CSE261: Computer Architecture

8. Logic Design Basics

Seongil Wi

Department of Computer Science and Engineering

The Score for HW1 will be Announced Soon

We discovered that several students committed plagiarism

Recap: Academic Integrity

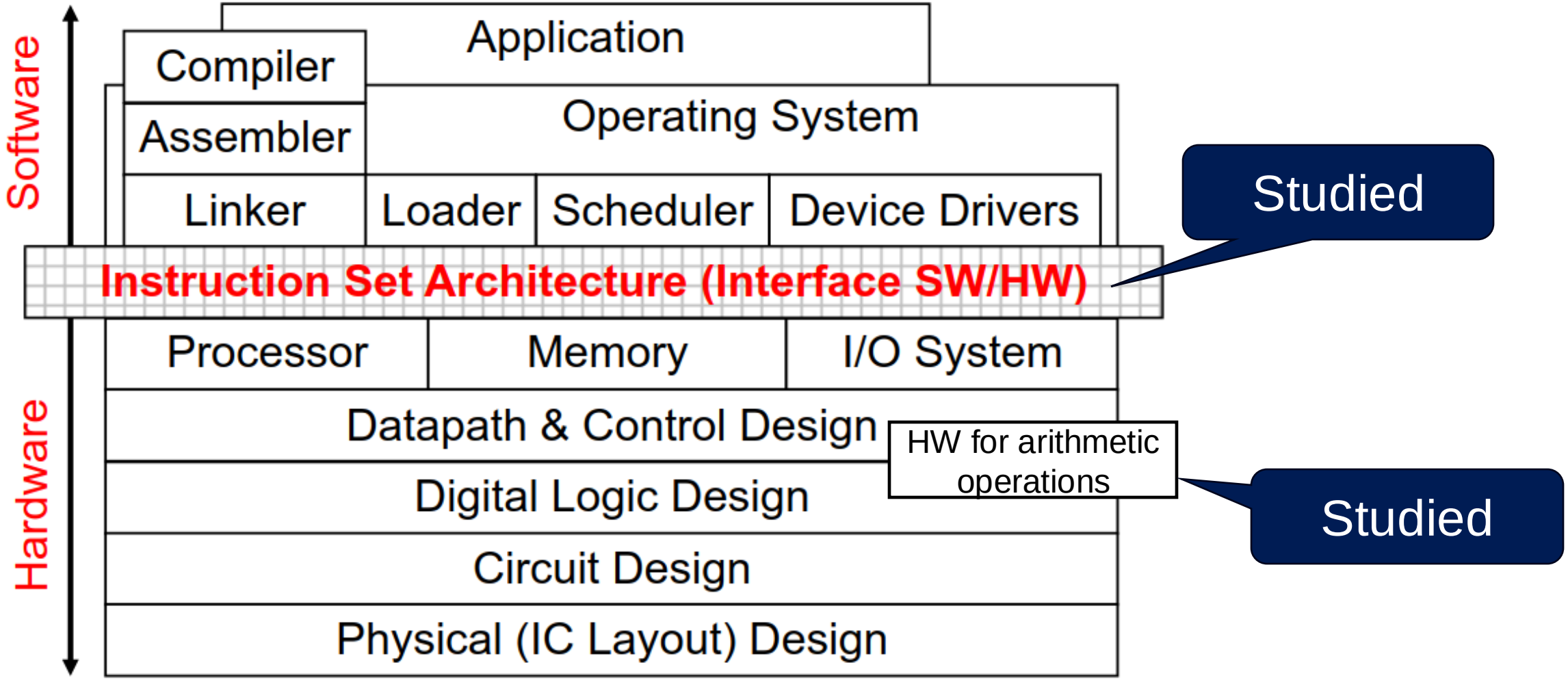
- DO NOT share the course contents (e.g., assignments or exams) with others
  - E.g., Github public repository, chegg.com, etc
- DO NOT discuss the **details of solutions** with others
- DO NOT plagiarize
  - Submit your own work
- Any integrity violation: at LEAST F

UNIST CSE Policy on Cheating and Plagiarism

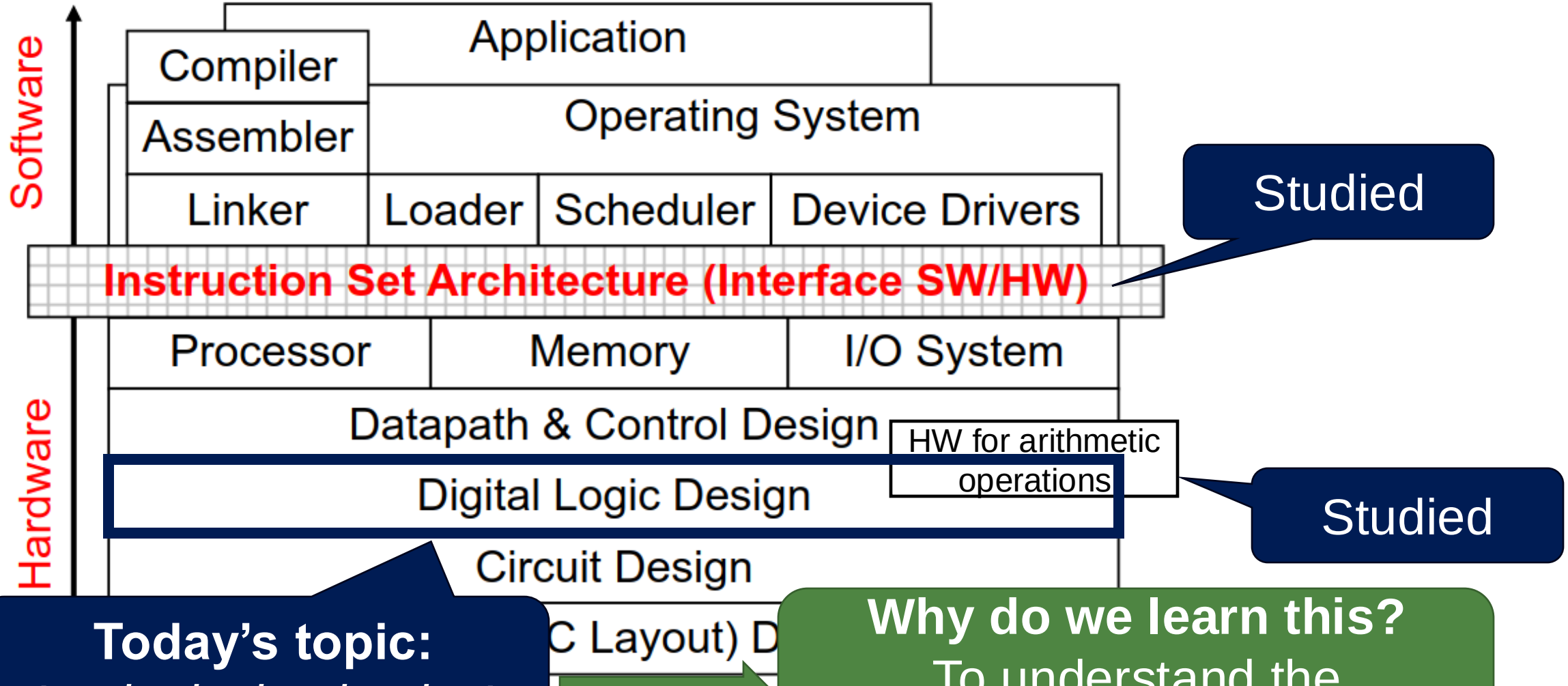
Note: The term **solution** means program code, mathematical derivation, experimental setup, etc., for any type of deliverable, homework assignment, or projects in class.

The purpose of this document is to make our expectations in CSE as clear as possible in regard to the Honor Code at UNIST. The basic principle under which we operate is that each of you is expected to **submit your own work in your courses**. In particular, attempting to take credit for someone else's work by turning it in as your own constitutes plagiarism, which is a serious violation of fundamental academic standards. However, you are also encouraged to work as a team and collaborate with each other, and it is usually appropriate to ask others—the TA, the instructor, or other students—for direction and debugging help or to talk generally about

Where Are We?


Software
Application
Compiler
Operating System
Assembler
Linker
Loader
Scheduler
Device Drivers
Instruction Set Architecture (Interface SW/HW)
Processor
Memory
I/O System
Datapath & Control Design
Digital Logic Design
Circuit Design
Physical (IC Layout) Design
Hardware
Studied
HW for arithmetic operations
Studied

Today's Topic

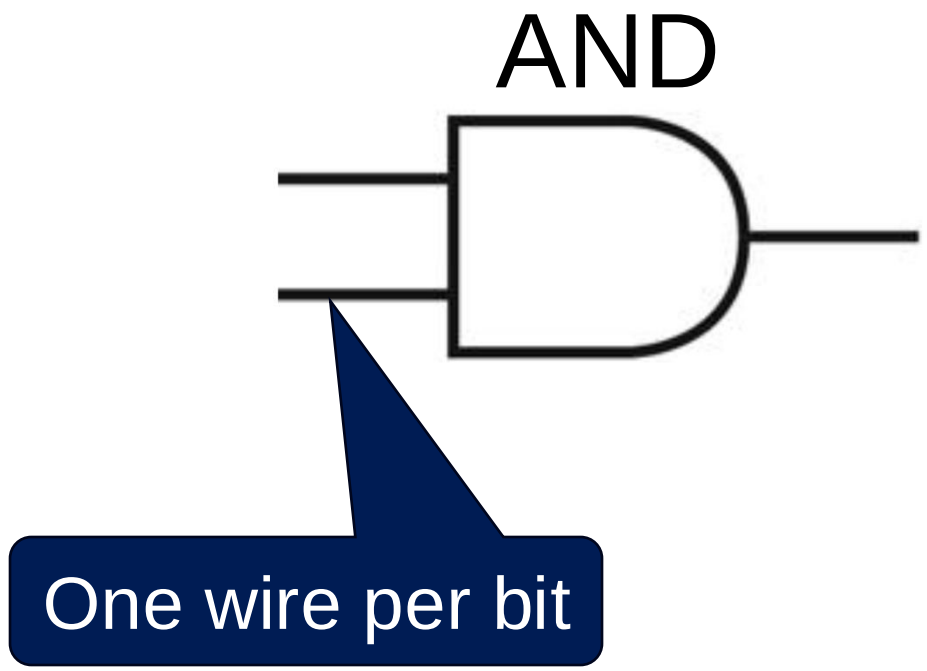


Logic Design

Basic organization of the circuitry of a digital computer

Logic Design Basics

- Information encoded in binary
  - Low voltage = 0, High voltage = 1
  - One wire per bit
  - Multi-bit data encoded on multi-wire buses



Two Types of Logic Circuits

- **Combinational circuit**

- **Sequential circuit**

Two Types of Logic Circuits

- **Combinational circuit**
  - Outputs only depends on the current inputs

- **Sequential circuit**

Combinational Logic Circuits

AND

Input		Output
a	b	
0	0	0
0	1	0
1	0	0
1	1	1

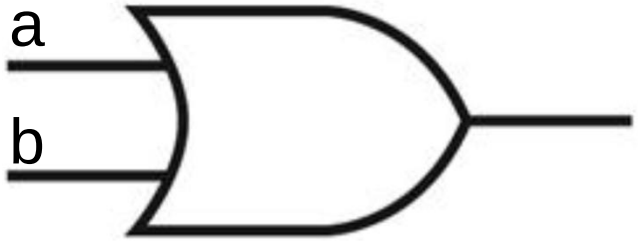
Outputs only depends on the current inputs

Combinational Circuits: AND, OR, NOT

AND

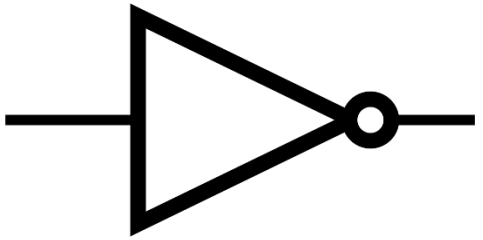
Input		Output
a	b	
0	0	0
0	1	0
1	0	0
1	1	1

OR



Input		Output
a	b	
0	0	0
0	1	1
1	0	1
1	1	1

NOT

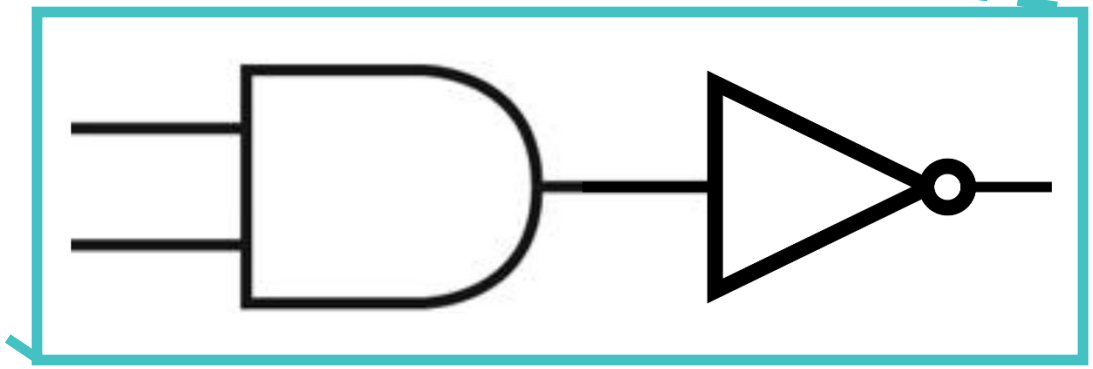


Input	Output
0	1
1	0

Basic blocks for creating combinational circuits

Combinational Circuits: NAND

NAND



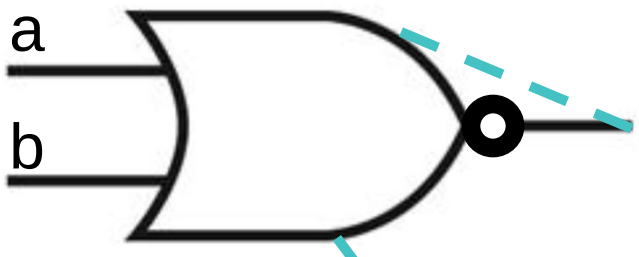
Input		Output
a	b	
0	0	1
0	1	1
1	0	1
1	1	0

Combinational Circuits: NOR

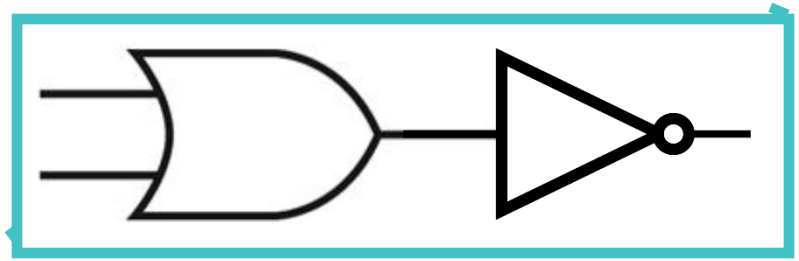
NAND

Input		Output
a	b	
0	0	1
0	1	1
1	0	1
1	1	0

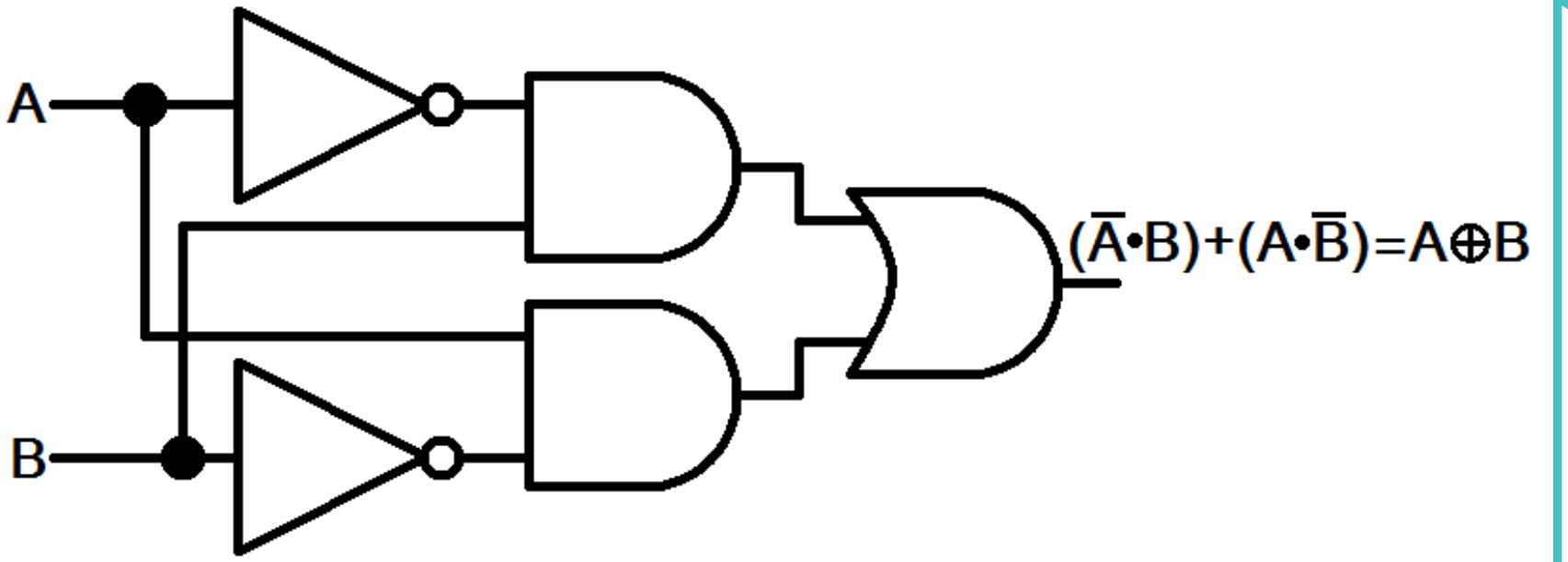
NOR

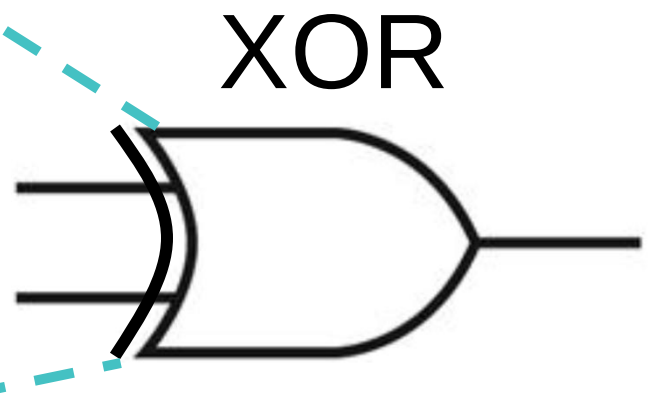


Input		Output
a	b	
0	0	1
0	1	0
1	0	0
1	1	0



Combinational Circuits: XOR





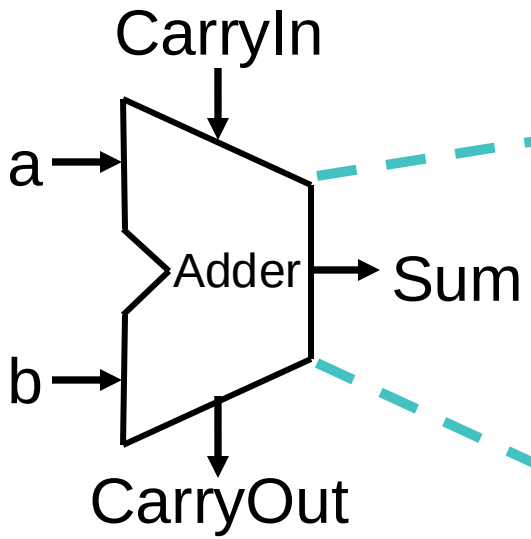
0	0	1
0	1	1
1	0	1
1	1	0

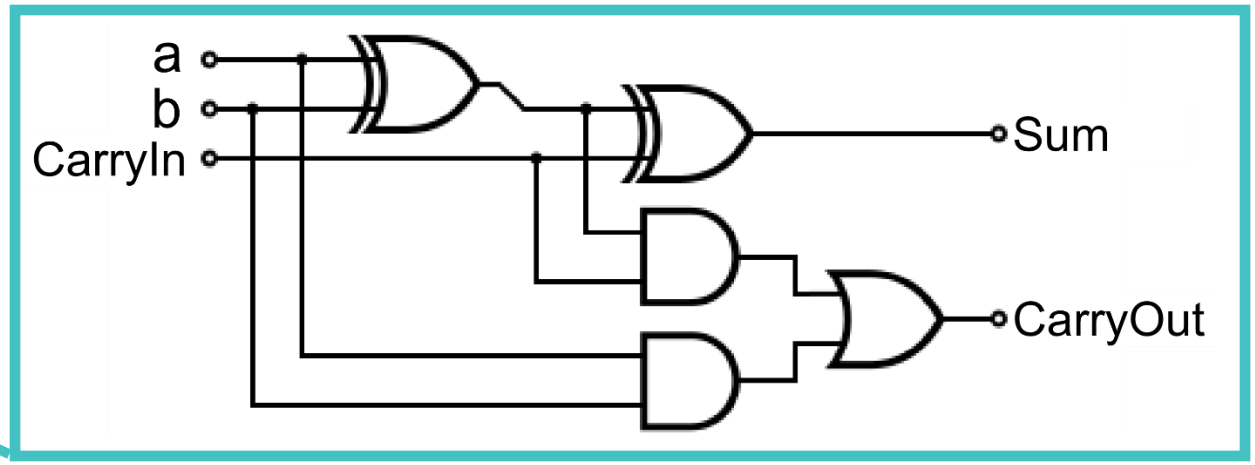
0	0	1
0	1	0
1	0	0
1	1	0

Input		Output
a	b	
0	0	0
0	1	1
1	0	1
1	1	0

Combinational Circuits: Adder

Adder

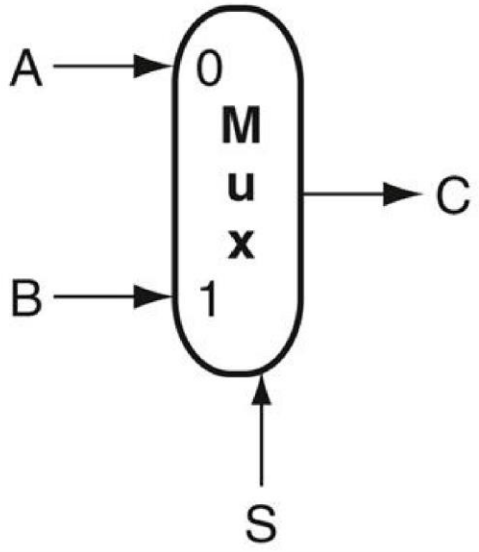




Inputs			Outputs		
a	b	CarryIn	CarryOut	Sum	Comments
0	0	0	0	0	$0 + 0 + 0 = 00_{two}$
0	0	1	0	1	$0 + 0 + 1 = 01_{two}$
0	1	0	0	1	$0 + 1 + 0 = 01_{two}$
0	1	1	1	0	$0 + 1 + 1 = 10_{two}$
1	0	0	0	1	$1 + 0 + 0 = 01_{two}$
1	0	1	1	0	$1 + 0 + 1 = 10_{two}$
1	1	0	1	0	$1 + 1 + 0 = 10_{two}$
1	1	1	1	1	$1 + 1 + 1 = 11_{two}$

Combinational Circuits: Multiplexor

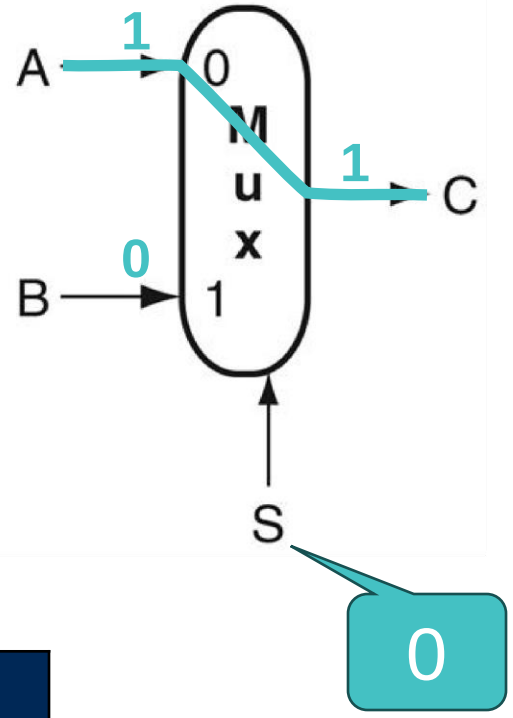
Multiplexor
(a.k.a., MUX)



Input S	Output
0	A's Input
1	B's Input

Combinational Circuits: Multiplexor

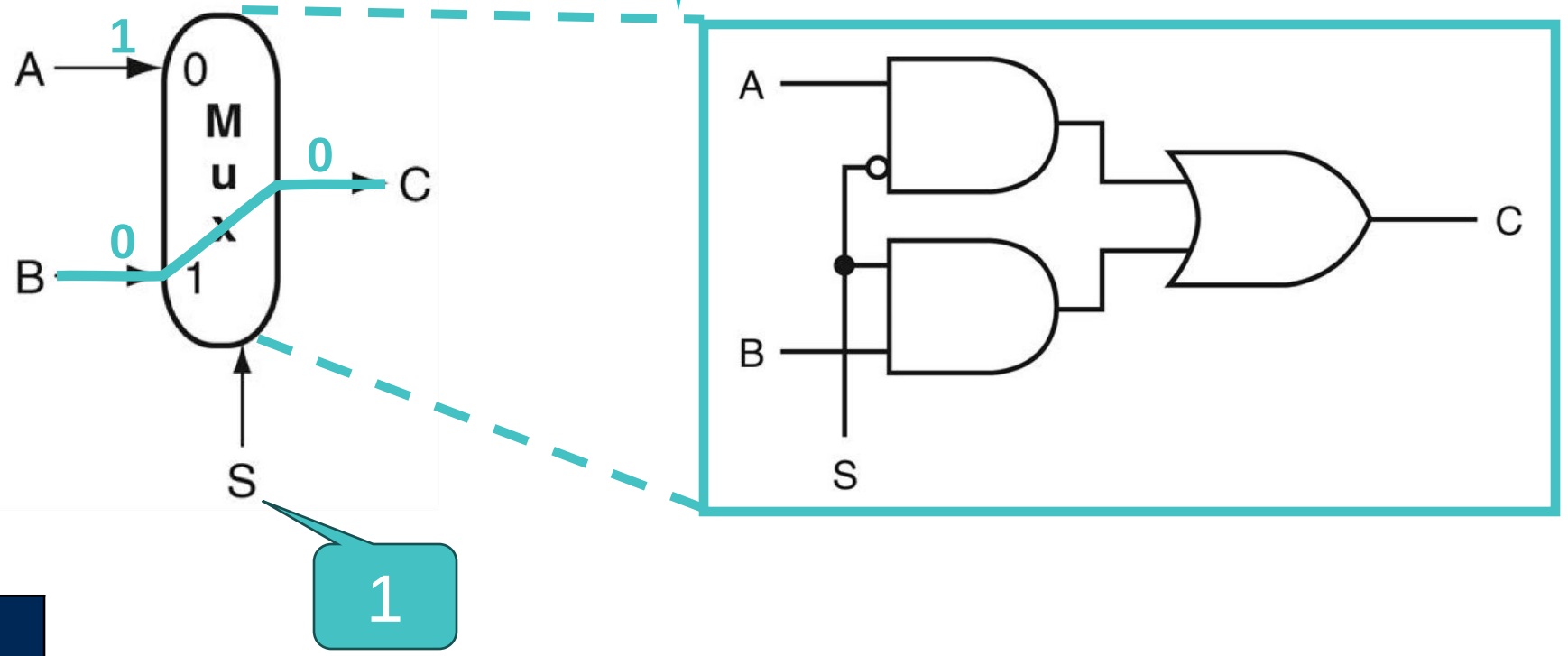
Multiplexor
(a.k.a., MUX)



Input S	Output
0	A's Input
1	B's Input

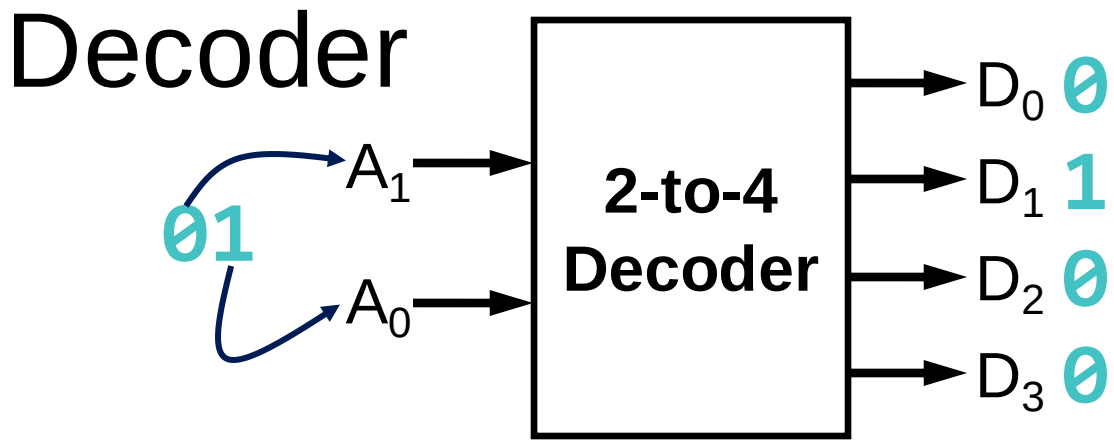
Combinational Circuits: Multiplexor

Multiplexor
(a.k.a., MUX)



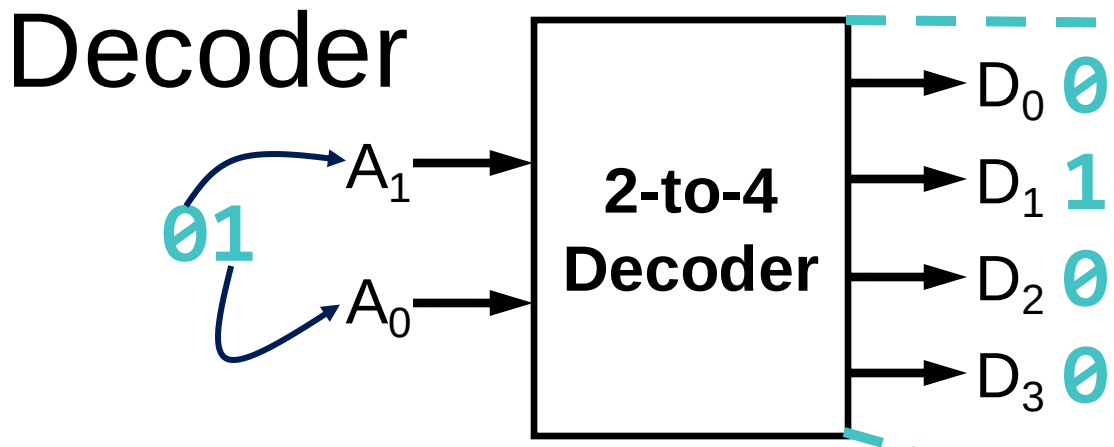
Input S	Output
0	A's Input
1	B's Input

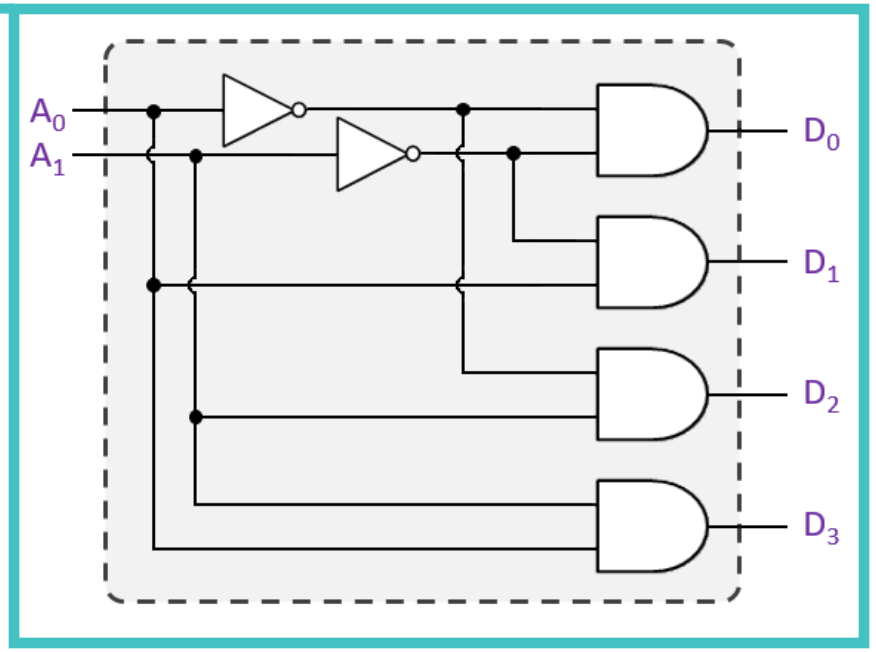
Combinational Circuits: Decoder



Input		Output			
A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Combinational Circuits: Decoder

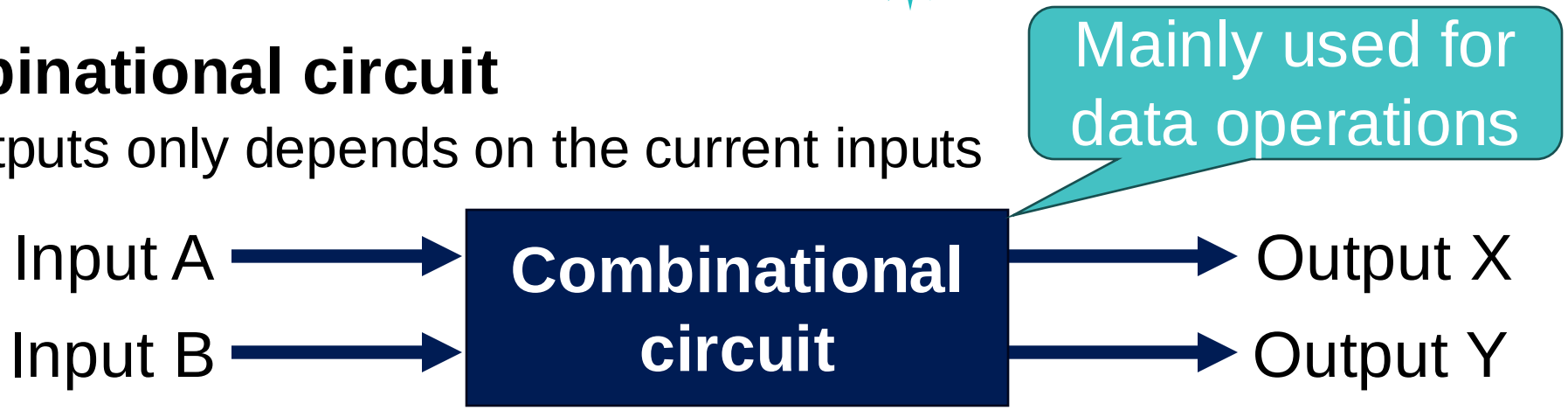




Input		Output			
A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Two Types of Logic Elements

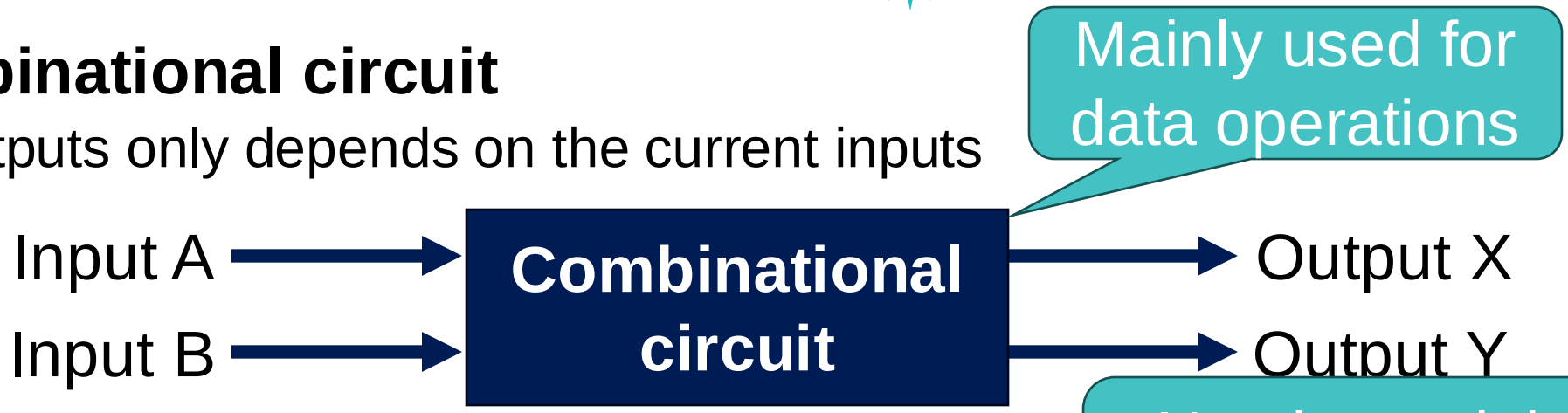
- **Combinational circuit**
  - Outputs only depends on the current inputs



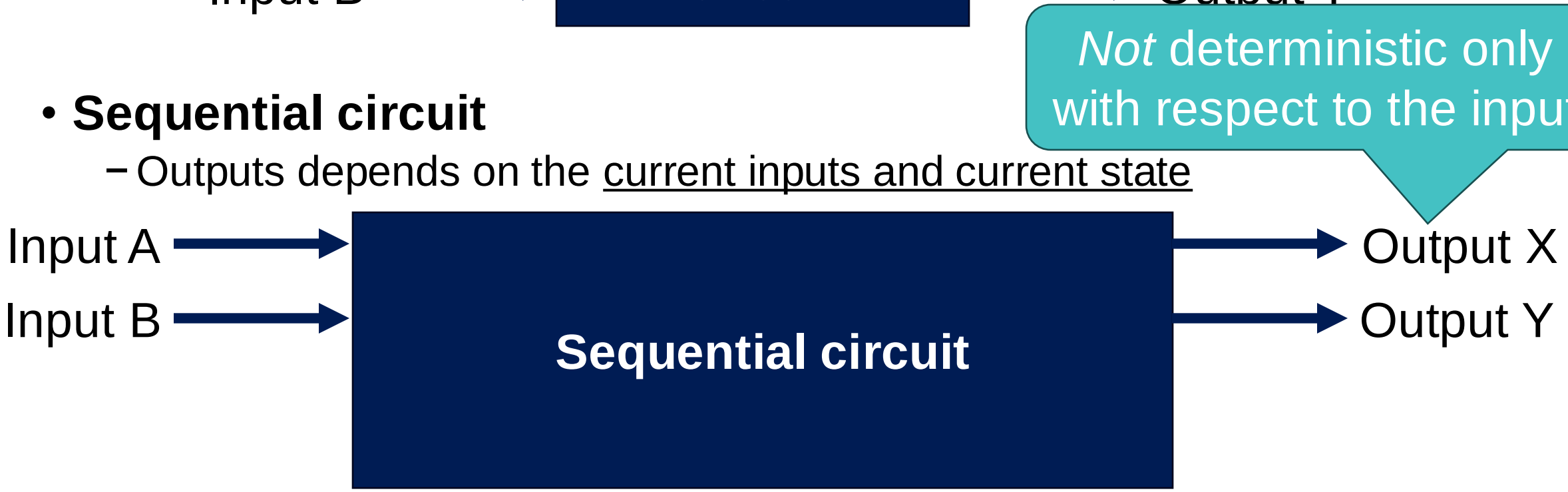
- **Sequential circuit**
  - Outputs depends on the current inputs and current state

Two Types of Logic Elements

- **Combinational circuit**
  - Outputs only depends on the current inputs

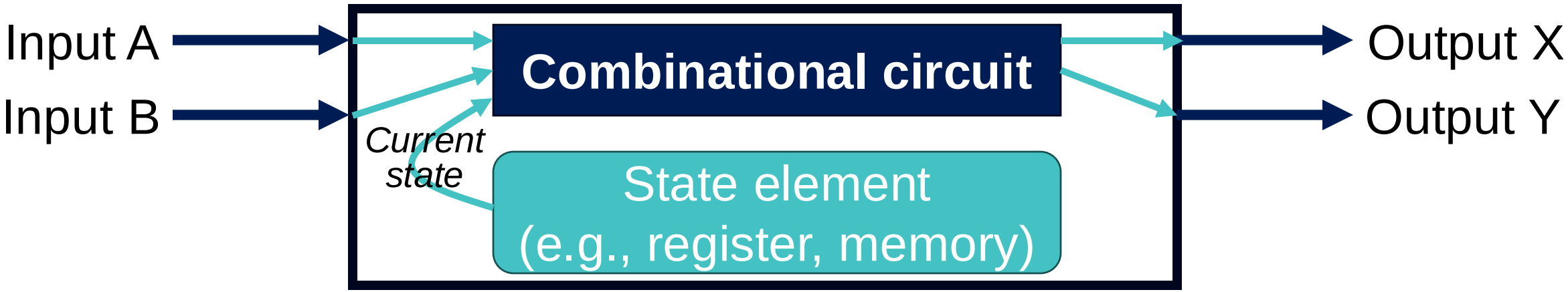


- **Sequential circuit**
  - Outputs depends on the current inputs and current state



Sequential Circuit

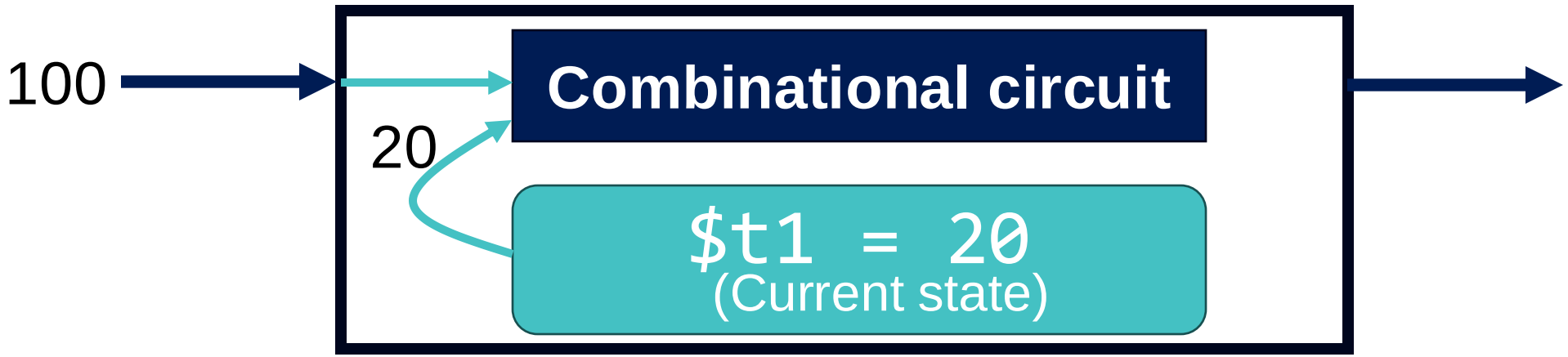
- **Sequential circuit**
  - Outputs depends on the current inputs and current state



Sequential Circuit: Example

```
addi $t1, $t1, 100
```

- **Sequential circuit**
  - Outputs depends on the current inputs and current state

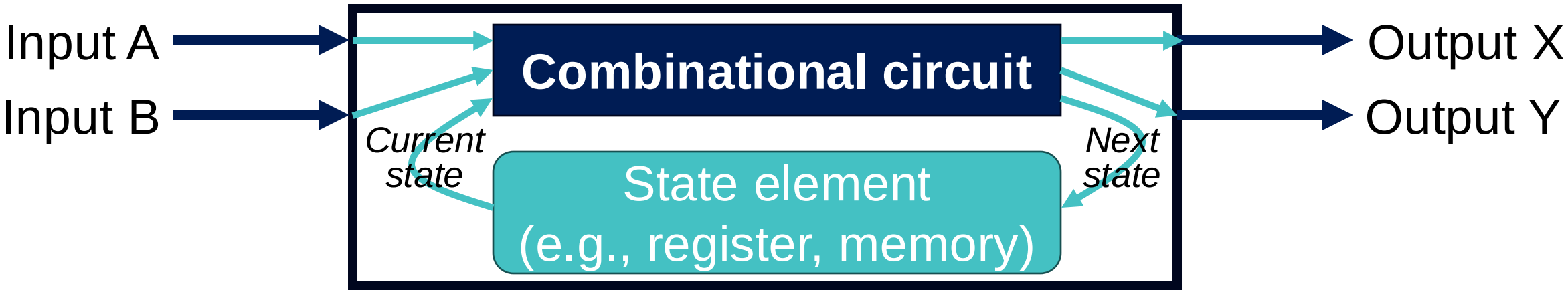


Sequential Circuit: Example

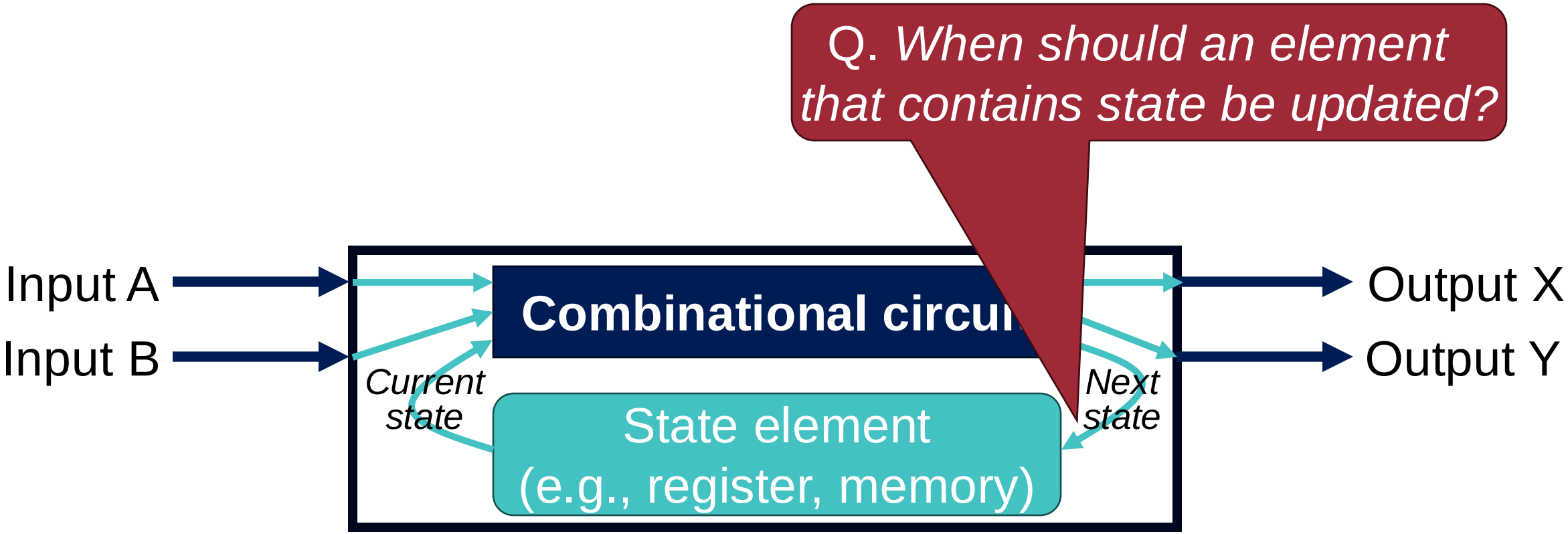

```
addi $t1, $t1, 100
```

- **Sequential circuit**
  - Outputs depends on the current inputs and current state

Sequential Circuit: Final View

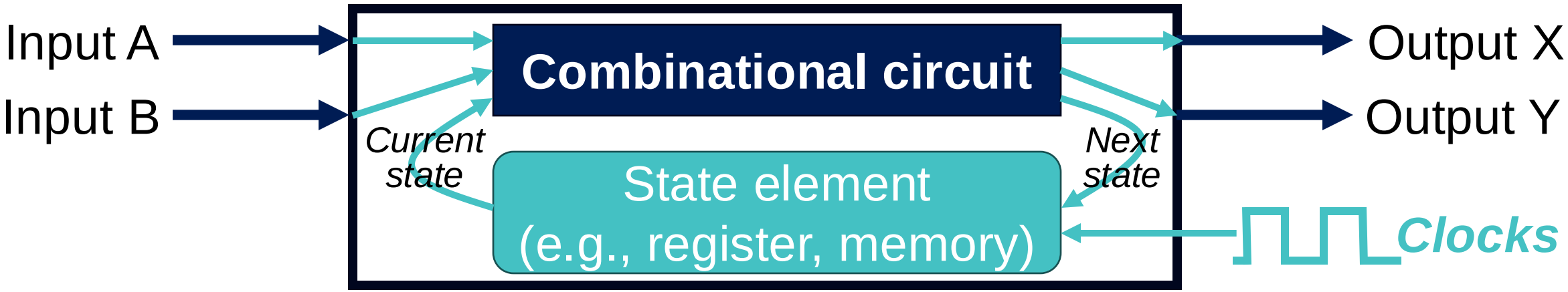


Motivation: Clocks

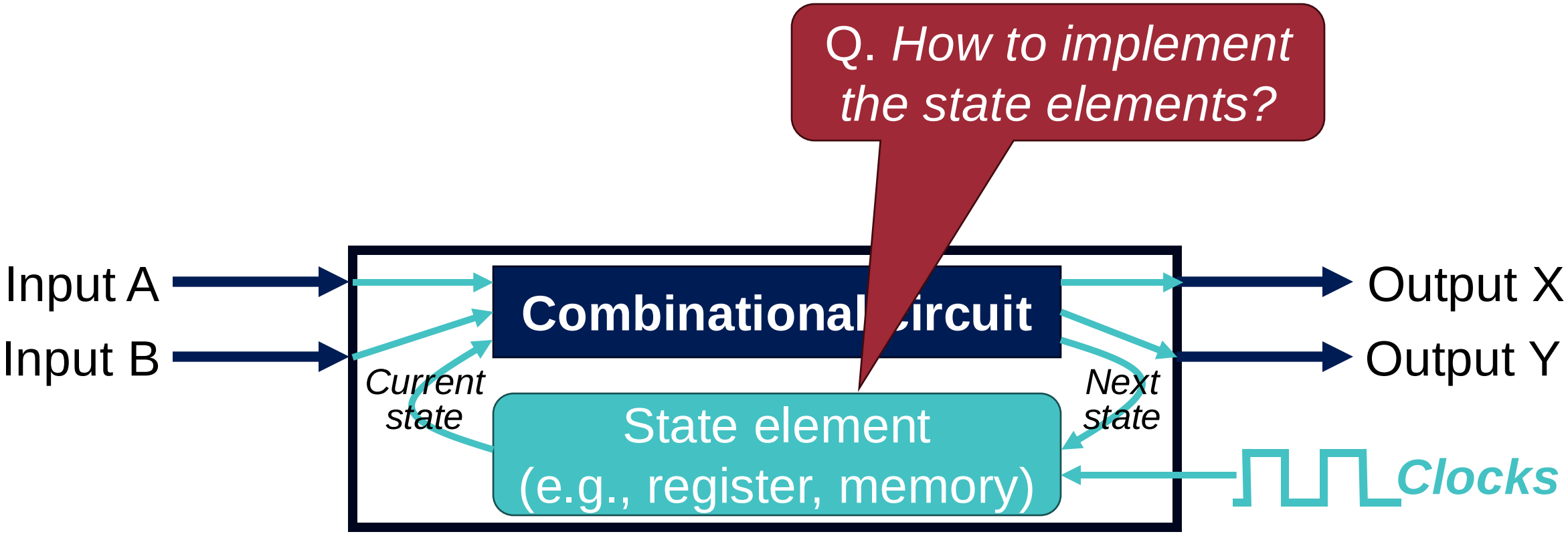


CPU Clocking

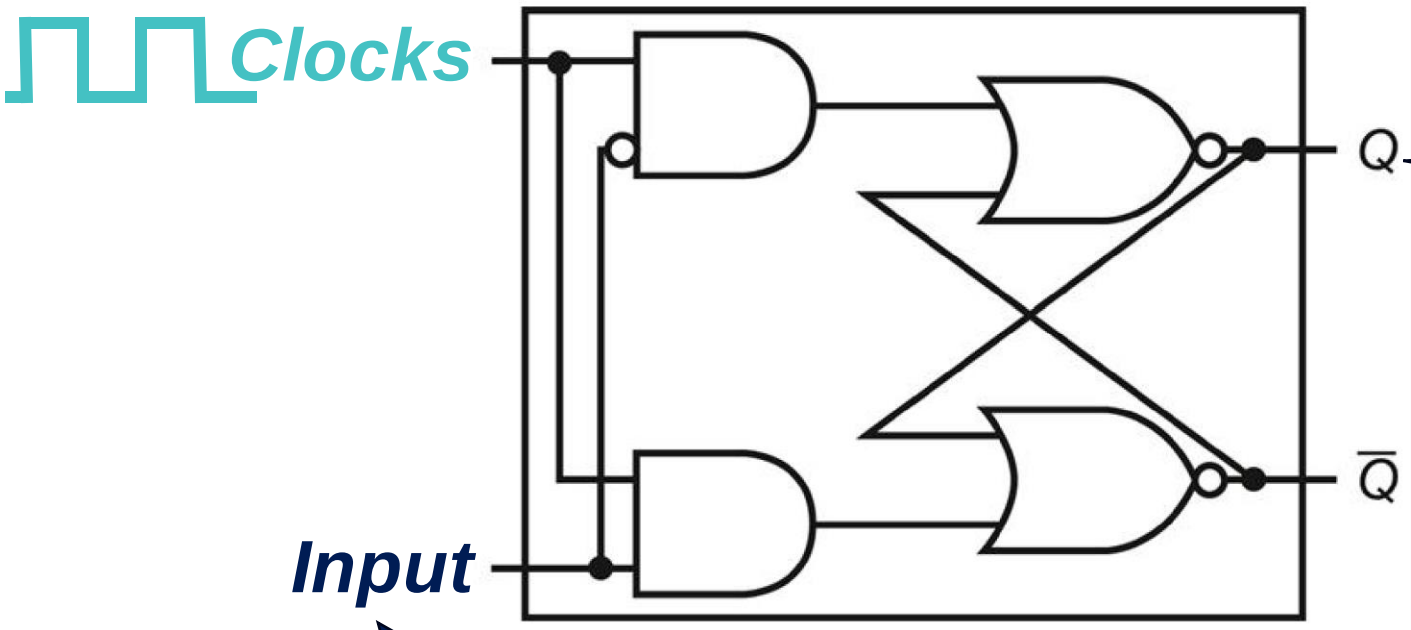
- Operation of digital hardware governed by a constant-rate clock



Motivation: D-Latch and D Flip-flop

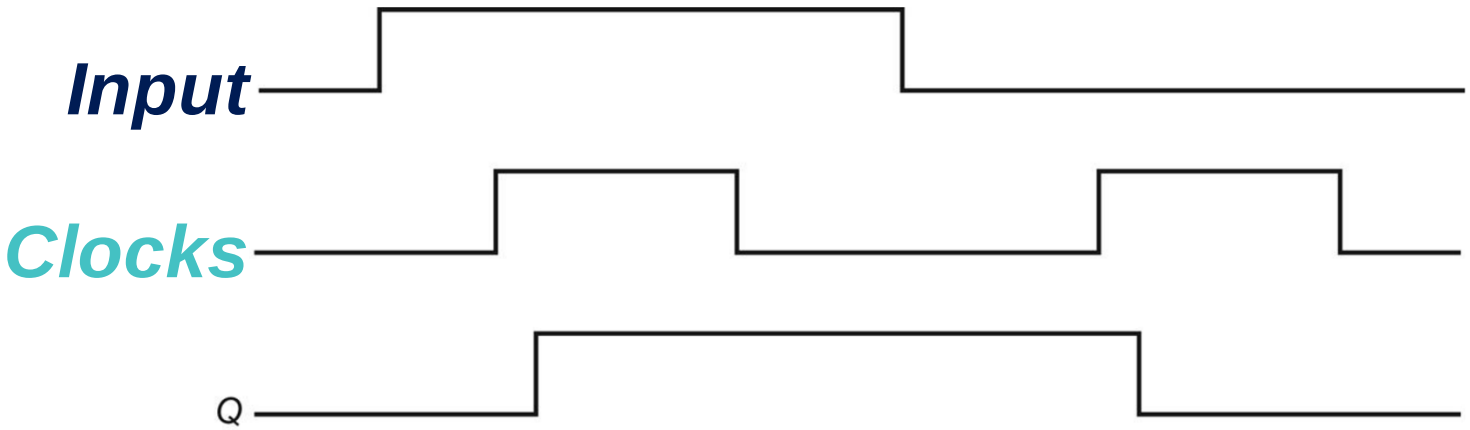


State Element #1: D-Latch

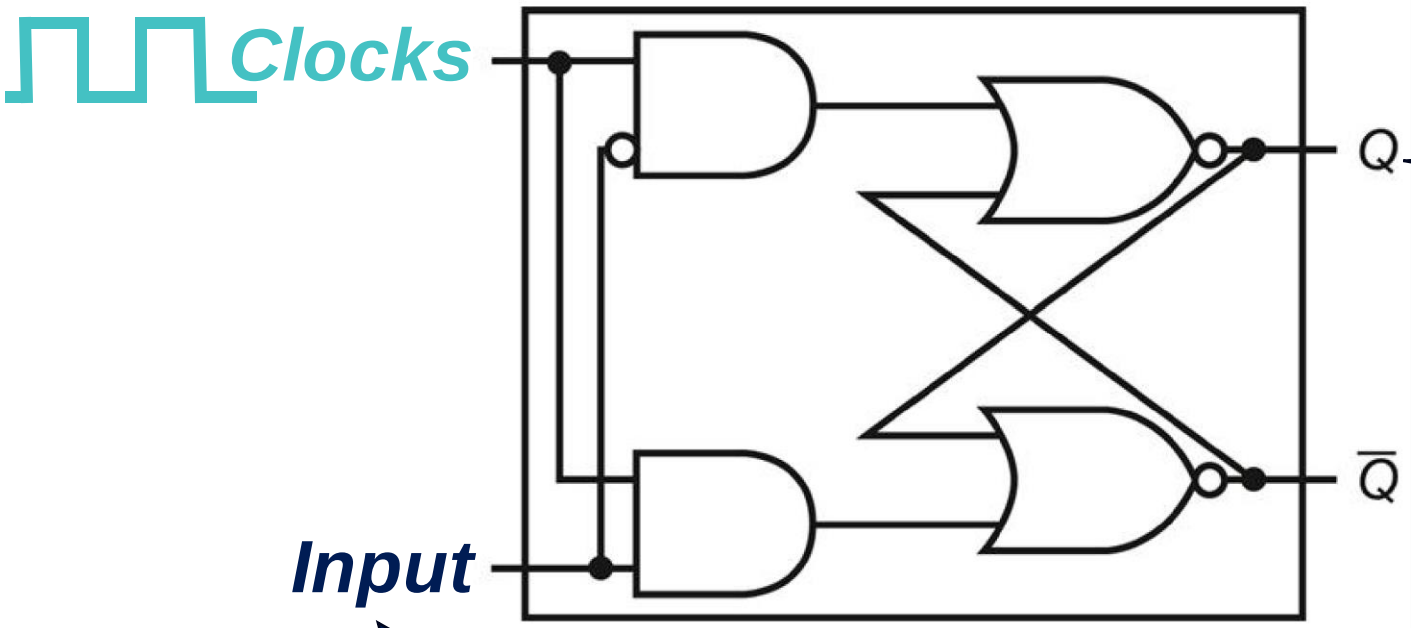


The value of the internal state

The data value to be stored



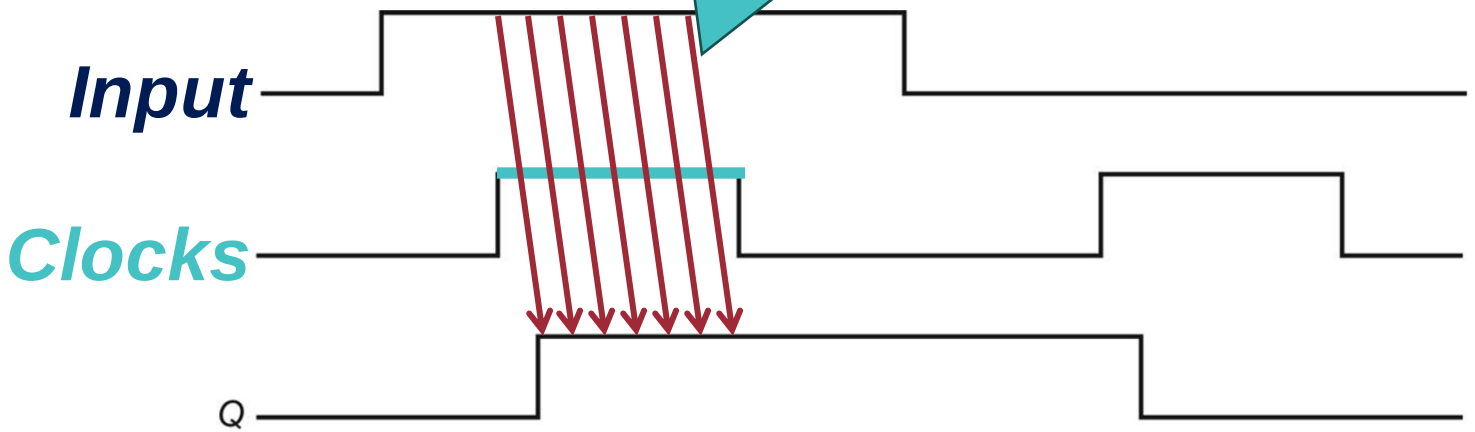
State Element #1: D-Latch



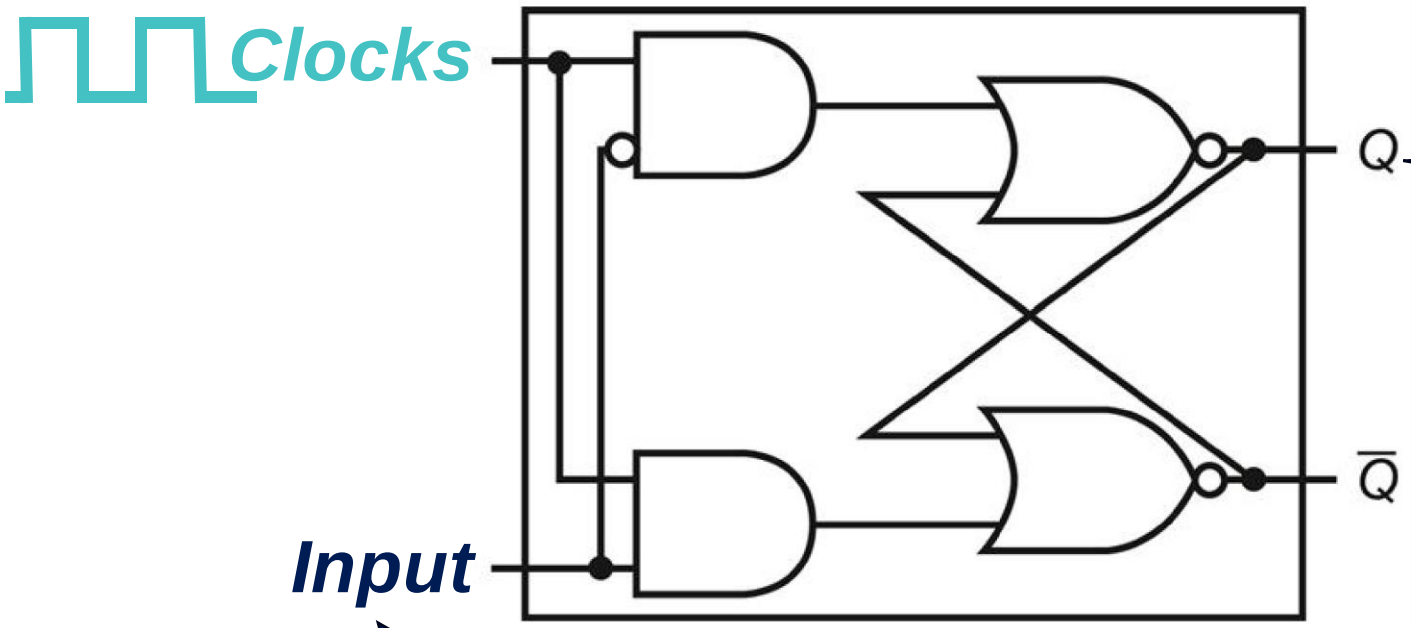
The value of the internal state

The data value to be stored

When the clock is 1, *Q* will reflect the input's value



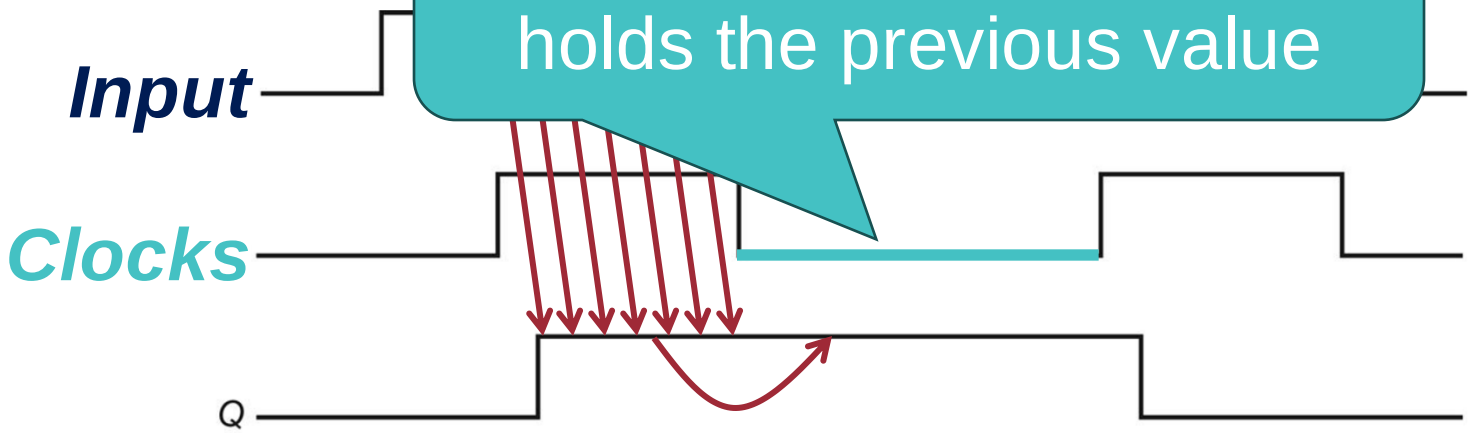
State Element #1: D-Latch



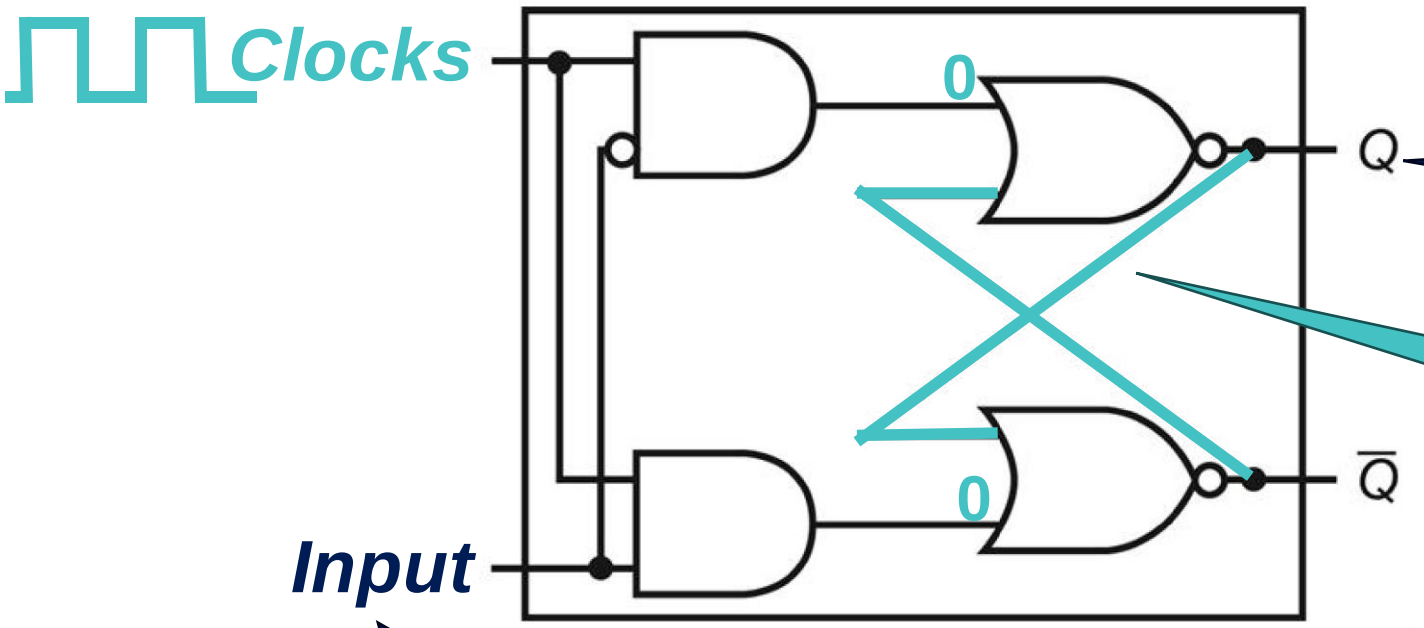
The value of the internal state

The data value to be stored

When the clock is 0, *Q* holds the previous value



State Element #1: D-Latch

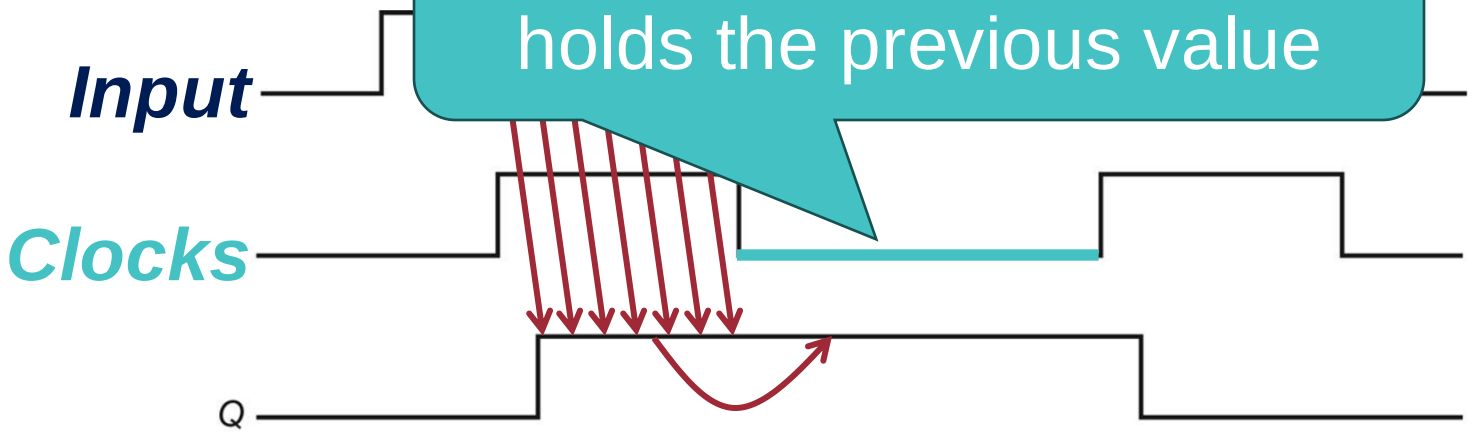


The value of the internal state

Store data in a circuit!

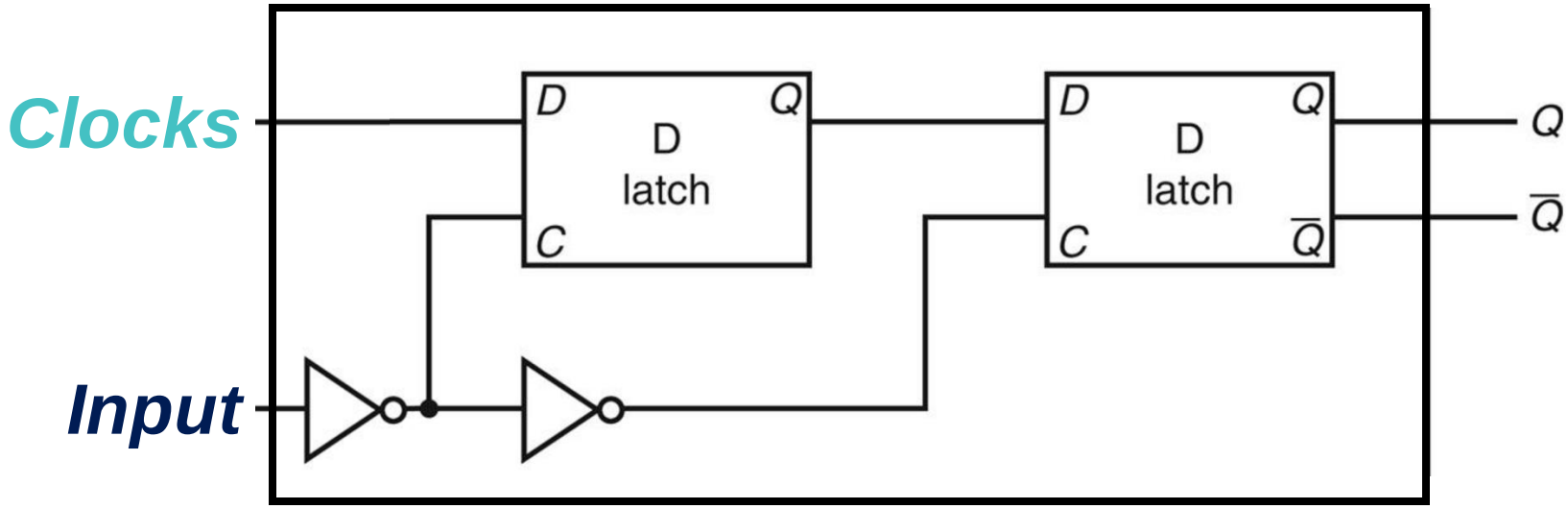
The data value to be stored

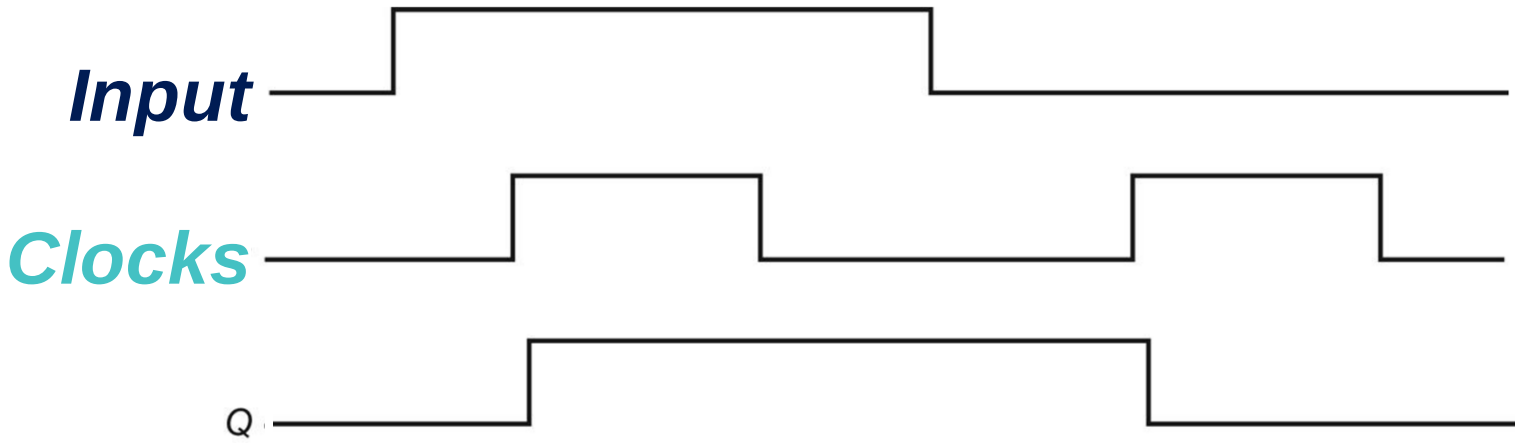
When the clock is 0, *Q* holds the previous value



State Element #2: D Flip-flop

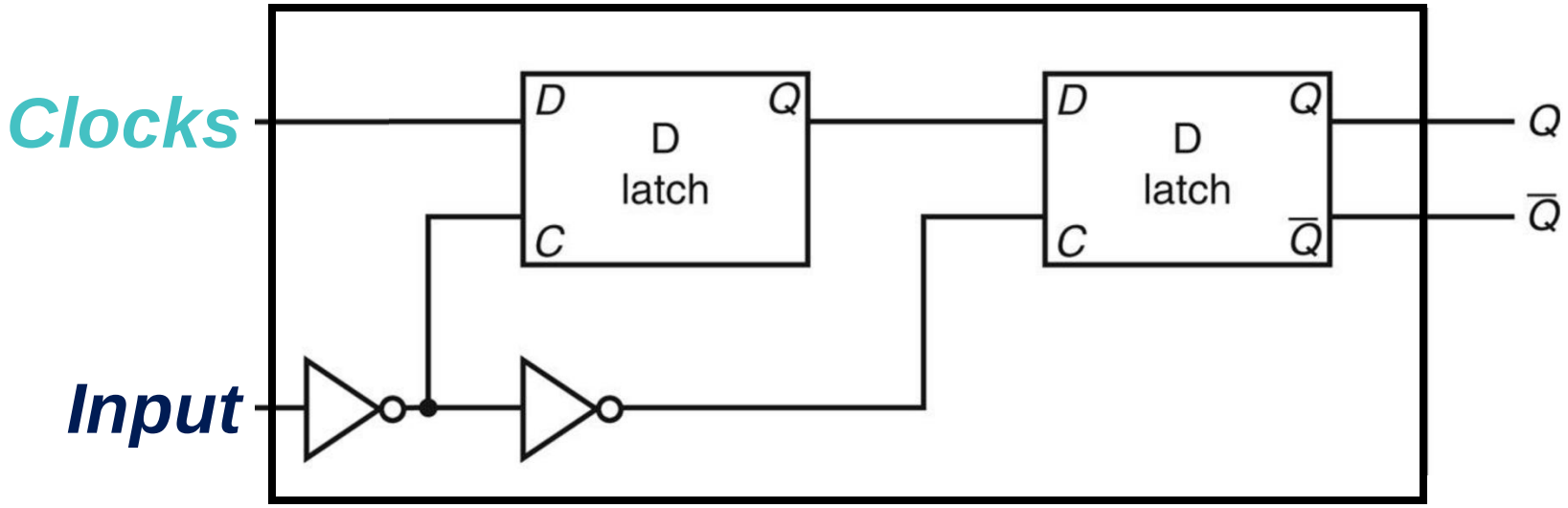
- Output changes ***only on the clock edge***

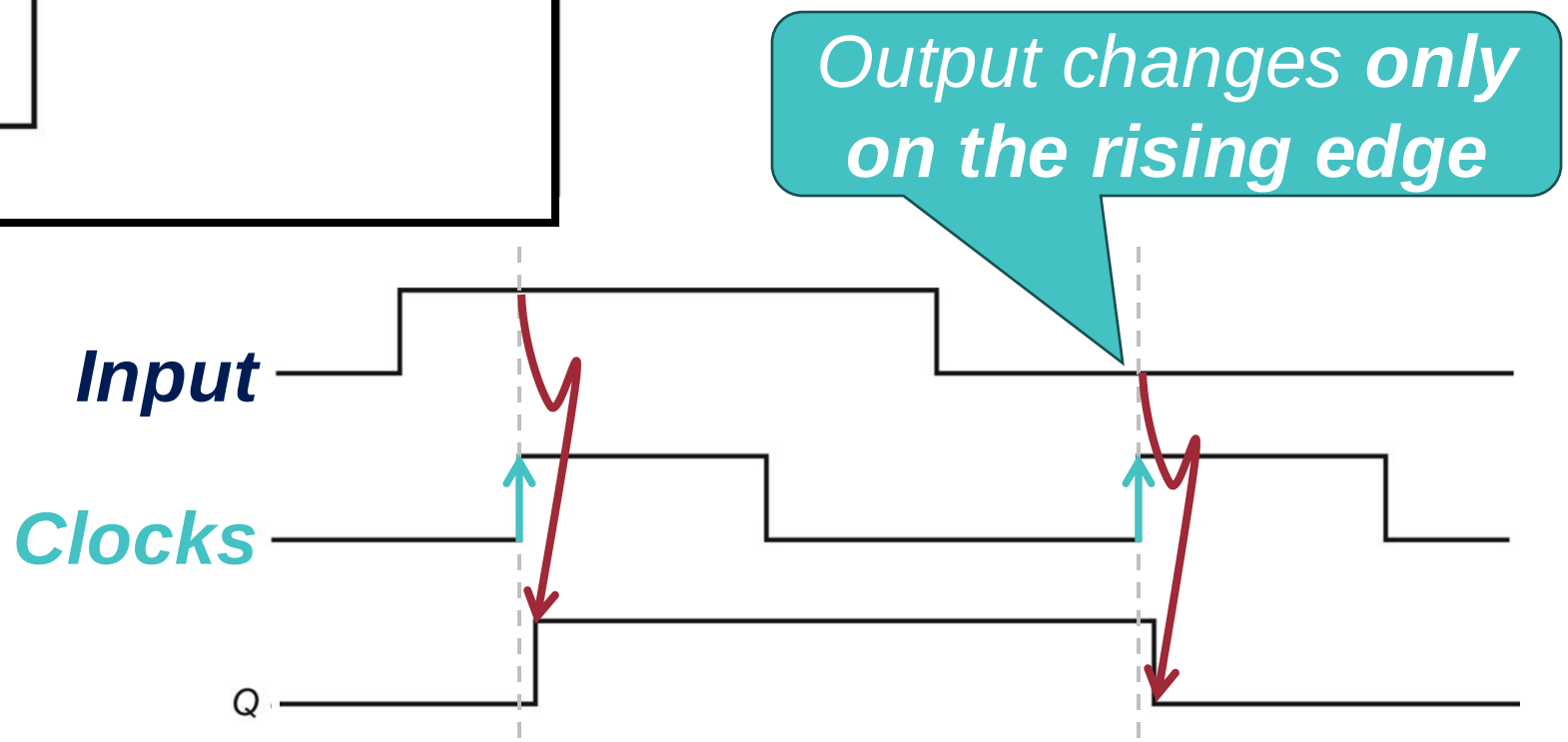




State Element #2: D Flip-flop

- Output changes ***only on the clock edge***

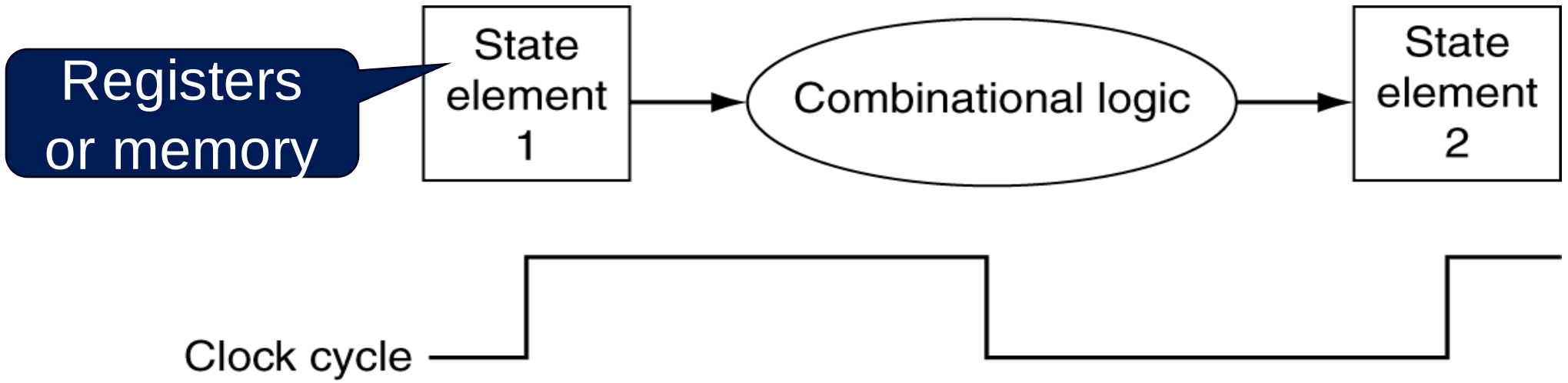




This Course

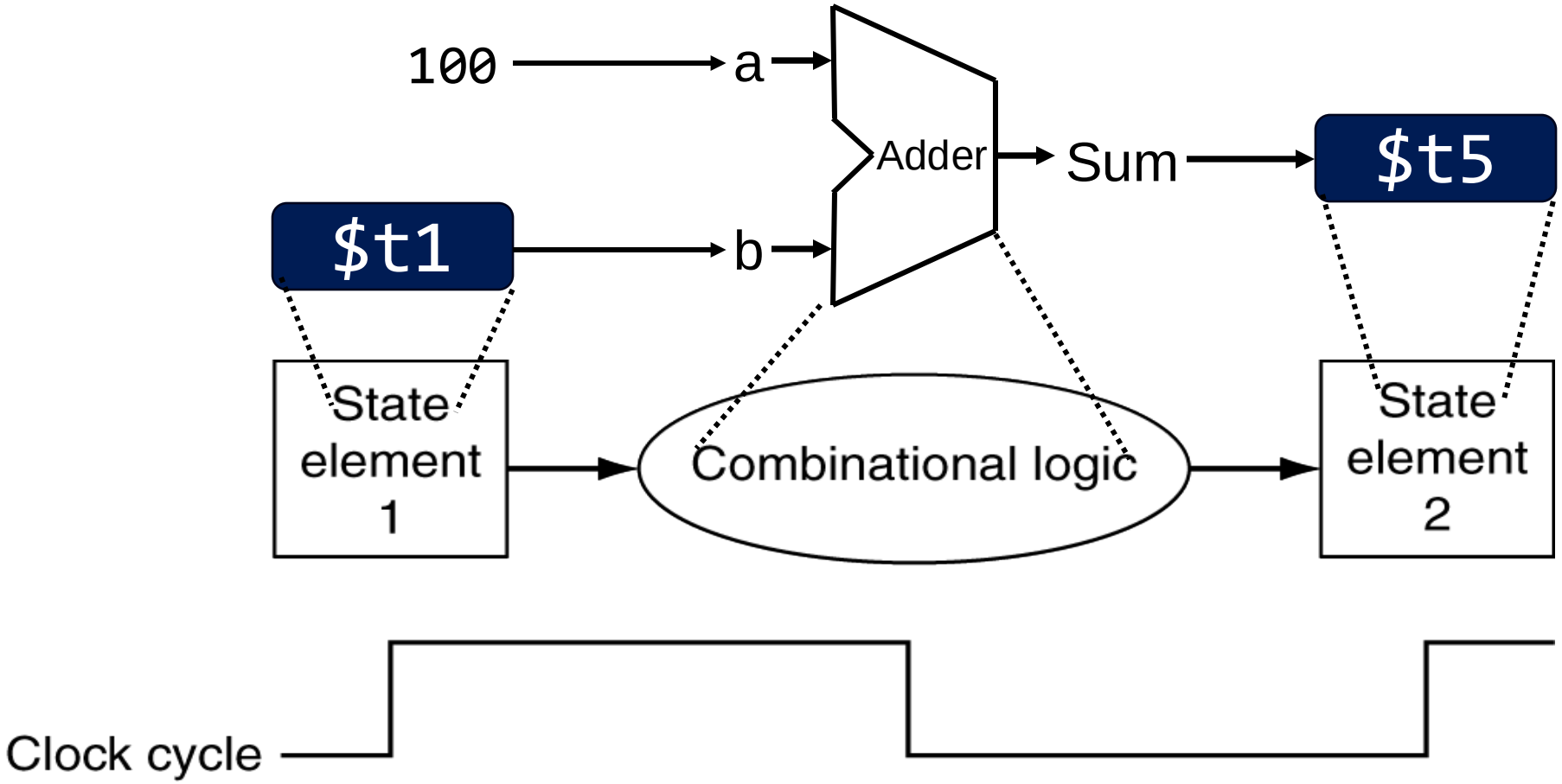
We consider about ***rising-edge triggered D flip-flop***

Clocking Methodology in Sequential Circuit



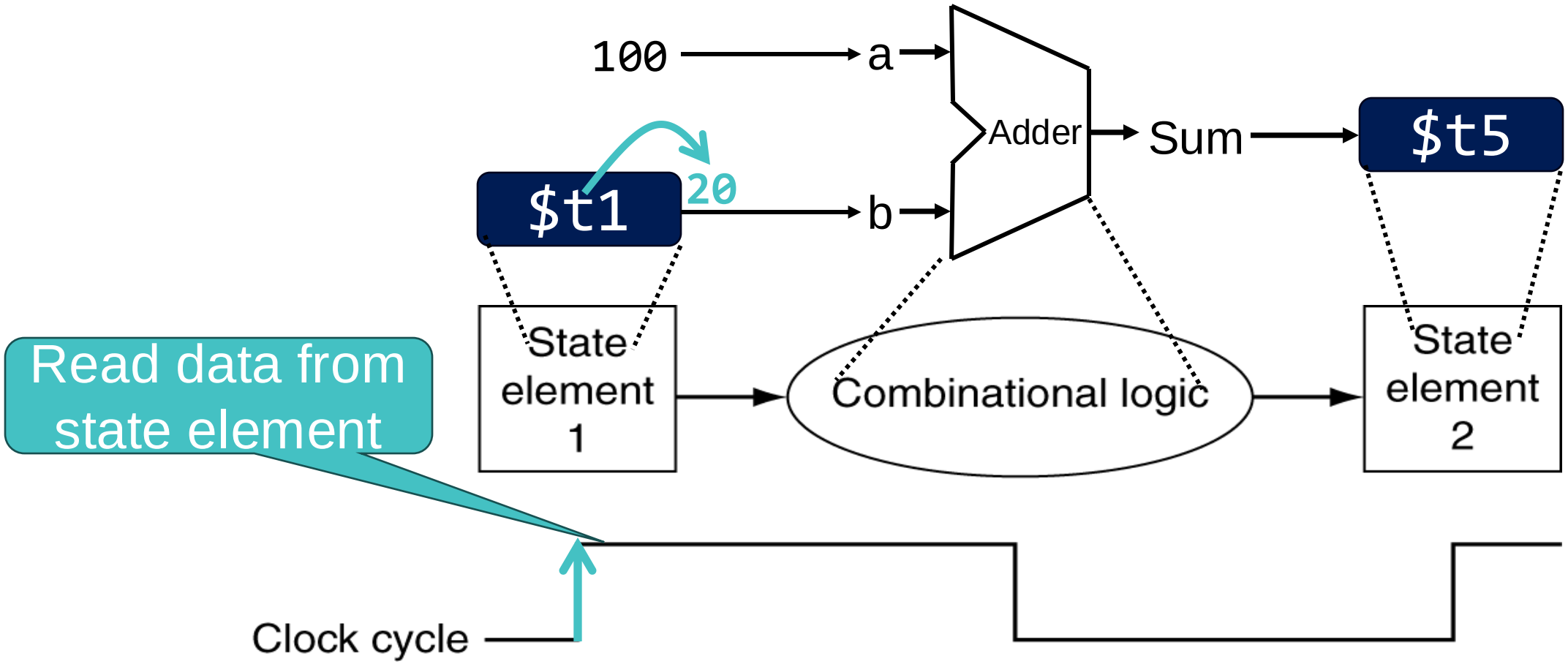
Clocking Methodology Example

```
addi $t5, $t1, 100
```

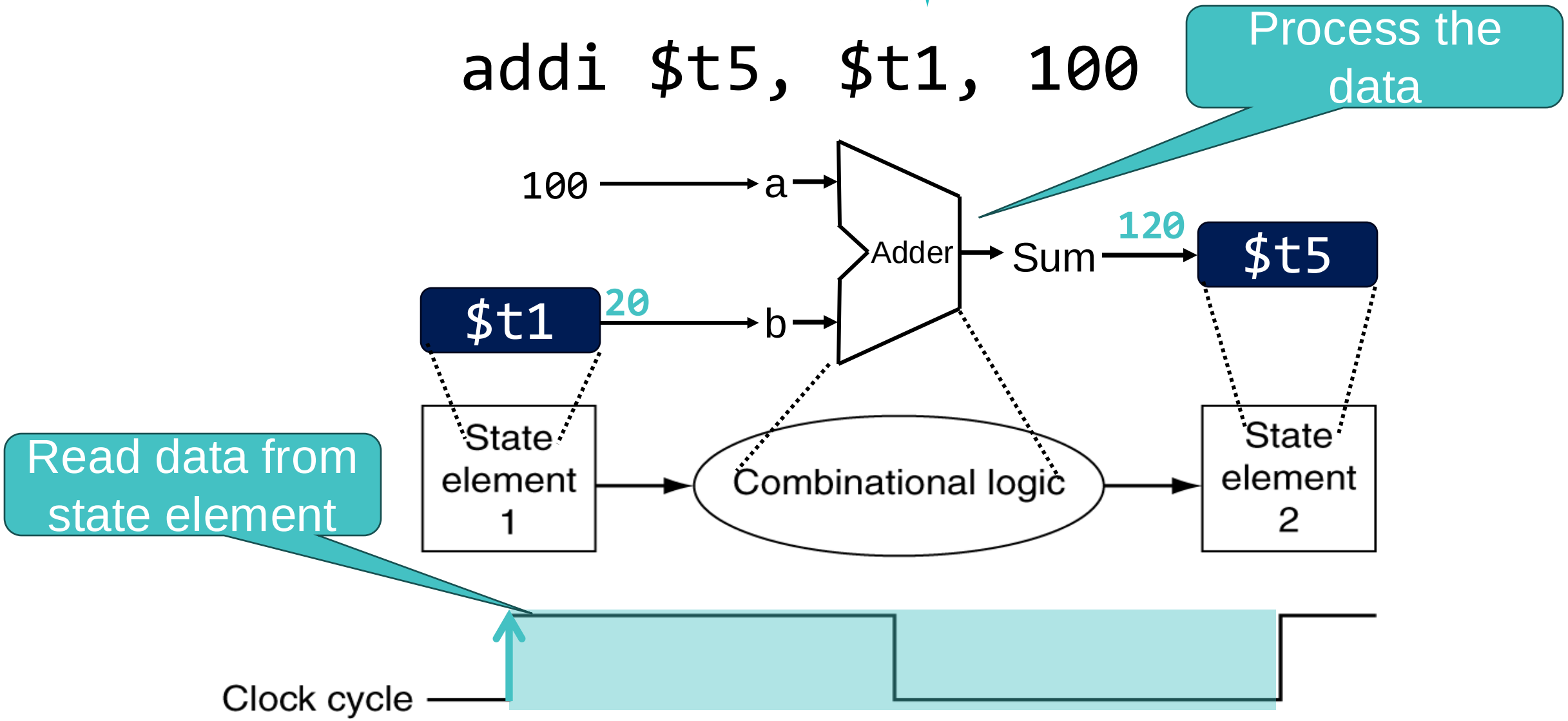


Clocking Methodology Example

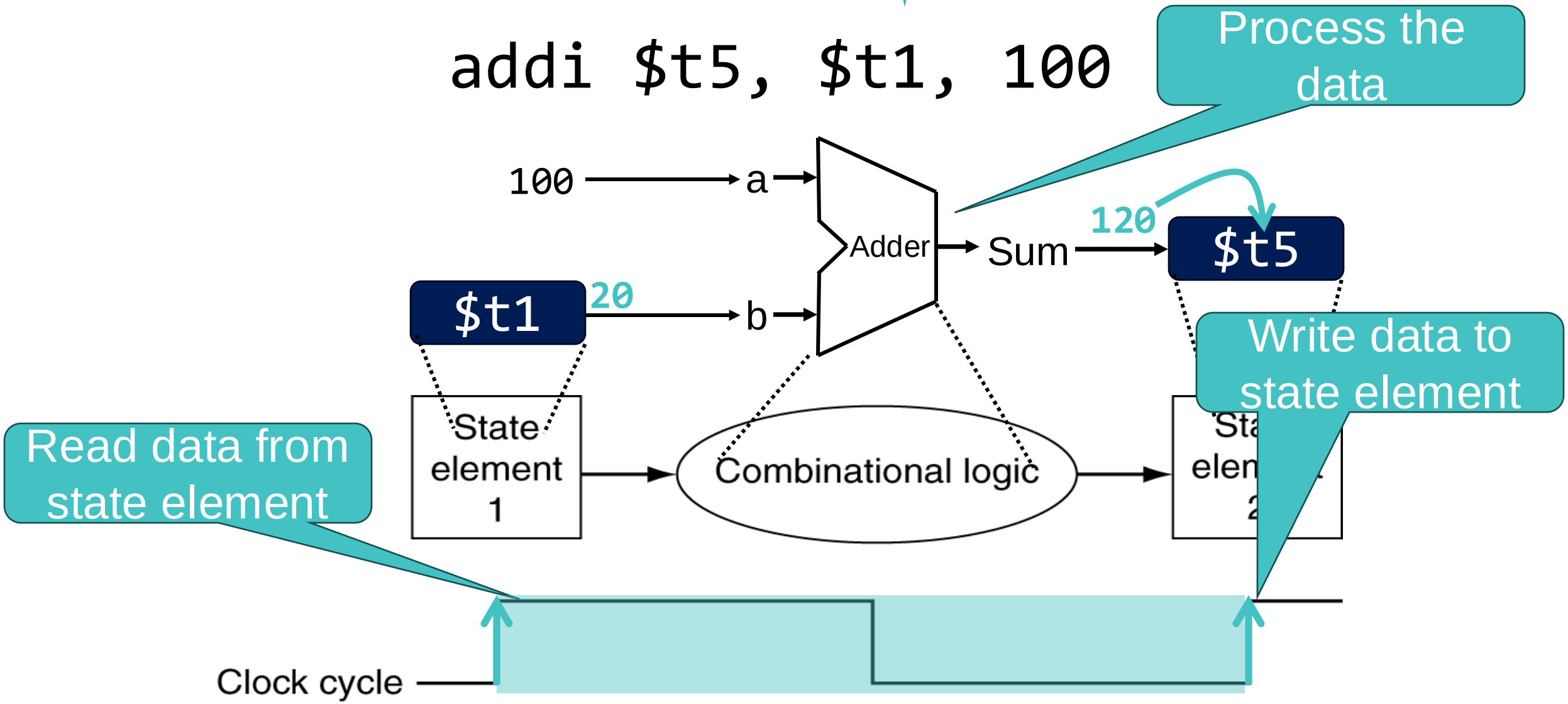
```
addi $t5, $t1, 100
```



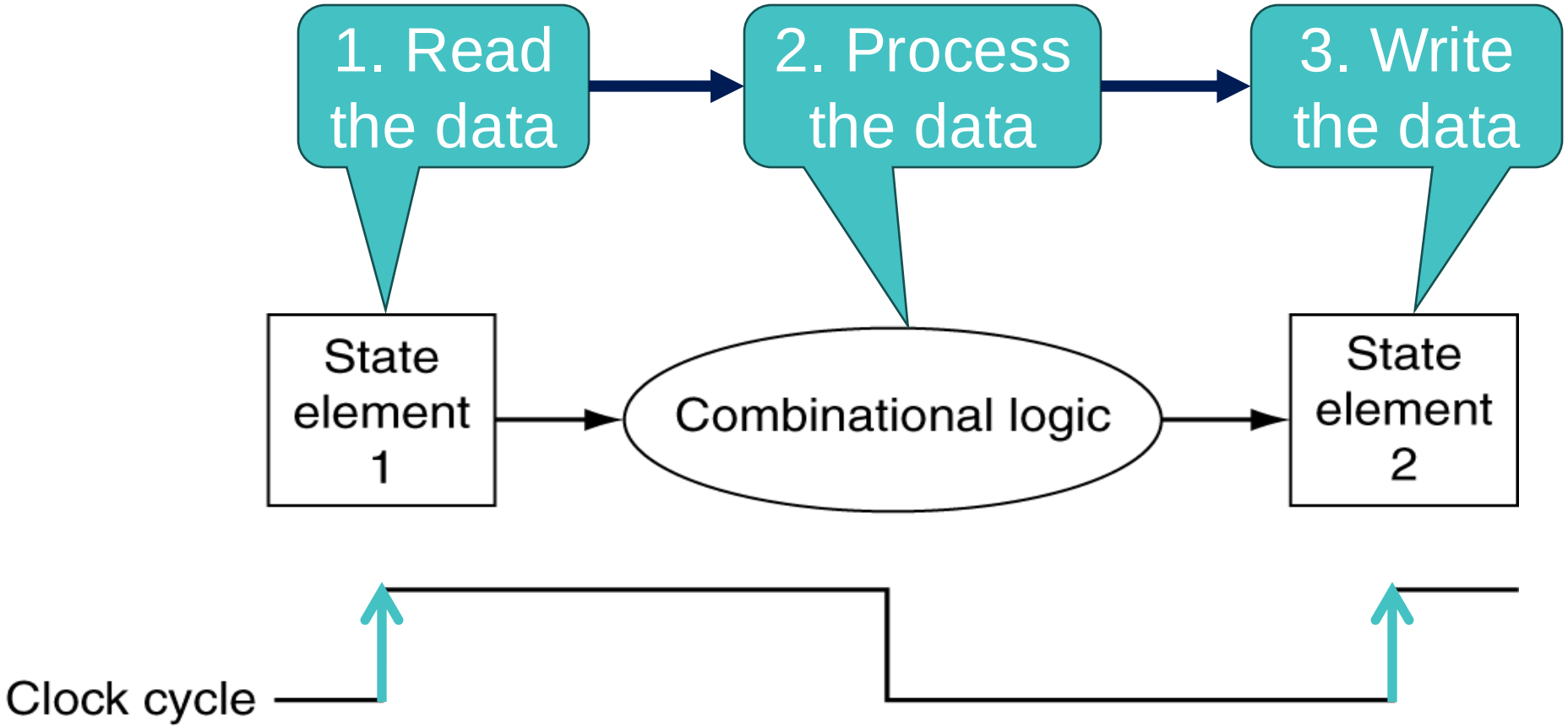
Clocking Methodology Example



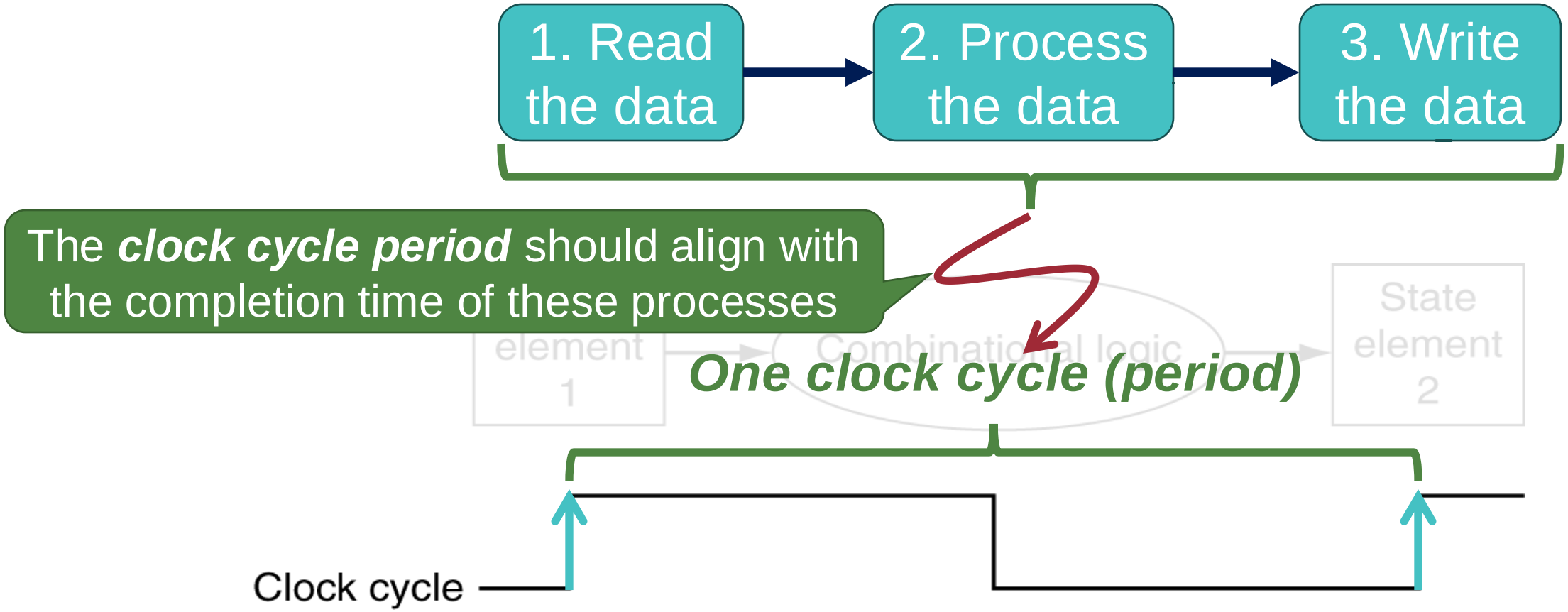
Clocking Methodology Example



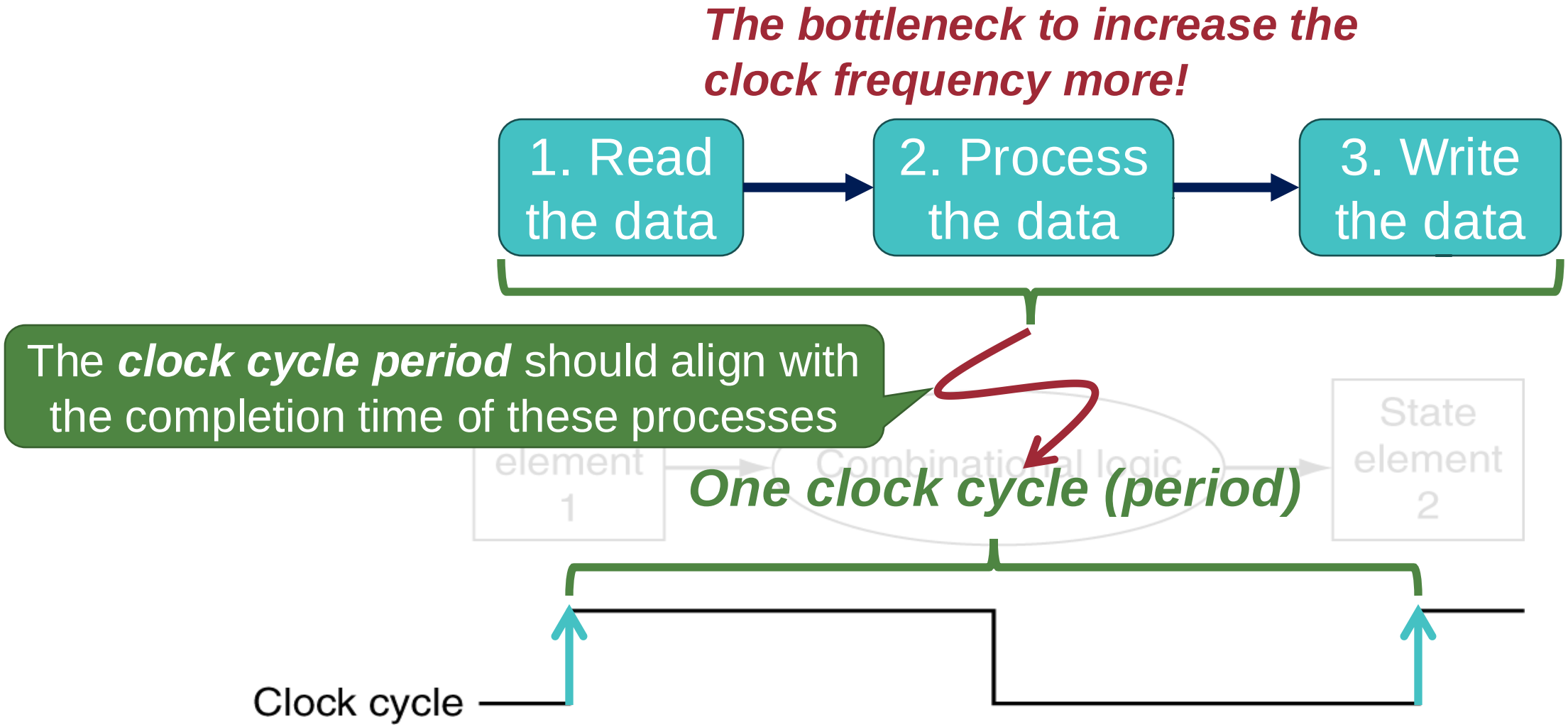
Clocking Methodology Summary



Critical Path



Critical Path



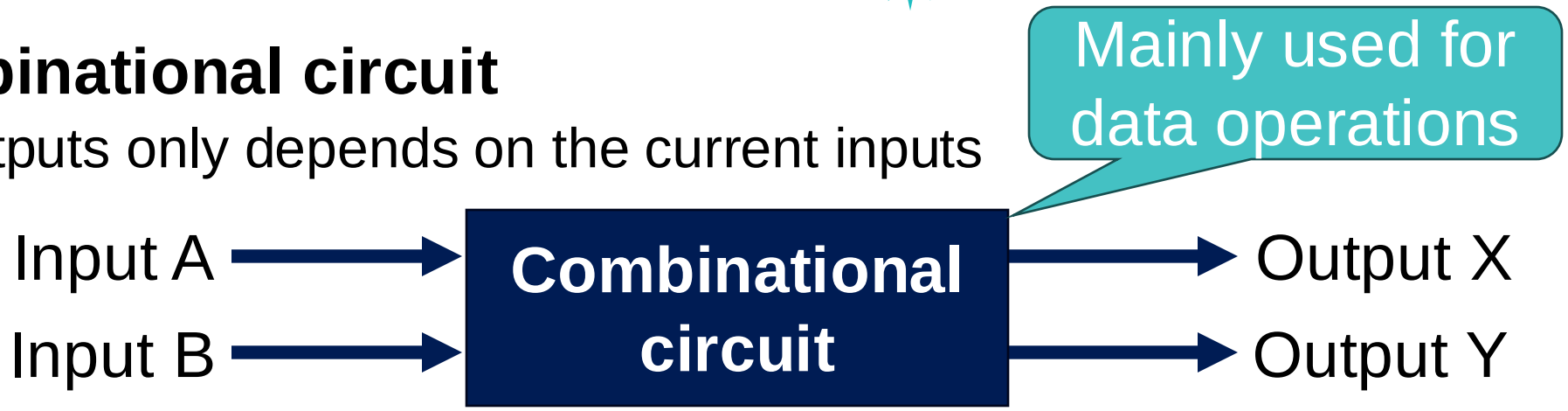
Critical Path

- The clock cycle period is fixed by **the longest delay (= Critical Path)**

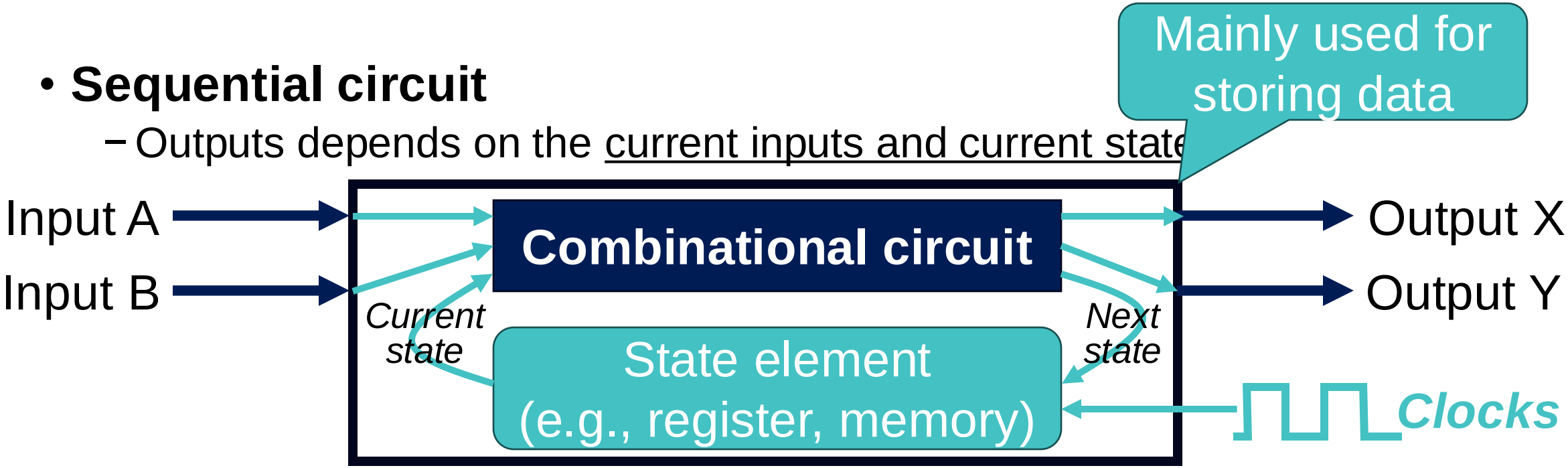
- CPU designers analyze the paths to ensure that all operations can complete within a single clock cycle

Summary

- **Combinational circuit**
  - Outputs only depends on the current inputs



- **Sequential circuit**
  - Outputs depends on the current inputs and current state



Summary

- **Combinational circuit**
  - Outputs only depends on the current inputs

For more details, refer to the
EEE202: Digital Logic and Laboratory course!

- **Sequential circuit**
  - Outputs depends on the current inputs and current state

Question?