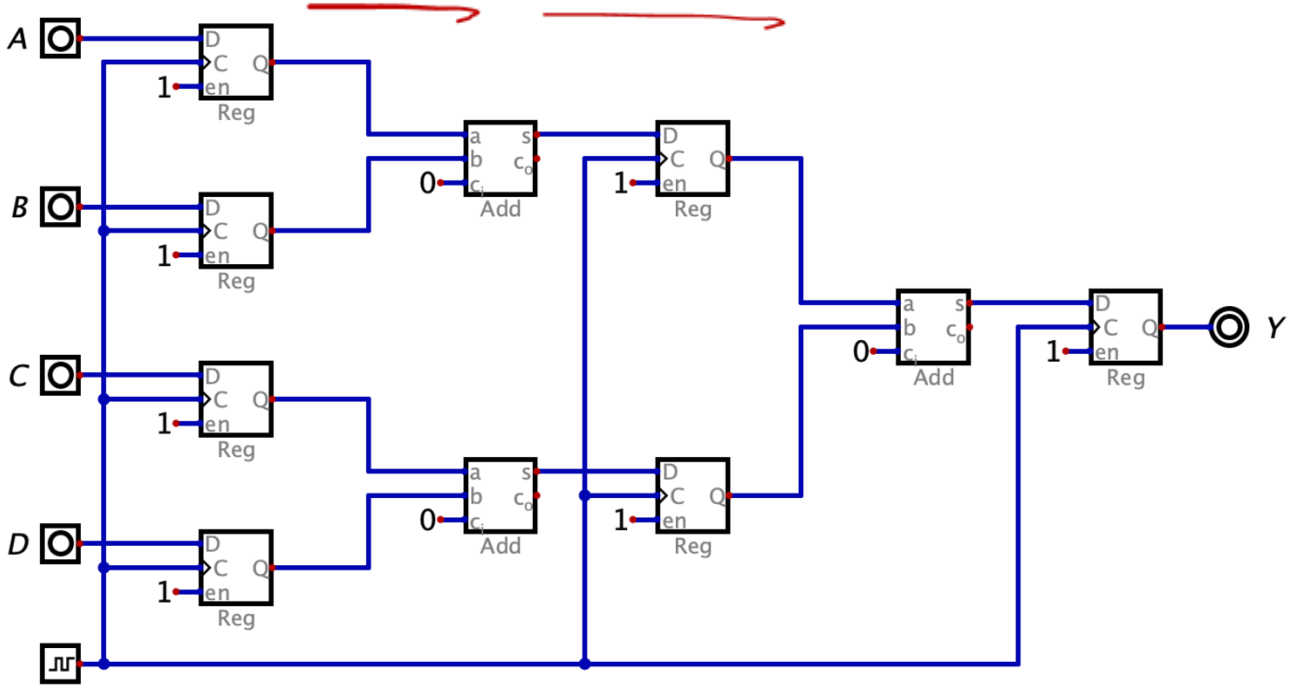
Single Cycle vs Pipelined

Clock-to-q delay = 20ps
Adder propagation delay = 480ps

Single-Cycle Implementation



Pipelined Implementation

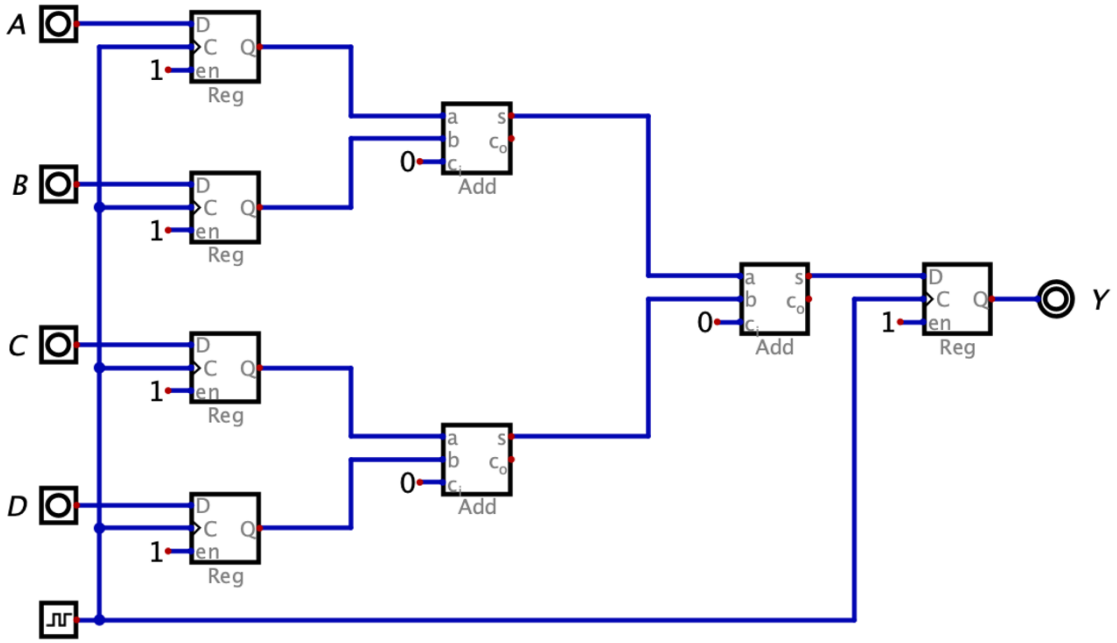


	Single-Cycle	Pipelined
Longest combinational path	960	480
Min clock period	980	500
# cycles to complete one operation	1	2
Time to complete 10 ops at min clock speed		

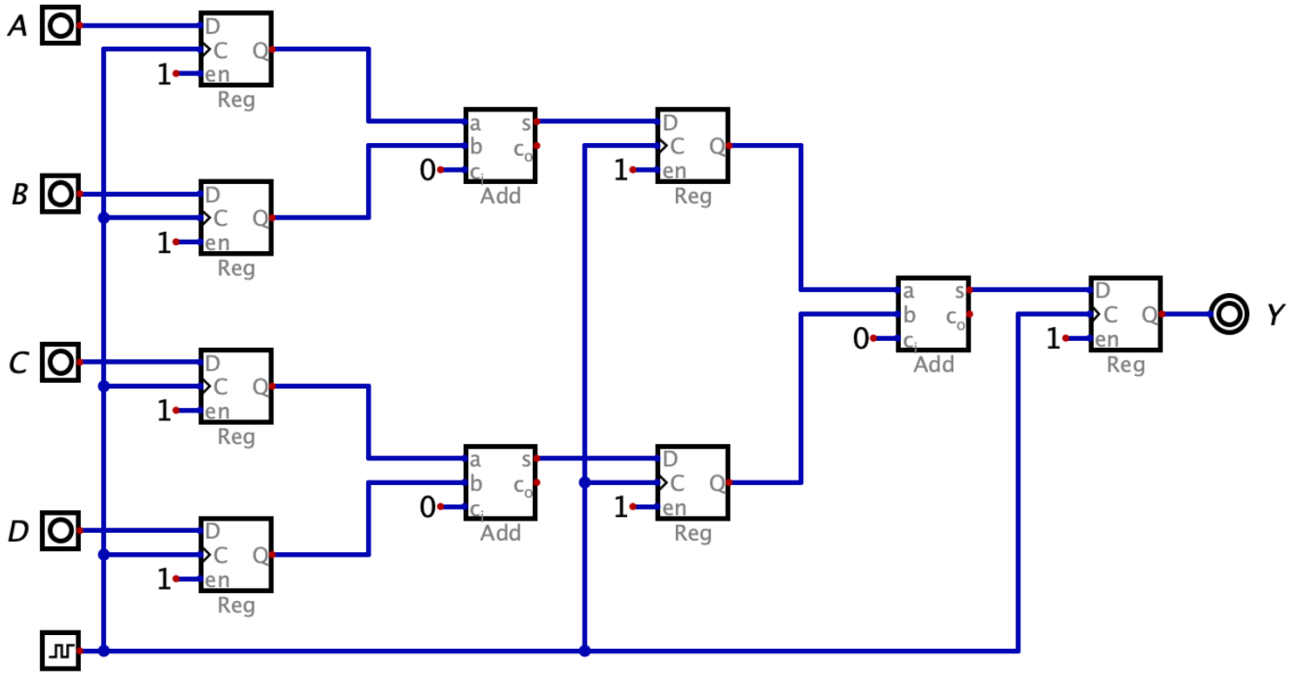
Single Cycle vs Pipelined

Clock-to-q delay = 20ps
Adder propagation delay = 480ps

Single-Cycle Implementation



Pipelined Implementation



	Single-Cycle	Pipelined
Longest combinational path	960ps	480ps
Min clock period	980ps	500ps
# cycles to complete one operation	1	2
Time to complete 10 ops at min clock speed	9800ps	5500ps

Single Cycle



1. What is the longest combinational path in this circuit?

2. What is the min clock period?

3. If we set the clock to the min period, how many clock cycles does it take to perform one operation?

4. If we run this circuit at the minimum clock period, how long will it take to complete 10 operations?

Single Cycle



1. What is the longest combinational path in this circuit?
   - The delay through the two adders
   - 480ps + 480ps = 960ps
2. What is the min clock period?
   - Clock-to-q delay + longest combinational delay
   - 20ps + 960ps = 980ps
3. If we set the clock to the min period, how many clock cycles does it take to perform one operation?
   - 1 cycle
4. If we run this circuit at the minimum clock period, how long will it take to complete 10 operations?
   - Final result is available at t = (980ps)(10) = 9800ps
   - Technically, the result is not available on Y until 9800ps + 20ps = 9820ps, but when we are talking about metrics, we keep it simple and just say 9800ps

Pipelined



1. What is the longest combinational path in this circuit?
2. What is the min clock period?
3. If we set the clock to the min period, how many clock cycles does it take to perform one operation?
4. If we run this circuit at the minimum clock period, how long will it take to complete 10 operations?

Pipelined



1. What is the longest combinational path in this circuit?
   - The longest combinational delay between any two registers is through one adder
   - 480ps
2. What is the min clock period?
   - Clock-to-q delay + longest combinational delay
   - 20ps + 480 = 500ps
3. If we set the clock to the min period, how many clock cycles does it take to perform one operation?
   - 2
4. If we run this circuit at the minimum clock period, how long will it take to complete 10 operations?
   - Final result is available at t = (500ps)(10+1) = 5500ps
   - Technically, the result is not available on Y until 5500ps + 20ps = 5520ps, but when we are talking about metrics, we keep it simple and just say 5500ps

Performance Metrics



- What is the latency of an operation in the single-cycle implementation?
- What is the latency of an operation in the the pipelined implementation?
- Is the throughput higher in the single cycle example or the pipelined example?
- What is the speedup for executing 10 operations?
- If we had performed 20 operations, would the speedup be higher?
  - Note, you shouldn't have to do any math for this

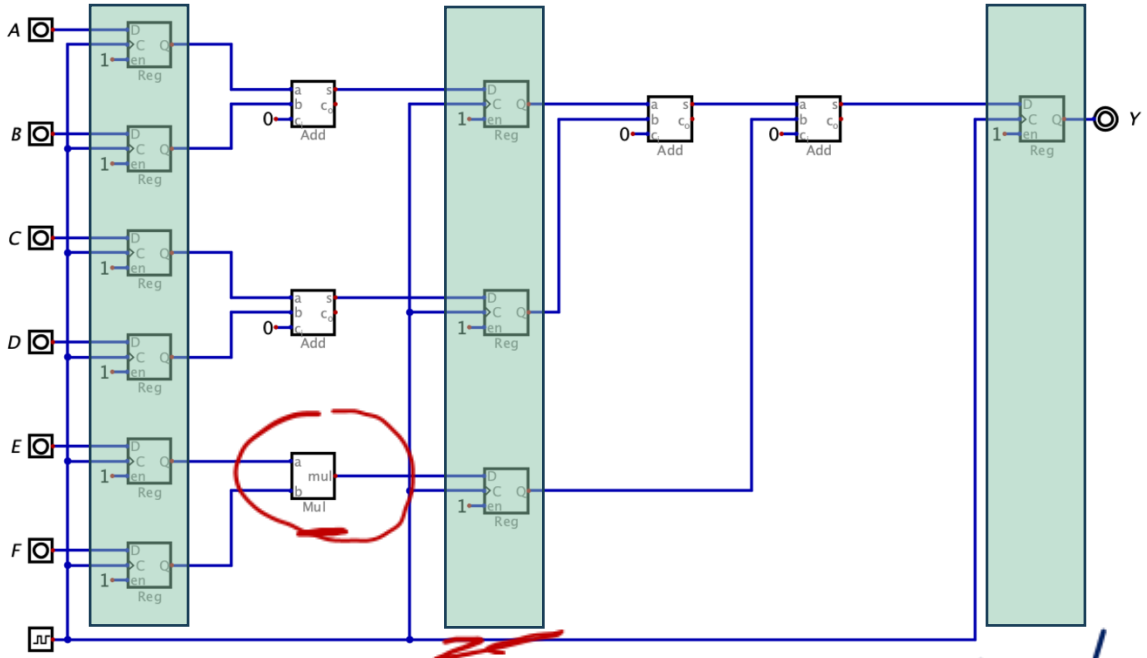
Performance Metrics



- What is the latency of an operation in the single-cycle implementation?
  - One clock cycle = 980ps
- What is the latency of an operation in the pipelined implementation?
  - Two clock cycles = 2(500ps) = 1000ps
- Is the throughput higher in the single cycle example or the pipelined example?
  - Pipelined
- Speedup
  - 9800ps/5500ps = 1.78
- If we had performed 20 operations, would the speedup be higher?
  - Yes

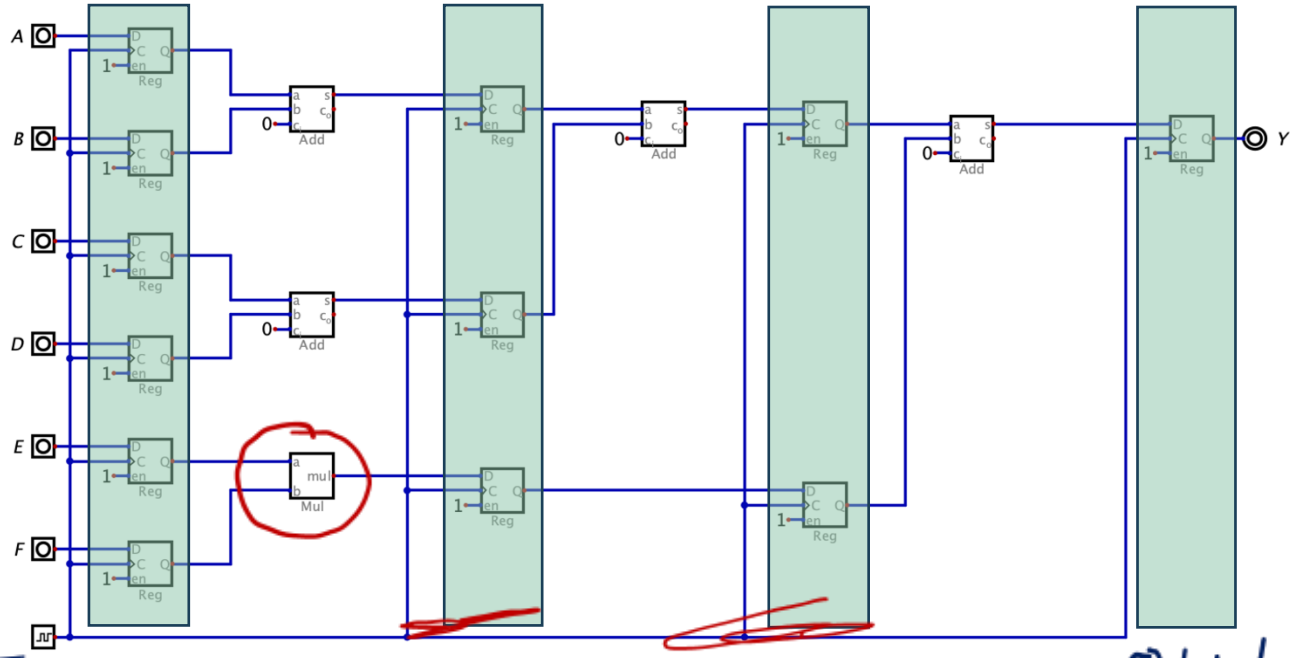
Y = A + B + C + D + EF

- Let's compare two circuits that perform the operation
  Y = A + B + C + D + EF



Stage X Stage Y

2-stage Pipeline



Stage X Stage Y Stage Z

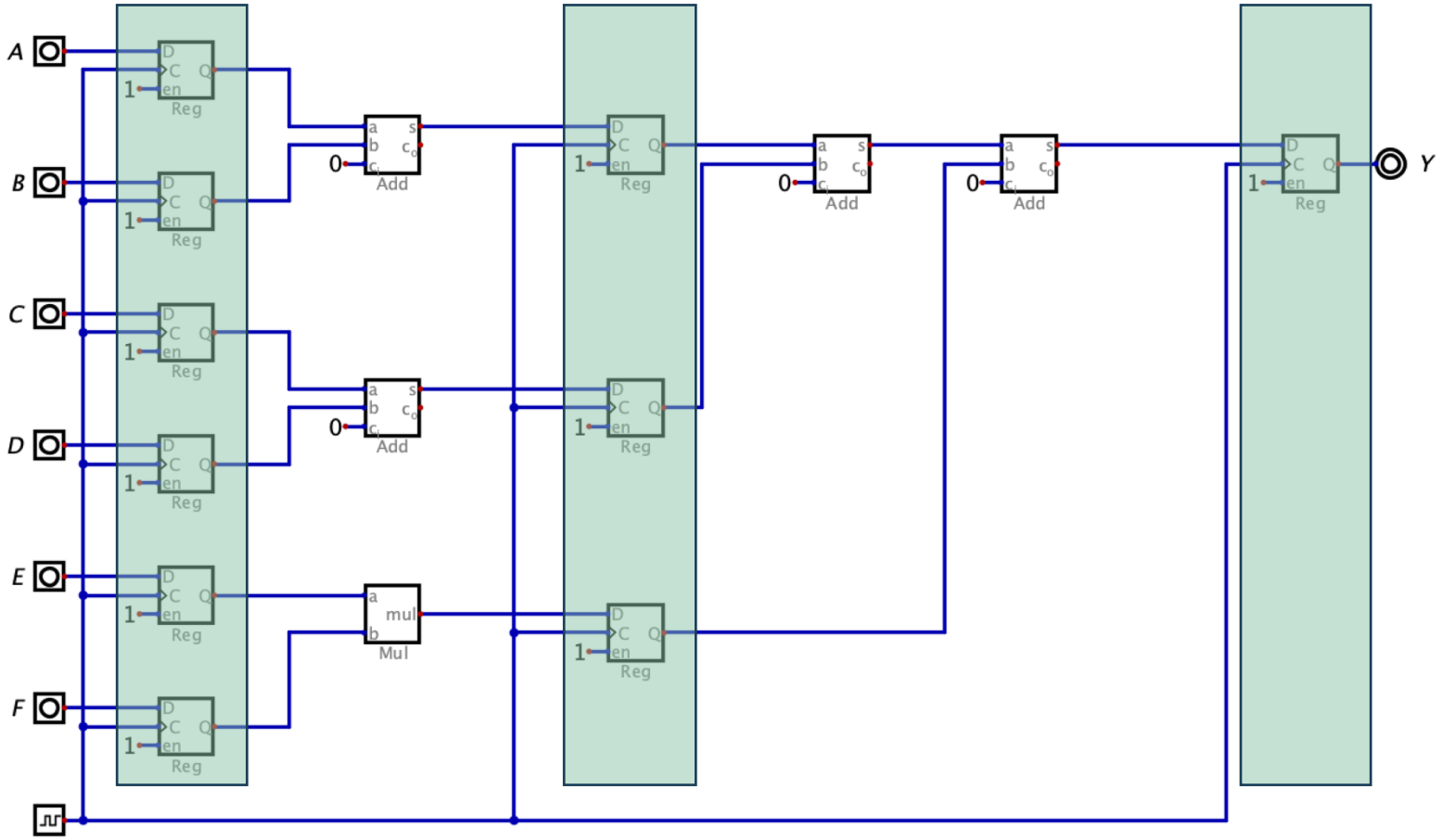
3-stage Pipeline

2-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

Let's say we want to perform the following operation:
Y = 1 + 2 + 3 + 4 + (5)(6)

- In Cycle 0, we will compute

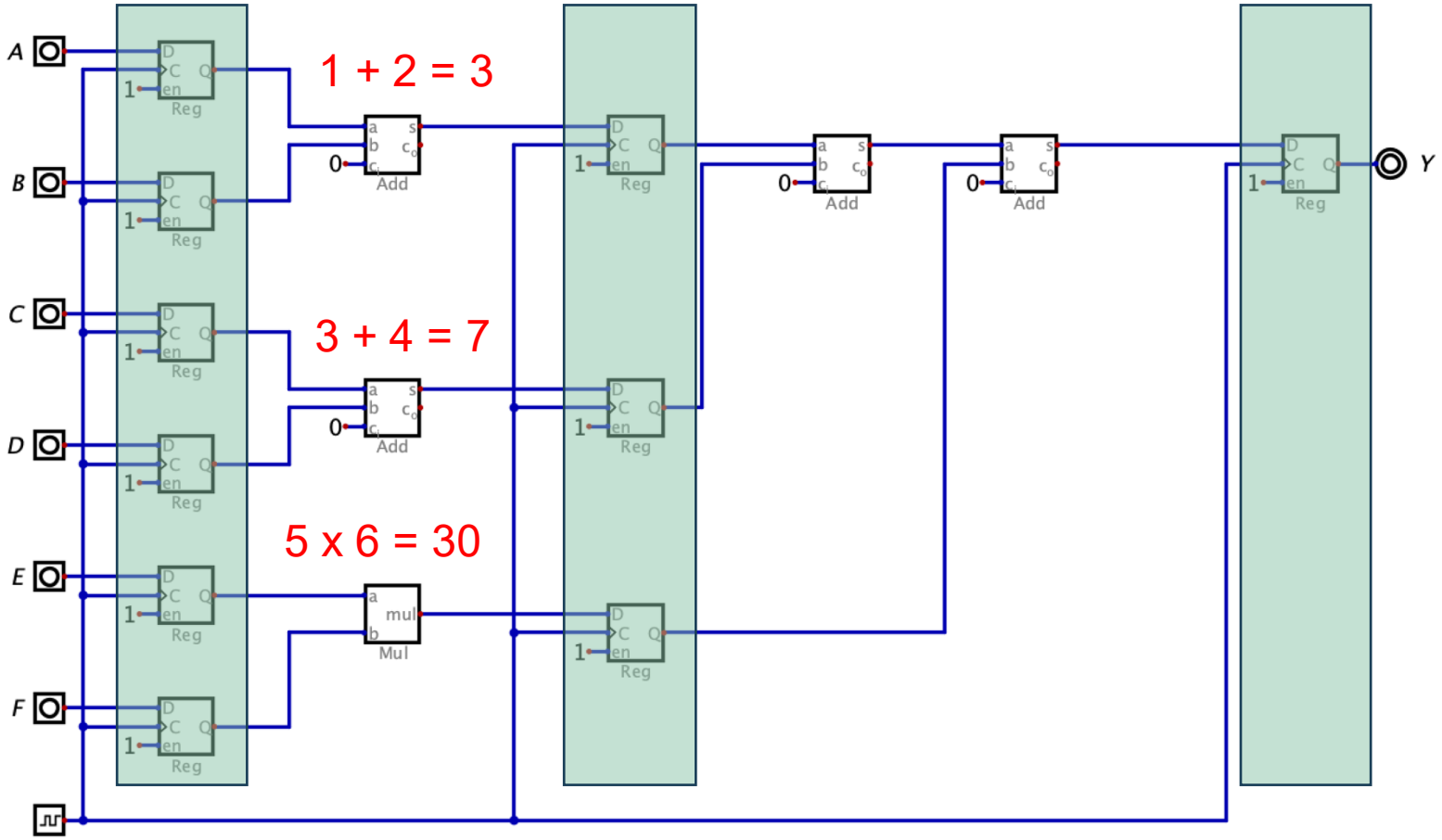


2-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

Let's say we want to perform the following operation:
Y = 1 + 2 + 3 + 4 + (5)(6)

- In Cycle 0, we will compute
  - 1 + 2 = 3
  - 3 + 4 = 7
  - 5 x 6 = 30

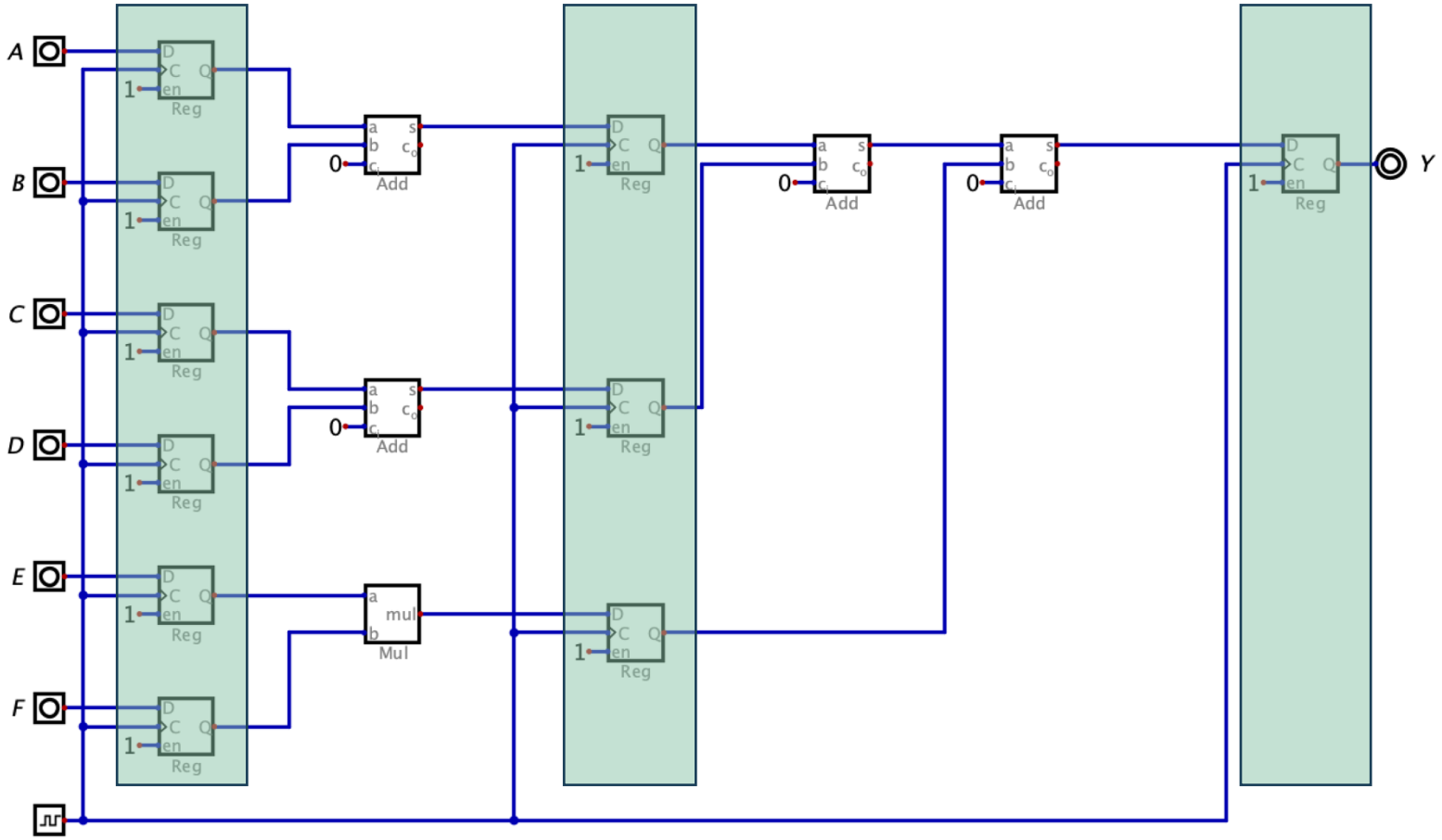


2-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

Let's say we want to perform the following operation:
Y = 1 + 2 + 3 + 4 + (5)(6)

- In Cycle 0, we will compute
  - 1 + 2 = 3
  - 3 + 4 = 7
  - 5 x 6 = 30
- In Cycle 1, we will compute

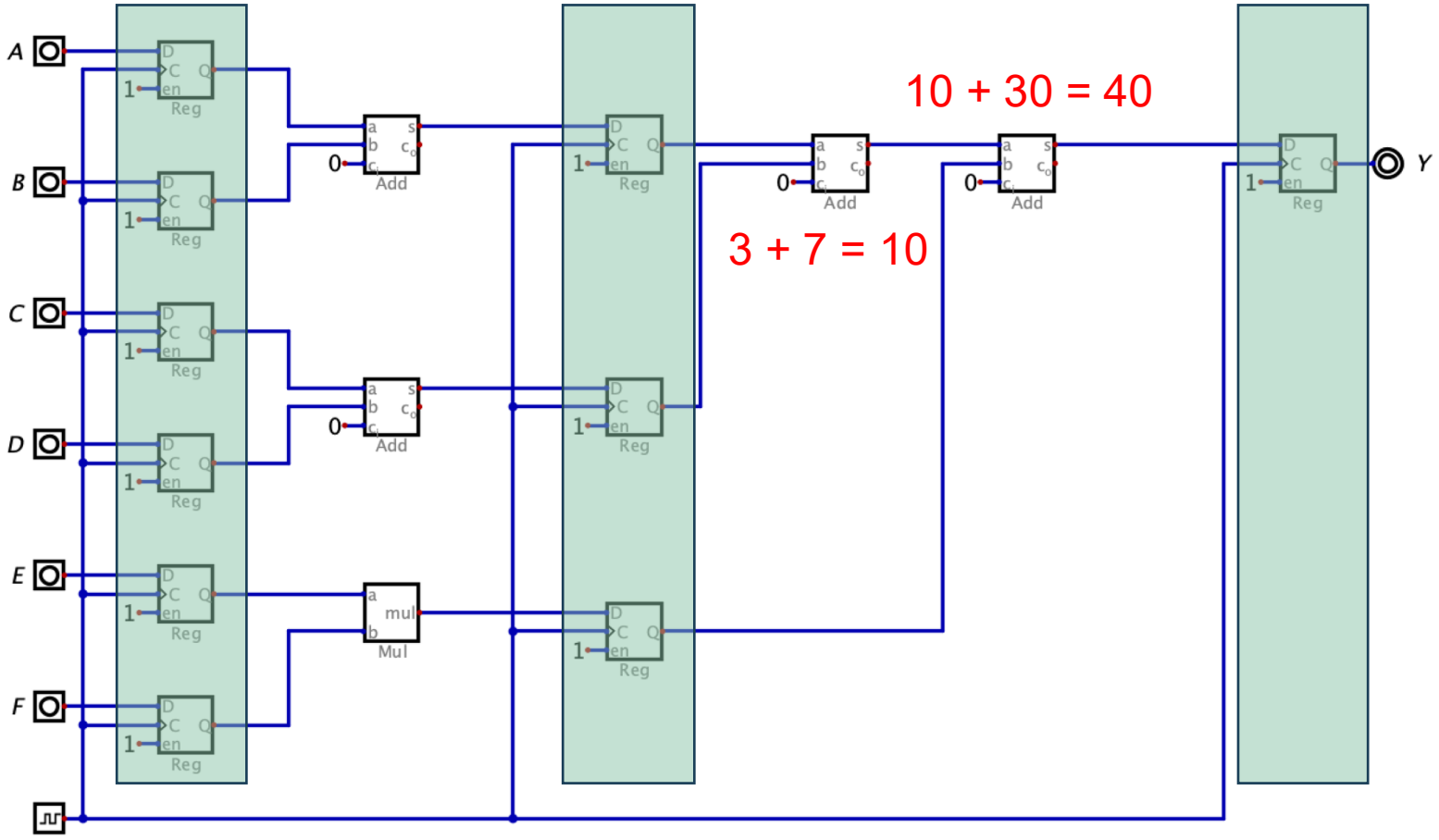


2-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

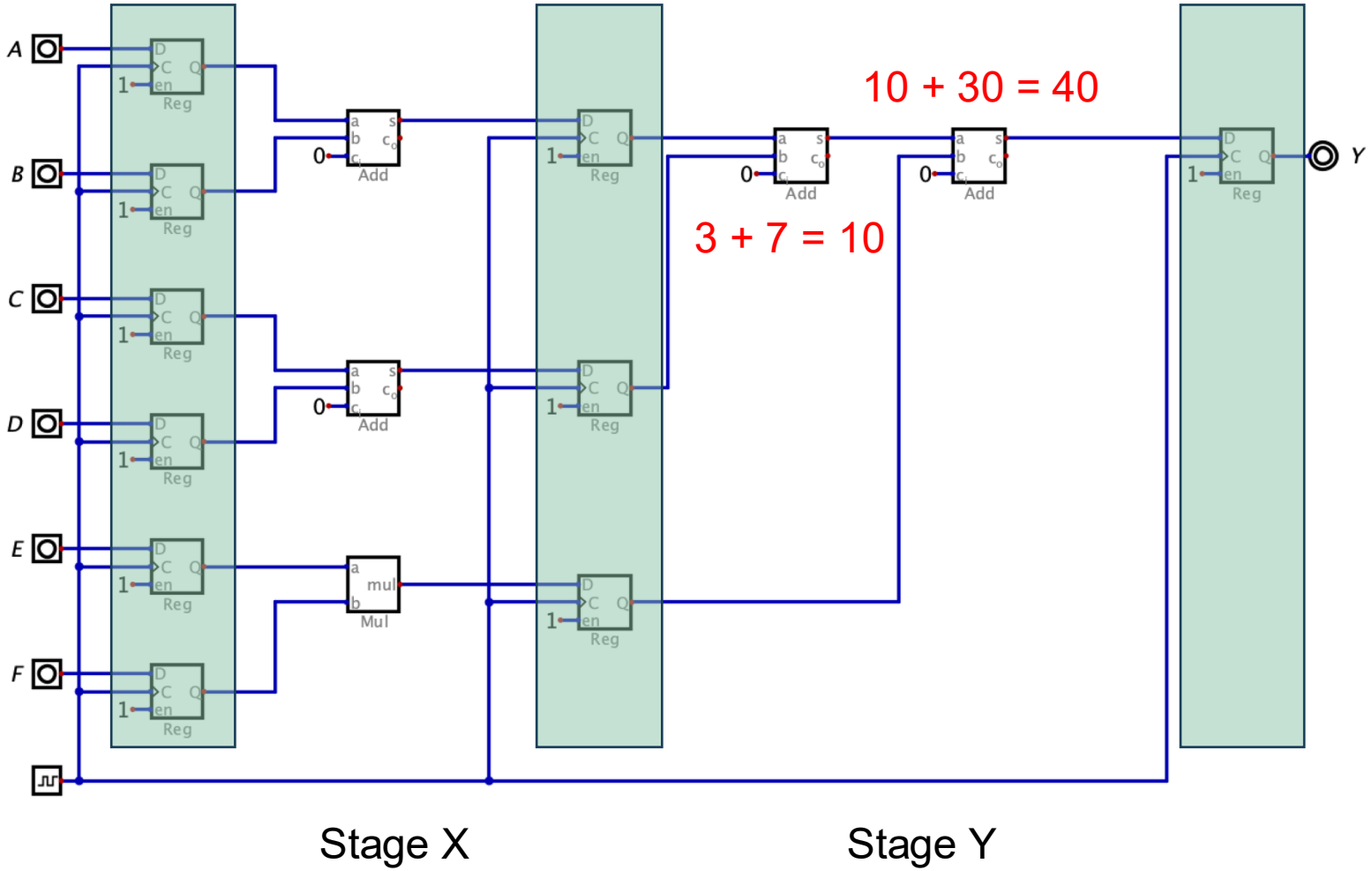
Let's say we want to perform the following operation:
Y = 1 + 2 + 3 + 4 + (5)(6)

- In Cycle 0, we will compute
  - 1 + 2 = 3
  - 3 + 4 = 7
  - 5 x 6 = 30
- In Cycle 1, we will compute
  - 3 + 7 + 30 = 40



2-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps



In Cycle 1, Stage X is empty. We can make better use of our resources and start another operation while the first operation is in Stage Y

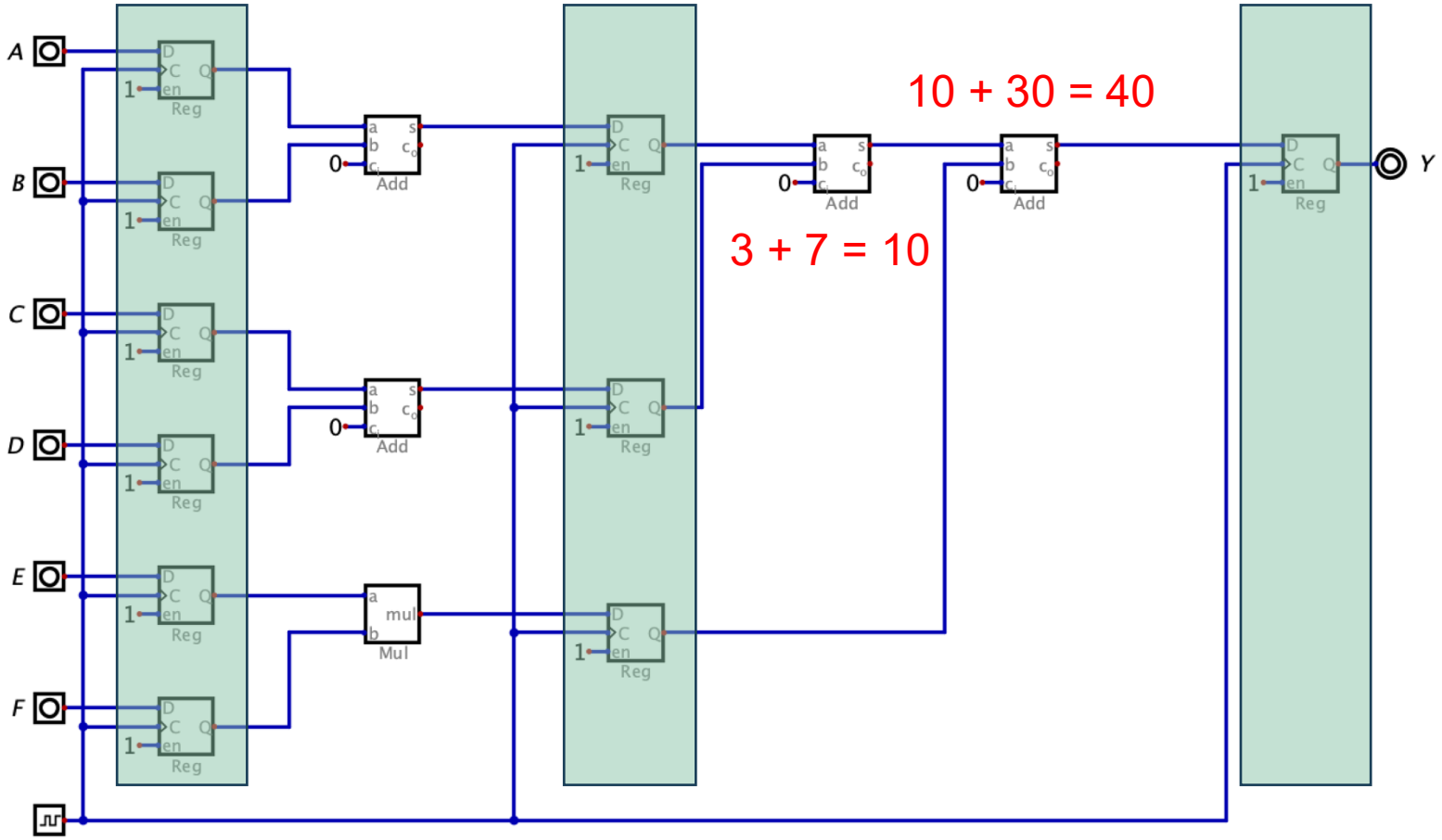
Let's say we also want to compute
Y = 4 + 2 + 7 + 10 + (9)(8)

2-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

Let's say we also want to perform the following operation:
Y = 4 + 2 + 7 + 10 + (9)(8)

- In Cycle 1, we will compute

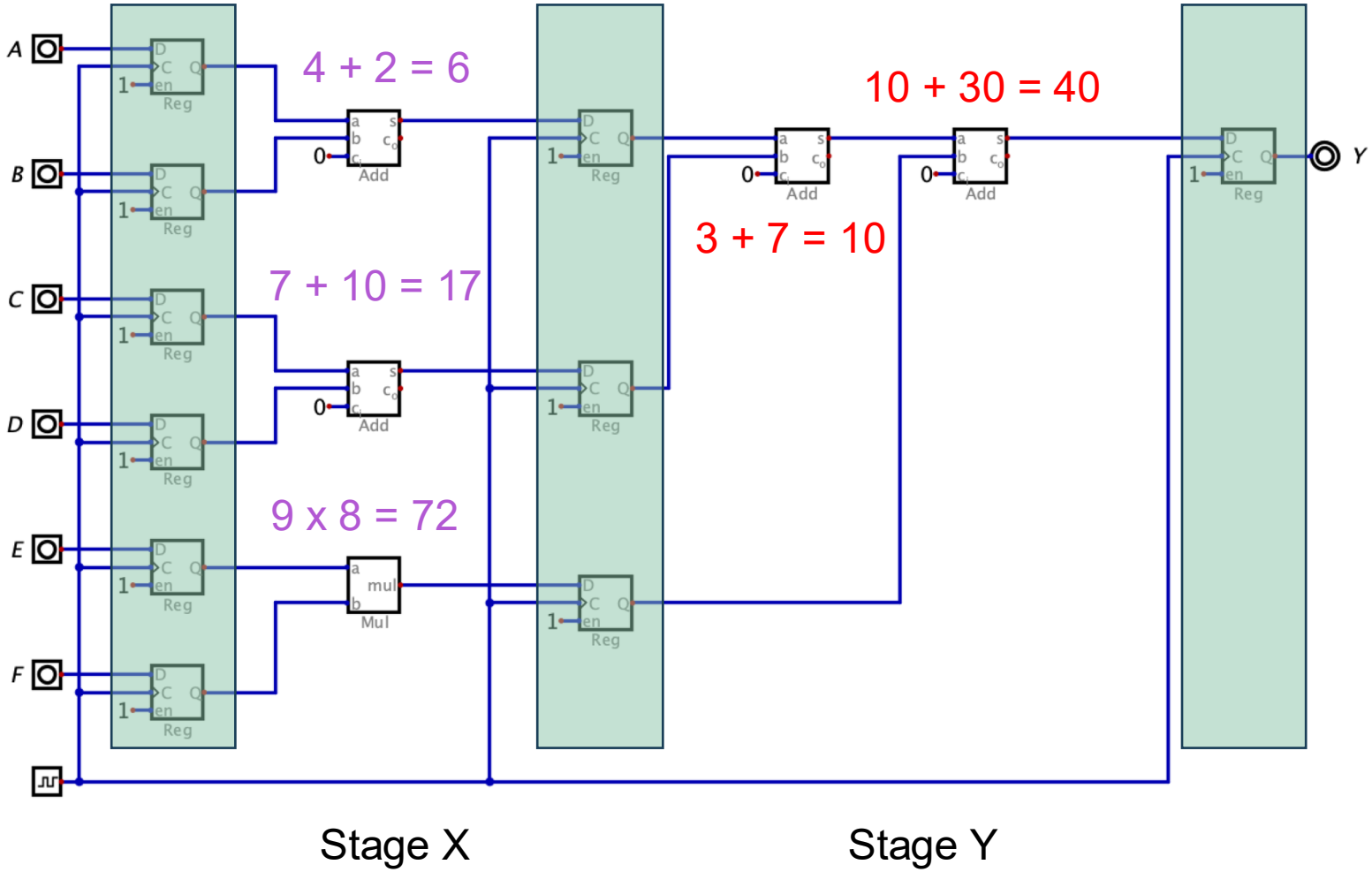


2-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

Let's say we also want to perform the following operation:
Y = 4 + 2 + 7 + 10 + (9)(8)

- In Cycle 1, we will compute
  - 4 + 2 = 6
  - 7 + 10 = 17
  - 9 x 8 = 72

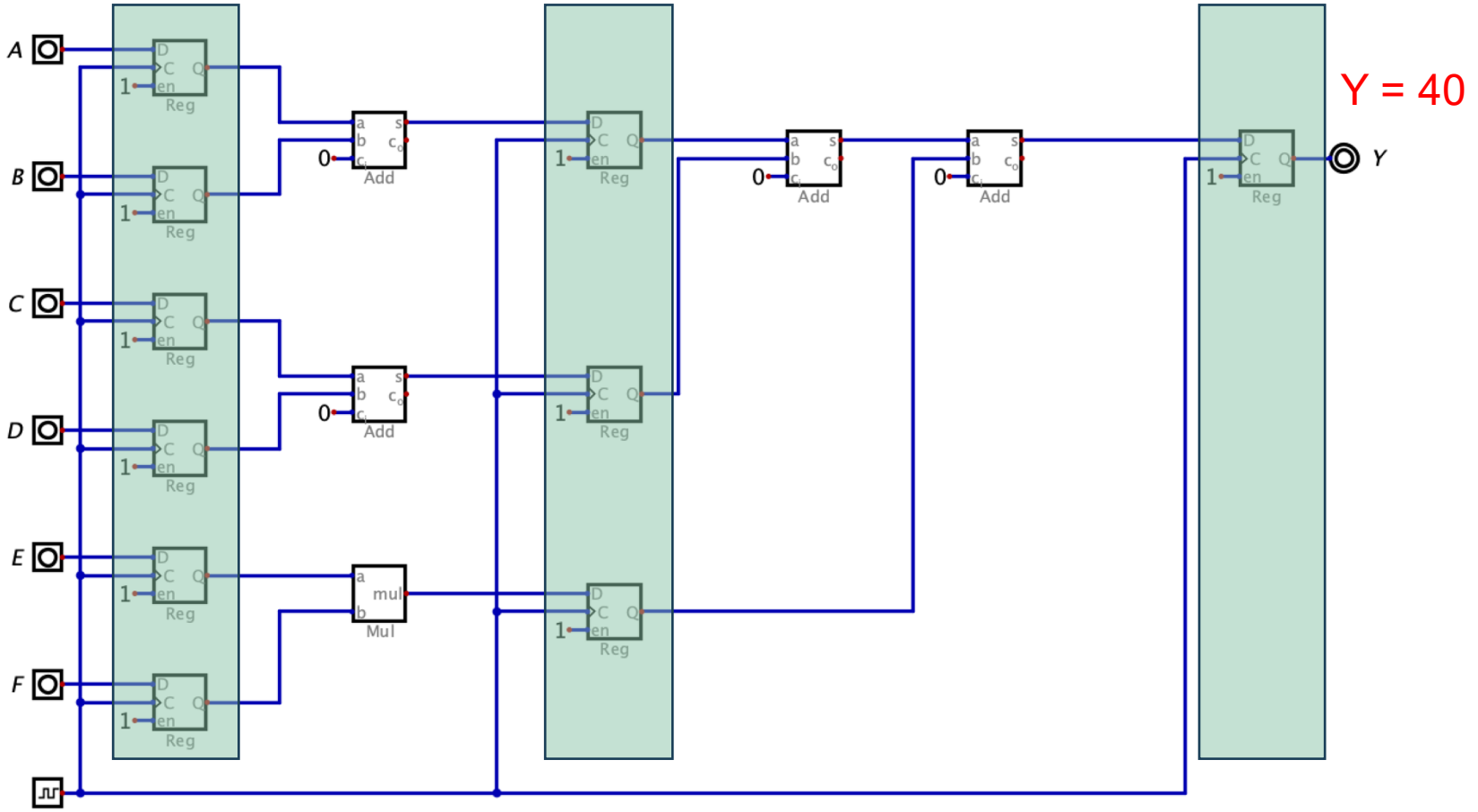


2-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

Let's say we also want to perform the following operation:
Y = 4 + 2 + 7 + 10 + (9)(8)

- In Cycle 1, we will compute
  - 4 + 2 = 6
  - 7 + 10 = 17
  - 9 x 8 = 72
- In Cycle 2, we will compute

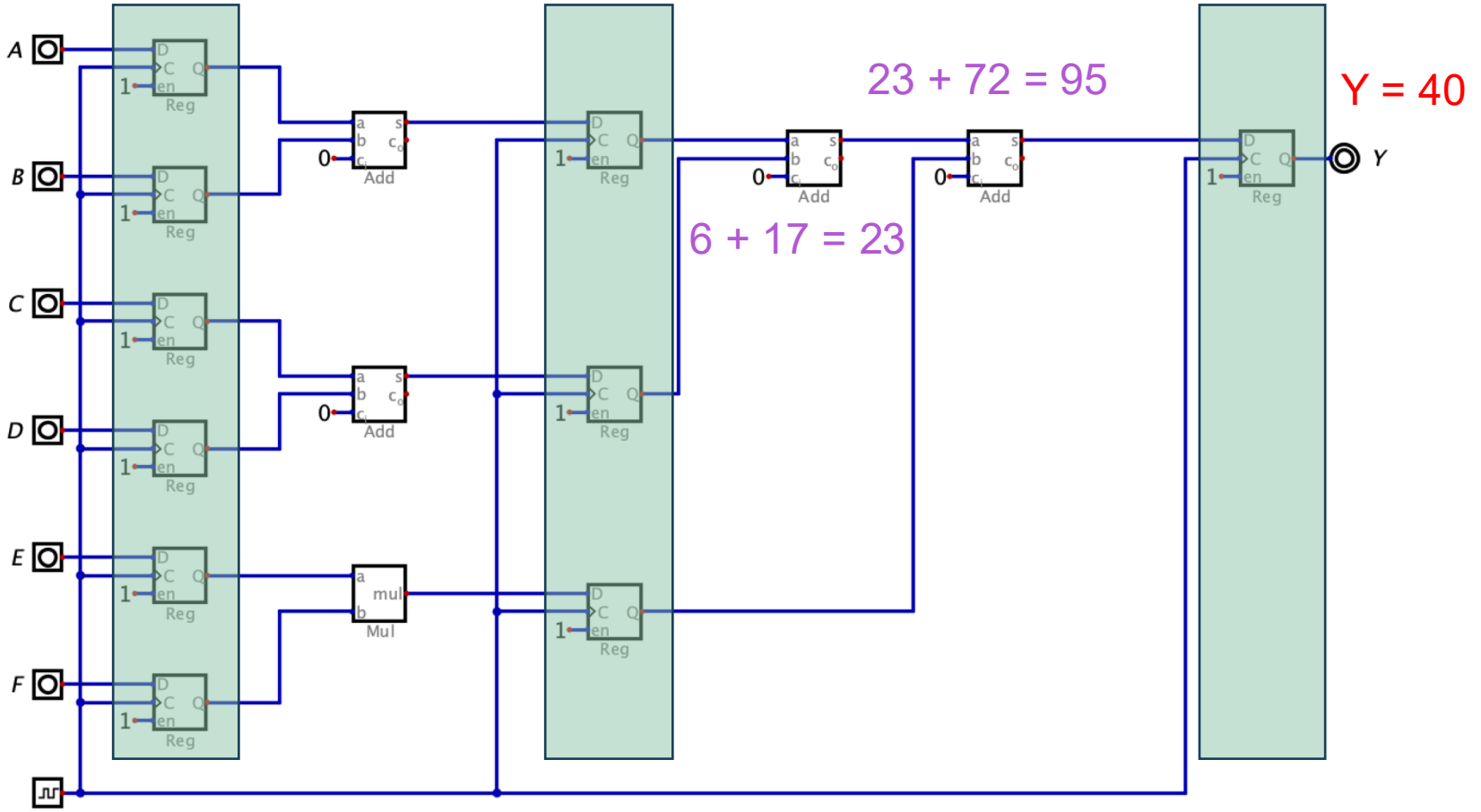


2-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

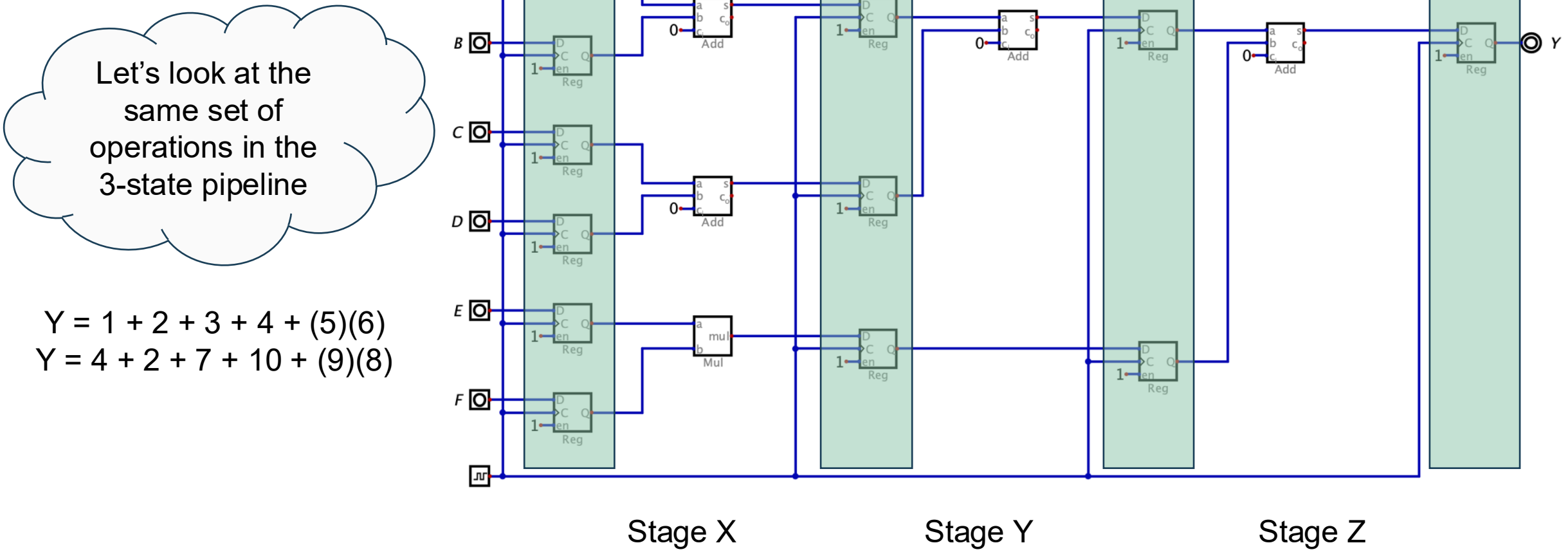
Let's say we also want to perform the following operation:
Y = 4 + 2 + 7 + 10 + (9)(8)

- In Cycle 1, we will compute
  - 4 + 2 = 6
  - 7 + 10 = 17
  - 9 x 8 = 72
- In Cycle 2, we will compute
  - 6 + 17 + 72 = 95



3-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

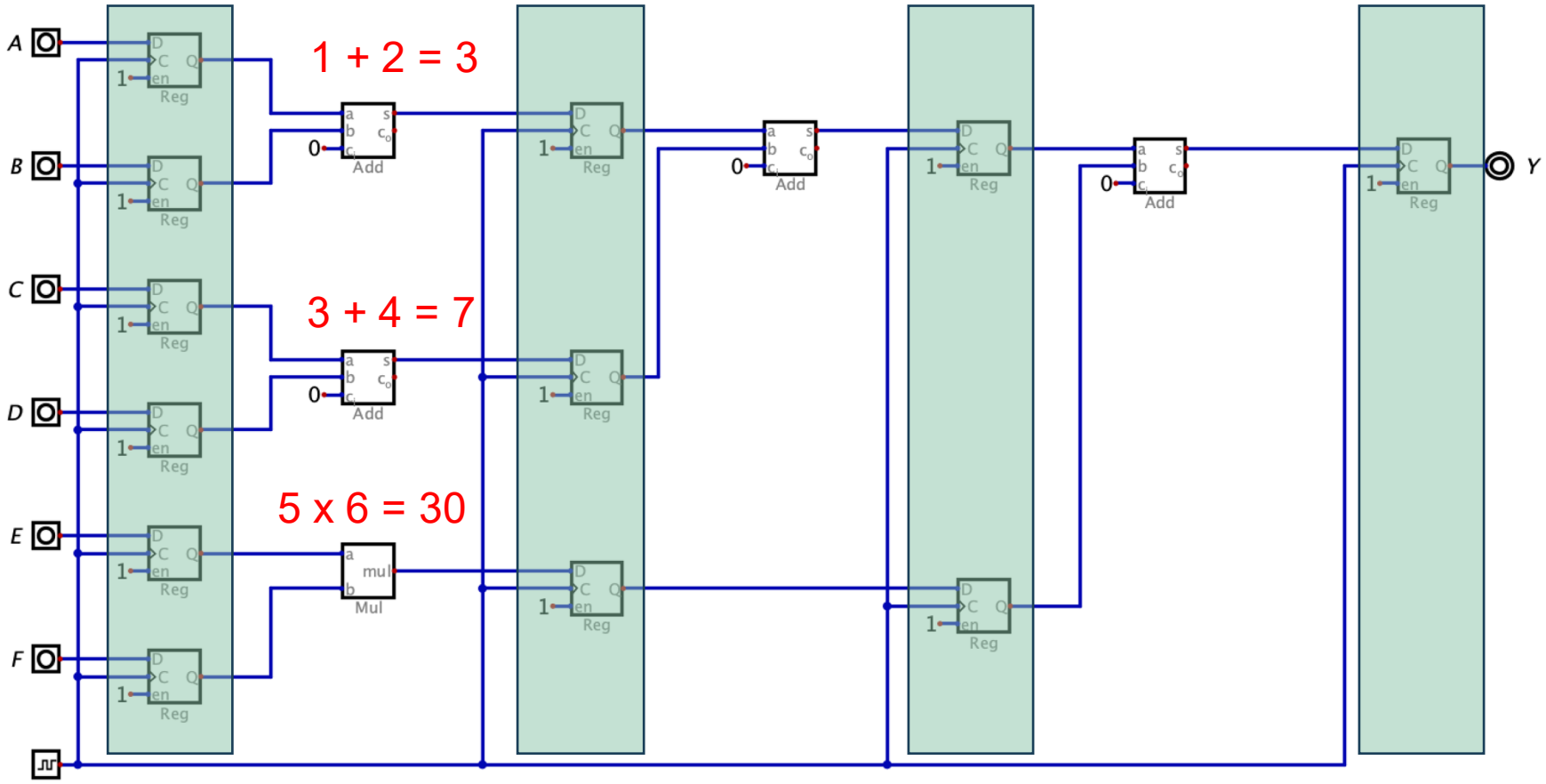


3-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

Op0: Y = 1 + 2 + 3 + 4 + (5)(6)
Op1: Y = 4 + 2 + 7 + 10 + (9)(8)

- In Cycle 0, we will compute
  - 1 + 2 = 3
  - 3 + 4 = 7
  - 5 x 6 = 30

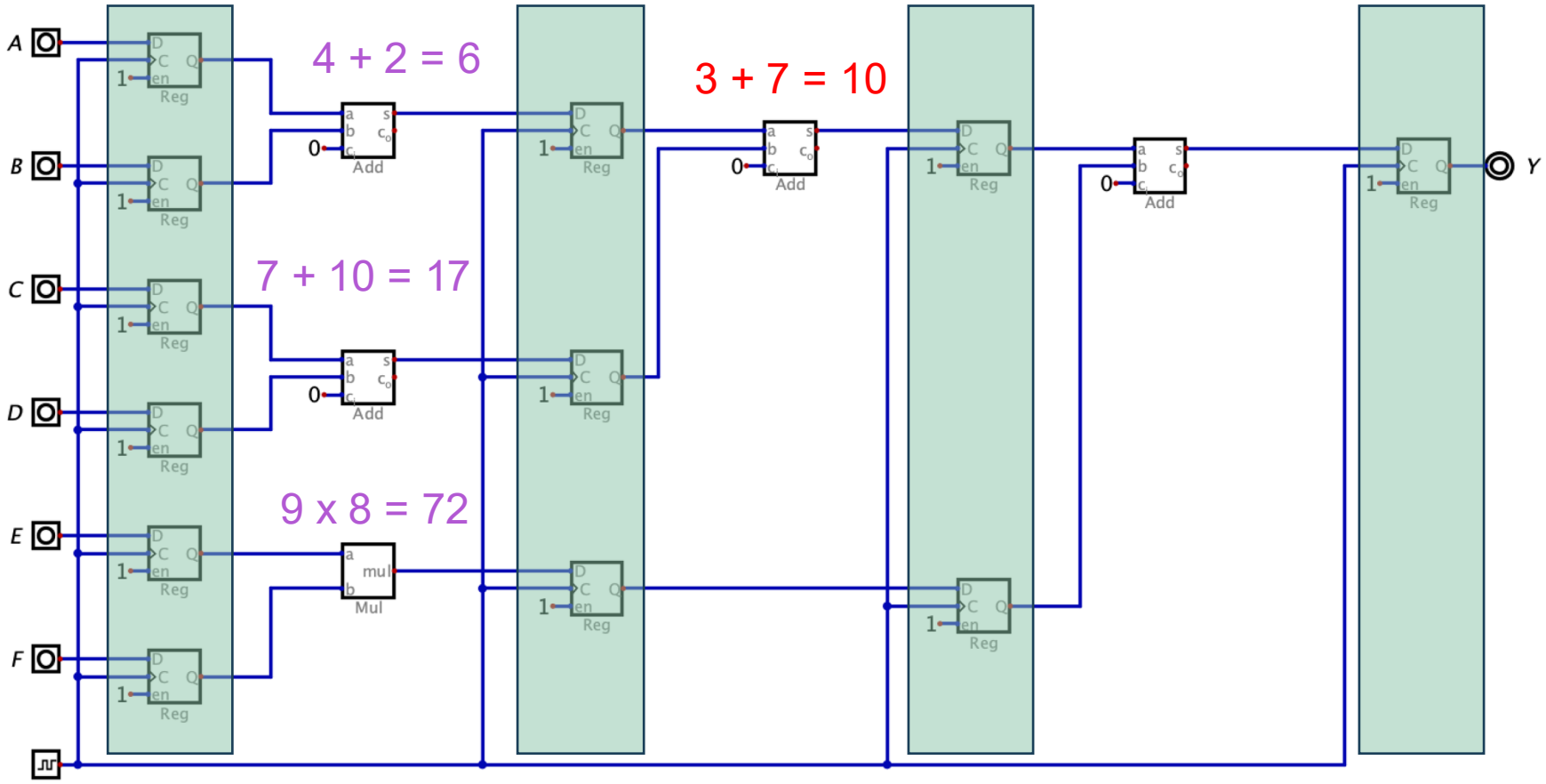


3-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

Op0: Y = 1 + 2 + 3 + 4 + (5)(6)
Op1: Y = 4 + 2 + 7 + 10 + (9)(8)

- In Cycle 0, we will compute
  - 1 + 2 = 3
  - 3 + 4 = 7
  - 5 x 6 = 30
- In Cycle 1, we will compute
  - 3 + 7 = 10
  - 4 + 2 = 6
  - 7 + 10 = 17
  - 9 x 8 = 72

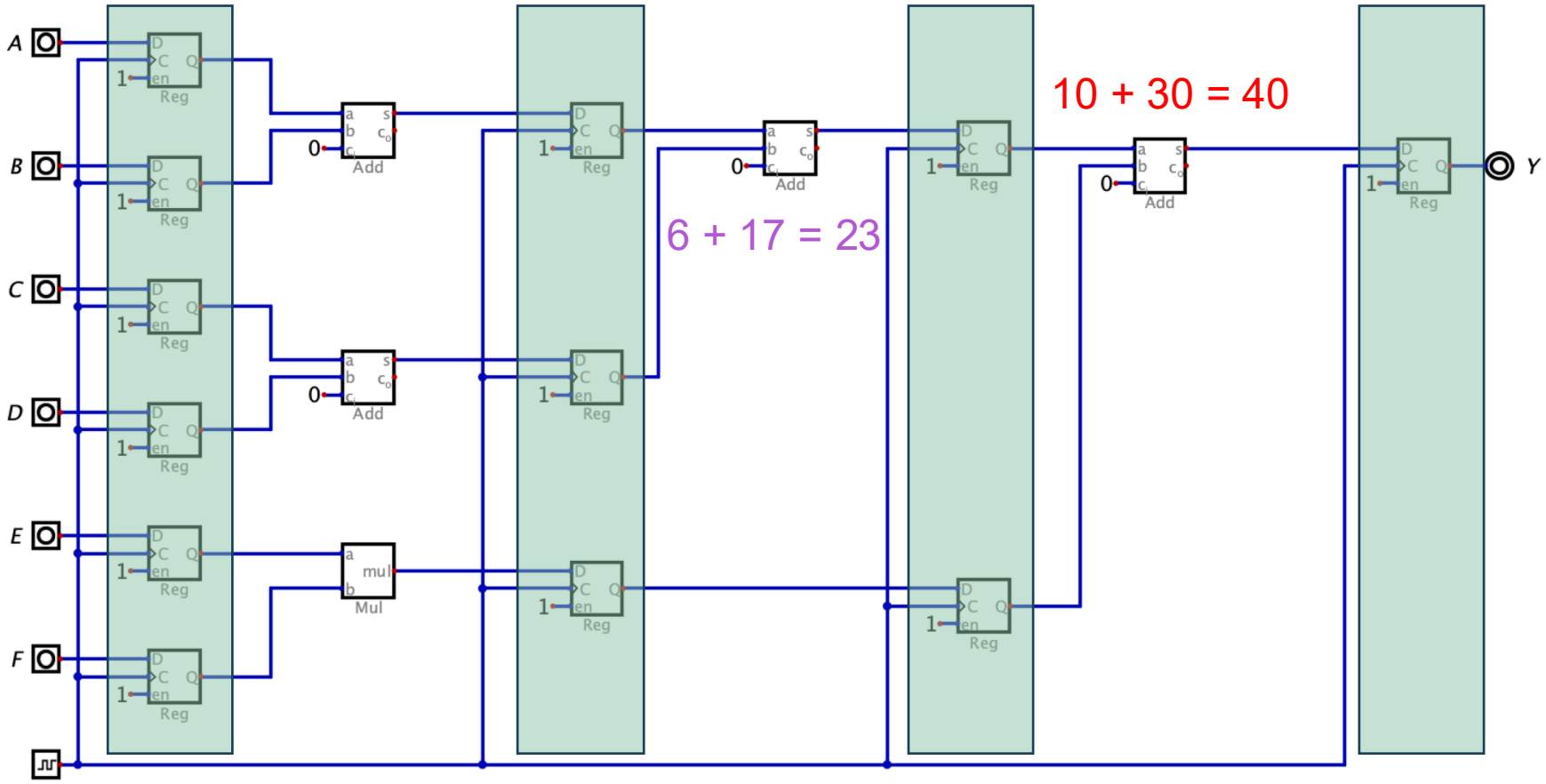


3-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

Op0: Y = 1 + 2 + 3 + 4 + (5)(6)
Op1: Y = 4 + 2 + 7 + 10 + (9)(8)

- In Cycle 0, we will compute
  - 1 + 2 = 3
  - 3 + 4 = 7
  - 5 x 6 = 30
- In Cycle 1, we will compute
  - 3 + 7 = 10
  - 4 + 2 = 6
  - 7 + 10 = 17
  - 9 x 8 = 72
- In Cycle 2, we will compute
  - 10 + 30 = 40
  - 6 + 17 = 23

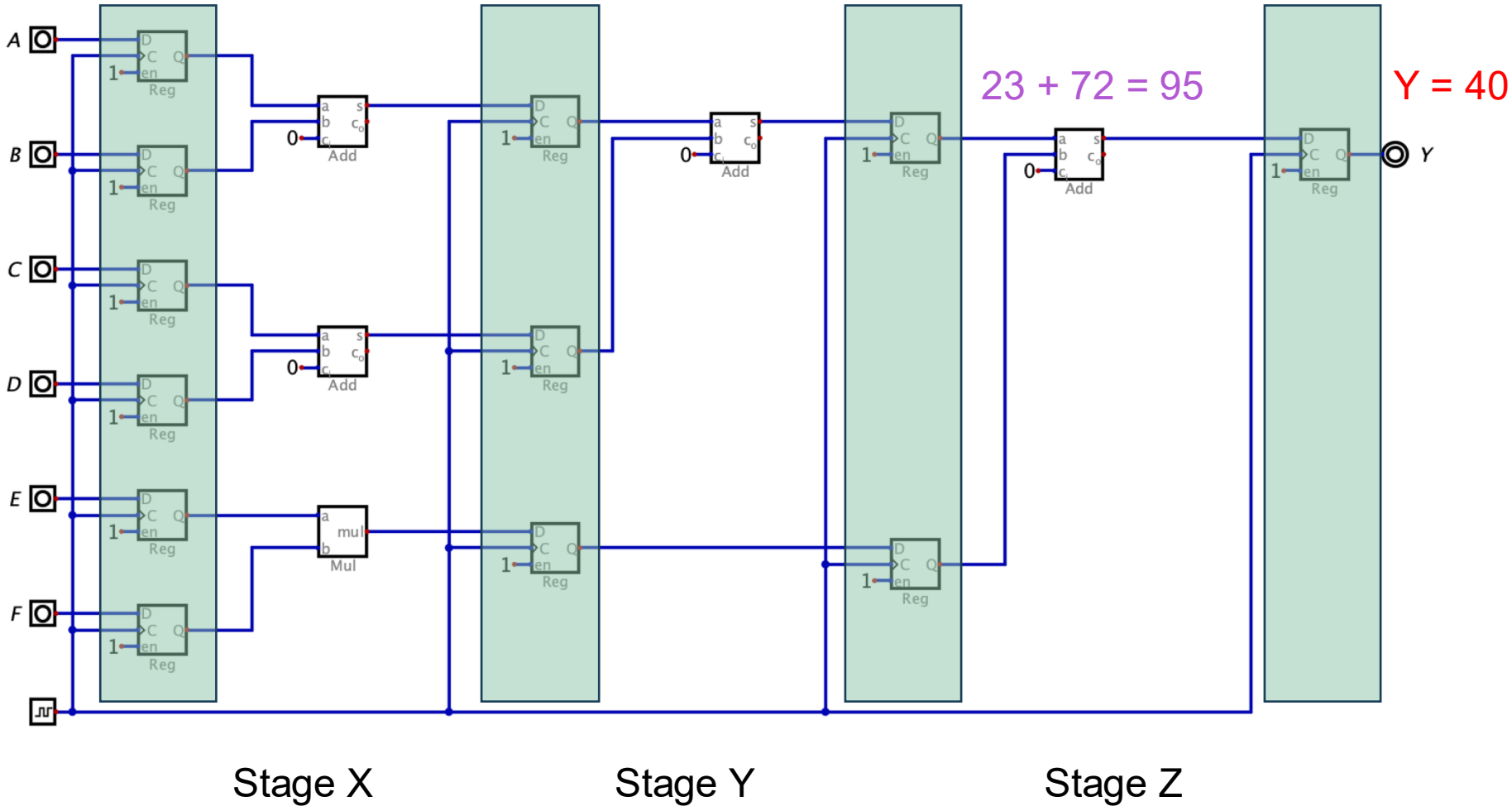


3-Stage Pipeline

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

Op0: Y = 1 + 2 + 3 + 4 + (5)(6)
Op1: Y = 4 + 2 + 7 + 10 + (9)(8)

- In Cycle 0, we will compute
  - 1 + 2 = 3
  - 3 + 4 = 7
  - 5 x 6 = 30
- In Cycle 1, we will compute
  - 3 + 7 = 10
  - 4 + 2 = 6
  - 7 + 10 = 17
  - 9 x 8 = 72
- In Cycle 2, we will compute
  - 10 + 30 = 40
  - 6 + 17 = 23
- In Cycle 3, we will compute
  - 23 + 72 = 95



2-stage Pipeline Diagram

Example Operations

Operation 0: 1 + 2 + 3 + 4 + (5)(6)
Operation 1: 4 + 2 + 7 + 10 + (9)(8)



Cycle #

	0	1	2	3	4	5	6	7	8	9	10	11	12
Op0	X	Y											
Op1													
Op2													
Op3													
Op4													
Op5													
Op6													
Op7													
Op8													
Op9													

2-stage Pipeline Diagram

Example Operations

Operation 0: 1 + 2 + 3 + 4 + (5)(6)
Operation 1: 4 + 2 + 7 + 10 + (9)(8)



Cycle #	0	1	2	3	4	5	6	7	8	9	10	11	12
Op0	X	Y											
Op1		X	Y										
Op2			X	Y									
Op3				X	Y								
Op4					X	Y							
Op5						X	Y						
Op6							X	Y					
Op7								X	Y				
Op8									X	Y			
Op9										X	Y		

3-stage Pipeline Diagram

Example Operations

Operation 0: 1 + 2 + 3 + 4 + (5)(6)
Operation 1: 4 + 2 + 7 + 10 + (9)(8)



Cycle #	0	1	2	3	4	5	6	7	8	9	10	11	12
Op0	X	Y	Z										
Op1		X	y	z									
Op2			x	y	z								
Op3				x	y	z							
Op4													
Op5													
Op6													
Op7													
Op8													
Op9													

3-stage Pipeline Diagram

Example Operations

Operation 0: 1 + 2 + 3 + 4 + (5)(6)
Operation 1: 4 + 2 + 7 + 10 + (9)(8)



Cycle #	0	1	2	3	4	5	6	7	8	9	10	11	12
Op0	X	Y	Z										
Op1		X	Y	Z									
Op2			X	Y	Z								
Op3				X	Y	Z							
Op4					X	Y	Z						
Op5						X	Y	Z					
Op6							X	Y	Z				
Op7								X	Y	Z			
Op8									X	Y	Z		
Op9										X	Y	Z	

For each circuit…

Clock-to-q delay = 20ps
Adder propagation delay = 480ps
Multiplier propagation delay = 1200ps

1. What is the longest combinational path?

1200 ; 1200

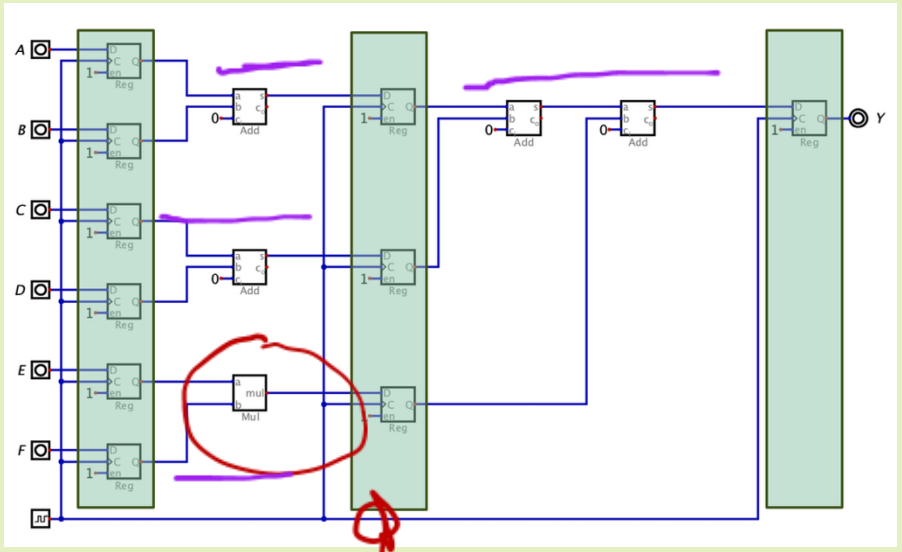
2. What is the min clock period (critical path)?

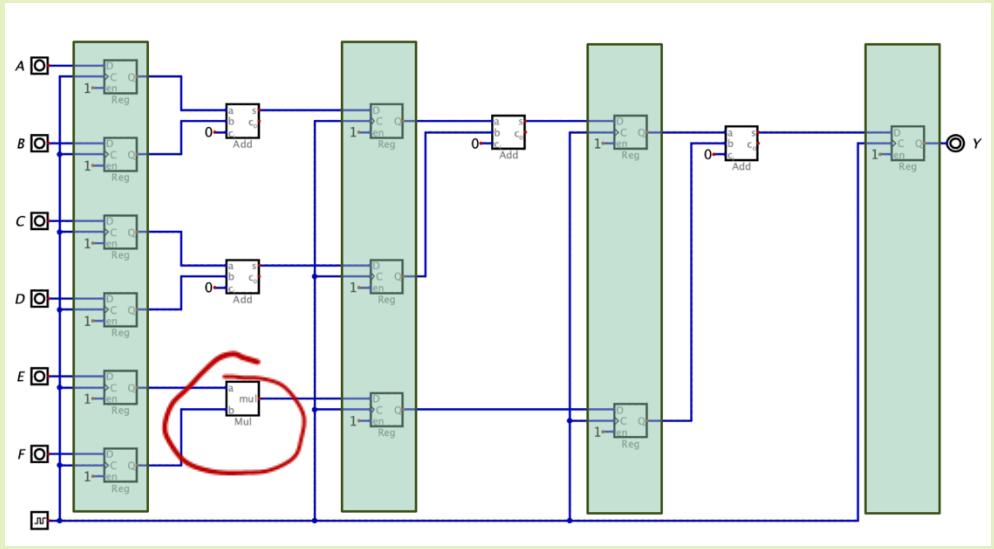
1220 ; 1220

3. If the clock period is at least the min period, how many clock cycles does it take to perform one operation?

2 ; 3

4. If we run this circuit at the minimum clock period, how long will it take to complete 10 operations?





Two Stage Pipeline



1. What is the longest combinational path?
   - Multiplier
   - 1200ps
2. What is the min clock period?
   - Clock-to-q delay + longest combinational delay
   - 20ps + 1200ps = 1220ps
3. If the clock period is at least the min period, how many clock cycles does it take to perform one operation?
   - two
4. If we run this circuit at the minimum clock period, how long will it take to complete 10 operations?
   - Final result is available at t = (1220ps)(10+1) = 13420ps

Three Stage Pipeline



1. What is the longest combinational path?
   - Multiplier
   - 1200ps
2. What is the min clock period?
   - Clock-to-q delay + longest combinational delay
   - 20ps + 1200ps = 1220ps
3. If the clock period is at least the min period, how many clock cycles does it take to perform one operation?
   - three
4. If we run this circuit at the minimum clock period, how long will it take to complete 10 operations?
   - Final result is available at t = (1220ps)(10+2) = 14640ps

Performance Metrics



- What is the latency of an operation in the 2-stage pipeline?
- What is the latency of an operation in the 3-stage pipeline?
- Is the throughput higher in the 2-stage or 3-stage pipeline?
- What is the speedup of completing 10 operations on the 2-stage pipeline vs 3-stage pipeline?

Performance Metrics

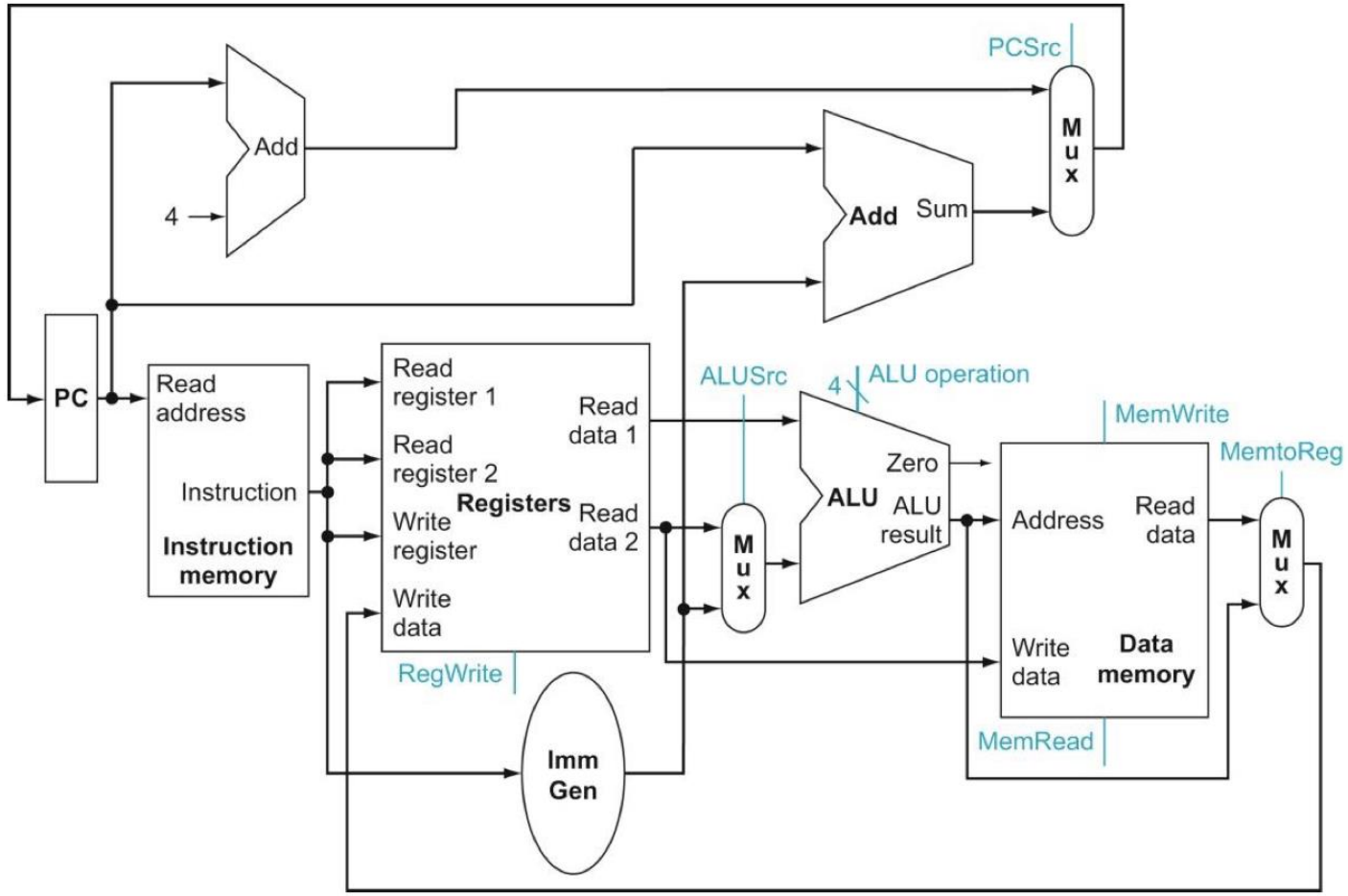


- What is the latency of an operation in the 2-stage pipeline?
  - (1220ps)(2) = 2440ps
- What is the latency of an operation in the 3-stage pipeline?
  - (1220ps)(3) = 3660ps
- Is the throughput higher in the 2-stage or 3-stage pipeline?
  - same **steady-state throughput.** The two-stage version has lower latency and finishes a finite batch sooner.
- What is the speedup of completing 10 operations on the 2-stage pipeline vs 3-stage pipeline?
  - 13420ps/14640ps = .9167. The two-stage pipeline is faster

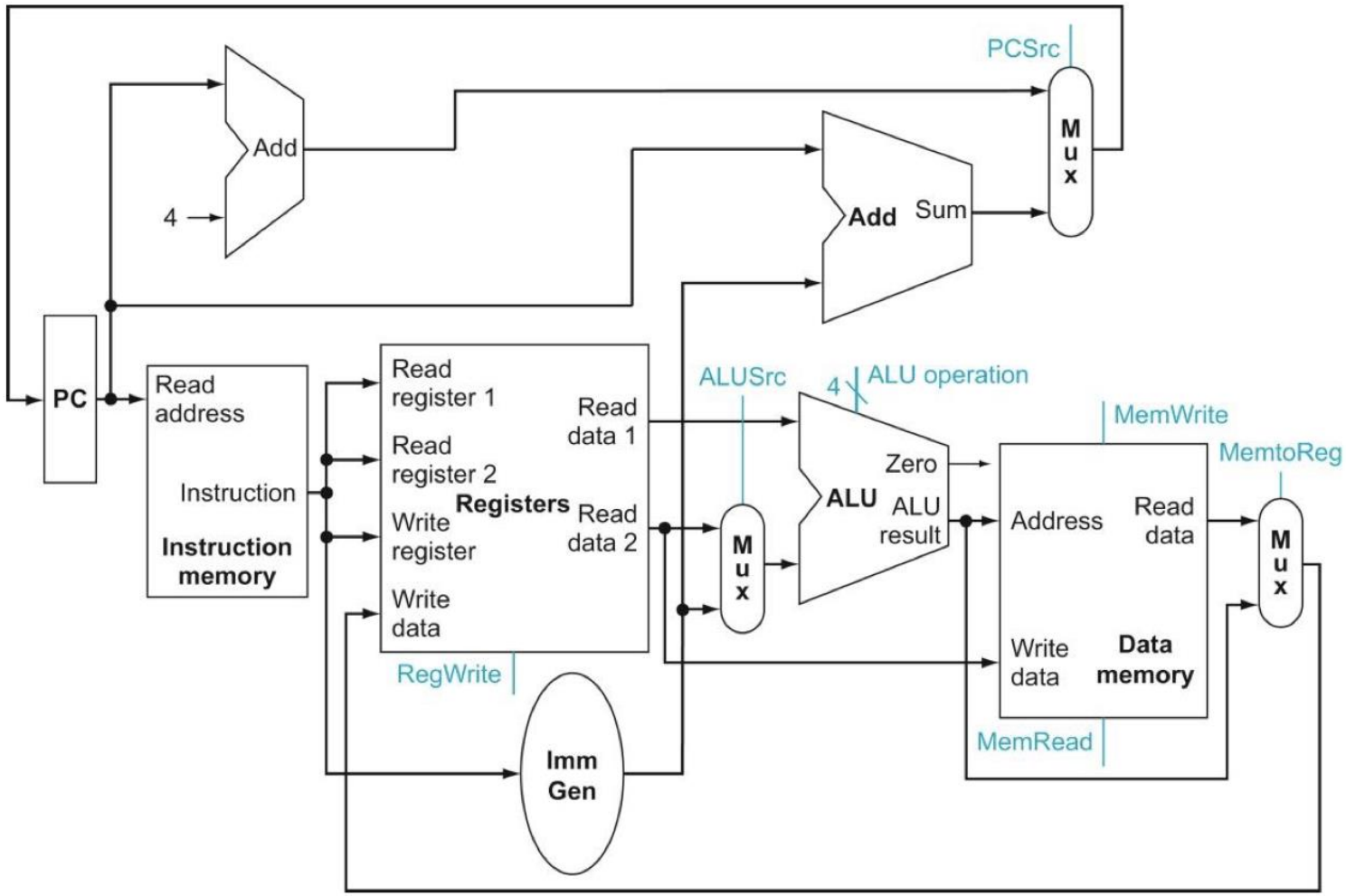


Review: The RISC-V Datapath

The RISC-V Datapath



ADD | Start with known state



add x5, x1, x2

add x5, x1, x2

PC = 0x1000

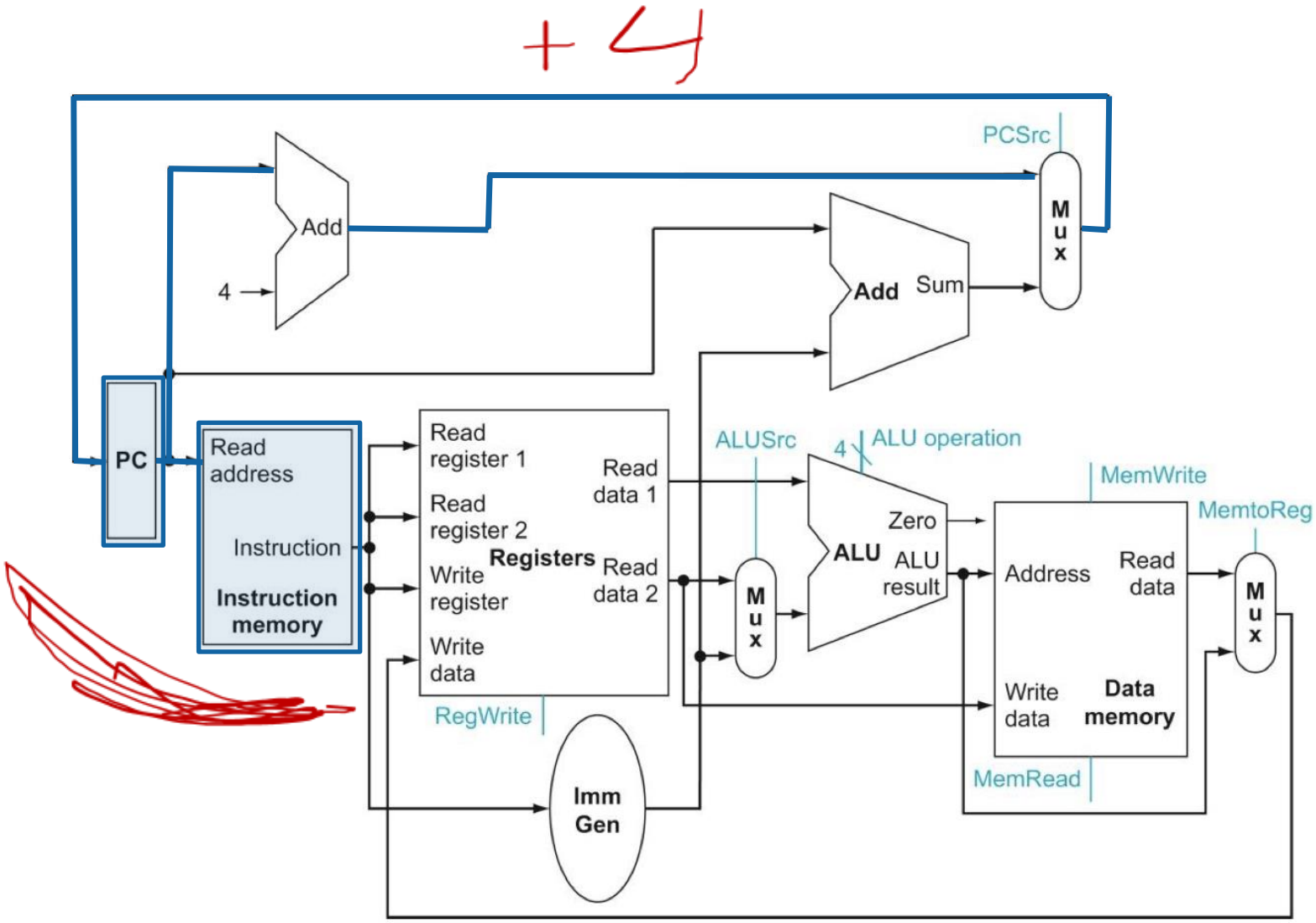
x1 = 7; x2 = 11

Predict: what changes?

Expected result: x5 = 18

Single-cycle datapath • steps show signal flow, not separate clock cycles.

ADD | 1 Fetch



add x5, x1, x2

PC addresses instruction memory.

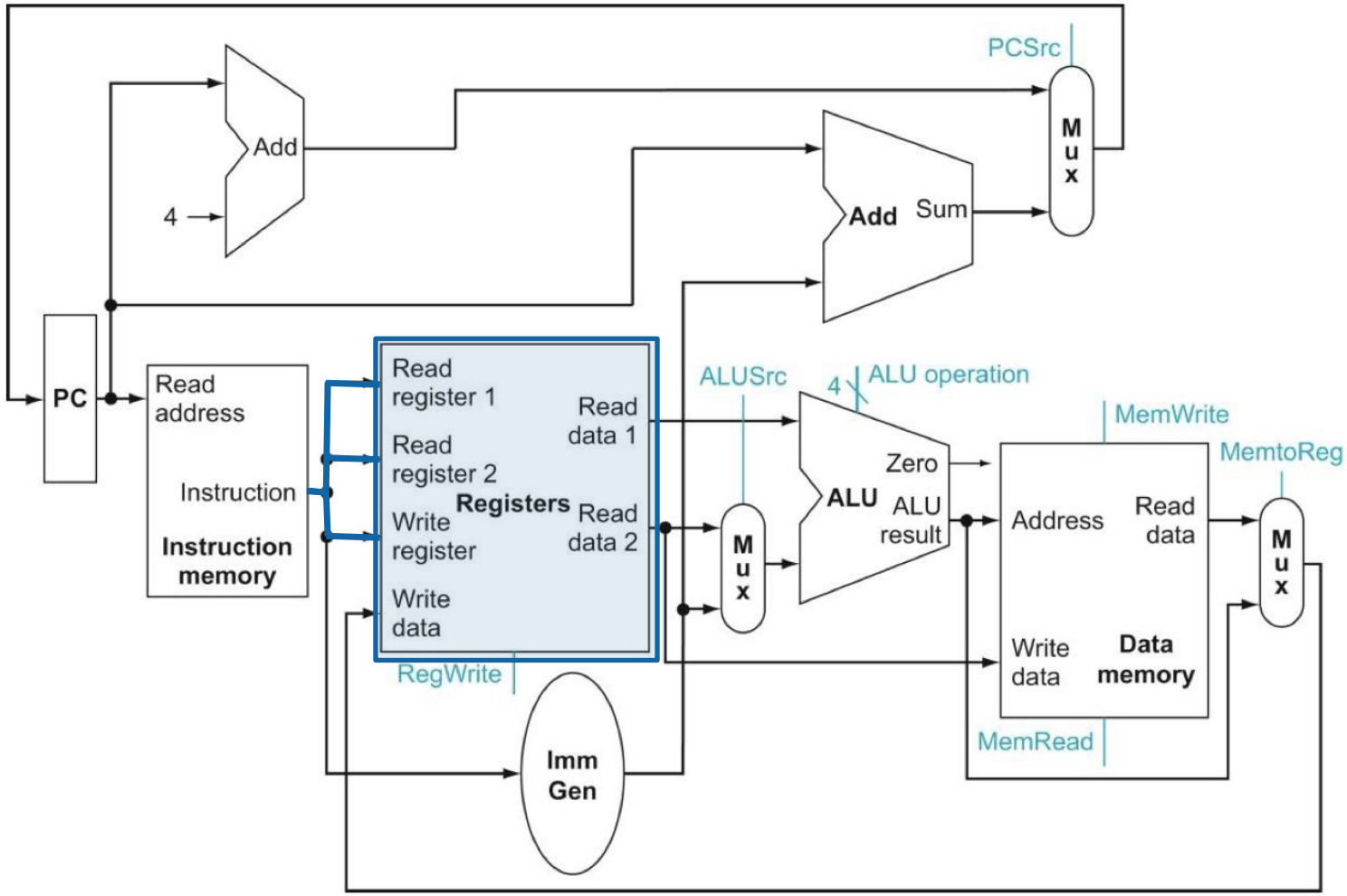
Fetch add x5, x1, x2.

The PC adder computes 0x1004.

PCSrc = 0 → select PC + 4

Single-cycle datapath • steps show signal flow, not separate clock cycles

ADD | 2 Decode / read



add x5, x1, x2

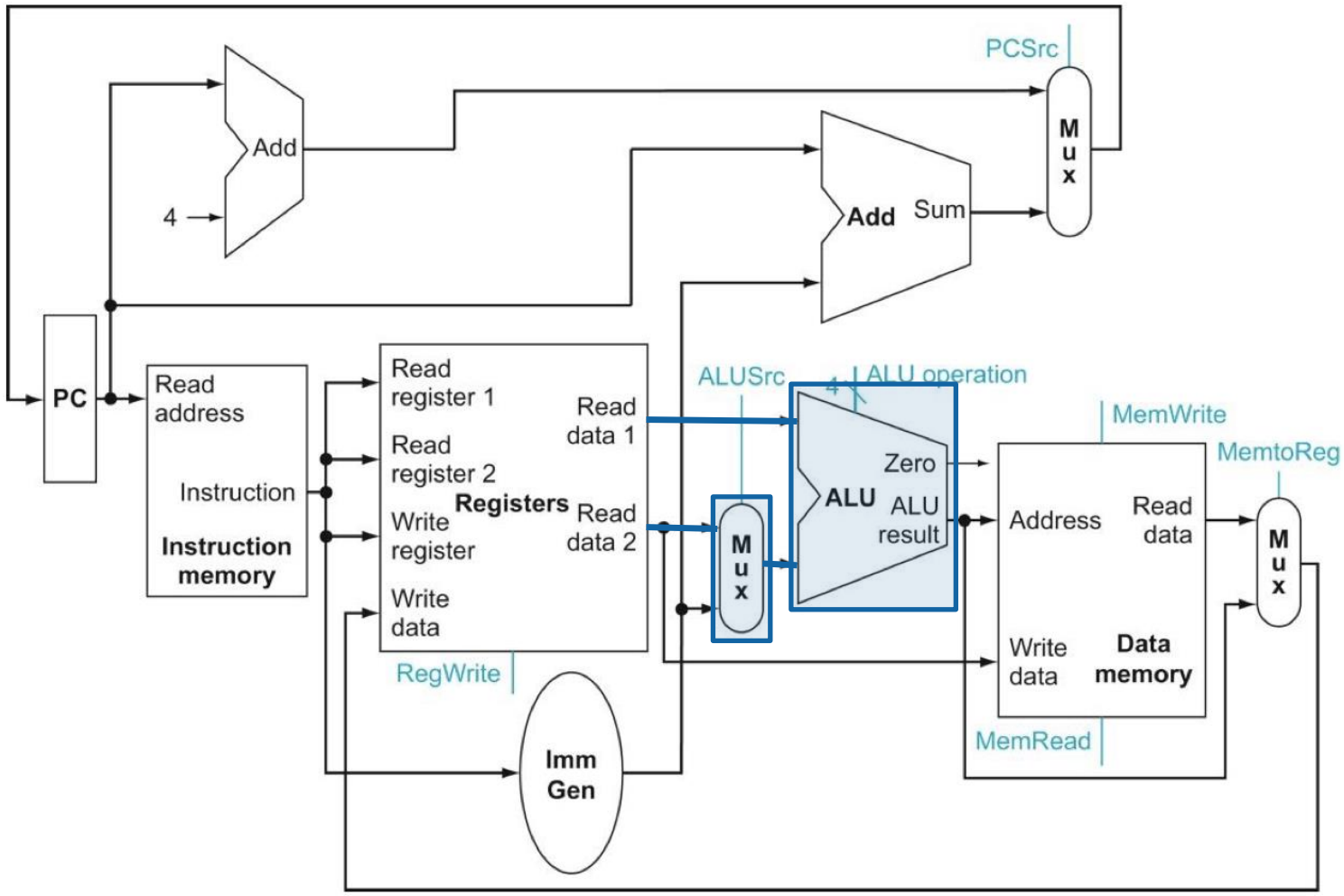
rs1 = 1 → read x1 = 7

rs2 = 2 → read x2 = 11

rd = 5 → destination x5

Control decodes an R-type ADD.

ADD | 3 Execute



add x5, x1, x2

ALU input A = 7

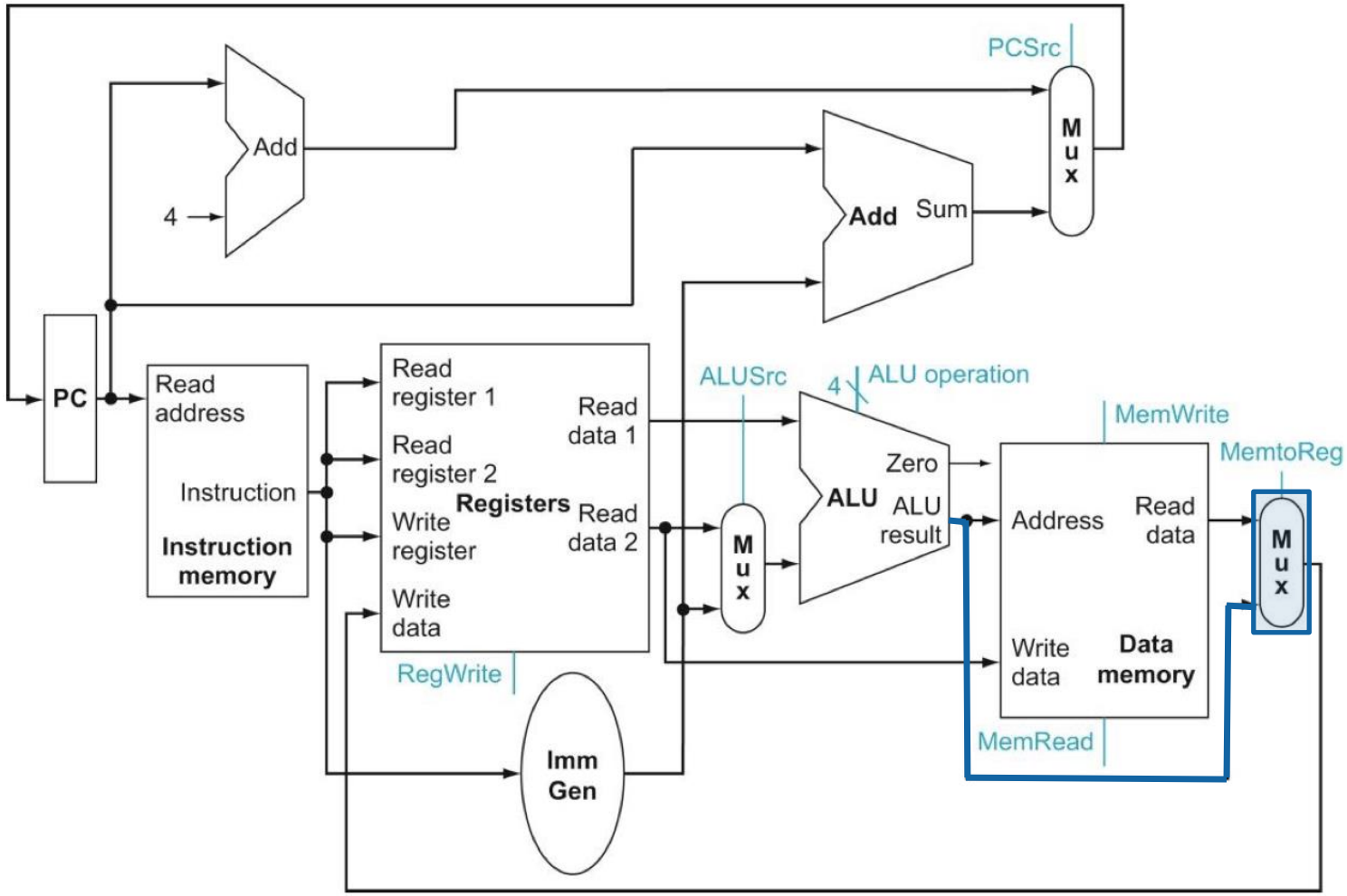
ALU input B = 11

ALU result = 7 + 11 = 18

ALUSrc = 0; ALU operation = ADD

Single-cycle datapath • steps show signal flow, not separate clock cycles.

ADD | 4 Bypass data memory



add x5, x1, x2

The result 18 travels around data memory.

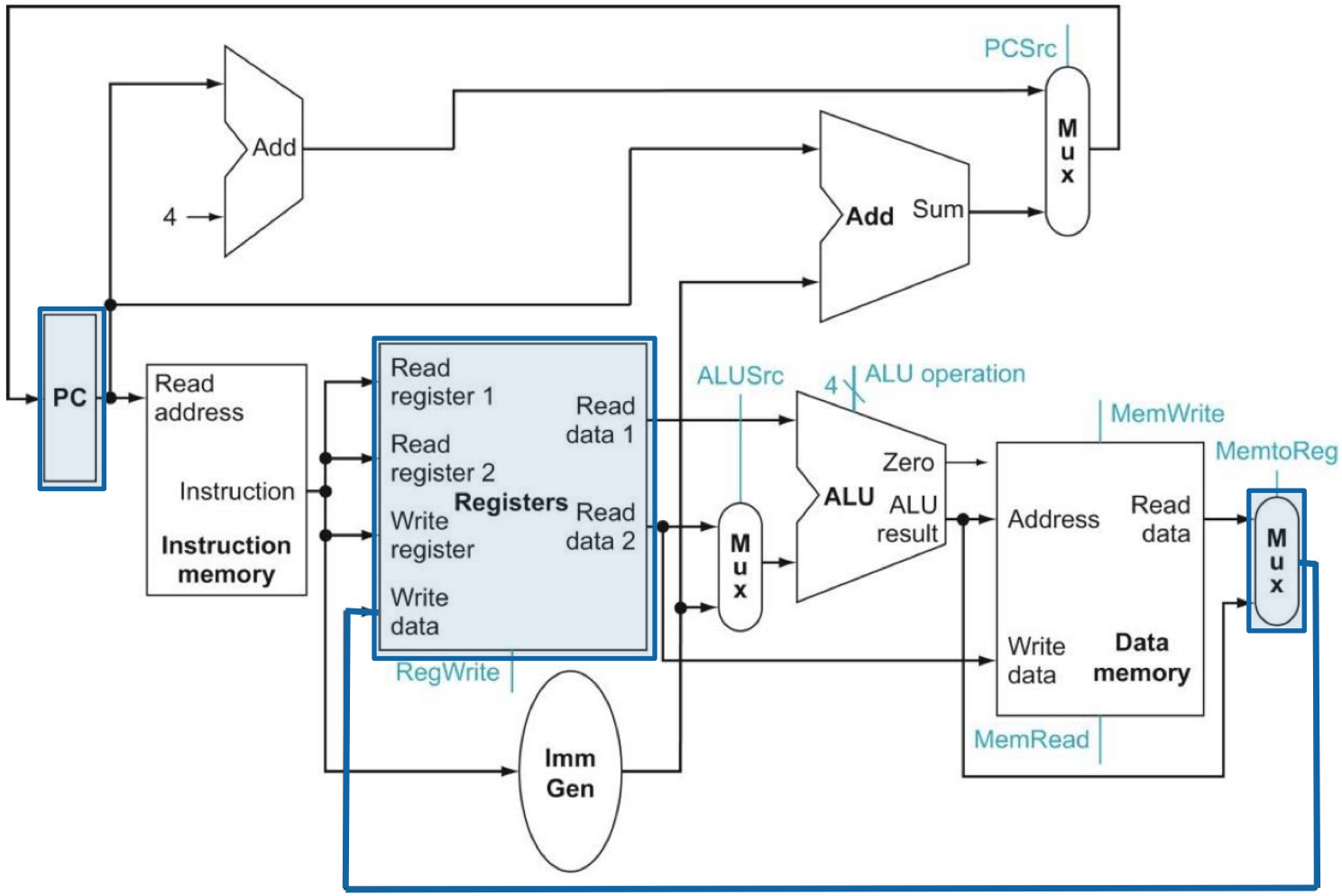
No data-memory read or write.

MemRead = 0; MemWrite = 0

MemtoReg = 0 → ALU result

Single-cycle datapath • steps show signal flow, not separate clock cycles.

ADD | 5 Write back at the edge



add x5, x1, x2

Write 18 to destination x5.

PC becomes 0x1004.

x1 = 7 and x2 = 11 stay unchanged.

RegWrite = 1; rd = 5

Single-cycle datapath • steps show signal flow, not separate clock cycles