

Logic Gates & Boolean Algebra

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Logic Gates - AND

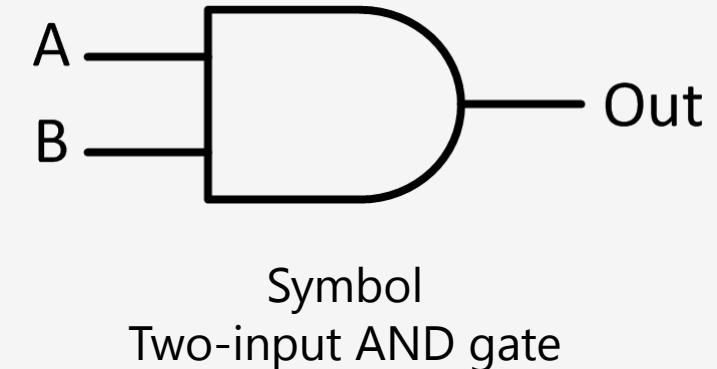
Both inputs of the two-input AND gate must be equal to logic 1 in order to produce an output at logic 1.

How about three-input AND?

How about n-input AND?

A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

Truth table
Two-input AND gate



$$\text{Out} = A \cdot B$$

Functional representation
Two-input AND gate

Logic Gates - OR

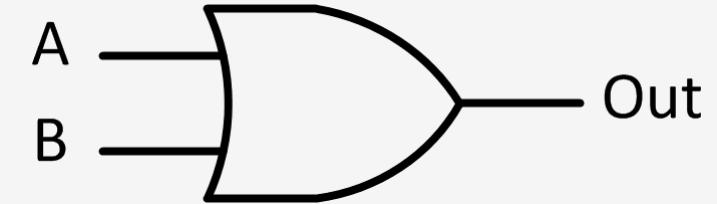
At least one input of the two-input OR gate must be equal to logic 1 in order to produce an output at logic 1.

How about three-input OR?

How about n-input OR?

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

Truth table
Two-input OR gate



Symbol
Two-input OR gate

$$\text{Out} = A + B$$

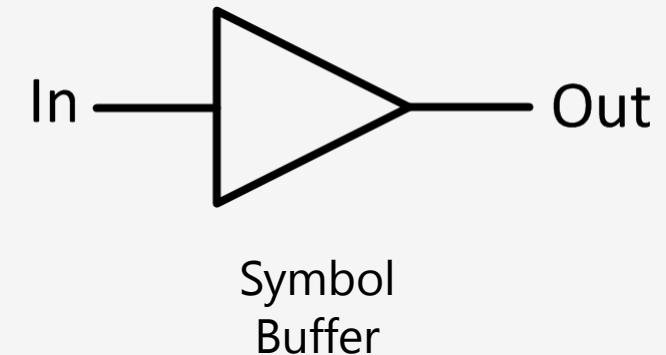
Functional representation
Two-input OR gate

Logic Gates - Buffer

Buffer passes its input, unchanged, to its output.

In	Out
0	0
1	1

Truth table
Buffer



$$\text{Out} = \text{In}$$

Functional representation
Buffer

Logic Gates - Inverter (INV)

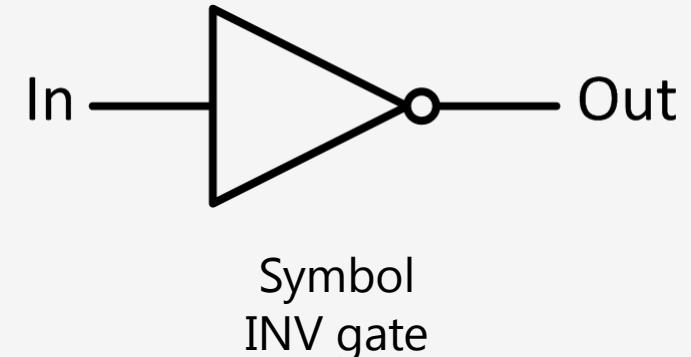
If a single input needs to be complemented, INV gate is used.

We also call it NOT gate.

Sometimes, we use just the bubble symbol.

In	Out
0	1
1	0

Truth table
INV gate



$$\text{Out} = \overline{\text{In}}$$

Functional representation
INV gate

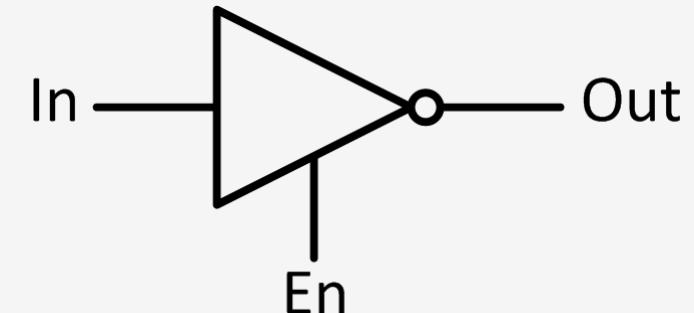
Logic Gates - Tri-State Inverter (E-INV)

E-INV behaves like an inverter when En input is at logic 1. However, when En is lowered to logic 0, its output disconnects from its input.

En	In	Out
0	0	HiZ
0	1	HiZ
1	0	1
1	1	0

Truth table
E-INV gate

A high impedance (HiZ) output is neither driven to a logical high nor low level. Such a signal can be seen as an open circuit.



Symbol
E-INV gate

$$\text{Out} = \begin{cases} \overline{\text{In}}, & \text{En}=1 \\ \text{HiZ}, & \text{En}=0 \end{cases}$$

Functional representation
E-INV gate

Logic Gates - Exclusive OR (XOR)

The two-input XOR gate produces a logic 1 output if the inputs are not equal.

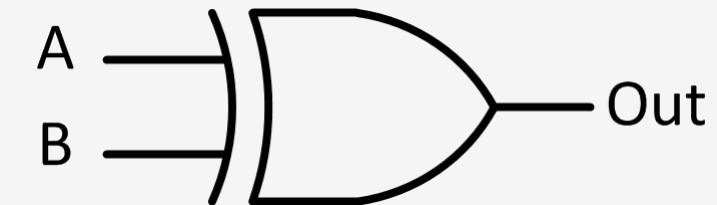
We can use XOR gate as a comparator.

How about three-input XOR?

How about n-input XOR?

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

Truth table
Two-input XOR gate



Symbol
Two-input XOR gate

$$\text{Out} = A \oplus B$$

Functional representation
Two-input XOR gate

Logic Gates - NAND

Both inputs of the two-input NAND gate must be equal to logic 1 in order to produce an output at logic 0.

How about three-input NAND?

How about n-input NAND?

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Truth table
Two-input NAND gate



Symbol
Two-input NAND gate

$$\text{Out} = \overline{A \cdot B}$$

Functional representation
Two-input NAND gate

Logic Gates - NOR

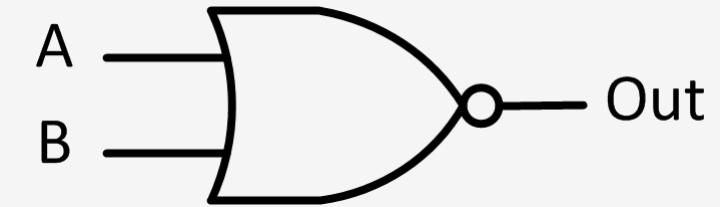
At least one input of the two-input NOR gate must be equal to logic 1 in order to produce an output at logic 0.

How about three-input NOR?

How about n-input NOR?

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Truth table
Two-input NOR gate



$$\overline{Out} = \overline{A + B}$$

Functional representation
Two-input NOR gate

Logic Gates - Exclusive NOR (XNOR)

The two-input XNOR gate produces a logic 0 output if the inputs are not equal.

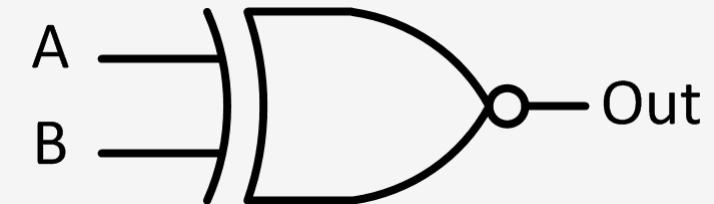
We can use XNOR gate as a comparator.

How about three-input XNOR?

How about n-input XNOR?

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

Truth table
Two-input XNOR gate



Symbol
Two-input XNOR gate

$$\text{Out} = \overline{A \oplus B}$$

Functional representation
Two-input XNOR gate

Logic Gates - Sample Question

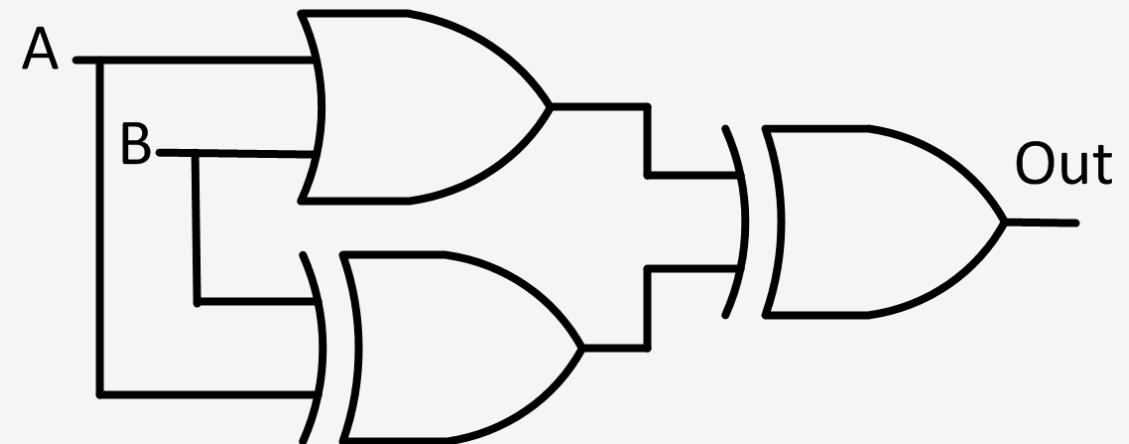
Implement a two-input AND gate using only two-input OR and two-input XOR gates.

Logic Gates - Sample Question - Solution

Implement a two-input AND gate using only two-input OR and two-input XOR gates.

A	B	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

No Flip



Flip

Boolean Algebra - Famous Rules

$$A \cdot 1 = A$$

$$A \cdot 0 = 0$$

$$A \cdot A = A$$

$$A \cdot \bar{A} = 0$$

$$A + 1 = 1$$

$$A + 0 = A$$

$$A + A = A$$

$$A + \bar{A} = 1$$

$$\bar{\bar{A}} = A$$

$$A \cdot B = B \cdot A$$

$$A + B = B + A$$

Identity

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

$$A + (B + C) = (A + B) + C$$

Associative

$$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$$

$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

Distributive

Commutative

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

DeMorgan's

Boolean Algebra - Sample Question

Reduce (=simplify) the following algebraic rule:

$$\text{Out} = A + (\bar{A} \cdot B)$$

Boolean Algebra - Sample Question - Solution

Reduce (=simplify) the following algebraic rules:

$$\begin{aligned}\text{Out} &= A + (\bar{A} \cdot B) \\ &= (A + \bar{A}) \cdot (A + B) \quad (\text{Distributive}) \\ &= 1 \cdot (A + B) \quad (\text{Identity}) \\ &= (A + B) \quad (\text{Identity})\end{aligned}$$

Design & Logic Minimization of Combinational Circuits

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Combinational Circuits - Definition

Common misconception: A combinational circuit is a cascaded form of logic gates without any feedback from the output to any of its inputs.

However, circuits with feedbacks can also be combinational in some cases.

Better definition: A combinational circuit is a combination of logic gates that its output depends only on its current inputs.

Truth Table

The logic function of a combinational circuit is obtained from a truth table that specifies the complete functionality of the circuit.

Sum-Of-Products (SOP): If the output is expressed in terms of AND gates, all output entries that are equal to **one** in the truth table must be grouped together as an OR gate.

Product-Of-Sums (POS): If the output is expressed in terms of OR gates, all the output entries that are equal to **zero** in the truth table must be grouped as an AND gate.

Each SOP term is called “minterm” while each POS term is called “maxterm”

Truth Table - Sample Question

Using the given truth table, determine both SOP and POS output functions.

A	B	C	Out
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Which gate does this truth table represent?

Truth Table - Sample Question - Solution

Using the given truth table, determine both SOP and POS output functions.

A	B	C	Out
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

minterm

$$\text{SOP: } \underbrace{(\bar{A} \cdot \bar{B} \cdot C)} + (\bar{A} \cdot B \cdot \bar{C}) + (A \cdot \bar{B} \cdot \bar{C}) + (A \cdot B \cdot C)$$

$$\text{POS: } \underbrace{(A + B + C) \cdot (A + \bar{B} + \bar{C}) \cdot (\bar{A} + B + \bar{C}) \cdot (\bar{A} + \bar{B} + C)}$$

maxterm

It represents a 3-input XOR gate.

Karnaugh Map

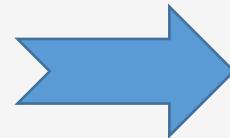
One of the most useful tools in logic design is the use of Karnaugh map (K-map) to minimize combinational logic functions.

Minimized SOP form: Logic 1 entries of the truth table must be grouped together in the K-map.

Minimized POS form: Logic 0 entries of the truth table must be grouped together in the K-map.

Convert Truth Table to Karnaugh Map

	A	B	C	Out
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1



Be Careful!



	BC=00	BC=01	BC=11	BC=10
A=0	0	1	3	2
A=1	4	5	7	6
	BC=00	BC=01	BC=11	BC=10
A=0	0	0	0	1
A=1	0	0	1	1

The sequence of numbers across the top of the Karnaugh map is not in binary sequence which would be 00, 01, 10, 11. It is 00, 01, 11 10, which is Gray code sequence.

3-Variable Karnaugh Map Grouping (SOP Form)

$$\text{Out} = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C$$

	BC	00	01	11	10
A	0	1	1		
	1				

$$\text{Out} = \bar{A}\bar{B}$$

$$\text{Out} = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}B\bar{C}$$

	BC	00	01	11	10
A	0	1	1	1	1
	1				

$$\text{Out} = \bar{A}$$

$$\text{Out} = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

	BC	00	01	11	10
A	0		1	1	
	1	1	1	1	

$$\text{Out} = C$$

$$\text{Out} = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C} + ABC$$

	BC	00	01	11	10
A	0	1		1	
	1	1		1	1

$$\text{Out} = \bar{C}$$

$$\text{Out} = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}B\bar{C} + ABC + ABC$$

	BC	00	01	11	10
A	0	1	1	1	1
	1		1	1	1

$$\text{Out} = \bar{A} + B$$

Karnaugh Map - Sample Question 1

Using the given truth table, determine both **minimized SOP** and **minimized POS** output functions.

A	B	C	Out
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

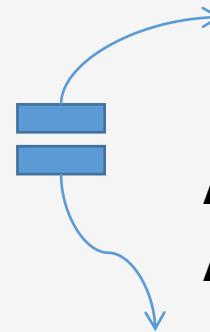
Karnaugh Map - Sample Question 1 - Solution

Using the given truth table, determine both **minimized SOP** and **minimized POS** output functions.

A	B	C	Out
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

	BC=00	BC=01	BC=11	BC=10
A=0	1	1	0	1
A=1	1	1	0	0

$$\text{Min SOP: } \text{Out} = \bar{B} + (\bar{A} \cdot \bar{C})$$



	BC=00	BC=01	BC=11	BC=10
A=0	1	1	0	1
A=1	1	1	0	0

$$\text{Min POS: } \text{Out} = (\bar{B} + \bar{C}) \cdot (\bar{A} + \bar{B})$$

Karnaugh Map - Sample Question 2

Using the given Karnaugh Map, determine both the **minimized SOP** and **minimized POS** output functions.

	CD=00	CD=01	CD=11	CD=10
AB=00	1	1	0	1
AB=01	0	0	0	0
AB=11	0	0	1	1
AB=10	1	0	1	1

Karnaugh Map - Sample Question 2 - Solution

Using the given Karnaugh Map, determine both the **minimized SOP** and **minimized POS** output functions.

	CD=00	CD=01	CD=11	CD=10
AB=00	1	1	0	1
AB=01	0	0	0	0
AB=11	0	0	1	1
AB=10	1	0	1	1

	CD=00	CD=01	CD=11	CD=10
AB=00	1	1	0	1
AB=01	0	0	0	0
AB=11	0	0	1	1
AB=10	1	0	1	1

$$\text{Min SOP: } \text{Out} = (A \cdot C) + (\bar{B} \cdot \bar{D}) + (\bar{A} \cdot \bar{B} \cdot \bar{C})$$

$$\text{Min POS: } \text{Out} = (\bar{B} + C) \cdot (A + \bar{B}) \cdot (\bar{A} + C + \bar{D}) \cdot (A + \bar{C} + \bar{D})$$

Show that
they are
equal.

Karnaugh Map - Sample Question 3

Using the given Karnaugh Map, determine **minimized SOP** output function. The “X” sign corresponds to a “don’t care” condition that represents either logic 0 or logic 1.

	CD=00	CD=01	CD=11	CD=10
AB=00	1	0	0	X
AB=01	1	0	X	0
AB=11	X	1	1	0
AB=10	1	0	1	0

Karnaugh Map - Sample Question 3 - Solution

Using the given Karnaugh Map, determine **minimized SOP** output function. The “X” sign corresponds to a “don’t care” condition that represents either logic 0 or logic 1.

	CD=00	CD=01	CD=11	CD=10
AB=00	1	0	0	X
AB=01	1	0	X	0
AB=11	X	1	1	0
AB=10	1	0	1	0

We use “don’t care”s, if they help us make larger groups while covering logic 1s in SOP form and logic 0s in POS form.

$$\text{Min SOP: } (\bar{C} \cdot \bar{D}) + (A \cdot B \cdot D) + (A \cdot C \cdot D)$$

Homework Assignment 1

Due Date: February 2, 11:30PM on BeachBoard Dropbox

Late Submission Policy:

One day delay (Second Due Date: Feb. 3, 11:30PM): 25% deduction

Two days delay (Third Due Date: Feb. 4, 11:30PM): 50% deduction

More than two days delay: No credit

Academic Integrity: There is zero tolerance for cheating, plagiarism, or any other act of violation of Academic Integrity.

Combinational Logic Blocks

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

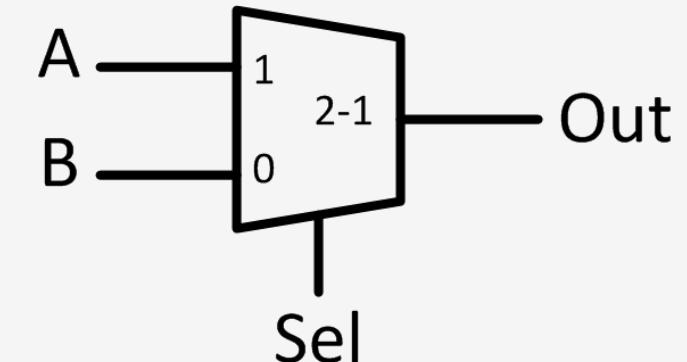
Multiplexer (MUX)

Multiplexer is a combinational logic block that allows one of its inputs to be routed to a single output.

In the given 2-1 MUX, when $\text{sel} = 1$, output OUT will be equal to input A. When $\text{sel} = 0$, output OUT will be equal to input B.

Sel	A	B	Out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Truth table 2-1 MUX



Symbol 2-1 MUX

$$\text{Out} = (\text{Sel} \cdot \text{A}) + (\overline{\text{Sel}} \cdot \text{B})$$

Functional representation
2-1 MUX

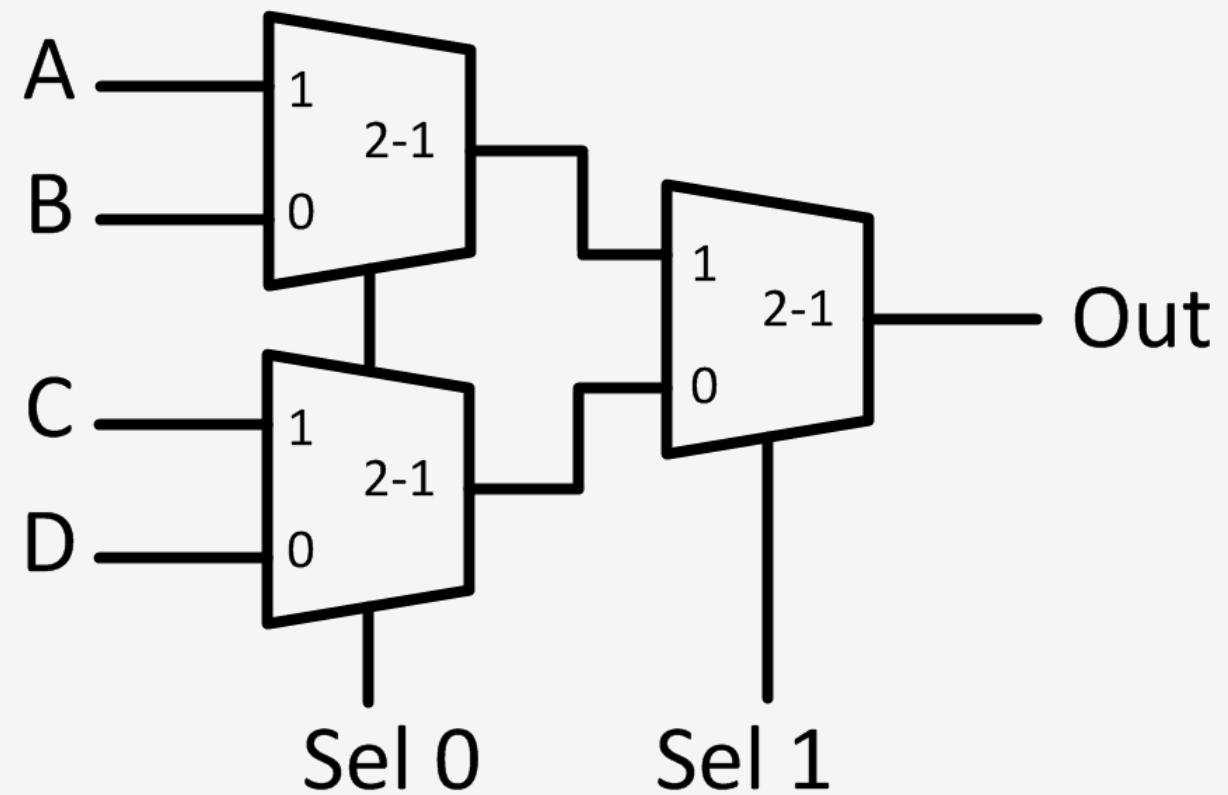
Multiplexer - Sample Question 1

Implement a 4-1 MUX using 2-1 MUXs.

Multiplexer - Sample Question 1 - Solution

Implement a 4-1 MUX using 2-1 MUXs.

Sel 1	Sel 0	Out
0	0	D
0	1	C
1	0	B
1	1	A



Multiplexer - Sample Question 2

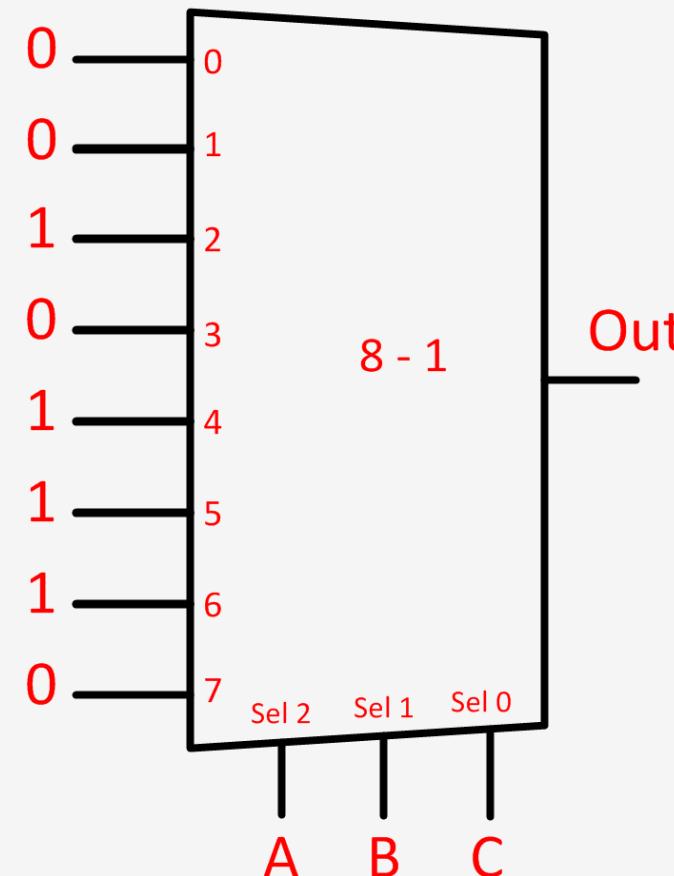
Implement the following function using one 8-1 MUX.

$$\text{Out} = (A \cdot \bar{B}) + (B \cdot \bar{C})$$

Multiplexer - Sample Question 2 - Solution

Implement the following function using one 8-1 MUX.

A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

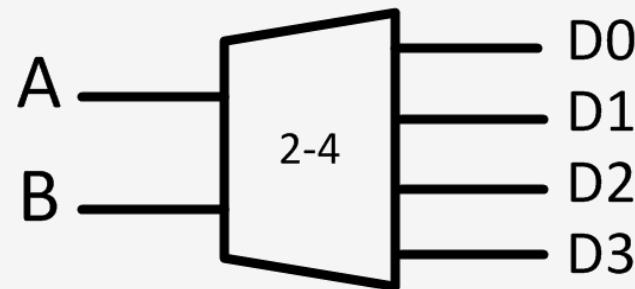


Decoder

Decoder is a combinational logic block that receives n inputs and produces 2^n decoded outputs.

A	B	D3	D2	D1	D0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Truth table 2-4 Decoder



Symbol 2-4 Decoder

$$D0 = \overline{A} \cdot \overline{B}, D1 = \overline{A} \cdot B, D2 = A \cdot \overline{B}, D3 = A \cdot B$$

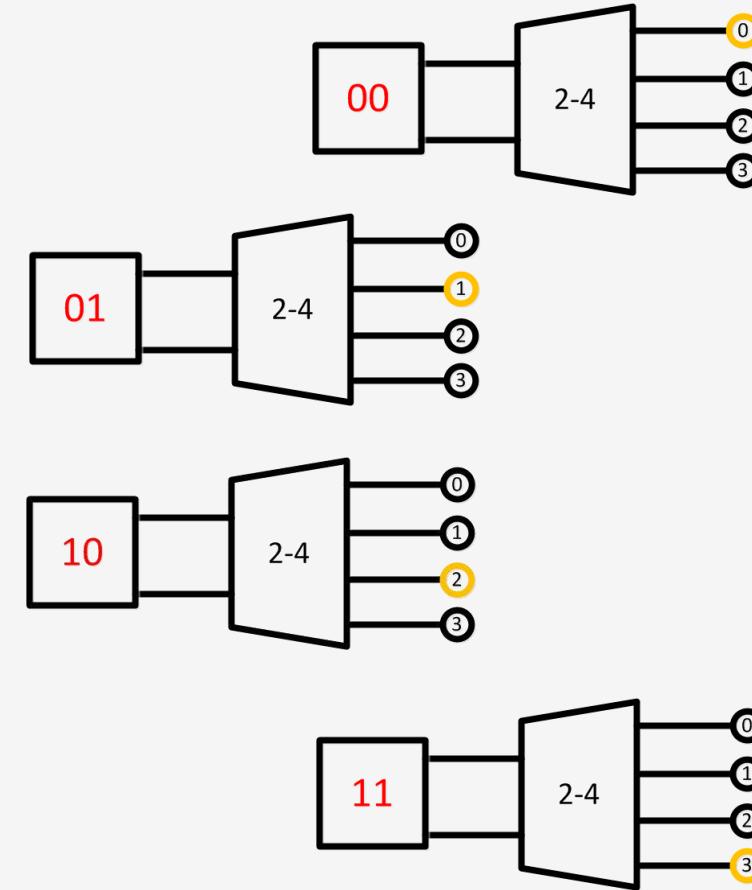
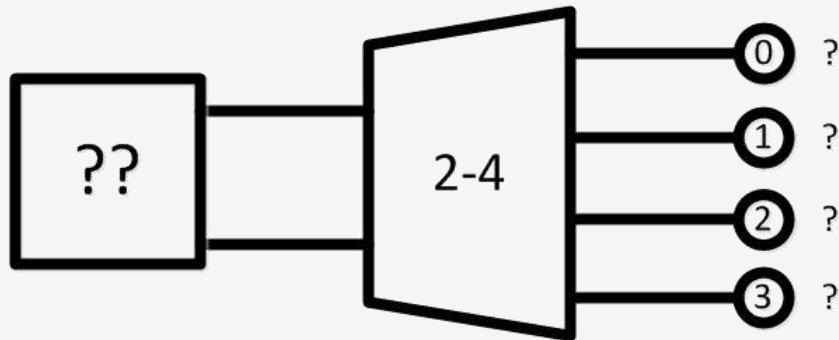
Functional representation 2-4 Decoder

Decoder - Sample Question

Implement a system that converts 2-bit binary numbers to their decimal forms.

Decoder - Sample Question - Solution

Implement a system that converts 2-bit binary numbers to their decimal forms.

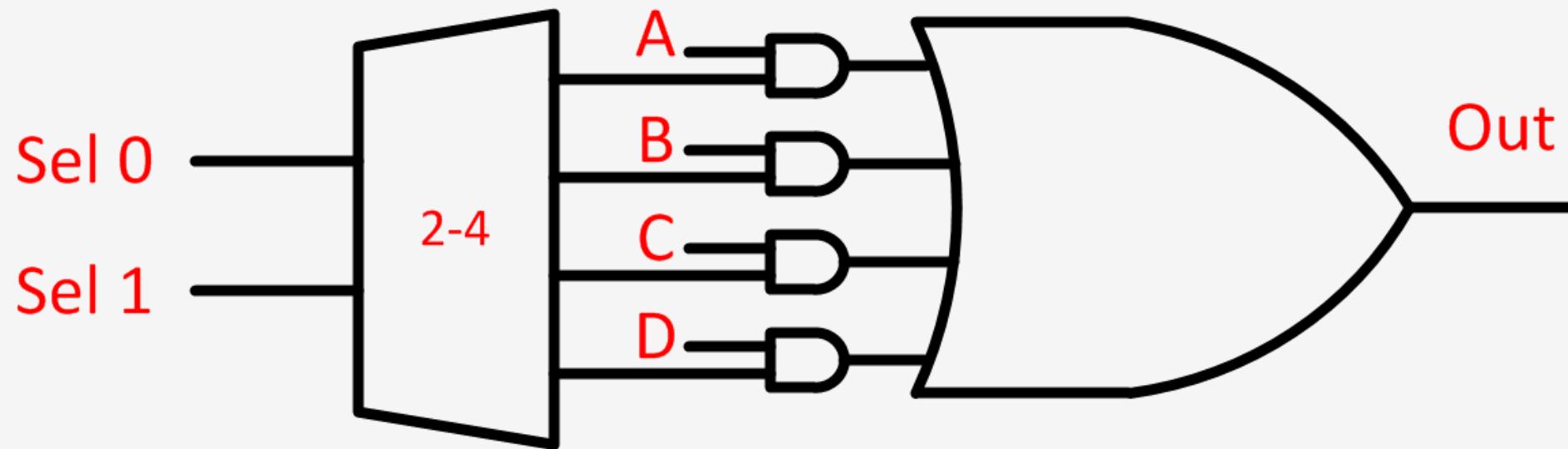


Decoder & Multiplexer- Sample Question

Implement a 4-1 MUX using a 2-4 decoder and logic gates.

Decoder & Multiplexer- Sample Question - Solution

Implement a 4-1 MUX using a 2-4 decoder and logic gates.

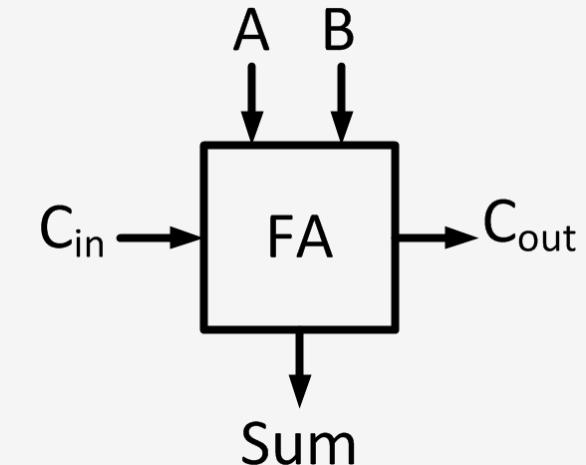


One-Bit Full Adder

The one-bit full adder simply adds the contents of its three inputs, A, B, and C_{in} .

A	B	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Truth table
1-Bit FA



$$\begin{aligned} \text{Sum} &= A \oplus B \oplus C_{in} \\ C_{out} &= C_{in} \cdot (A \oplus B) + (A \cdot B) \end{aligned}$$

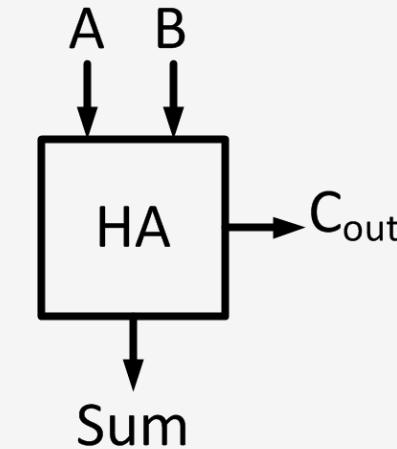
Functional representation
1-Bit FA

One-Bit Half Adder

The one-bit half adder has only two inputs, A and B with no C_{in} . The A and B inputs are added to generate the Sum and C_{out} outputs.

A	B	C_{out}	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Truth table
1-Bit HA



Symbol 1-Bit HA

$$\begin{aligned} \text{Sum} &= A \oplus B \\ C_{out} &= A \cdot B \end{aligned}$$

Functional representation
1-Bit HA

Adders & Subtractors

Amin Rezaei

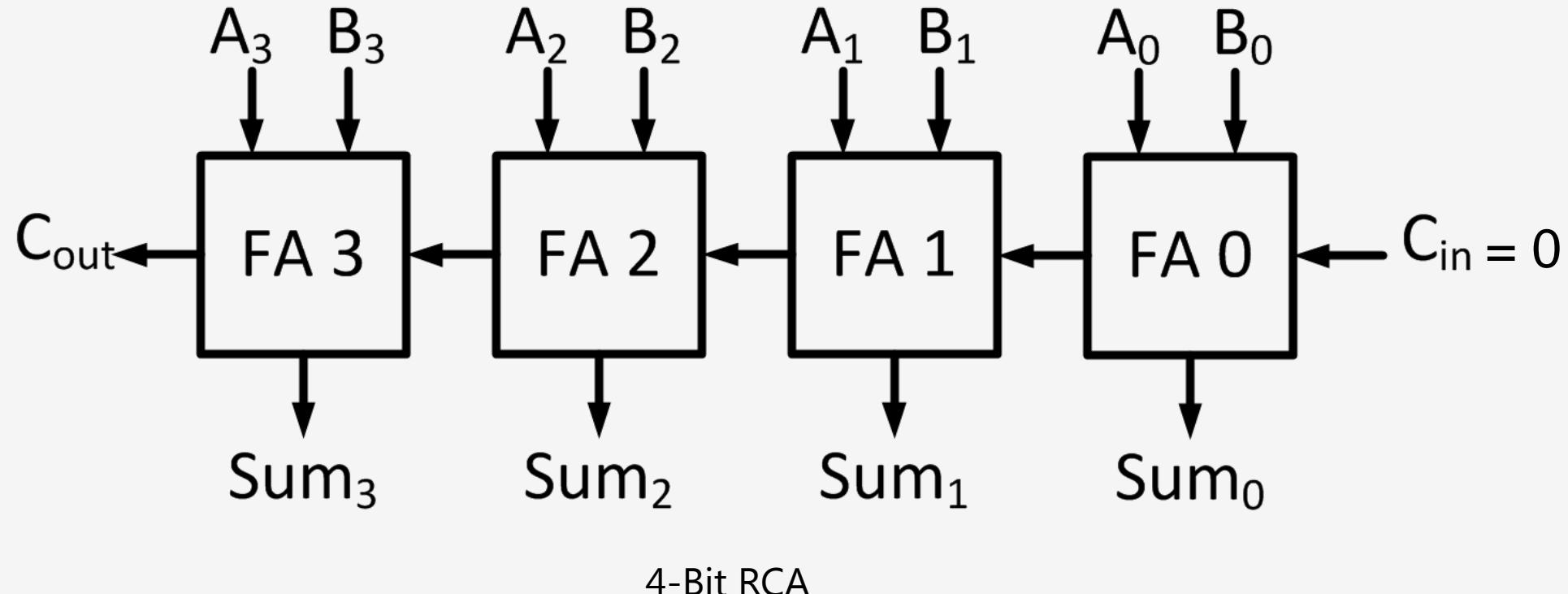
CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Ripple-Carry Adder (RCA)

RCA is a cascaded configuration of multiple one-bit full adders.



Ripple-Carry Adder - Formula

$$P_i = A_i \oplus B_i, \quad G_i = A_i \cdot B_i$$

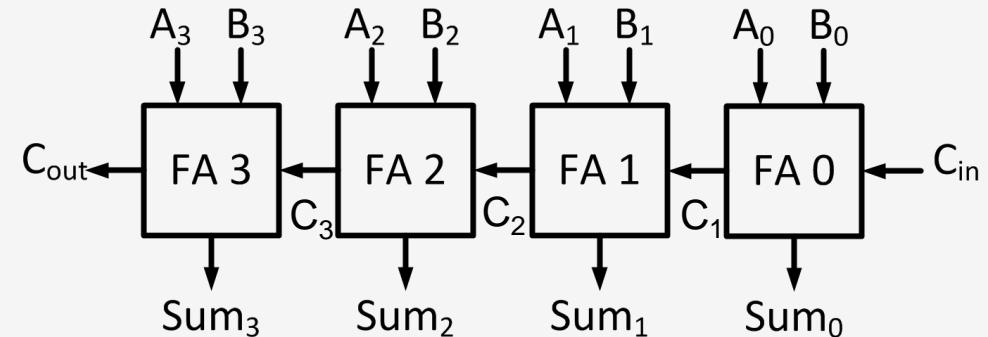
$$\text{Sum}_0 = P_0 \oplus C_{\text{in}}$$

$$\text{Sum}_1 = P_1 \oplus (G_0 + P_0 C_{\text{in}}) \rightarrow C_1 = C_{\text{out}} \text{ of FA 0} = C_{\text{in}} \text{ of FA 1}$$

$$\text{Sum}_2 = P_2 \oplus (G_1 + P_1 G_0 + P_1 P_0 C_{\text{in}}) \rightarrow C_2 = C_{\text{out}} \text{ of FA 1} = C_{\text{in}} \text{ of FA 2}$$

$$\text{Sum}_3 = P_3 \oplus (G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{\text{in}}) \rightarrow C_3 = C_{\text{out}} \text{ of FA 2} = C_{\text{in}} \text{ of FA 3}$$

$$C_{\text{out}} = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{\text{in}}$$



What is the formula for Sum_i ?

Ripple-Carry Adder - Propagation Delay

Propagation delay is time elapsed between the insertion of an input and occurrence of the corresponding output.

Sum_0 is valid only after the propagation delay of FA 0. In the same way, Sum_3 is valid only after the joint propagation delays of FA 0 to FA 3.

In simple words, the final result of RCA is valid only after the joint propagation delays of all the FAs inside it.

For an n-Bit RCA, the propagation delay is n.

RCA - Sample Question

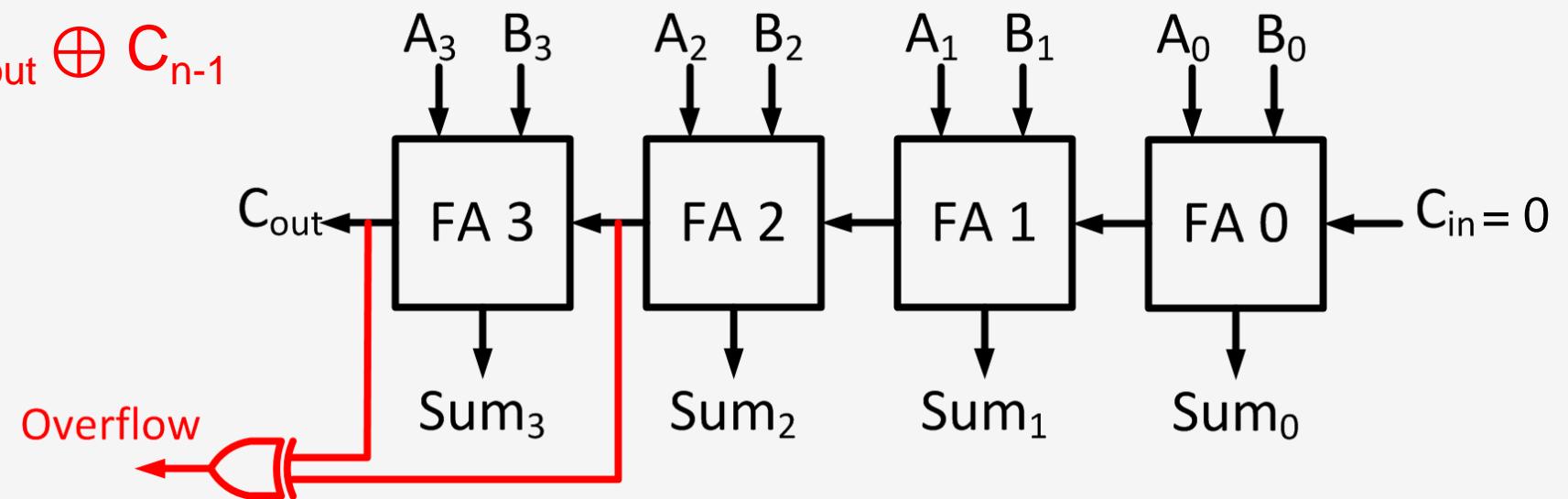
Design a 4-bit RCA with overflow flag. Overflow flag indicates that the signed result is too large to fit in the 4-bit output.

RCA - Sample Question - Solution

Design a 4-bit RCA with overflow flag. Overflow flag indicates that the signed result is too large to fit in the 4-bit output.

$$4\text{-bit Overflow} = C_{\text{out}} \oplus C_3$$

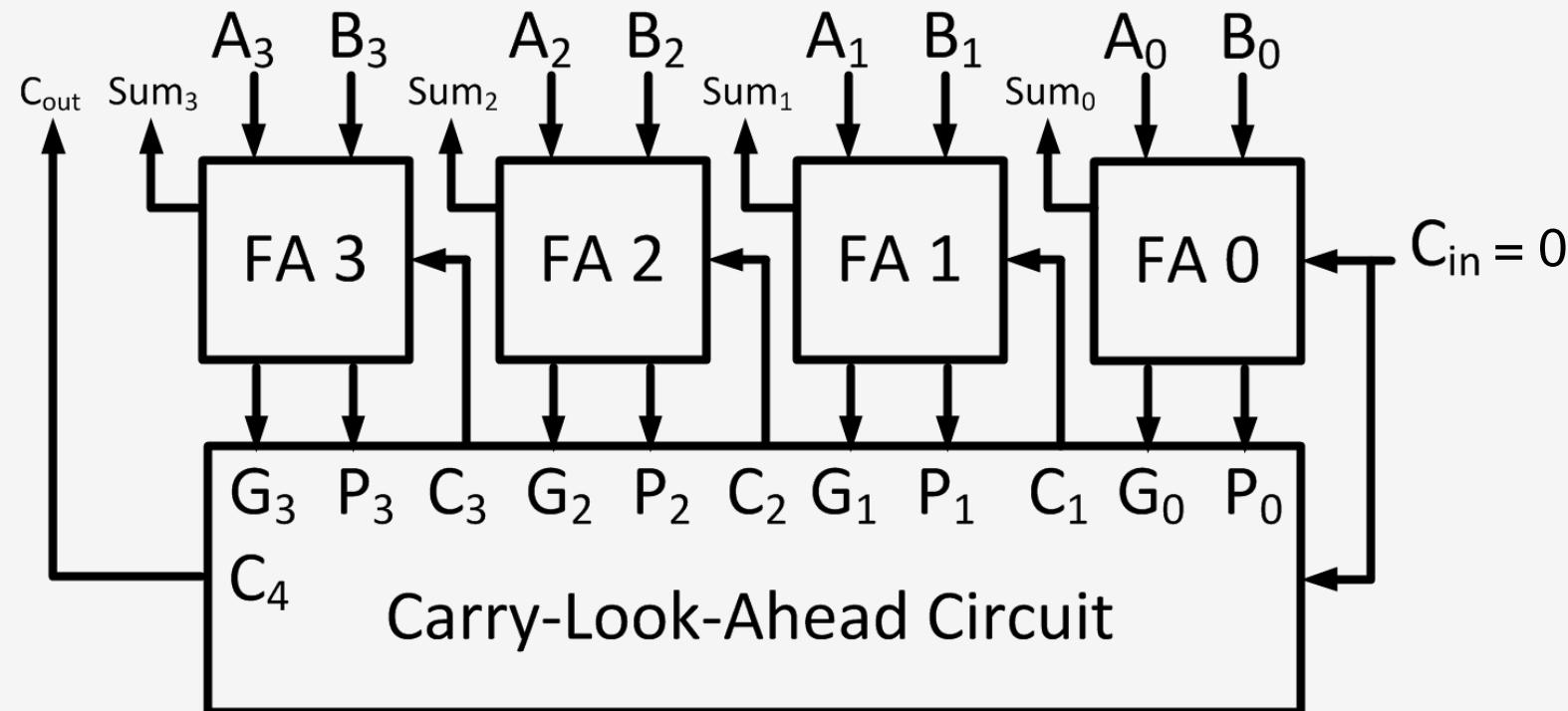
$$n\text{-bit Overflow} = C_{\text{out}} \oplus C_{n-1}$$



The overflow flag is meaningless when unsigned numbers are added or subtracted.

Carry-Look-Ahead Adder (CLA)

The idea behind CLA is to create a topology where carry-in bits to all one-bit full adders are available simultaneously.



Carry-Look-Ahead Adder - Formula

$$P_i = A_i \oplus B_i, \quad G_i = A_i \cdot B_i$$

$$\text{Sum}_i = P_i \oplus C_i$$

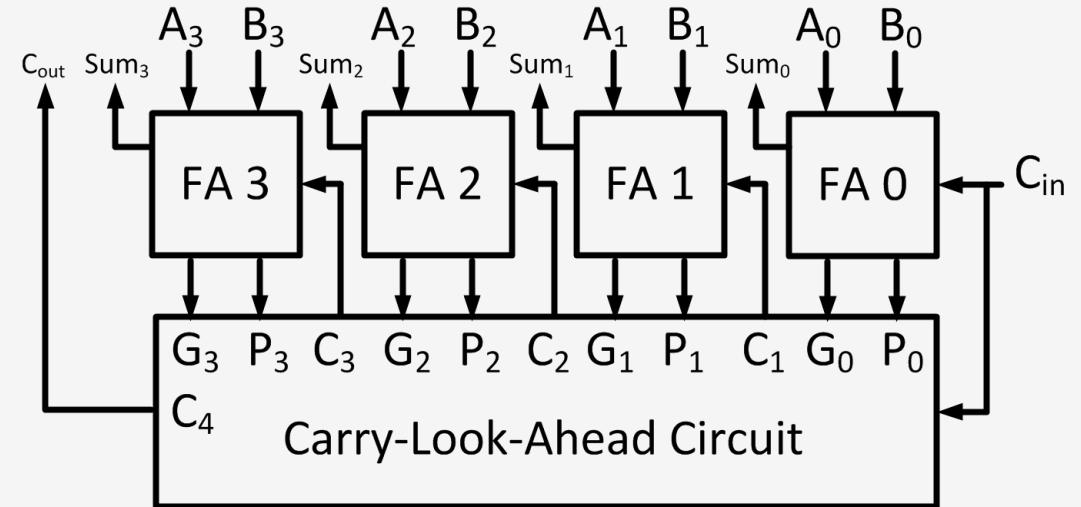
$$C_0 = C_{\text{in}}$$

$$C_1 = G_0 + P_0 C_{\text{in}}$$

$$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_{\text{in}}$$

$$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{\text{in}}$$

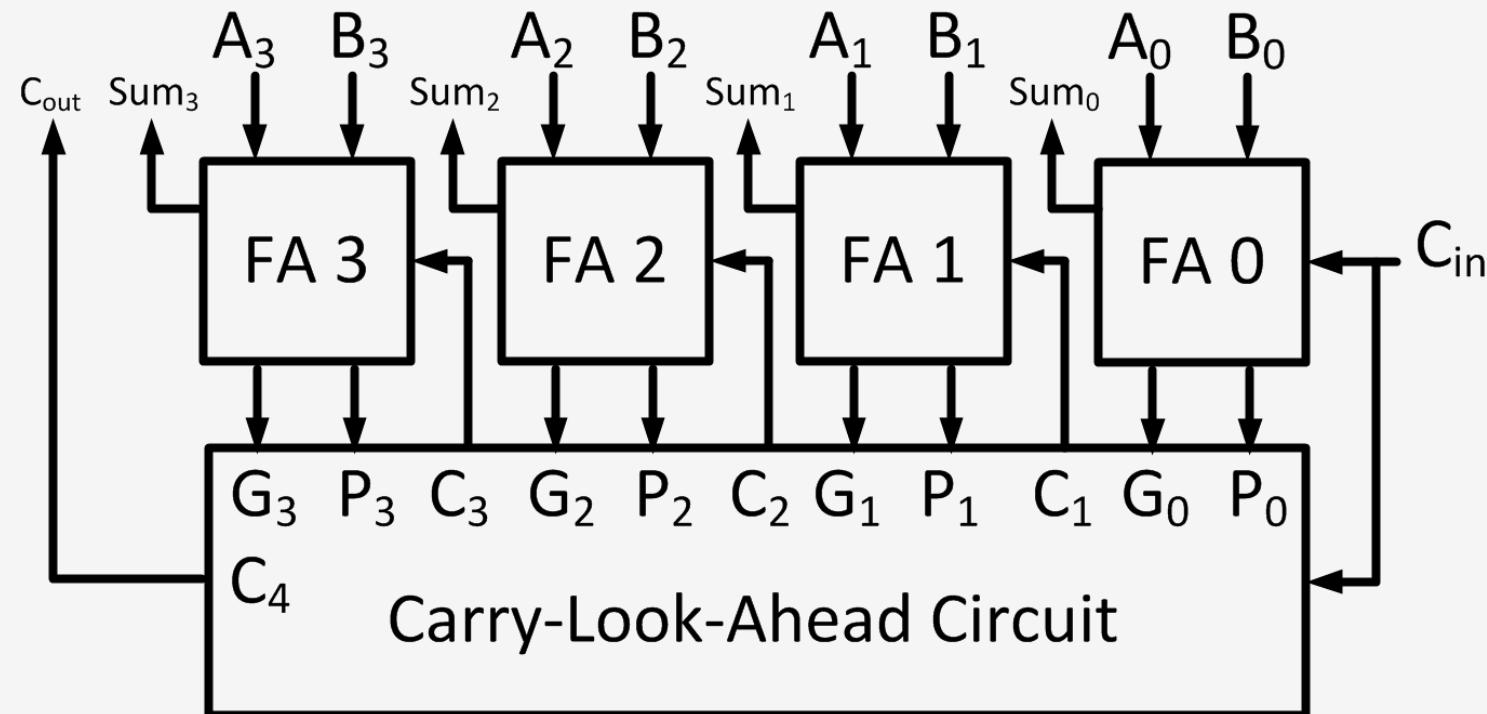
$$C_{\text{out}} = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{\text{in}}$$



What is the propagation delay of CLA?

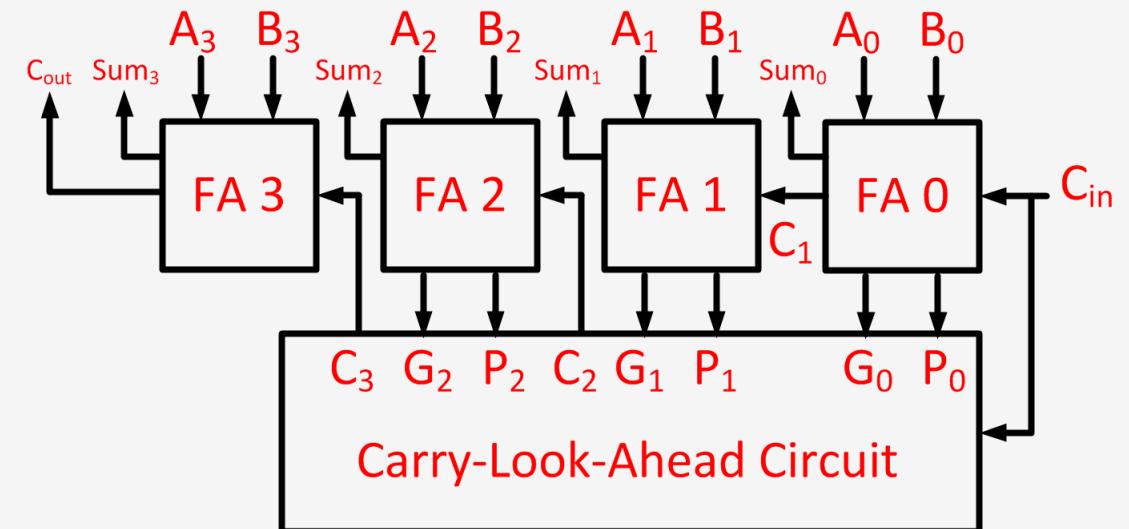
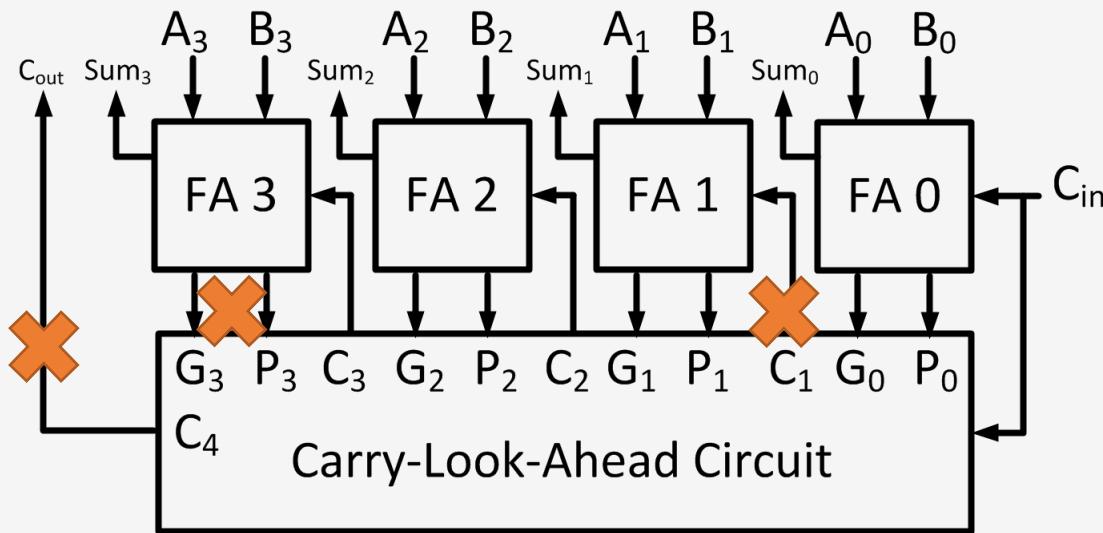
CLA - Sample Question

Reduce the area of the given 4-bit CLA by removing redundant components.



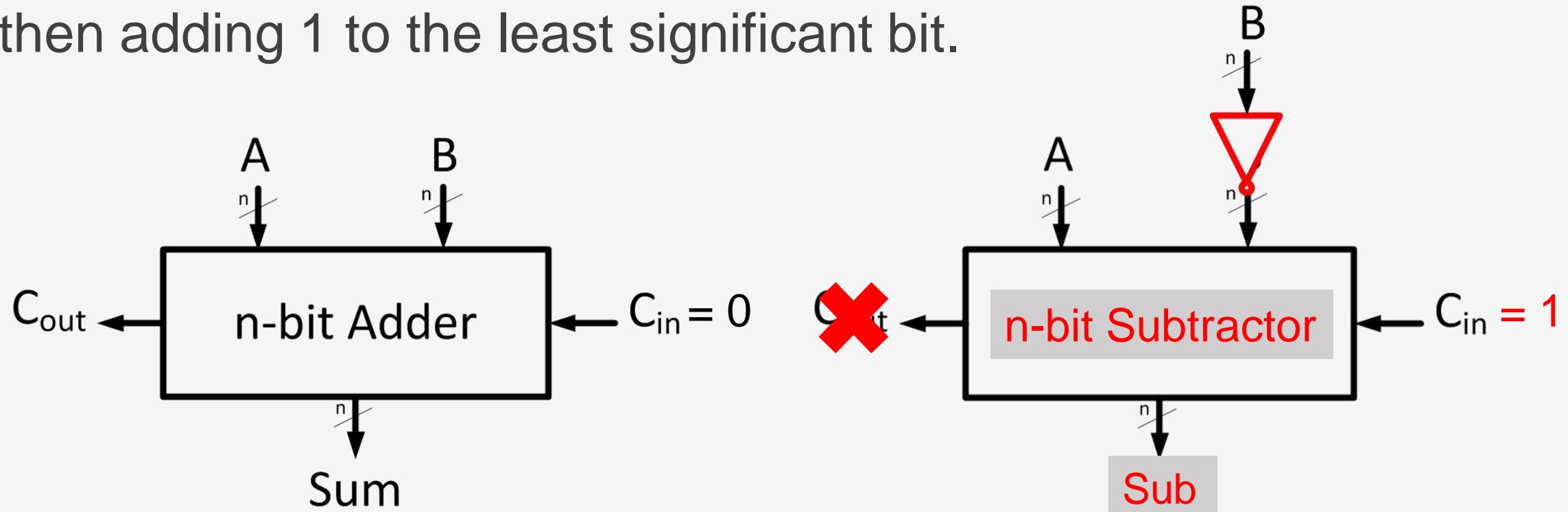
CLA - Sample Question - Solution

Reduce the area of the given 4-bit CLA by removing redundant components.



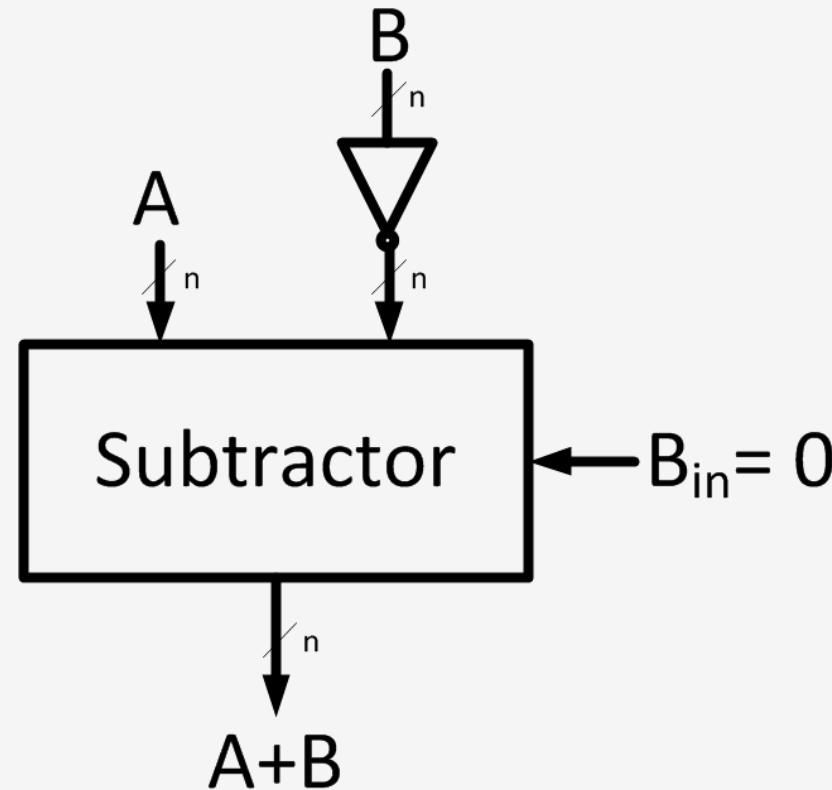
Subtractors

Subtraction is performed by a technique called twos (2s) complement addition. Twos complement addition first requires complementing one of the adder inputs (1s complement) and then adding 1 to the least significant bit.



Subtractors - Sample Question

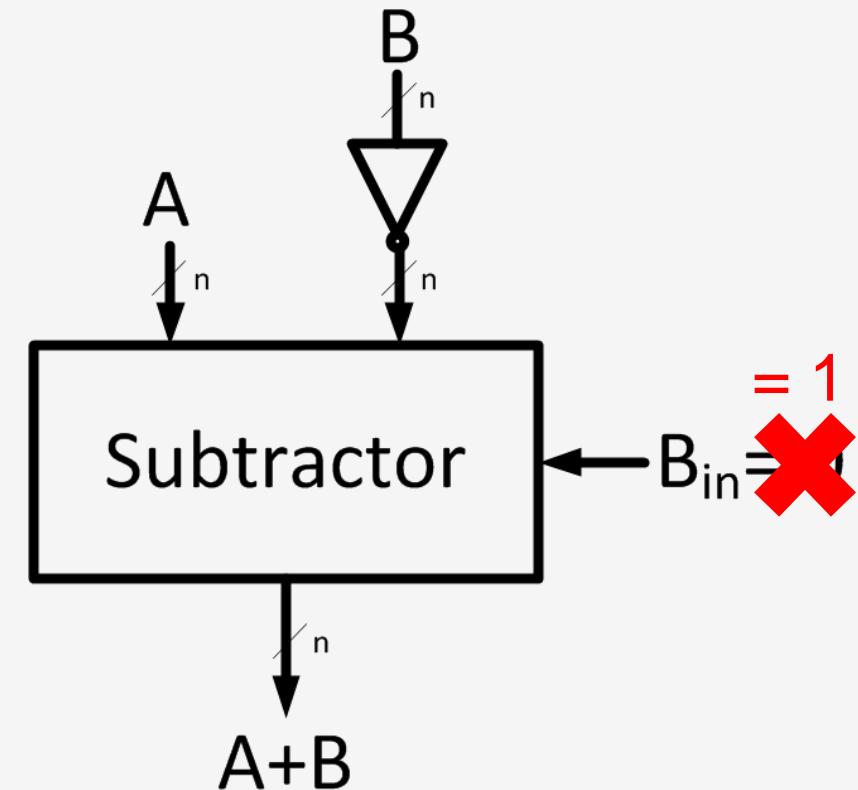
Kevin has designed the following adder given a subtractor.
What is the design problem?



Subtractors - Sample Question - Solution

Kevin has designed the following adder given a subtractor.
What is the design problem?

In order to convert a subtractor to an adder, we need to use 2s complement of one of the operands. Thus, B_{in} should be connected to 1.



Laboratory Assignment 1

Due Date: February 9, 11:30PM on BeachBoard Dropbox

Late Submission Policy:

One day delay (Second Due Date: Feb. 10, 11:30PM): 25% deduction

Two days delay (Third Due Date: Feb. 11, 11:30PM): 50% deduction

More than two days delay: No credit

Demo Time:

Feb. 12, 8:00AM - 10:45AM

Feb. 12, 1:00PM - 3:45PM

Academic Integrity: There is zero tolerance for cheating, plagiarism, or any other act of violation of Academic Integrity.

Basic Memory Devices

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Quiz 1

Please take Quiz 1 on BeachBoard.

Objective: Evaluate your knowledge on combinational circuits.

Time: 10 Minutes.

Note: Attendance guarantees 50%.

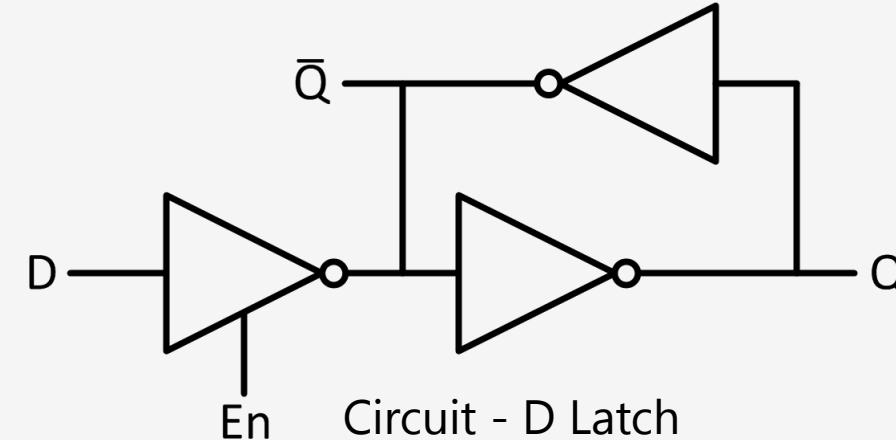
D Latch

D Latch is the most basic memory element in logic design.

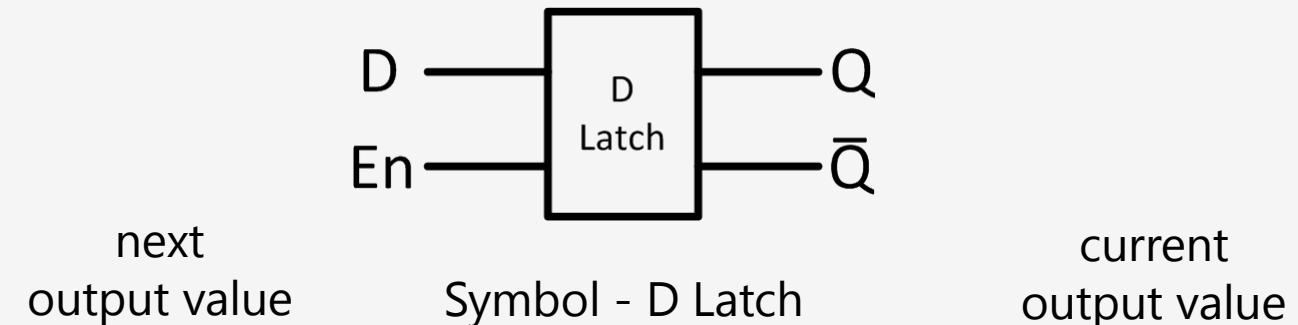
Latch is level-sensitive.

En	D	Q	\bar{Q}
0	0	Q	\bar{Q}
0	1	Q	\bar{Q}
1	0	0	1
1	1	1	0

Truth table - D Latch



Circuit - D Latch



next
output value

Symbol - D Latch

current
output value

$$Q' = (En \cdot D) + (\overline{En} \cdot \bar{Q})$$

Functional representation - D Latch

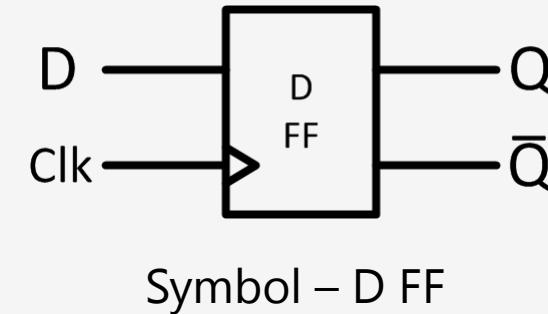
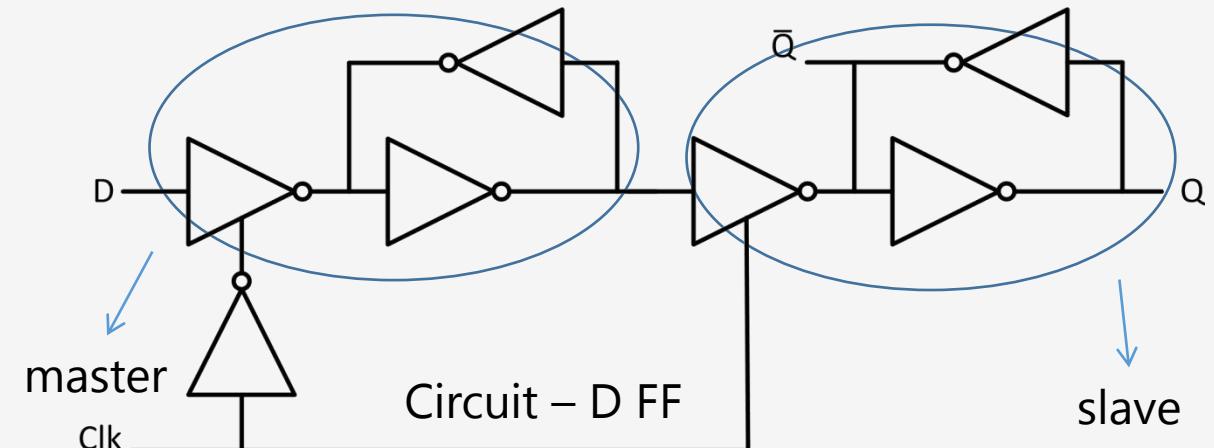
D Flip Flop (FF)

D Flip Flop is another important memory element in logic design.

Flip Flop is edge-triggered.

Clk	D	Q	\bar{Q}
\uparrow	0	Q	\bar{Q}
\uparrow	1	Q	\bar{Q}
\uparrow	0	0	1
\uparrow	1	1	0

Truth table – D FF

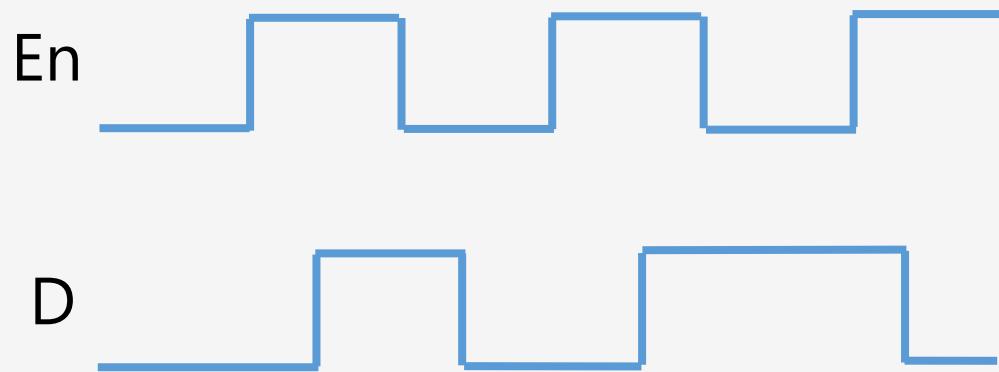


$$Q' = (\uparrow \text{Clk} \cdot D) + (\uparrow \text{Clk} \cdot \bar{Q})$$

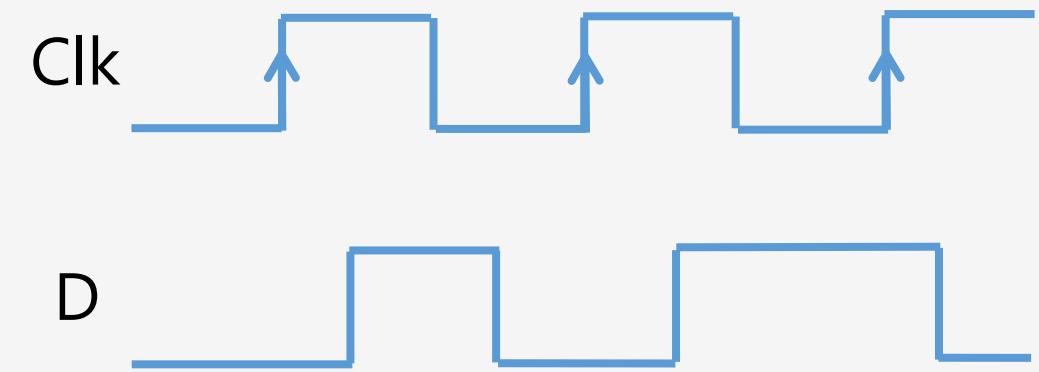
Functional representation – D FF

Latch & Flip Flop - Sample Question

Consider the following input signal D. Compute output signal Q for D Latch and D Flip Flop.



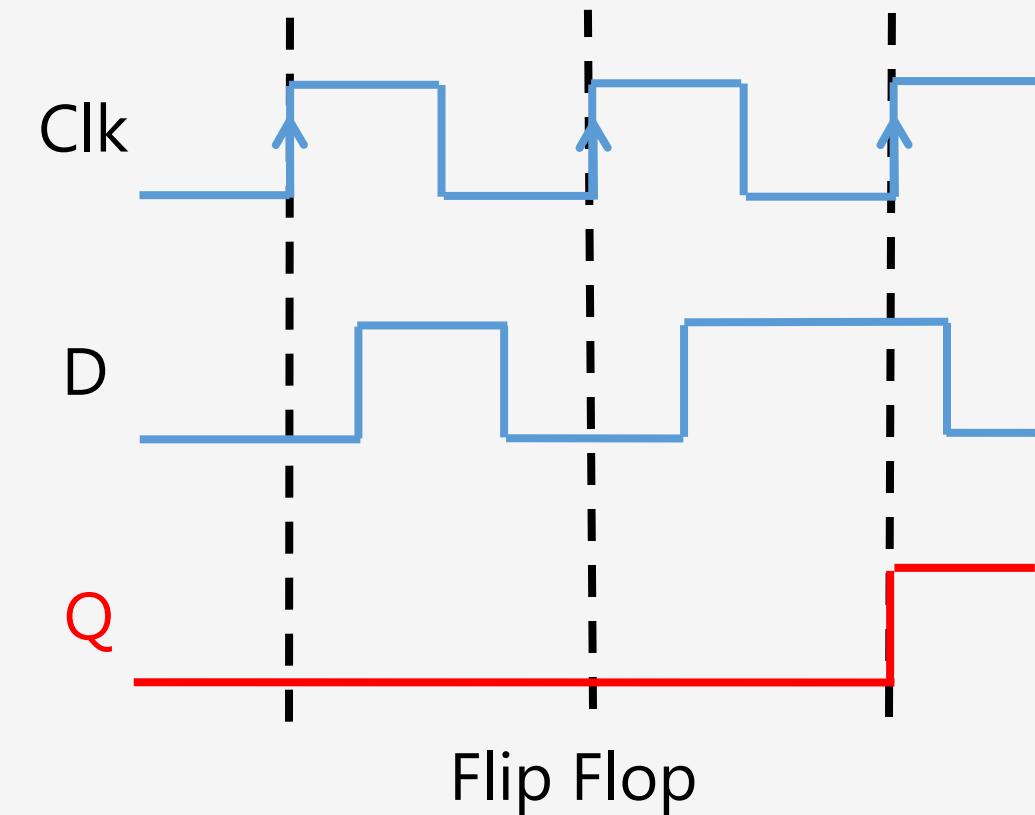
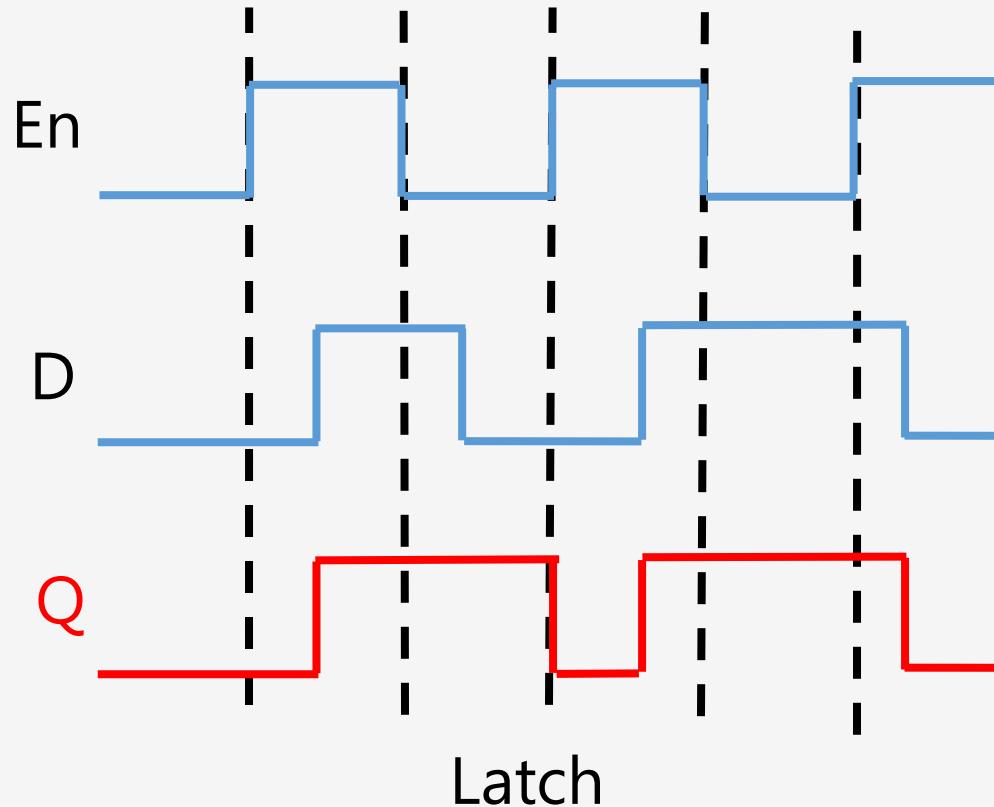
Latch



Flip Flop

Latch & Flip Flop - Sample Question - Solution

Consider the following input signal D. Compute output signal Q for D Latch and D Flip Flop.



Timing Violation

We need to watch out possible timing violations because of the unexpected delays in the data-path and the system clock.

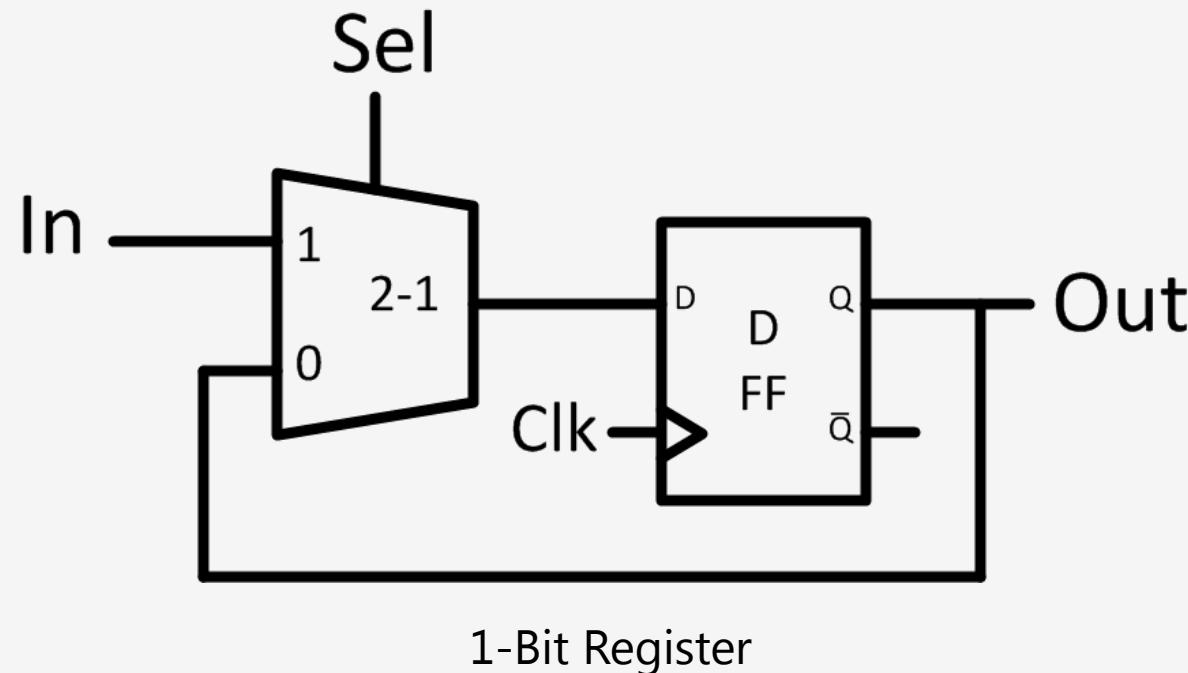
Setup Time: The minimum amount of time BEFORE the clock's active edge by which the data must be stable.

Hold Time: The minimum amount of time AFTER the clock's active edge by which the data must be stable.

We will talk about digital timing analysis in future lectures.

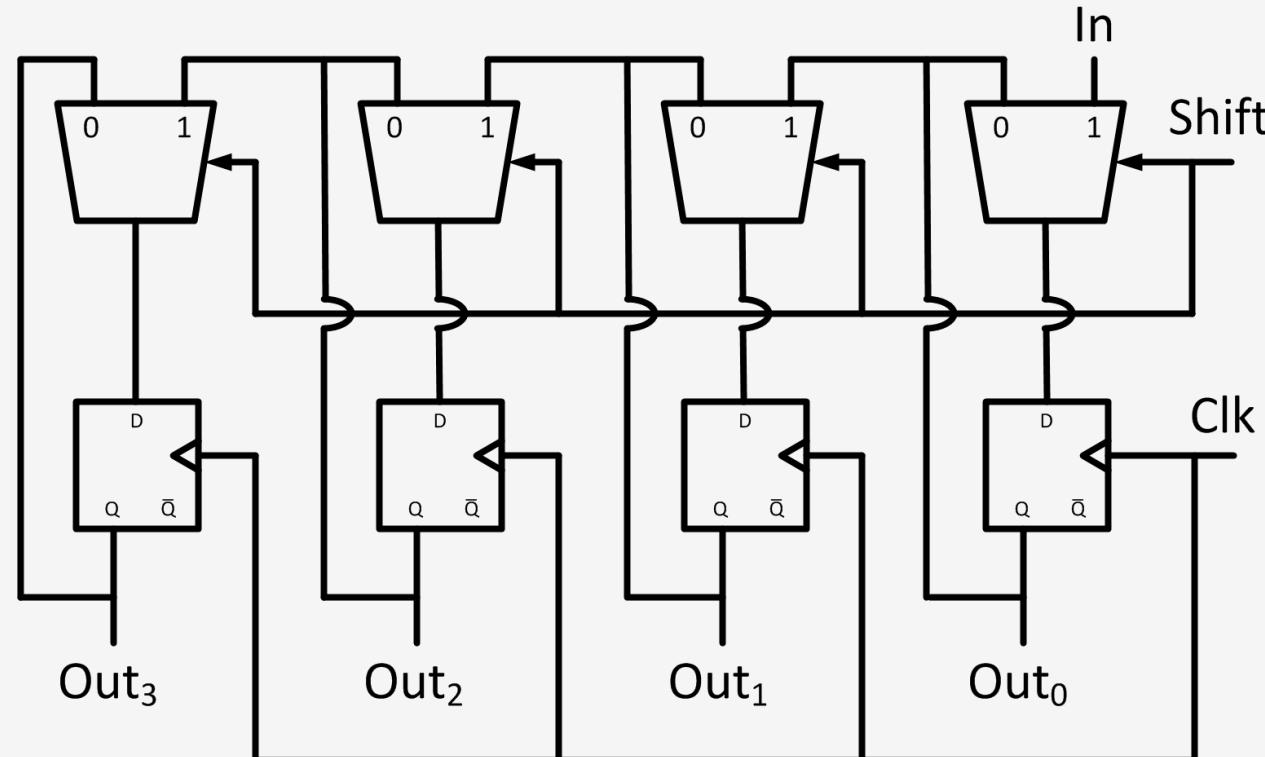
Register

While the flip-flop holds data for only one clock cycle until new data arrives at the next clock edge, the register can hold the same data perpetually until the power is turned off.



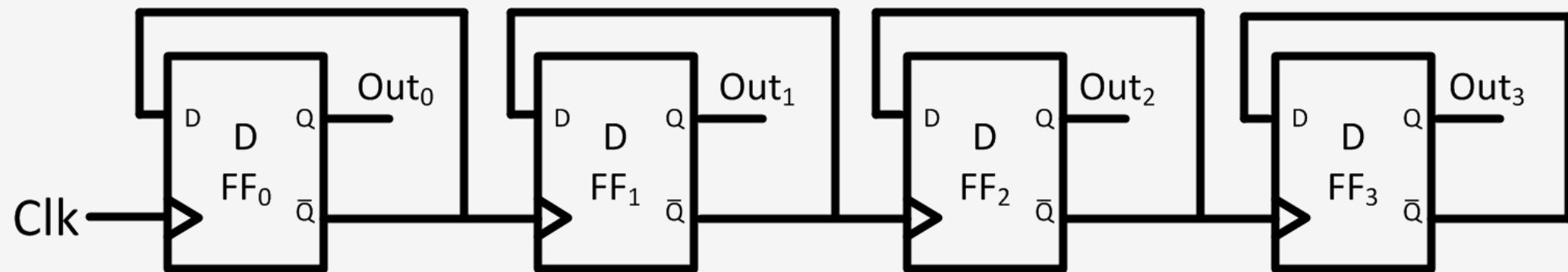
Shift Register

Shift register is a special form of register designed to shift data to the right (or to the left) according to the design needs.



Counter

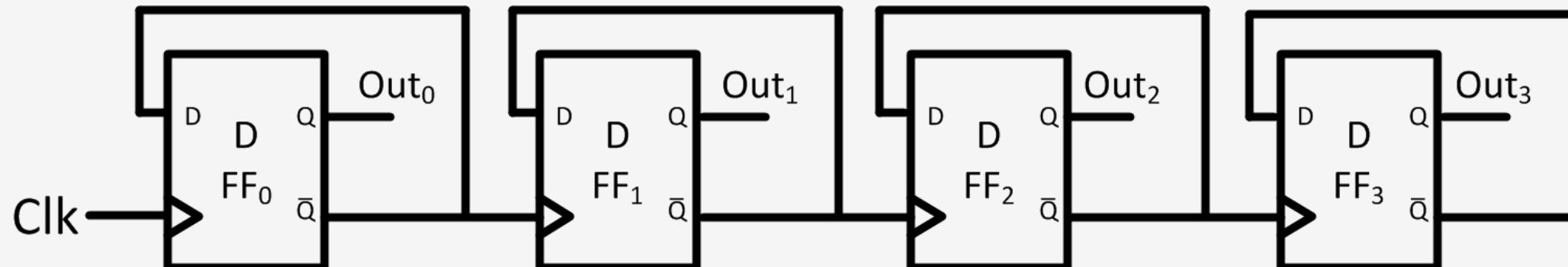
Counter is a special form of register designed to count up (or down) at each rising (or falling) edge of the clock.



4-Bit Asynchronous Up Binary Counter

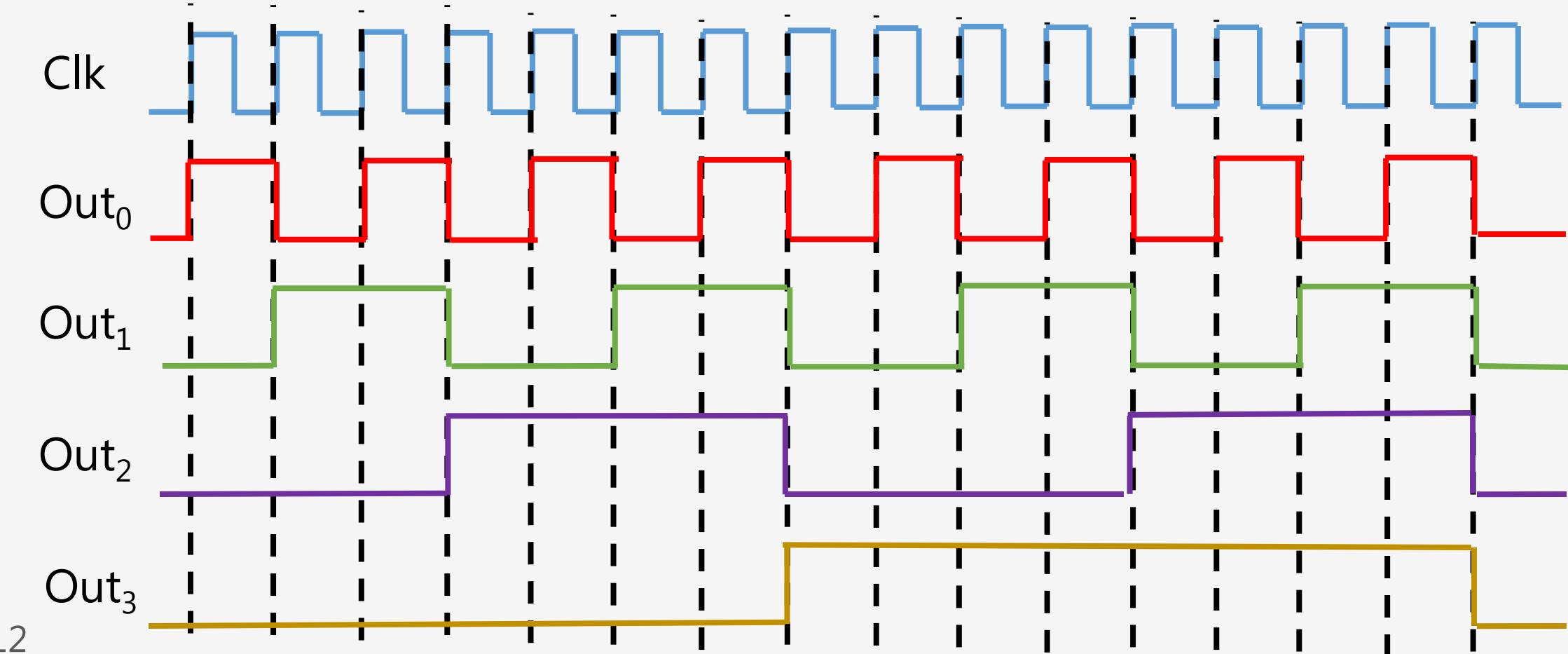
Counter - Sample Question

Suppose all the FFs are initialized to zero, draw the waveforms for given 4-Bit Asynchronous Up Binary Counter.



Counter - Sample Question - Solution

Draw the waveforms for given 4-Bit Asynchronous Up Binary Counter.



Asynchronous vs. Synchronous Counter

Asynchronous Counter:

- ✓ FFs are connected in a way that the output of first FF drives the clock of next one.
- ✓ FFs are NOT clocked simultaneously.
- ✓ Circuit is simple for large designs.
- ✓ Speed is slow as clock is propagated through all FFs.

Synchronous Counter:

- ✓ There is no connection between the output of first FF and the clock of next one.
- ✓ FFs are clocked simultaneously.
- ✓ Circuit becomes complicated for large designs.
- ✓ Speed is high as clock is given at a same time.

Finite State Machines 1

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Finite State Machine (FSM)- Definition

Synchronous sequential circuit is a digital circuit in which the changes in the state of memory elements are synchronized by a common clock signal.

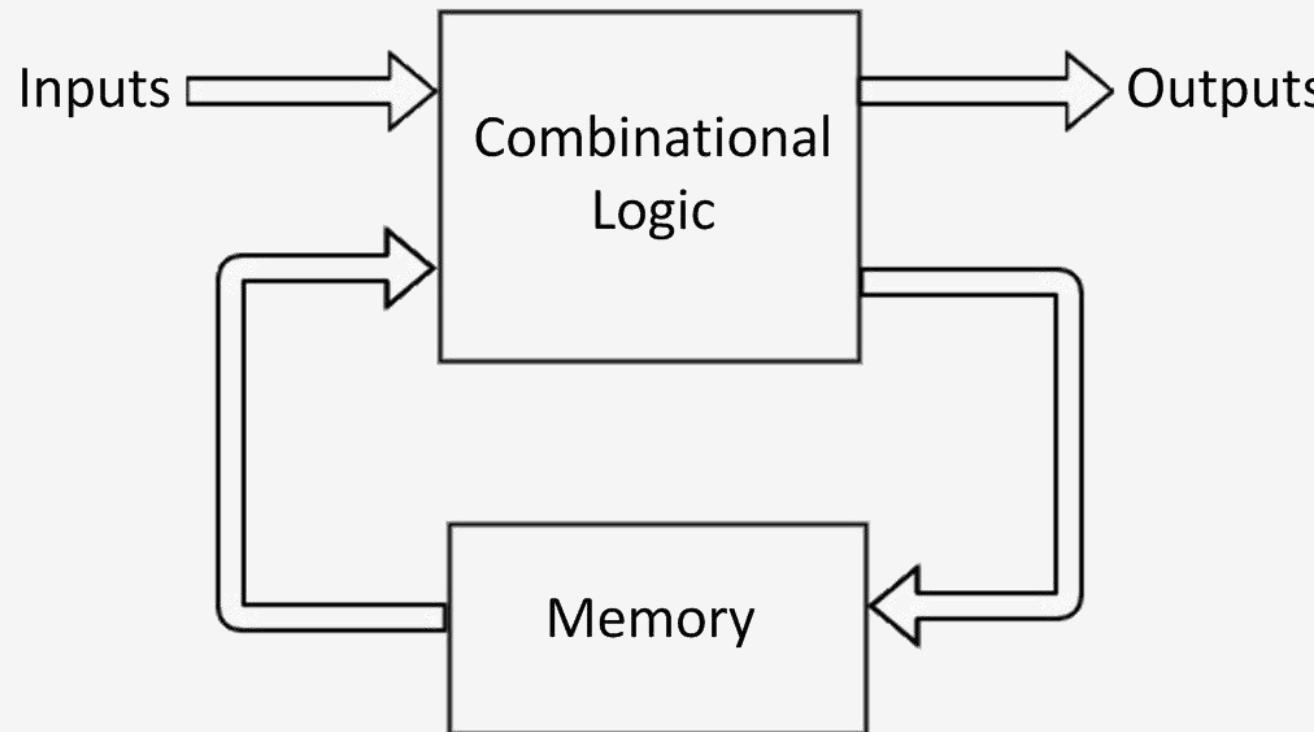
The behavior of synchronous sequential circuits can be represented in the graphical form known as state diagram.

A synchronous sequential circuit is called Finite State Machine (FSM) if it has finite number of states.

There are two types of FSMs: **Mealy Machine & Moore Machine**

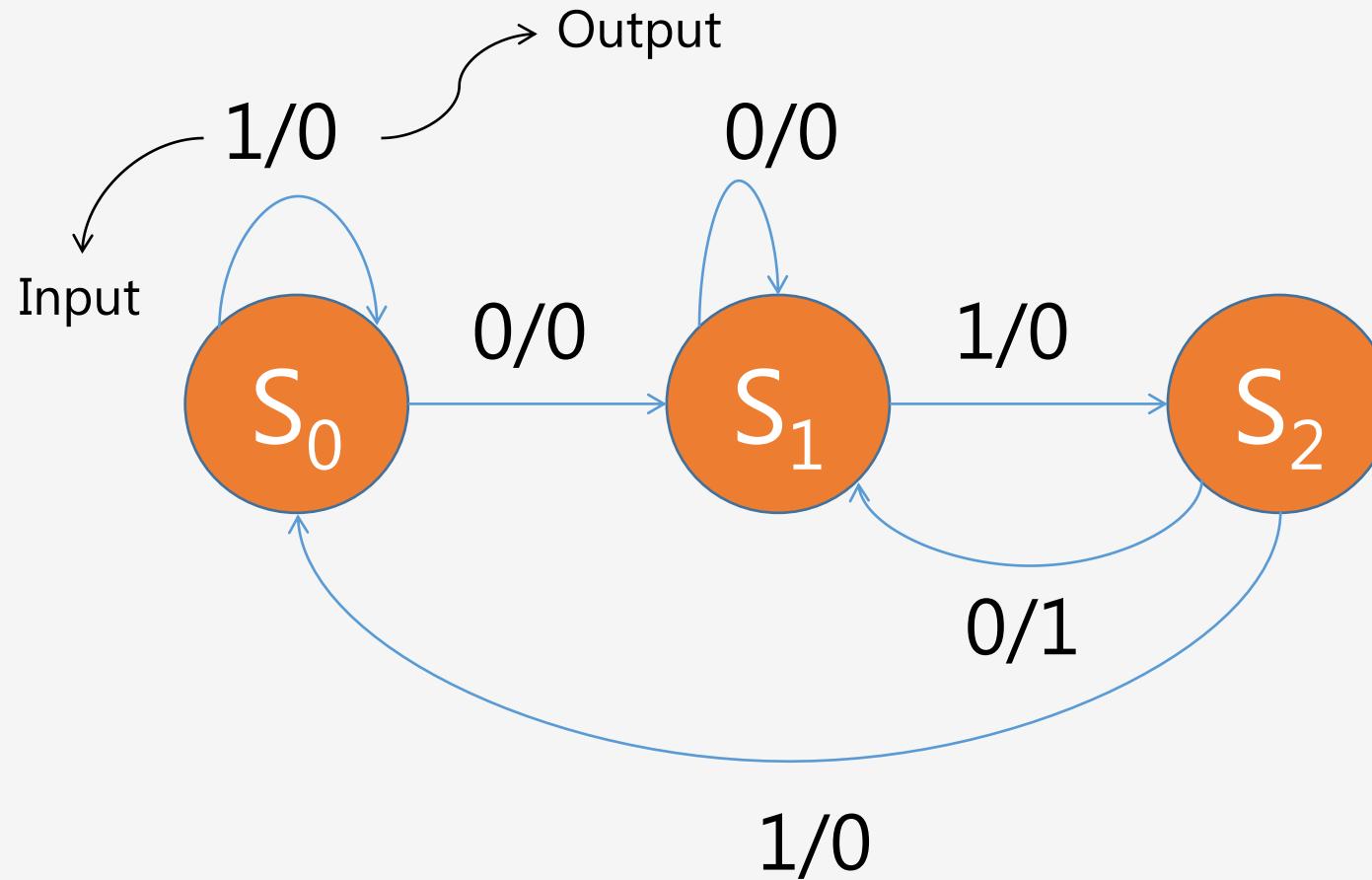
Mealy Machine

A FSM is said to be Mealy machine, if outputs depend on both current inputs & current states.



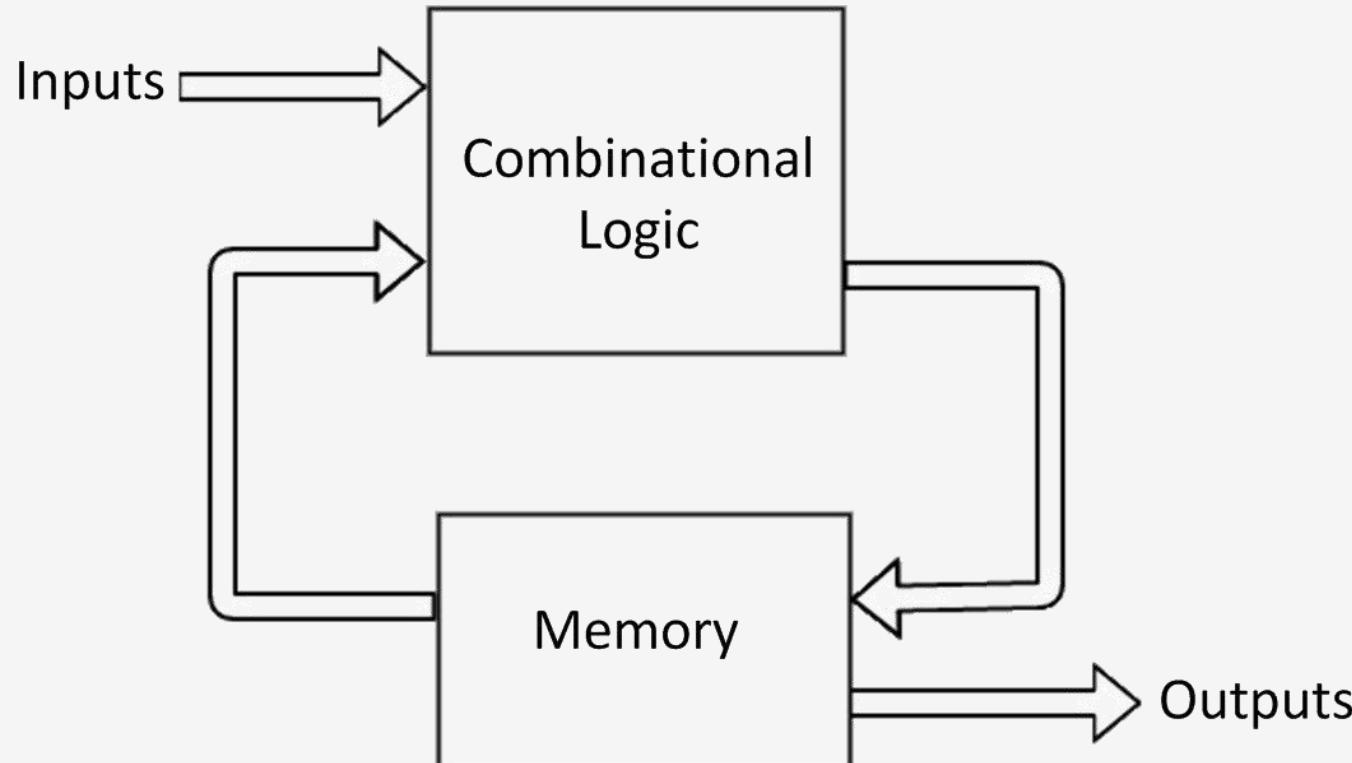
The outputs are valid only at positive (or negative) transition of the clock signal.

Mealy State Diagram



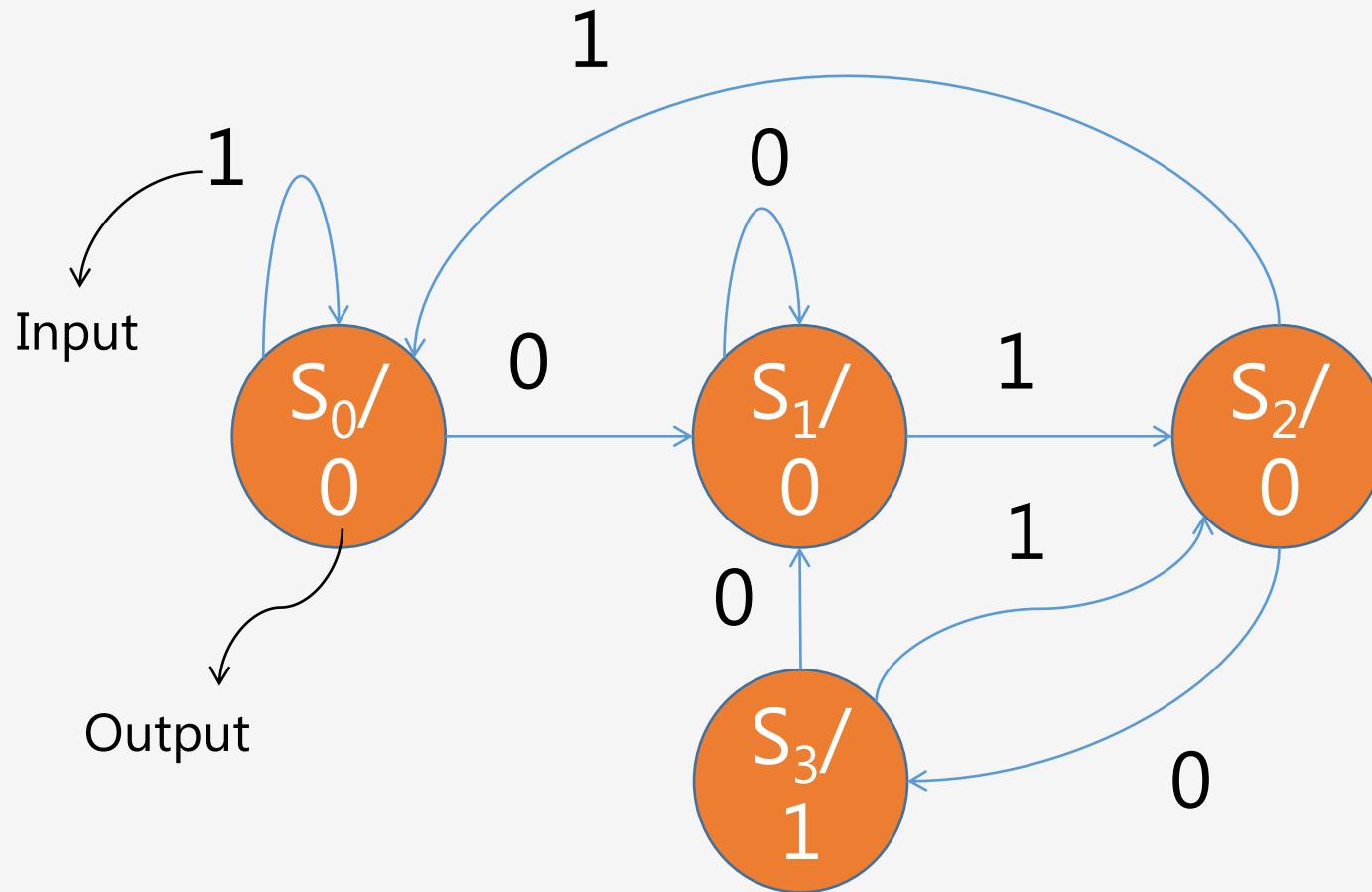
Moore Machine

A FSM is said to be Moore machine, if outputs depend only on current states.



The outputs are valid only after transition of the state.

Moore State Diagram



Mealy to Moore Conversion

Step 1: Draw the transition table of Mealy machine and calculate the number of different outputs for each state S_i .

Step 2: Draw the transition table of Moore machine. If all the outputs of S_i are same, copy state S_i . If it has n distinct outputs, break S_i into n states as S_{in} where $n=0,1,2,\text{etc.}$

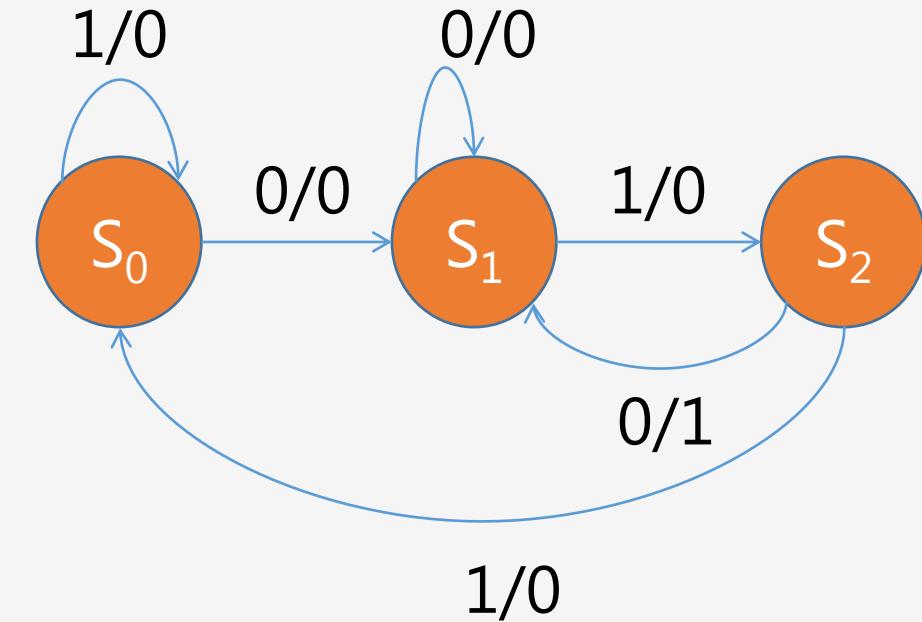
Step 3: Draw the Moore state diagram. If the output of the initial state is 1, insert a new initial state at the beginning which gives 0 output.

Mealy to Moore Conversion - Example

Step 1: Draw the transition table of Mealy machine.

Current State	Input = 0		Input = 1	
	Next State	Output	Next State	Output
S_0	S_1	0	S_0	0
S_1	S_1	0	S_2	0
S_2	S_1	1	S_0	0

Mealy Transition Table



Mealy State Diagram

Mealy to Moore Conversion - Example

Step 2: Draw the transition table of Moore machine.

Current State	Input = 0	Input = 1	Output
	Next State	Next State	
S_0	S_{10}	S_0	0
S_{10}	S_{10}	S_2	0
S_{11}	S_{10}	S_2	1
S_2	S_{11}	S_0	0

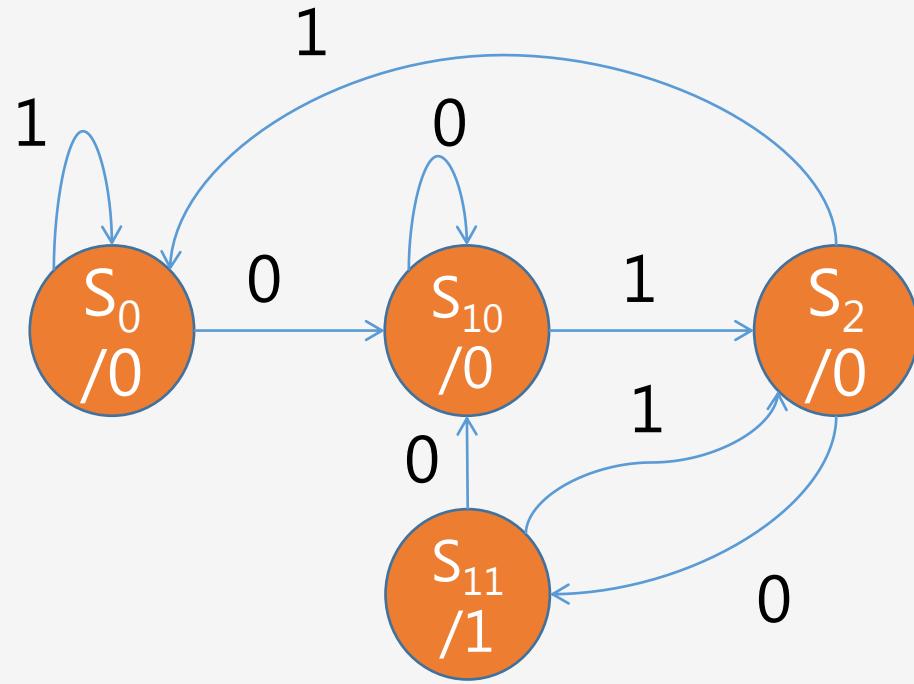
Moore Transition Table

Current State	Input = 0		Input = 1	
	Next State	Output	Next State	Output
S_0	S_1	0	S_0	0
S_1	S_1	0	S_2	0
S_2	S_1	1	S_0	0

Mealy Transition Table

Mealy to Moore Conversion - Example

Step 3: Draw Moore state diagram.



Moore State Diagram

Current State	Input = 0	Input = 1	Output
	Next State	Next State	
S_0	S_{10}	S_0	0
S_{10}	S_{10}	S_2	0
S_{11}	S_{10}	S_2	1
S_2	S_{11}	S_0	0

Moore Transition Table

Moore to Mealy Conversion

Step 1: Draw the transition table of Moore machine

Step 2: Draw the transition table of Mealy machine. Copy all the Moore transition states. If for a state S_i output is m, copy it into the output columns of the transition table of Mealy machine wherever S_i appears in the next state.

Step 3: Draw the Mealy state diagram.

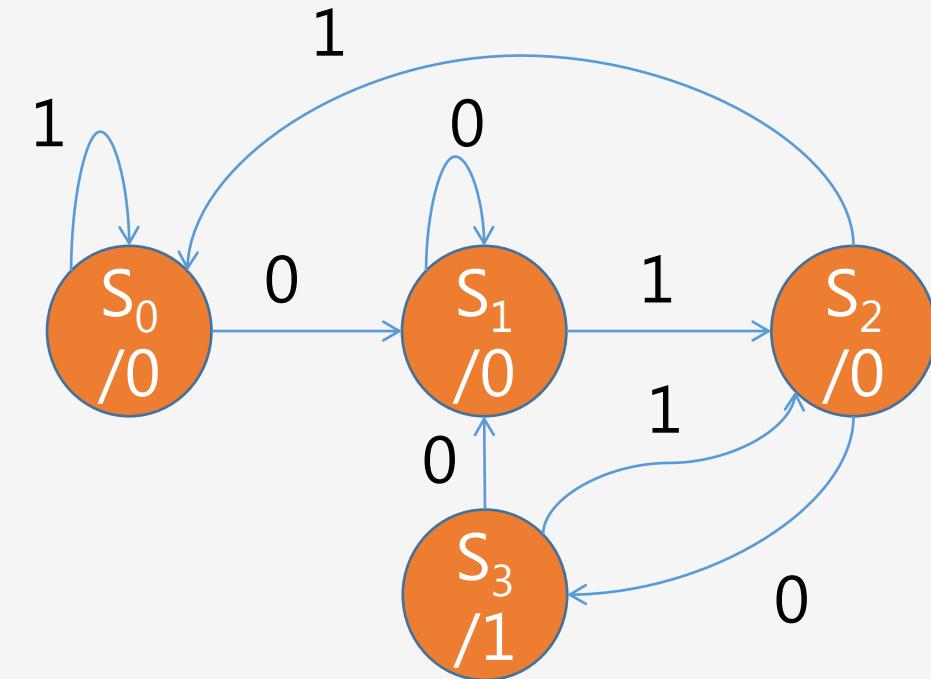
Step 4 (Optional): Combine states that have equivalent behavior.

Moore to Mealy Conversion - Example

Step 1: Draw the transition table of Moore machine.

Current State	Input = 0	Input = 1	Output
	Next State	Next State	
S_0	S_1	S_0	0
S_1	S_1	S_2	0
S_2	S_3	S_0	0
S_3	S_1	S_2	1

Moore Transition Table



Moore State Diagram

Moore to Mealy Conversion - Example

Step 2: Draw the transition table of Mealy machine.

Current State	Input = 0		Input = 1	
	Next State	Output	Next State	Output
S_0	S_1	0	S_0	0
S_1	S_1	0	S_2	0
S_2	S_3	1	S_0	0
S_3	S_1	0	S_2	0

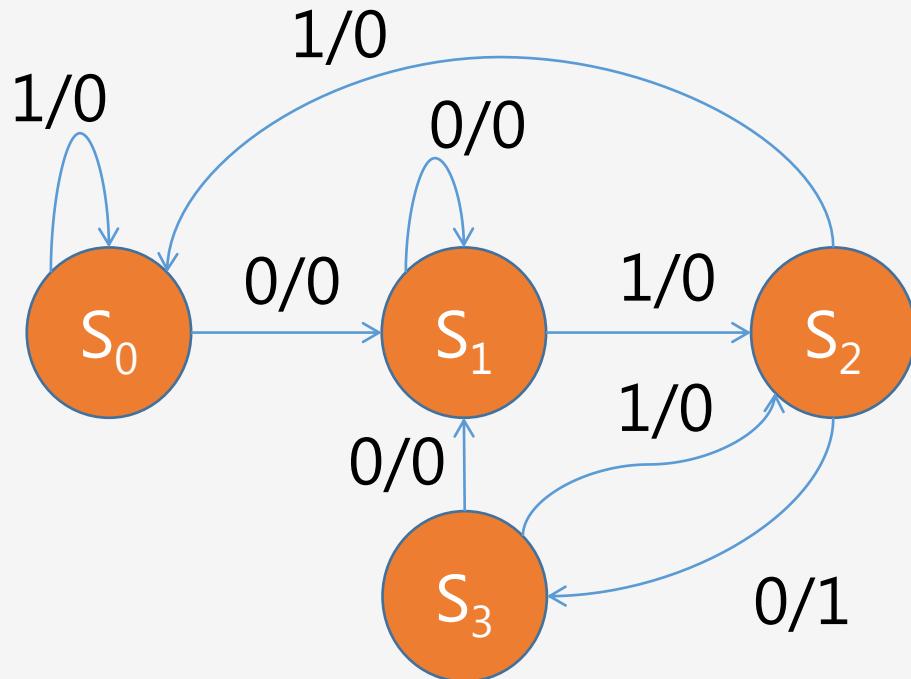
Mealy Transition Table

Current State	Input = 0	Input = 1	Output
	Next State	Next State	
S_0	S_1	S_0	0
S_1	S_1	S_2	0
S_2	S_3	S_0	0
S_3	S_1	S_2	1

Moore Transition Table

Moore to Mealy Conversion - Example

Step 3: Draw Mealy state diagram.



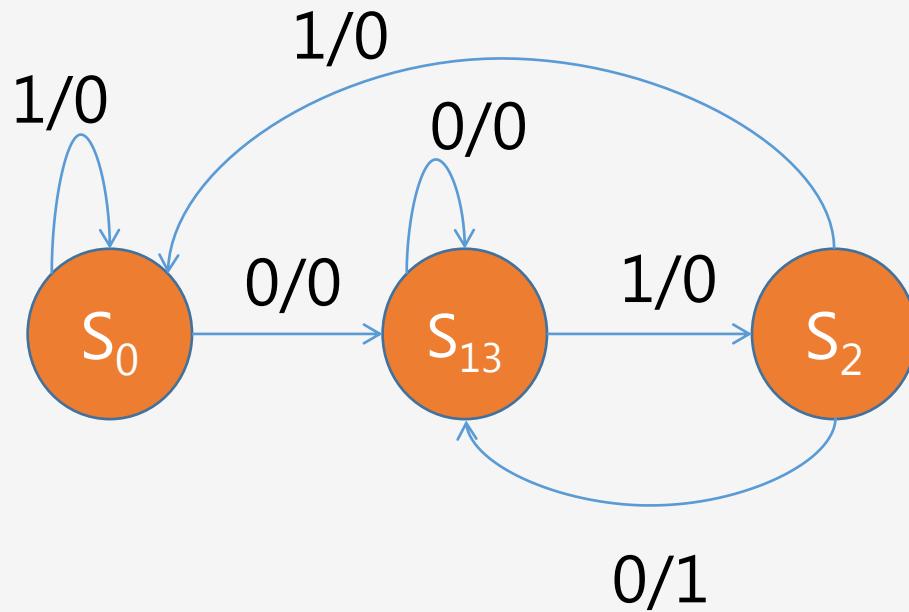
Mealy State Diagram

Current State	Input = 0		Input = 1	
	Next State	Output	Next State	Output
S ₀	S ₁	0	S ₀	0
S ₁	S ₁	0	S ₂	0
S ₂	S ₃	1	S ₀	0
S ₃	S ₁	0	S ₂	0

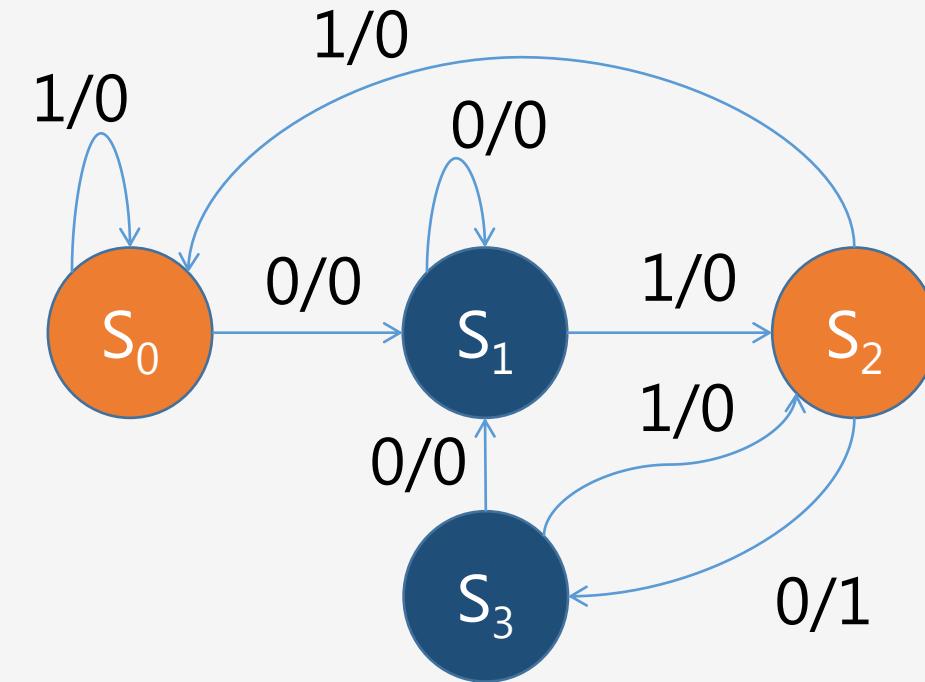
Mealy Transition Table

Moore to Mealy Conversion - Example

Step 4 (Optional): Combine states that have equivalent behavior.



Reduced Mealy State Diagram



Mealy State Diagram

Mealy vs. Moore Machine

Mealy Machine:

- ✓ Outputs depend both on current states and inputs.
- ✓ Generally, it has fewer states than Moore Machine.
- ✓ Generally, it reacts in the same clock cycle.
- ✓ It is more difficult to design than Moore Machine.

Moore Machine:

- ✓ Outputs depend only on current states.
- ✓ Generally, it has more states than Mealy Machine.
- ✓ Generally, it reacts one clock cycle later.
- ✓ It is easier to design than Mealy Machine.

Homework Assignment 2

Due Date: February 16, 11:30PM on BeachBoard Dropbox

Late Submission Policy:

One day delay (Second Due Date: Feb. 17, 11:30PM): 25% deduction

Two days delay (Third Due Date: Feb. 18, 11:30PM): 50% deduction

More than two days delay: No credit

Academic Integrity: There is zero tolerance for cheating, plagiarism, or any other act of violation of Academic Integrity.

Finite State Machines 2

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Sequential Circuit Design Procedure

Step 1: Make a state diagram based on the problem statement.

Step 2: Assign binary codes to the states and then convert the state diagram to a truth table with current states, current inputs, next states, and outputs.

Note: If you have n states, your binary codes will have at least $\log_2 n$ digits. (i.e., Your circuit will have at least $\log_2 n$ D FFs.)

Step 3: Find simplified equations for next states (i.e., D FF inputs) and the outputs.

Step 4: Build the circuit.

Sequential Circuit Design - Sample Question

Design a single-input, single-output sequential circuit with the following description:

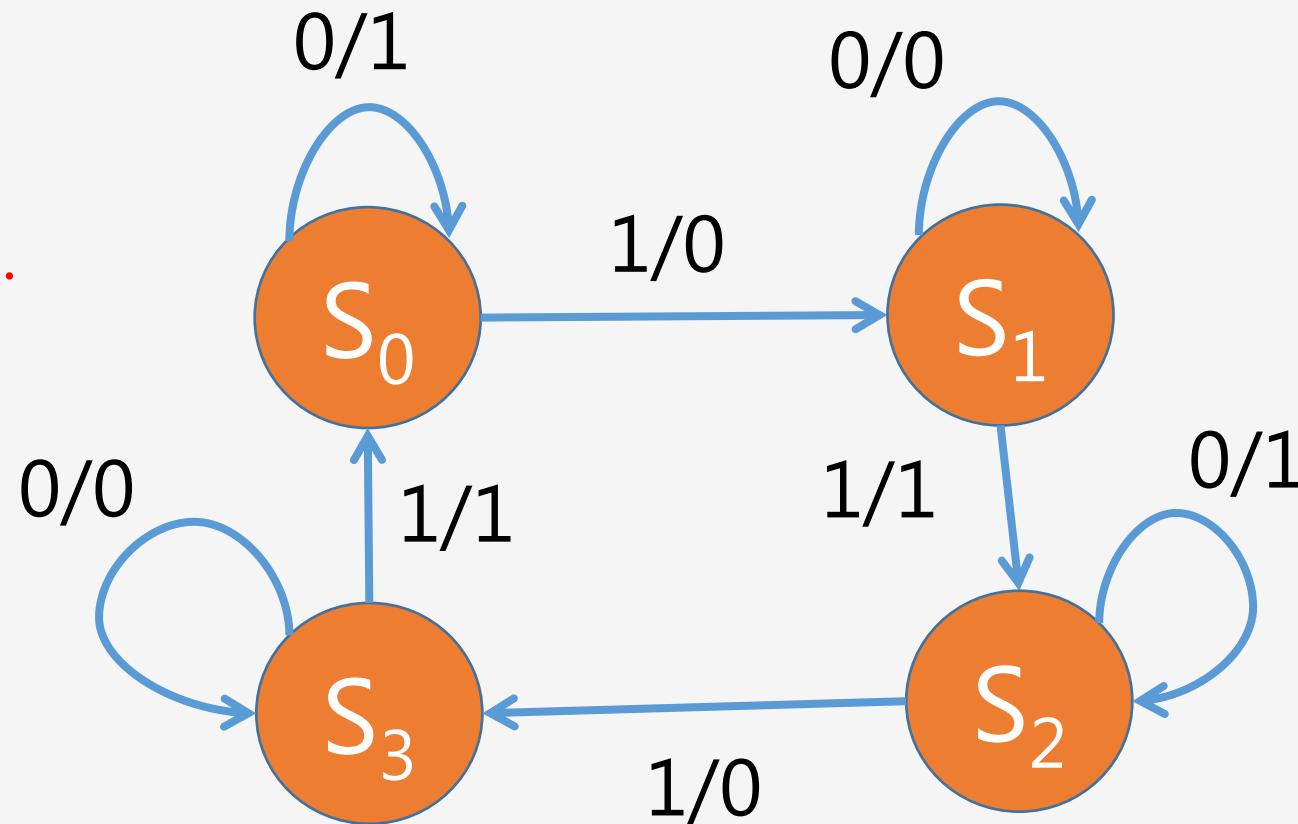
- 1) When input = 0, the state of the circuit remains the same;
- 2) When input = 1, the circuit goes through the state transitions from S_0 to S_1 to S_2 to S_3 , back to S_0 , and repeats;
- 3) When the circuit is in even states, output = 1 regardless of the input value.

Sequential Circuit Design - Sample Question - Solution

Step 1: Make a state diagram based on the problem statement.

There are 4 states, so we need 2 D FFs.

Can we implement it with Moore Machine?

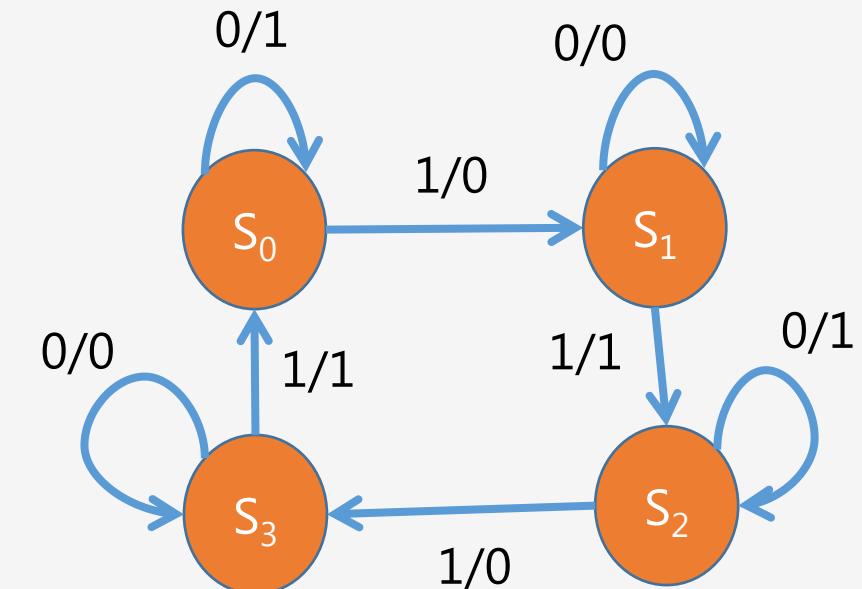


Sequential Circuit Design - Sample Question - Solution

Step 2: Convert the state diagram to a truth table.

	Q_1	Q_0	I	$Q'_1=D_1$	$Q'_0=D_0$	O
S_0	0	0	0	0	0	1
S_1	0	0	1	0	1	0
S_2	0	1	0	0	1	0
S_3	0	1	1	1	0	1
	1	0	0	1	0	1
	1	0	1	1	1	0
	1	1	0	1	1	0
	1	1	1	0	0	1

Current State Next State



Sequential Circuit Design - Sample Question - Solution

Step 3: Find simplified equations.

Q₁	Q₀	I	D₁	D₀	O
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	1
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	0	0	1

	Q₀I=00	Q₀I=01	Q₀I=11	Q₀I=10
Q₁=0	0	0	1	0
Q₁=1	1	1	0	1

$$D_1 = Q_1 \bar{Q}_0 + Q_1 I + \bar{Q}_1 Q_0 I$$

	Q₀I=00	Q₀I=01	Q₀I=11	Q₀I=10
Q₁=0	0	1	0	1
Q₁=1	0	1	0	1

$$D_0 = Q_0 \oplus I$$

	Q₀I=00	Q₀I=01	Q₀I=11	Q₀I=10
Q₁=0	1	0	1	0
Q₁=1	1	0	1	0

$$O = \overline{Q_0 \oplus I}$$

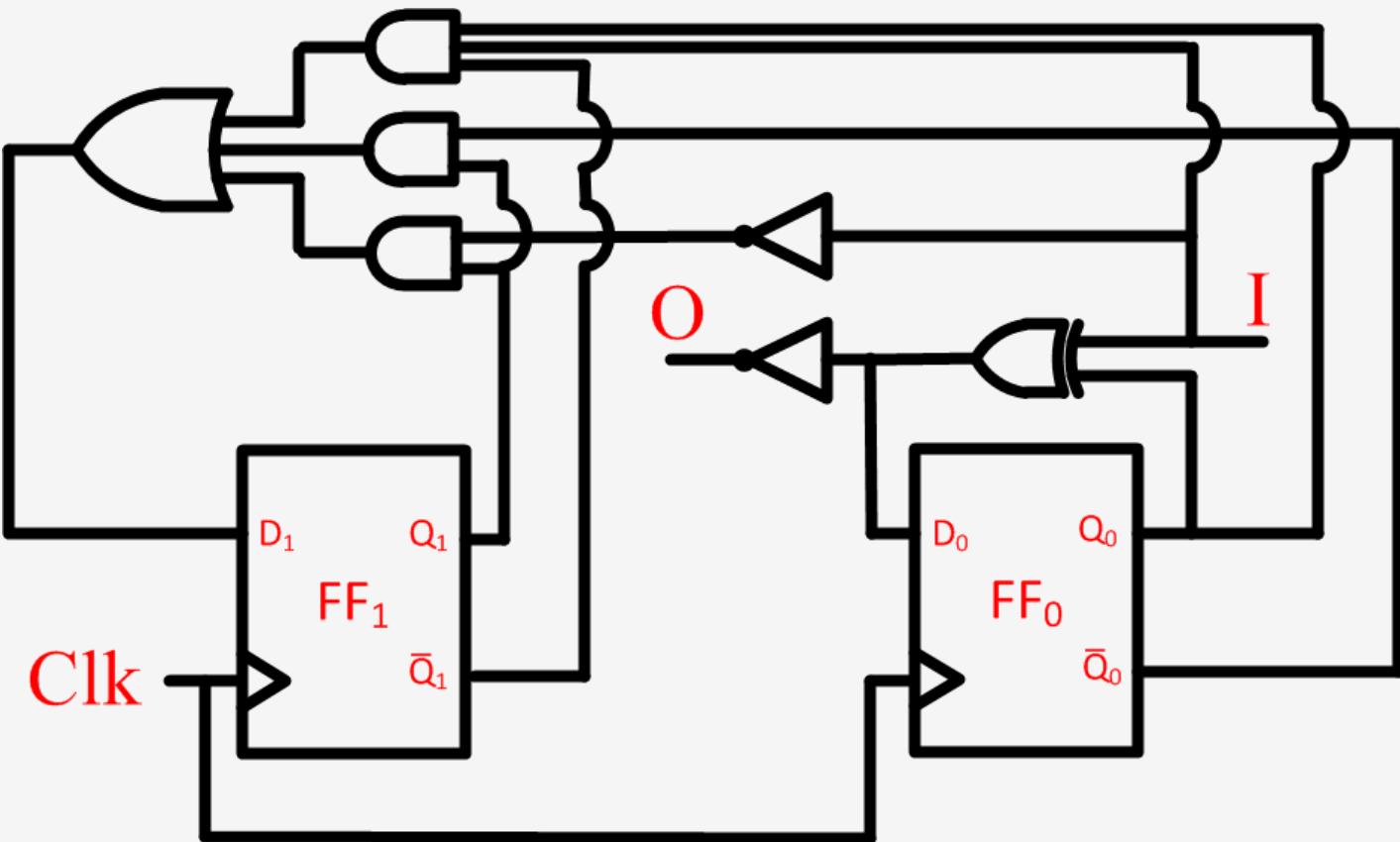
Sequential Circuit Design - Sample Question - Solution

Step 4: Build the circuit.

$$D_1 = Q_1 \bar{Q}_0 + Q_1 \bar{I} + \bar{Q}_1 Q_0 I$$

$$D_0 = Q_0 \oplus I$$

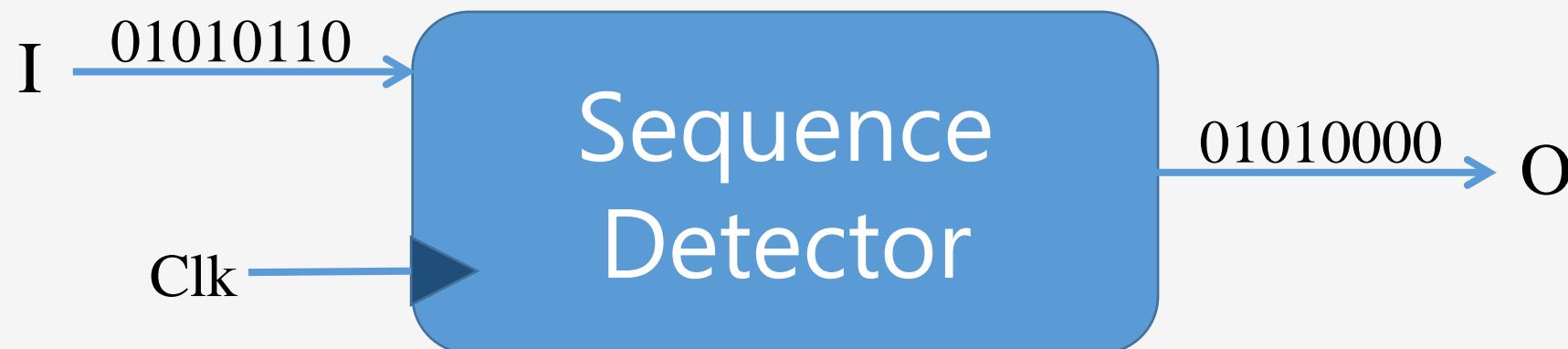
$$O = \overline{Q_0 \oplus I}$$



Sequence Detector

Sequence detector is a sequential circuit that outputs 1 when a particular pattern of bits sequentially arrives at its input.

Overlapping sequence detector is capable of detecting overlapping sequences while **non-overlapping** sequence detector is not.



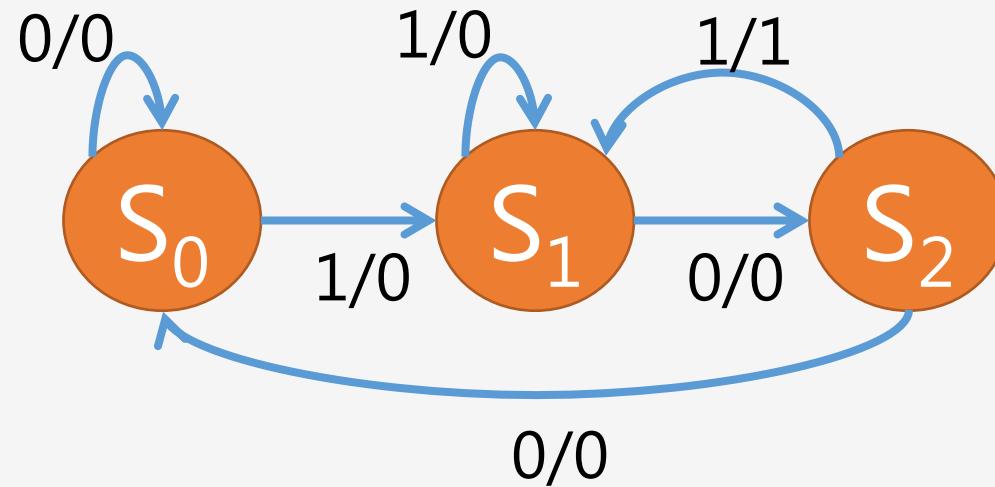
Sequence Detector - Sample Question

Design an overlapping sequence detector in a way that each time there is a sequence of 101, the output will generate 1. Otherwise, the output will be 0.

Sequence Detector - Sample Question - Solution

Step 1: Make a state diagram based on the problem statement.

There are 3 states, so we need 2 D FFs.



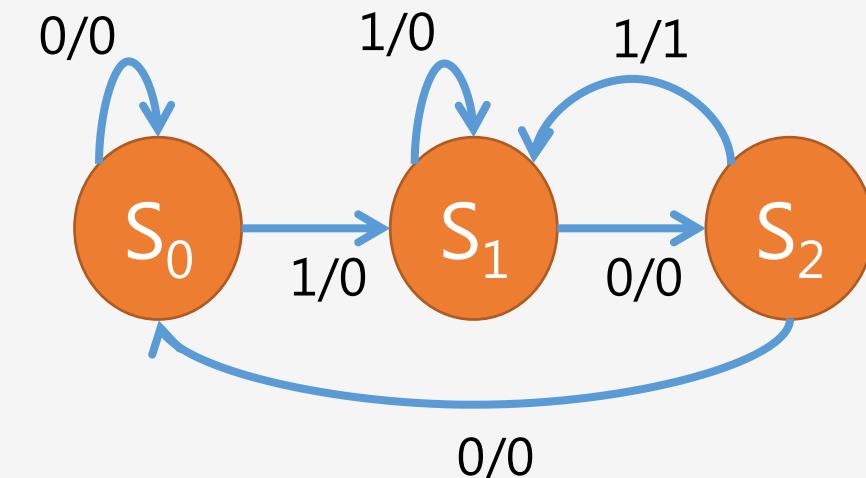
Current State	S_0	S_1	S_1	S_2	S_1	S_2	S_1	S_1	S_2	S_0
Input	1	1	0	1	0	1	1	0	0	0
Output	0	0	0	1	0	1	0	0	0	0
Next State	S_1	S_1	S_2	S_1	S_2	S_1	S_1	S_2	S_0	S_0

Sequence Detector - Sample Question - Solution

Step 2: Convert the state diagram to a truth table.

	Q_1	Q_0	I	$Q'_1=D_1$	$Q'_0=D_0$	O
S_0	0	0	0	0	0	0
	0	0	1	0	1	0
S_1	0	1	0	1	0	0
	0	1	1	0	1	0
S_2	1	0	0	0	0	0
	1	0	1	0	1	1
S_3	1	1	0	X	X	X
	1	1	1	X	X	X

Current State Next State



Sequence Detector - Sample Question - Solution

Step 3: Find simplified equations.

Q₁	Q₀	I	D₁	D₀	O
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	X	X	X
1	1	1	X	X	X

	Q₀I=00	Q₀I=01	Q₀I=11	Q₀I=10
Q₁=0	0	0	0	1
Q₁=1	0	0	X	X

$$D_1 = Q_0 \bar{I}$$

	Q₀I=00	Q₀I=01	Q₀I=11	Q₀I=10
Q₁=0	0	1	1	0
Q₁=1	0	1	X	X

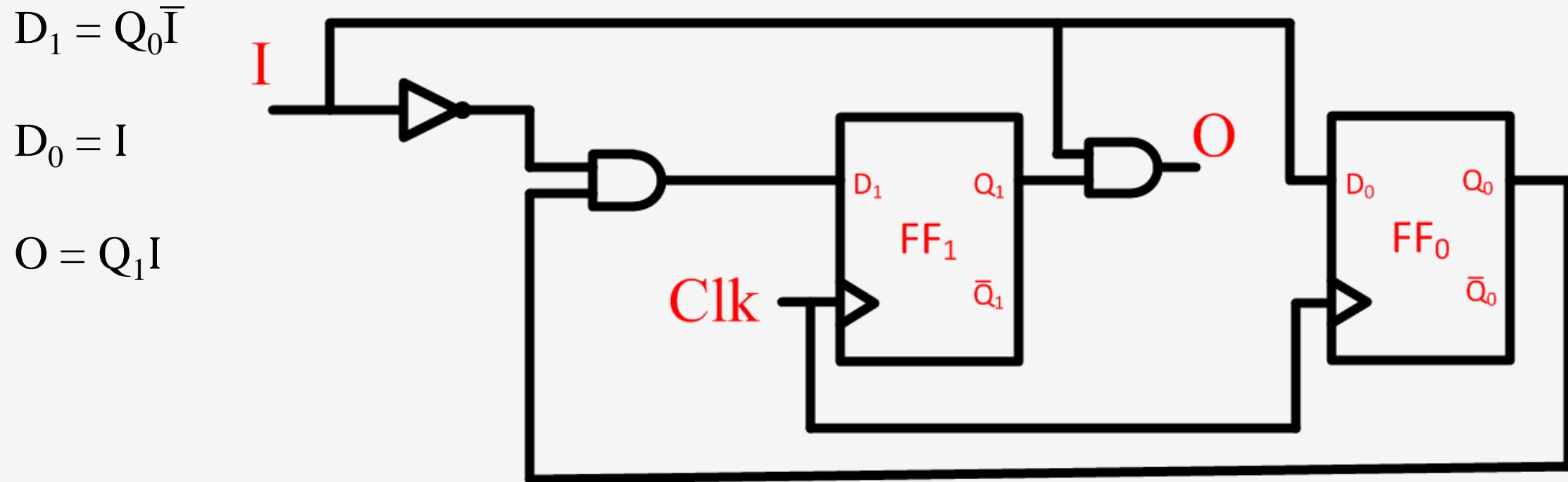
$$D_0 = I$$

	Q₀I=00	Q₀I=01	Q₀I=11	Q₀I=10
Q₁=0	0	0	0	0
Q₁=1	0	1	X	X

$$O = Q_1 I$$

Sequence Detector - Sample Question - Solution

Step 4: Build the circuit.



BCD to 7 Segment Counter

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

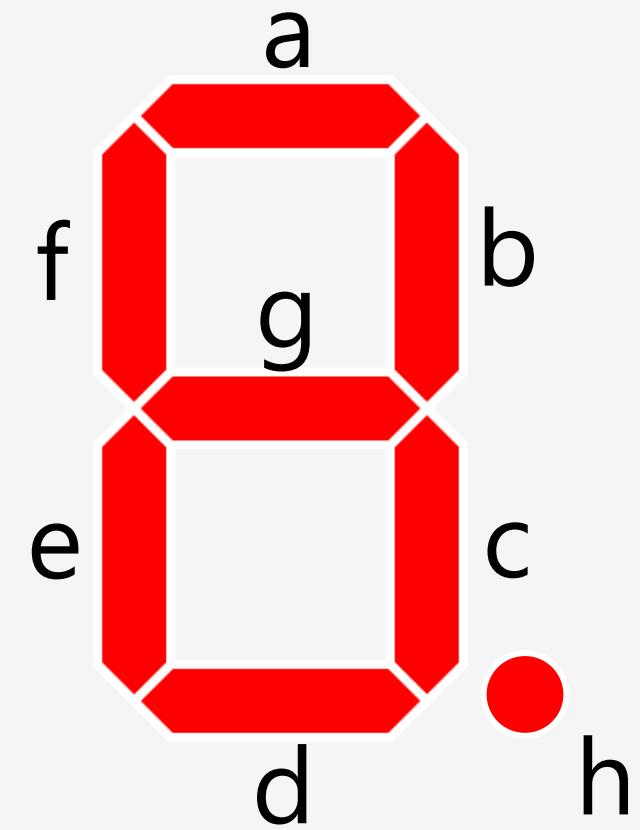
Spring 2021

7 Segment Display

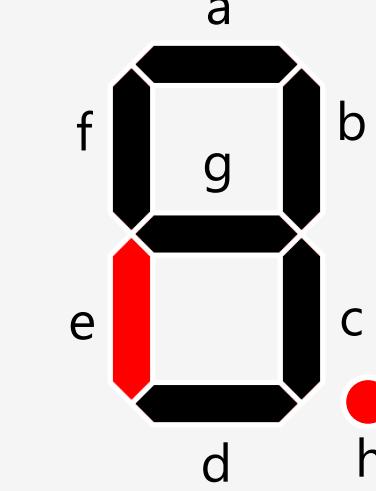
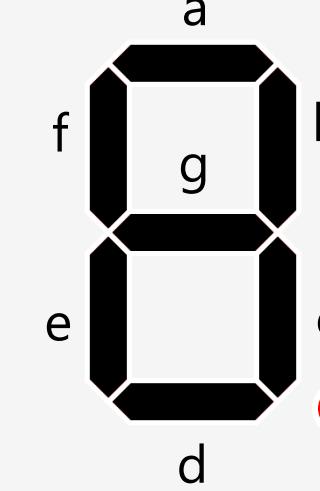
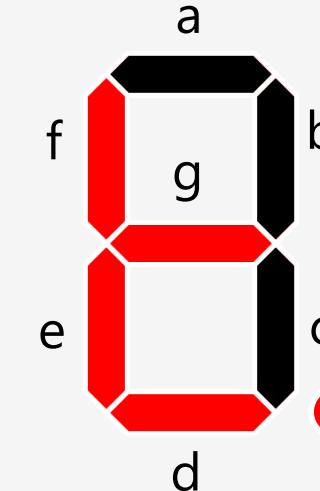
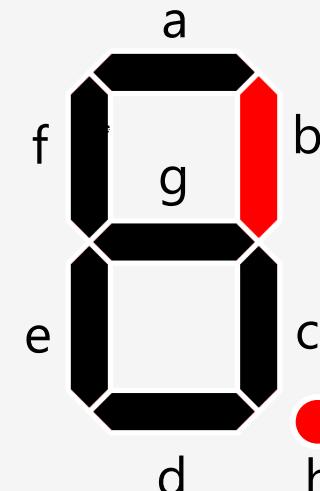
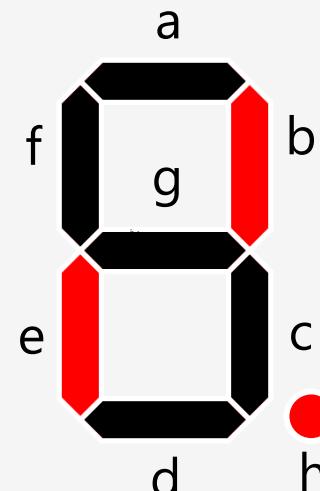
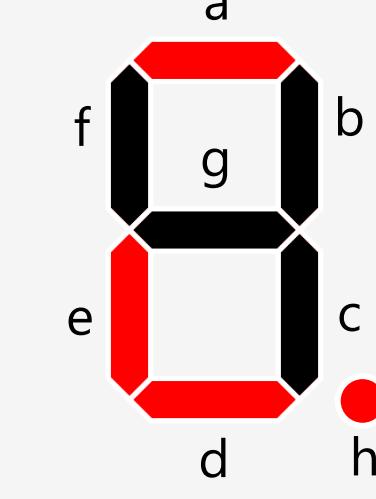
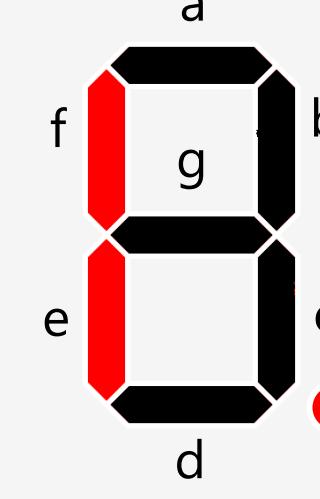
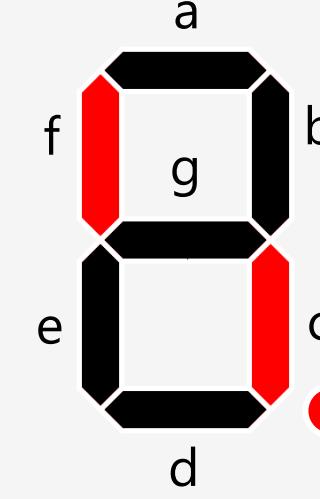
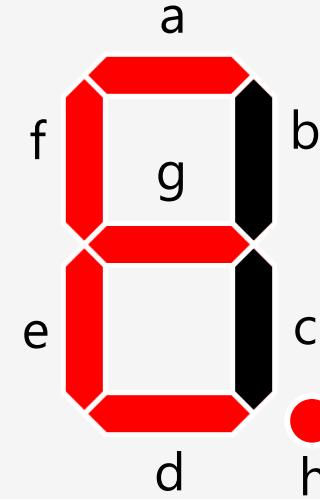
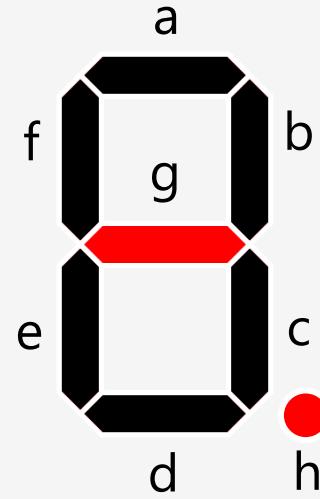
7 segment is an electronic display device for displaying decimal numerals.

Common Cathode → A logic '1' applied to the anode terminal of the individual segment illuminates it.

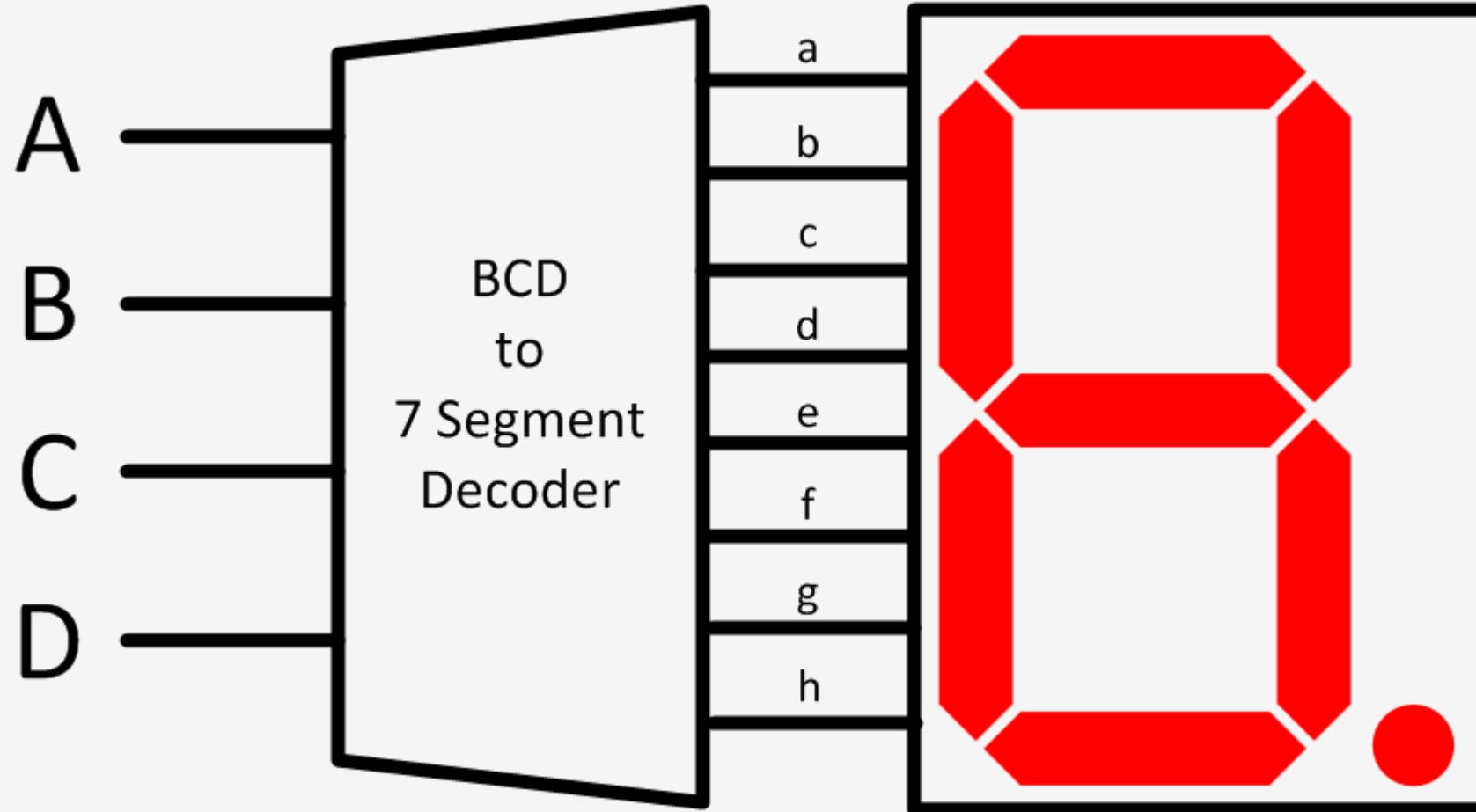
Common Anode → A logic '0' applied to the cathode terminal of the individual segment illuminates it.



7 Segment Decimal Representation



BCD to 7 Segment Decoder - Scheme



BCD to 7 Segment Decoder - Truth Table for Common Anode

A	B	C	D	a	b	c	d	e	f	g	h	Dis
0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	1	1	0	0	1	1	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0	1	2
0	0	1	1	0	0	0	0	1	1	0	1	3
0	1	0	0	1	0	0	1	1	0	0	1	4
0	1	0	1	0	1	0	0	1	0	0	1	5
0	1	1	0	0	1	0	0	0	0	0	1	6
0	1	1	1	0	0	0	1	1	1	1	1	7
1	0	0	0	0	0	0	0	0	0	0	1	8
1	0	0	1	0	0	0	0	1	0	0	1	9
1	0	1	0	1	1	1	1	1	1	1	0	DP

BCD to 7 Segment Decoder - K-Map for a

	CD=00	CD=01	CD=11	CD=10
AB=00	0	1	0	0
AB=01	1	0	0	0
AB=11	X	X	X	X
AB=10	0	0	X	1

$$a = (A \cdot C) + (\overline{A} \oplus \overline{C}) \cdot (B \oplus D)$$

XOR functions create stripes and checkerboards in K-map. You may think about what combinations of stripes and checkerboards would be needed to create patterns of ones.

BCD to 7 Segment Decoder - K-Map for b

	CD=00	CD=01	CD=11	CD=10
AB=00	0	0	0	0
AB=01	0	1	0	1
AB=11	X	X	X	X
AB=10	0	0	X	1

$$b = (A \cdot C) + (B \cdot \bar{C} \cdot D) + (B \cdot C \cdot \bar{D})$$

BCD to 7 Segment Decoder - K-Map for c

	CD=00	CD=01	CD=11	CD=10
AB=00	0	0	0	1
AB=01	0	0	0	0
AB=11	X	X	X	X
AB=10	0	0	X	1

$$c = \bar{B} \cdot C \cdot \bar{D}$$

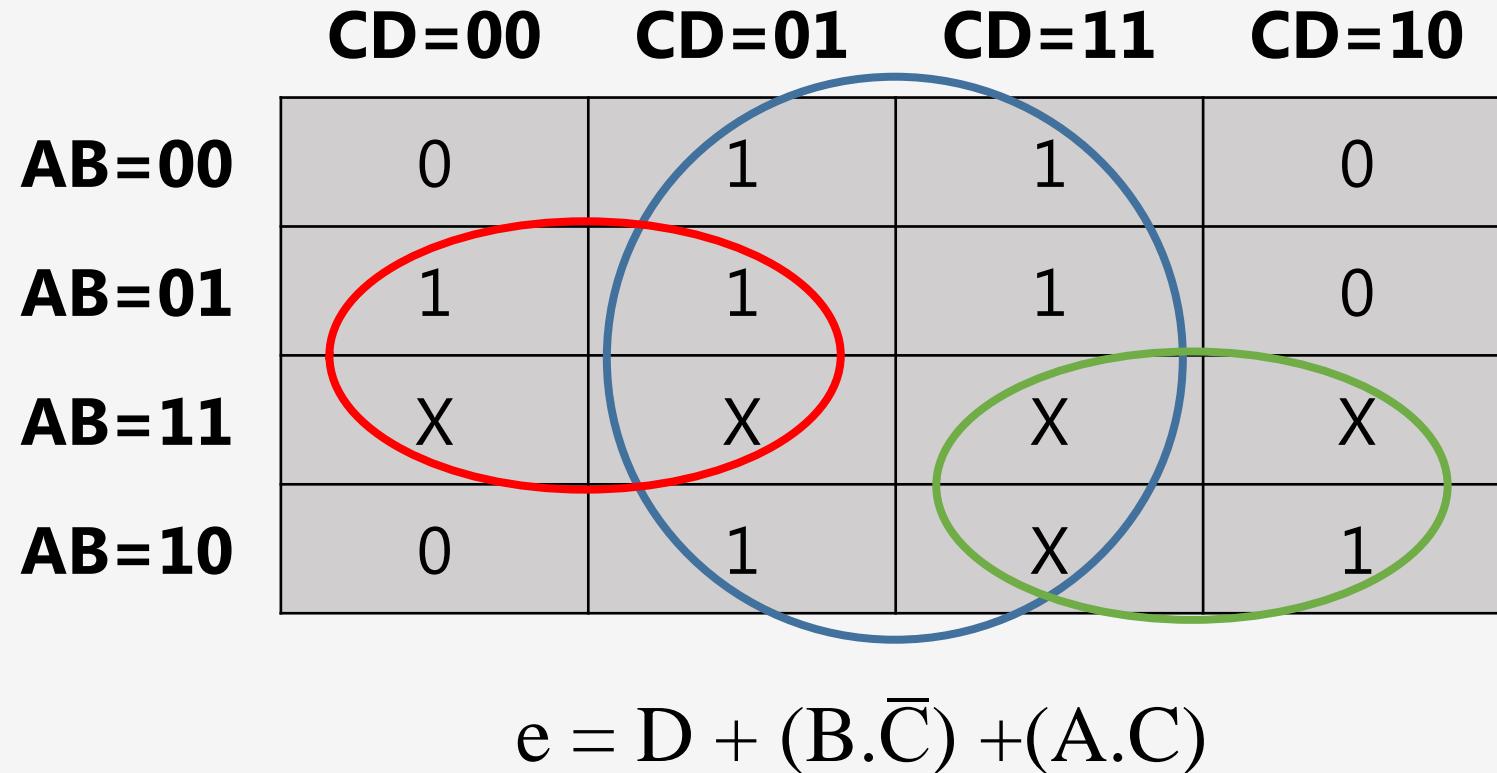
BCD to 7 Segment Decoder - K-Map for d

	CD=00	CD=01	CD=11	CD=10
AB=00	0	1	0	0
AB=01	1	0	1	0
AB=11	X	X	X	X
AB=10	0	0	X	1

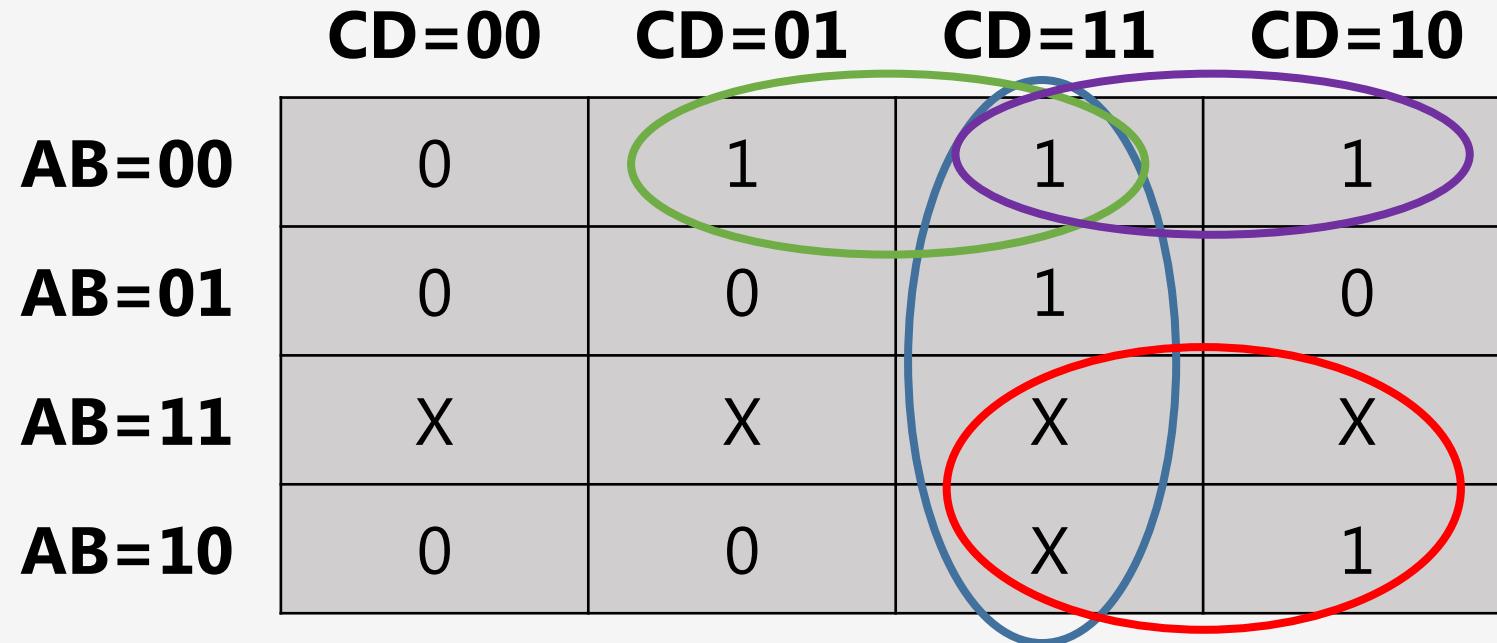
The K-Map shows the following minterms:
- AB=01, CD=00 (green oval)
- AB=00, CD=01 (purple oval)
- AB=10, CD=11 (red oval)
- AB=11, CD=10 (blue oval)

$$d = (A \cdot C) + (B \cdot C \cdot D) + (B \cdot \bar{C} \cdot \bar{D}) + (\bar{A} \cdot \bar{B} \cdot \bar{C} \cdot D)$$

BCD to 7 Segment Decoder - K-Map for e



BCD to 7 Segment Decoder - K-Map for f



$$f = (C \cdot D) + (A \cdot C) + (\bar{A} \cdot \bar{B} \cdot D) + (\bar{A} \cdot \bar{B} \cdot C)$$

BCD to 7 Segment Decoder - K-Map for g

	CD=00	CD=01	CD=11	CD=10
AB=00	1	1	0	0
AB=01	0	0	1	0
AB=11	X	X	X	X
AB=10	0	0	X	1

$$g = (A \cdot C) + (\bar{A} \cdot \bar{B} \cdot \bar{C}) + (B \cdot C \cdot D)$$

BCD to 7 Segment Decoder - K-Map for h

	CD=00	CD=01	CD=11	CD=10
AB=00	1	1	1	1
AB=01	1	1	1	1
AB=11	X	X	X	X
AB=10	1	1	X	0

$$h = \bar{A} + \bar{C} = \overline{A \cdot C}$$

Synchronous BCD Counter - Truth Table

BCD counter starts from 0, counts up to 9 and then resets to 0.

Q_3	Q_2	Q_1	Q_0	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0

Synchronous BCD Counter - K-Maps

	$Q_1Q_0=00$	$Q_1Q_0=01$	$Q_1Q_0=11$	$Q_1Q_0=10$
$Q_3Q_2=00$	0	0	0	0
$Q_3Q_2=01$	0	0	1	0
$Q_3Q_2=11$	0	0	0	0
$Q_3Q_2=10$	1	0	0	0

$$D_3 = (\bar{Q}_3 \cdot Q_2 \cdot Q_1 \cdot Q_0) + (Q_3 \cdot \bar{Q}_2 \cdot \bar{Q}_1 \cdot \bar{Q}_0)$$

	$Q_1Q_0=00$	$Q_1Q_0=01$	$Q_1Q_0=11$	$Q_1Q_0=10$
$Q_3Q_2=00$	0	0	1	0
$Q_3Q_2=01$	1	1	0	1
$Q_3Q_2=11$	0	0	0	0
$Q_3Q_2=10$	0	0	0	0

$$D_2 = (\bar{Q}_3 \cdot Q_2 \cdot \bar{Q}_1) + (\bar{Q}_3 \cdot Q_2 \cdot \bar{Q}_0) + (\bar{Q}_3 \cdot \bar{Q}_2 \cdot Q_1 \cdot Q_0)$$

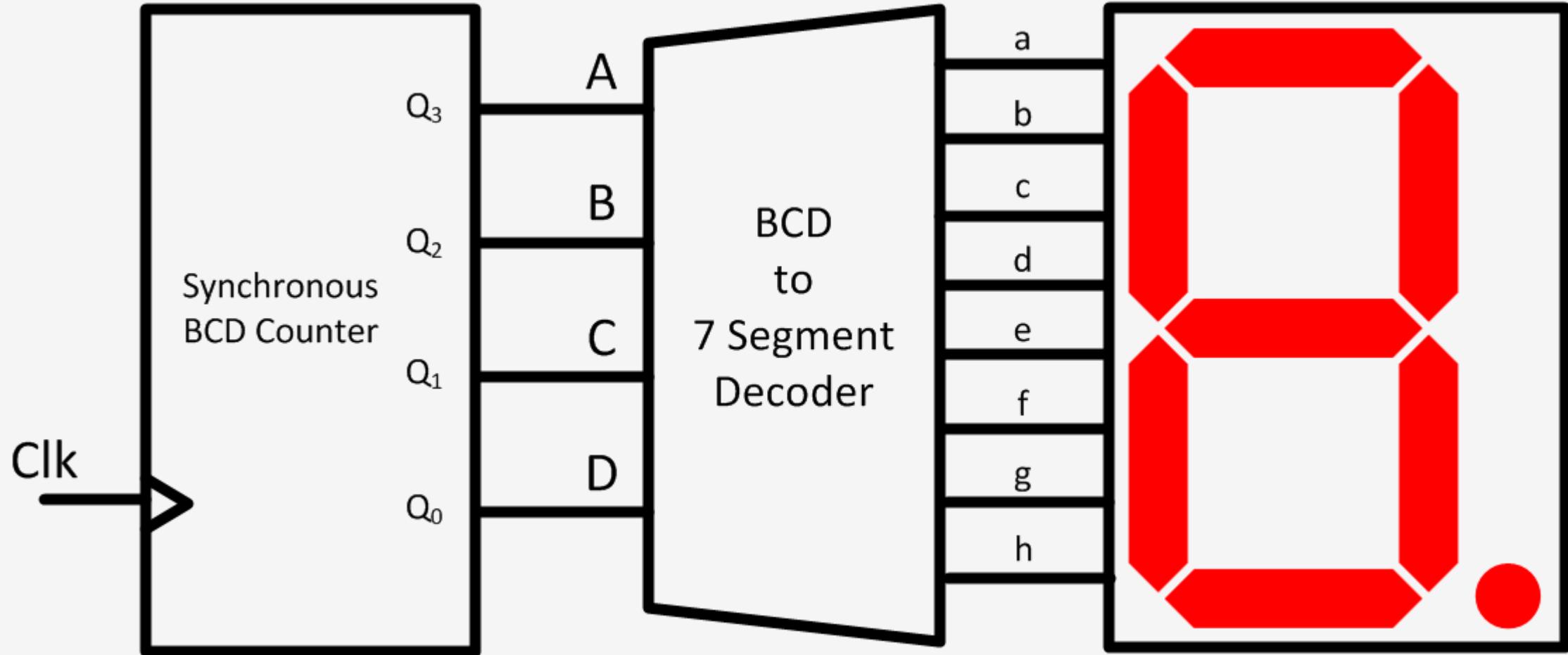
	$Q_1Q_0=00$	$Q_1Q_0=01$	$Q_1Q_0=11$	$Q_1Q_0=10$
$Q_3Q_2=00$	0	1	0	1
$Q_3Q_2=01$	0	1	0	1
$Q_3Q_2=11$	0	0	0	0
$Q_3Q_2=10$	0	0	0	0

$$D_1 = \bar{Q}_3 \cdot (Q_1 \oplus Q_0)$$

	$Q_1Q_0=00$	$Q_1Q_0=01$	$Q_1Q_0=11$	$Q_1Q_0=10$
$Q_3Q_2=00$	1	0	0	1
$Q_3Q_2=01$	1	0	0	1
$Q_3Q_2=11$	0	0	0	0
$Q_3Q_2=10$	1	0	0	0

$$D_0 = (\bar{Q}_3 \cdot \bar{Q}_0) + (\bar{Q}_2 \cdot \bar{Q}_1 \cdot \bar{Q}_0)$$

BCD to 7 Segment Counter - Scheme



Laboratory Assignment 2

Due Date: March 2, 11:30PM on BeachBoard Dropbox

Late Submission Policy:

One day delay (Second Due Date: Mar. 3, 11:30PM): 25% deduction

Two days delay (Third Due Date: Mar. 4, 11:30PM): 50% deduction

More than two days delay: No credit

Demo Time:

Mar. 5, 8:00AM - 10:45AM

Mar. 5, 1:00PM - 3:45PM

Academic Integrity: There is zero tolerance for cheating, plagiarism, or any other act of violation of Academic Integrity.

Midterm 1

Date & Time: Friday, Feb. 19, 11:00AM on BeachBoard Quizzes.

Review Session: Friday, Feb 19, 9:00AM (Bring your breakfast!)

Coverage: Topic 1 (Combinational Circuits) and Topic 2 (Sequential Circuits.)

Type: Open book/notes/slides and you are allowed to use internet (but no communication with others)

Questions: True/False & Multiple Choices (& maybe Short Answer)

Note1: Time is limited.

Note2: You are not allowed to move backwards through questions.

Note3: Everyone has to start the exam with at most 5 minutes delay. For every 31 minute after 11:05AM, you will receive a penalty of -2pt.

Quiz 2

Please take Quiz 2 on BeachBoard.

Objective: Evaluate your knowledge sequential circuits.

Time: 10 Minutes.

Note: Attendance guarantees 50%.

Midterm 1 Review

Amin Rezaei

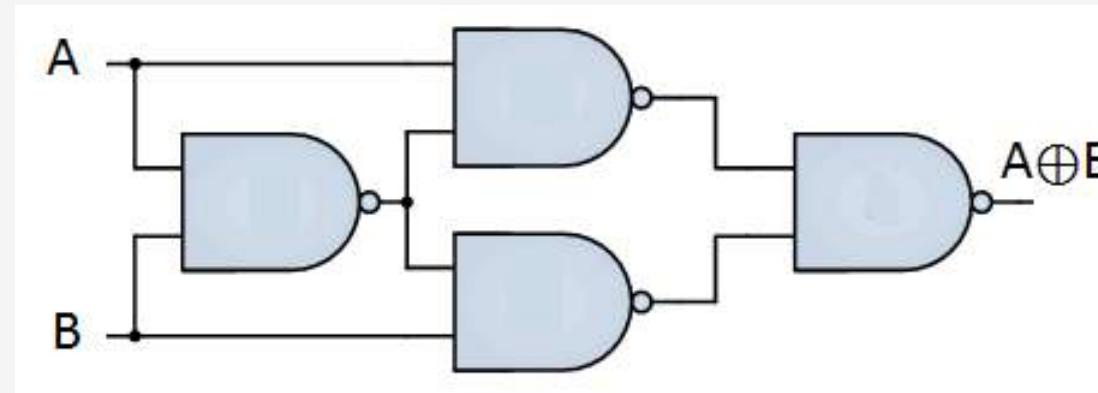
CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Universal Gates

Design a two-input XOR gate using two-input NAND gates.



Reducing Boolean Expressions

Reduce each of the following expressions:

$$(a) \text{Out} = A + \overline{(B \cdot A)}$$

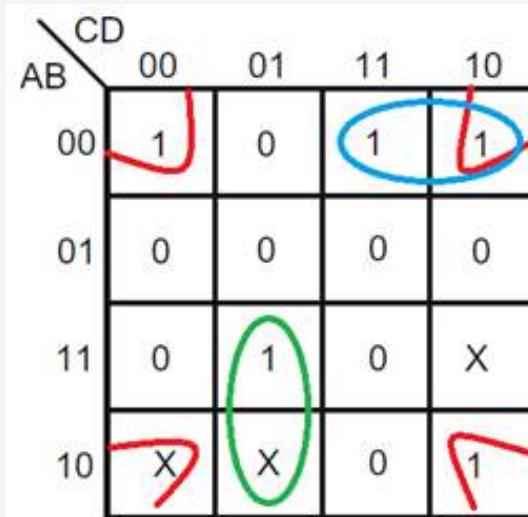
$$\begin{aligned}&= A + (\overline{B} + \overline{A}) \quad (\text{DeMorgan's}) \\&= A + (\overline{A} + \overline{B}) \quad (\text{Commutative}) \\&= (A + \overline{A}) + \overline{B} \quad (\text{Associative}) \\&= 1 + \overline{B} \quad (\text{Identity}) \\&= 1 \quad (\text{Identity})\end{aligned}$$

$$(b) \text{Out} = \overline{(A \cdot B)} \cdot (\overline{A} + B) \cdot (\overline{B} + B)$$

$$\begin{aligned}&= \overline{(A \cdot B)} \cdot (\overline{A} + B) \cdot 1 \quad (\text{Identity}) \\&= (\overline{A} + \overline{B}) \cdot (\overline{A} + B) \quad (\text{Identity & DeMorgan's}) \\&= \overline{A} + (\overline{B} \cdot B) \quad (\text{Inverse Distributive}) \\&= \overline{A} + 0 \quad (\text{Identity}) \\&= \overline{A} \quad (\text{Identity})\end{aligned}$$

Karnaugh Map

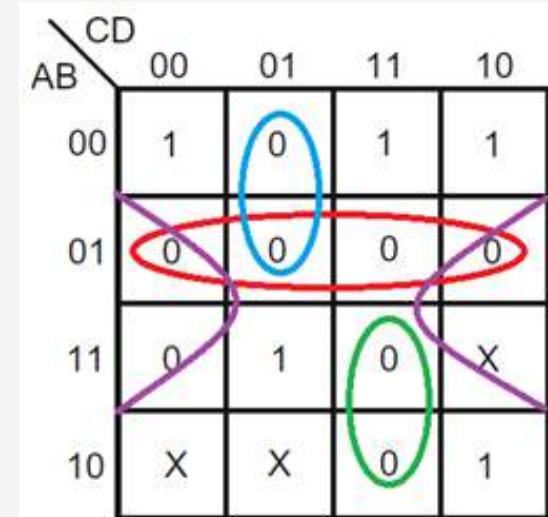
Determine the minimized SOP and minimized POS forms of the output function from the following K-map.



$$\text{Out} = (\bar{B} \cdot \bar{D}) + (A \cdot \bar{C} \cdot D) + (\bar{A} \cdot \bar{B} \cdot C)$$

AB\CD	00	01	11	10
00	1	0	1	1
01	0	0	0	0
11	0	1	0	X
10	X	X	0	1

$$\text{Out} = (A + \bar{B}) \cdot (\bar{B} + D) \cdot (A + C + \bar{D}) \cdot (\bar{A} + \bar{C} + \bar{D})$$

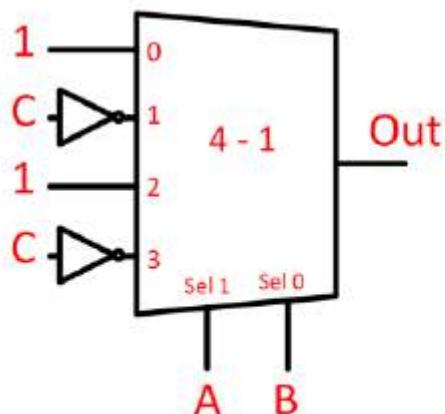


Multiplexer

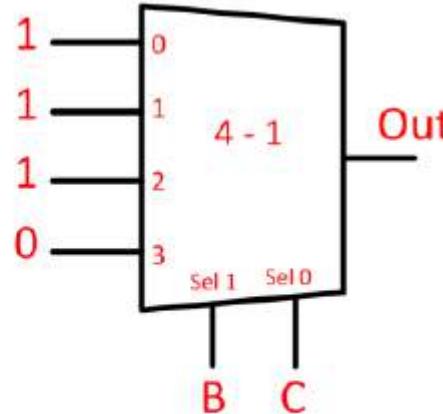
Implement the following function using: (a) One 8-1 MUX (b) One 4-1 MUX (and Inverters if you need.)

			8:1	4:1
A	B	C	Out	Out
0	0	0	1	1
0	0	1	1	
0	1	0	1	\bar{C}
0	1	1	0	
1	0	0	1	1
1	0	1	1	
1	1	0	1	\bar{C}
1	1	1	0	

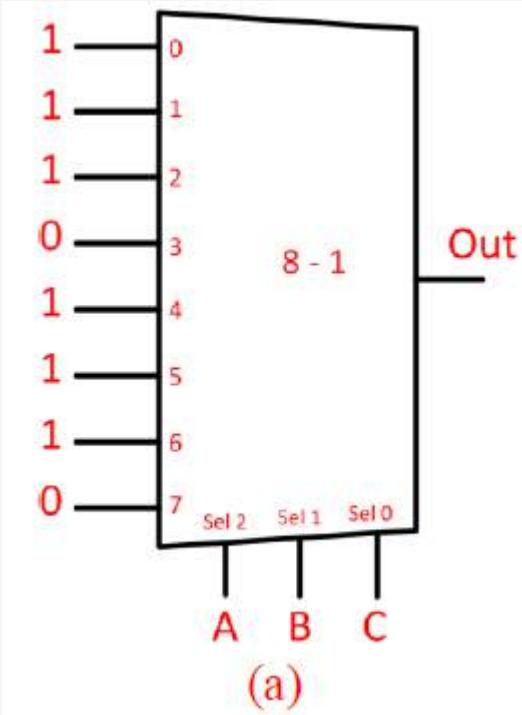
$$\text{Out} = (\bar{A} \cdot B \cdot \bar{C}) + (\bar{B} \cdot C)$$



(b) Solution 1



(b) Solution 2



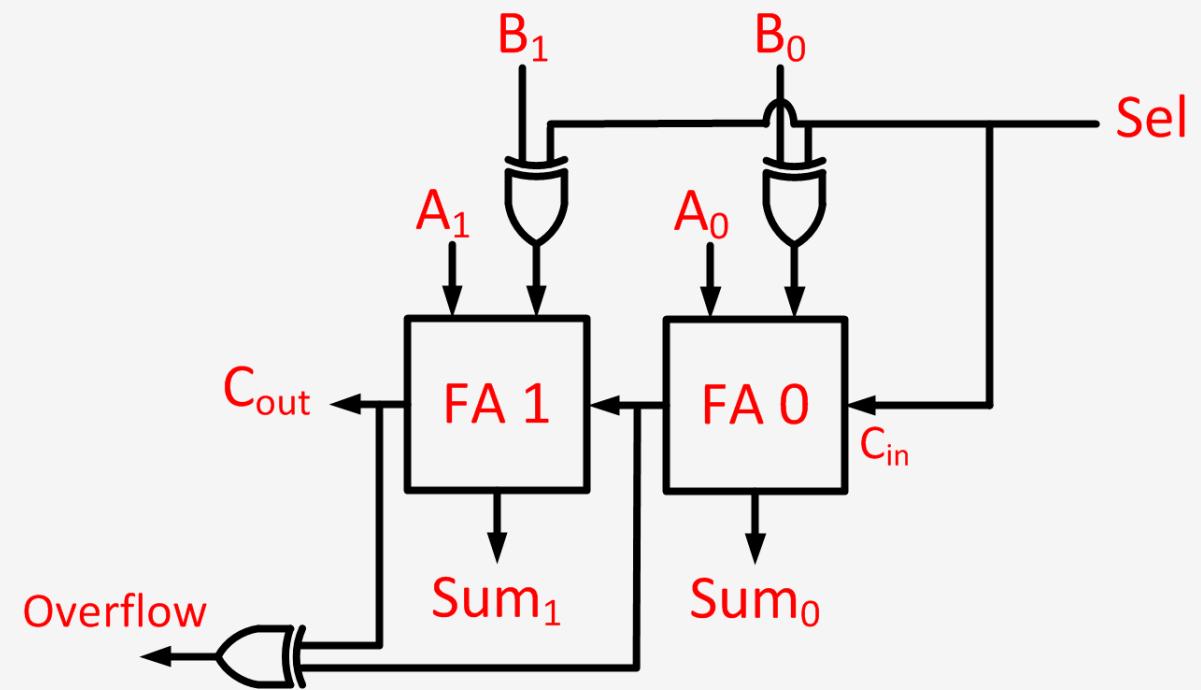
(a)

Adder

Design a two-bit ripple-carry adder & subtractor using one-bit full-adders and logic gates. The design should have a selection bit to choose between addition & subtraction. The design should also preserve the overflow flag.

$\text{Sel} = 0 \rightarrow \text{2-bit adder}$

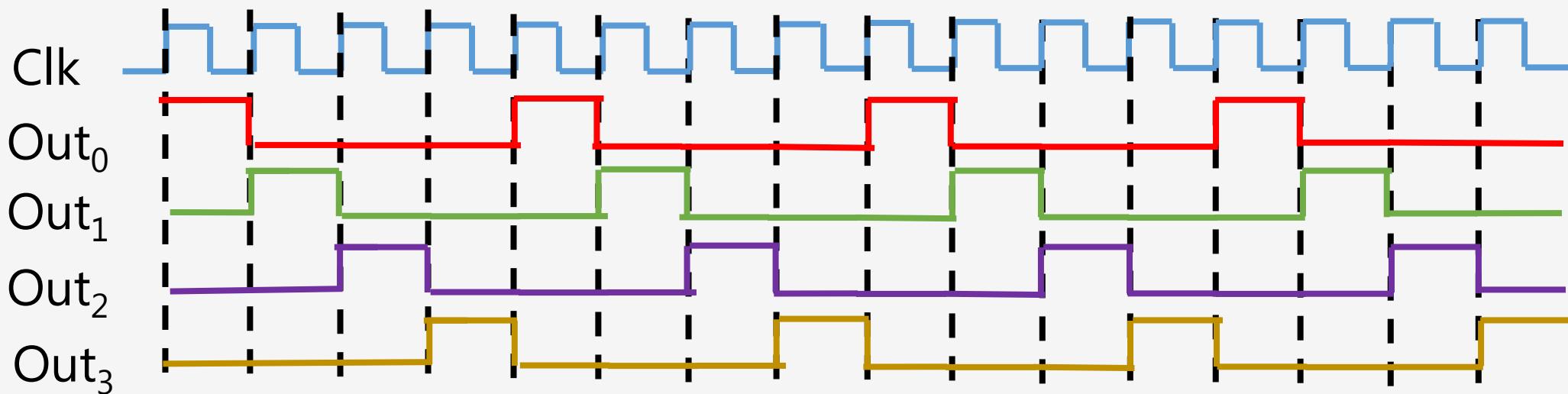
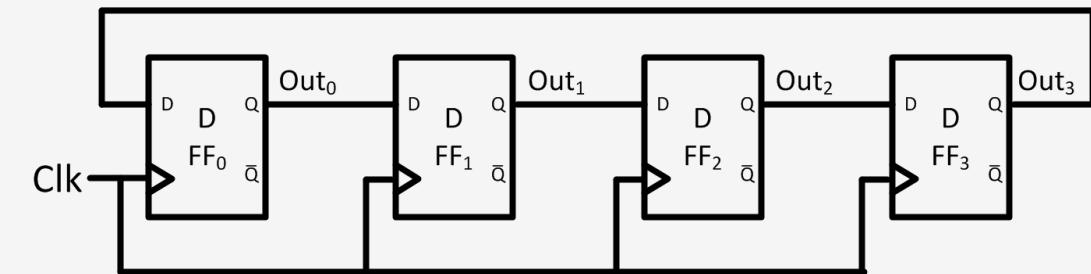
$\text{Sel} = 1 \rightarrow \text{2-bit subtractor}$



Counter 1

Draw the waveforms for four-bit ring counter. Suppose FF0 is initialized to 1 and all the other FFs are initialized to 0. How many numbers can be distinguished by four-bit ring counter?

It can count up to 4 distinct numbers.

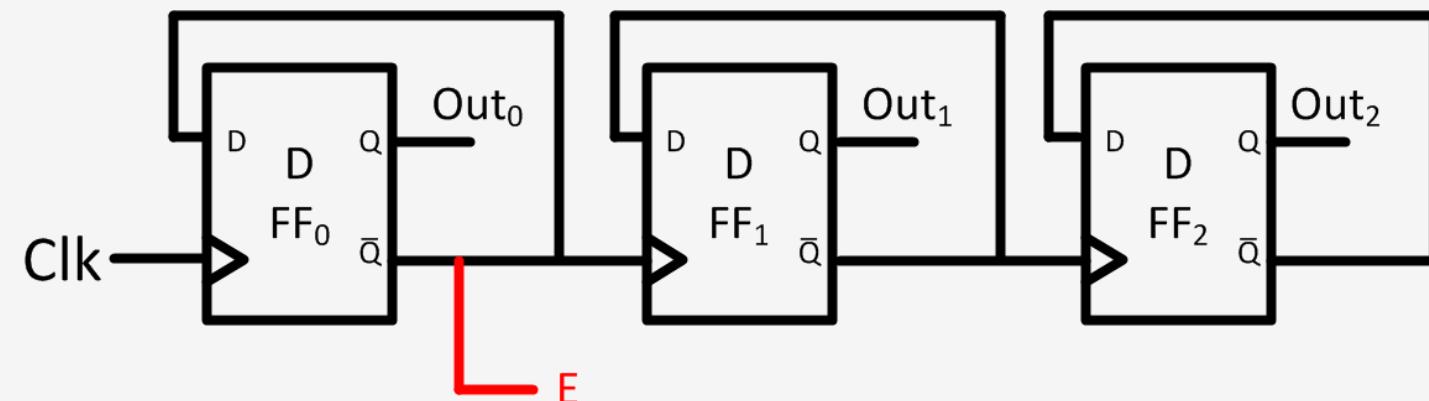


Counter 2

Design a three-bit asynchronous up binary counter that detects even numbers.

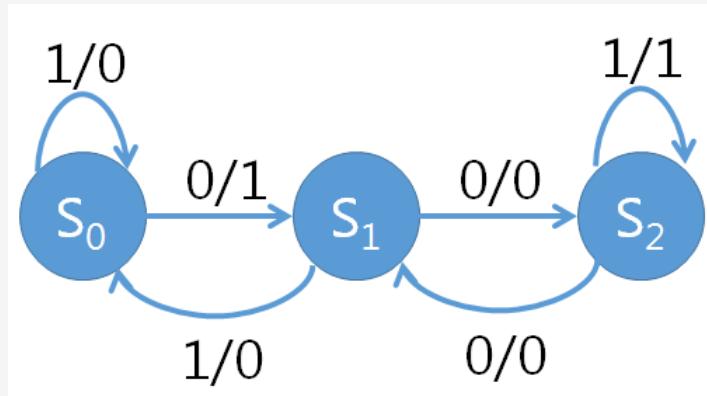
Out2	Out1	Out0	E
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Out ₂ =0	Out ₁ Out ₀ =00	Out ₁ Out ₀ =01	Out ₁ Out ₀ =11	Out ₁ Out ₀ =10
Out ₂ =0	1	0	0	1
Out ₂ =1	1	0	0	1

$$E = \overline{Out_0}$$


Finite State Machine 1

Convert the given Mealy machine to Moore machine.



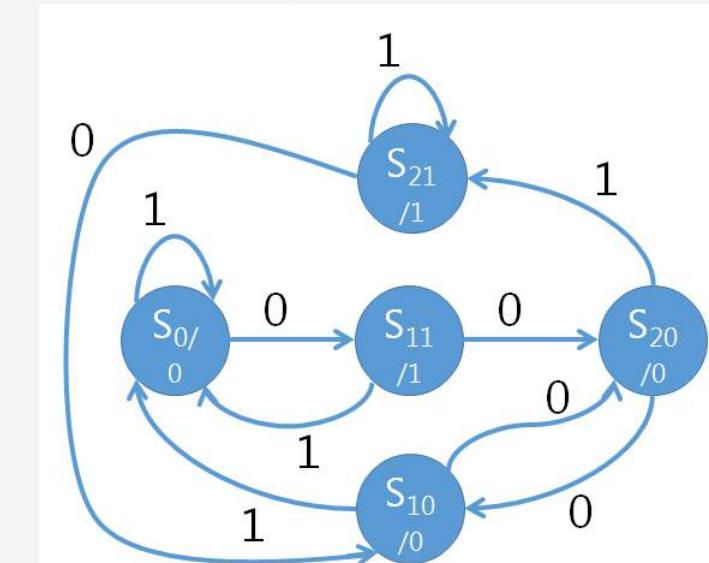
(2) Drawing Moore transition table

Current State	Input = 0		Input = 1	
	Next State	Output	Next State	Output
S_0	S_1	1	S_0	0
S_1	S_2	0	S_0	0
S_2	S_1	0	S_2	1

Current State	Input = 0	Input = 1	Output
	Next State	Next State	
S_0	S_{11}	S_0	0
S_{10}	S_{20}	S_0	0
S_{11}	S_{20}	S_0	1
S_{20}	S_{10}	S_{21}	0
S_{21}	S_{10}	S_{21}	1

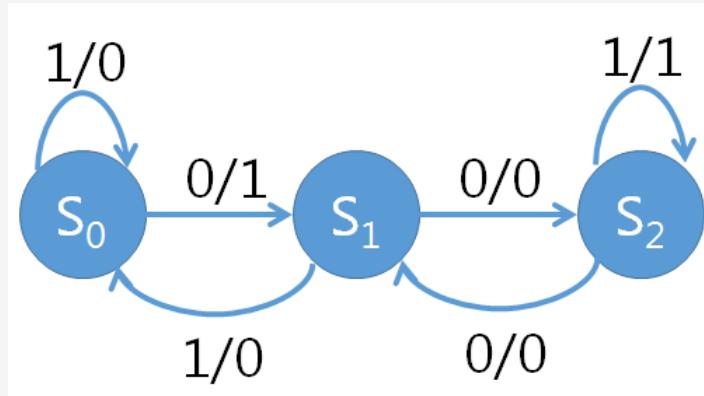
(1) Drawing Mealy transition table

(3) Drawing Moore state diagram



Finite State Machine 2

Design the given Mealy machine using D FFs and logic gates.



Q_1	Q_0	I	$Q'_1 = D_1$	$Q'_0 = D_0$	O
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	1	0	1
1	1	0	X	X	X
1	1	1	X	X	X

(1) Drawing truth table

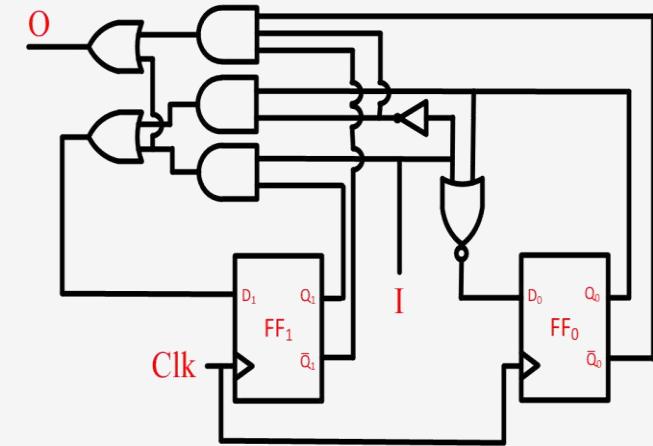
$Q_0 I = 00$	$Q_0 I = 01$	$Q_0 I = 11$	$Q_0 I = 10$
$Q_1 = 0$	0	0	0
$Q_1 = 1$	0	1	X

$$D_1 = Q_1 I + Q_0 \bar{I}$$

$Q_0 I = 00$	$Q_0 I = 01$	$Q_0 I = 11$	$Q_0 I = 10$
$Q_1 = 0$	1	0	0
$Q_1 = 1$	1	0	X

$$D_0 = \bar{Q}_0 \bar{I} = \overline{Q_0 + I}$$

$Q_0 I = 00$	$Q_0 I = 01$	$Q_0 I = 11$	$Q_0 I = 10$
$Q_1 = 0$	1	0	0
$Q_1 = 1$	0	1	X

$$O = Q_1 I + \bar{Q}_1 \bar{Q}_0 \bar{I}$$


(3) Building circuit

(2) Obtaining simplified equations

Combinational Timing Analysis

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Design Trade-Off

Electronic devices require small, low-power, and very fast logic circuits to work with advanced applications.

✓ **Area:** The smaller the better.

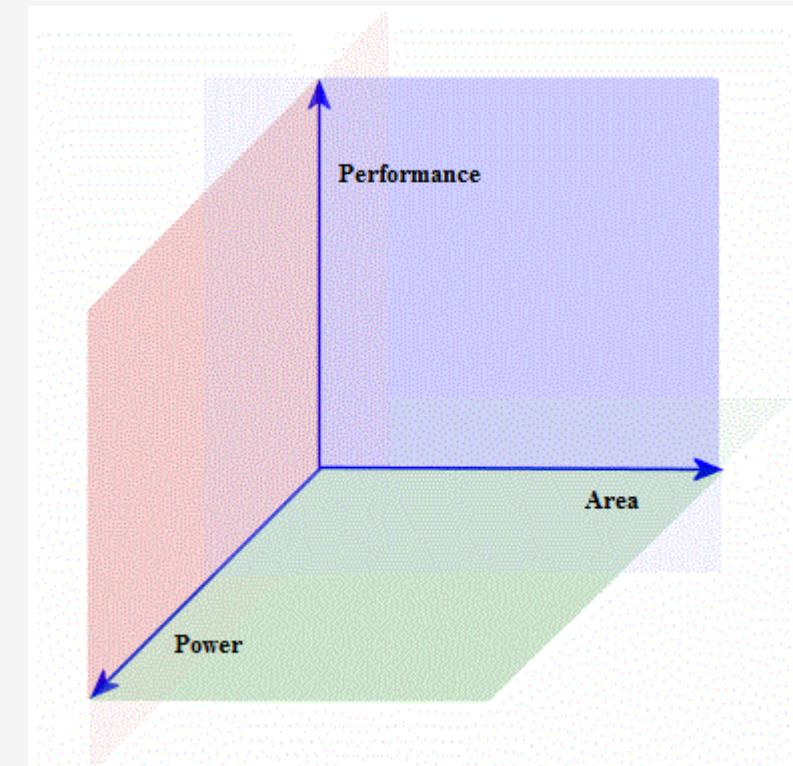
Application → Biomedical Chips

✓ **Power:** The lower the better.

Application → Smartphones

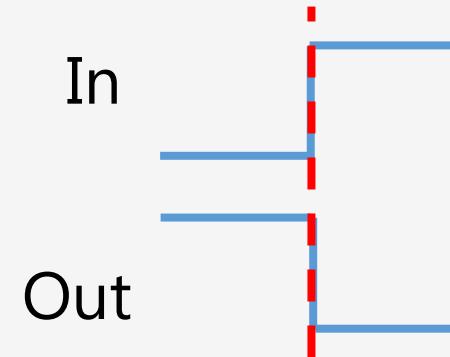
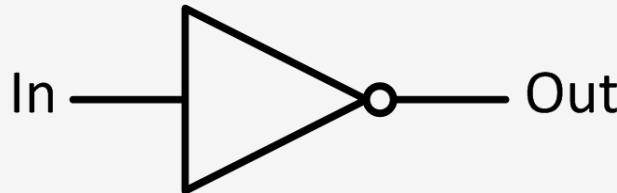
✓ **Speed:** The faster the better.

Application → Spacecrafts

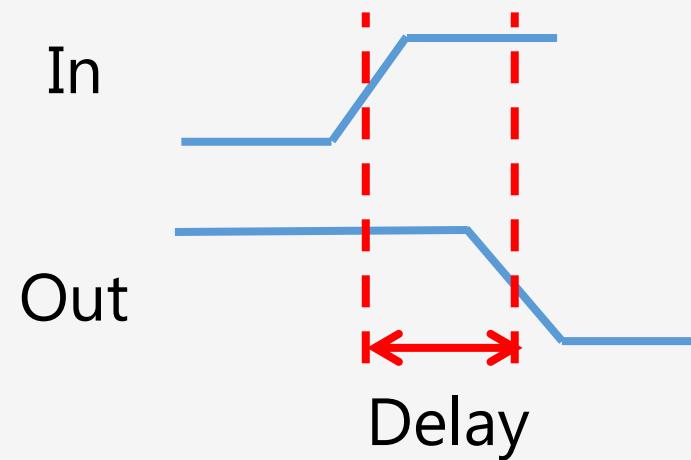
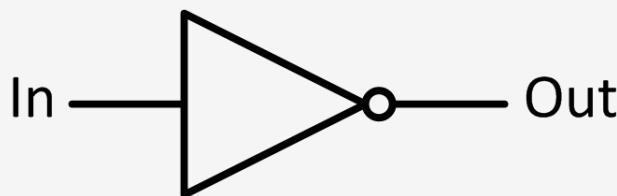


Combinational Circuit Delay

In theory, we assume that output changes **immediately** with the input in combinational circuits.



However, in practice, output changes with some **delays** from the input.



Delay Sources

Delay is fundamentally caused by capacitance and resistance in a circuit.

- ✓ **Rising** (i.e., $0 \rightarrow 1$) vs. **falling** (i.e., $1 \rightarrow 0$) signals
- ✓ **Process variation** (i.e., variation in the attributes of transistors during fabrication)
- ✓ **Temperature** changes
- ✓ **Aging** of the circuit

Types of Combinational Delay

Contamination delay (t_{cd}): Minimum time from when an input changes until any output **starts to change** its value.

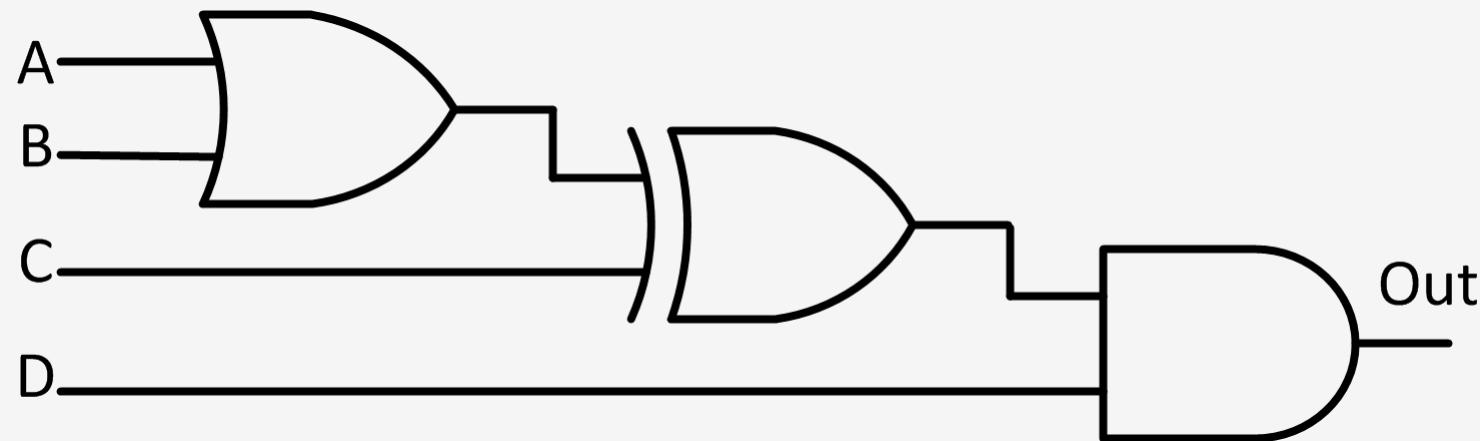
The contamination delay of a combinational circuit is the sum of the contamination delays through each element on the **shortest** path.

Propagation delay (t_{pd}): Maximum time from when an input changes until any output **finishes to change** its value.

The propagation delay of a combinational circuit is the sum of the propagation delays through each element on the **longest** (i.e., critical) path.

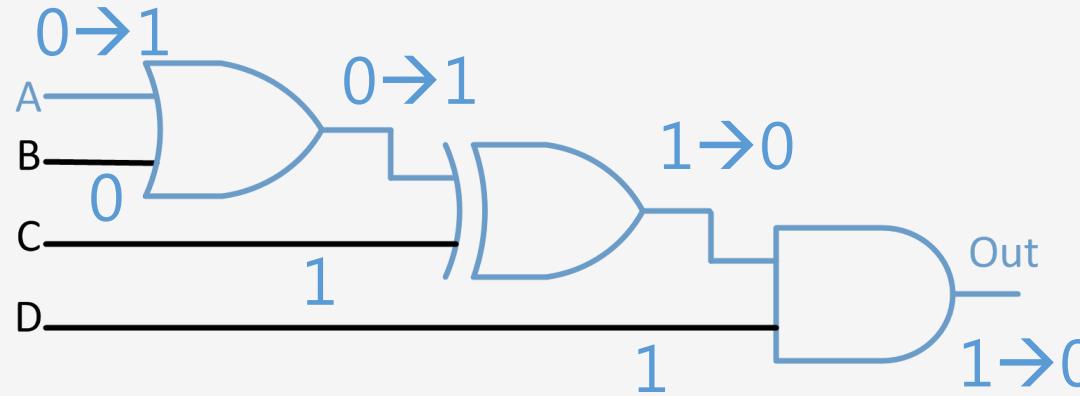
Combinational Delay - Sample Question

Suppose each gate has a propagation delay of 100ps and contamination delay of 60ps. Find the propagation delay and the contamination delay of the following circuit.



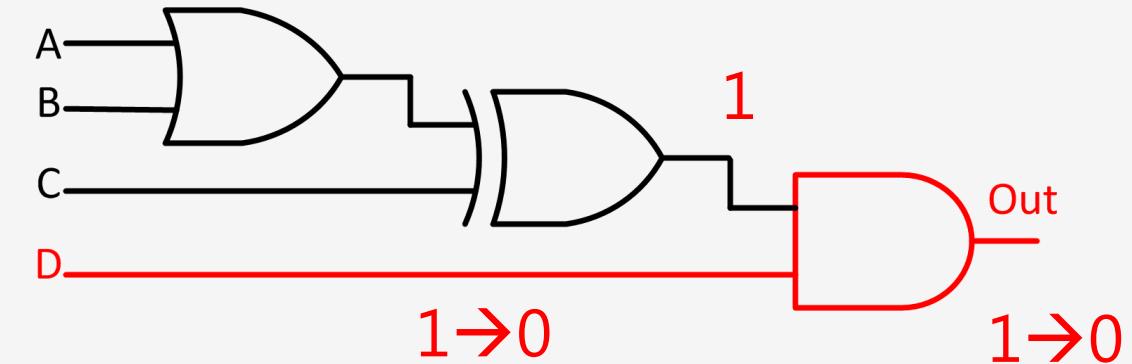
Combinational Delay - Sample Question - Solution

Suppose each gate has a propagation delay of 100ps and contamination delay of 60ps. Find the propagation delay and the contamination delay of the following circuit.



$$t_{pd} = t_{pd-OR} + t_{pd-XOR} + t_{pd-AND}$$

$$t_{pd} = 3 \times 100 = 300\text{ps}$$

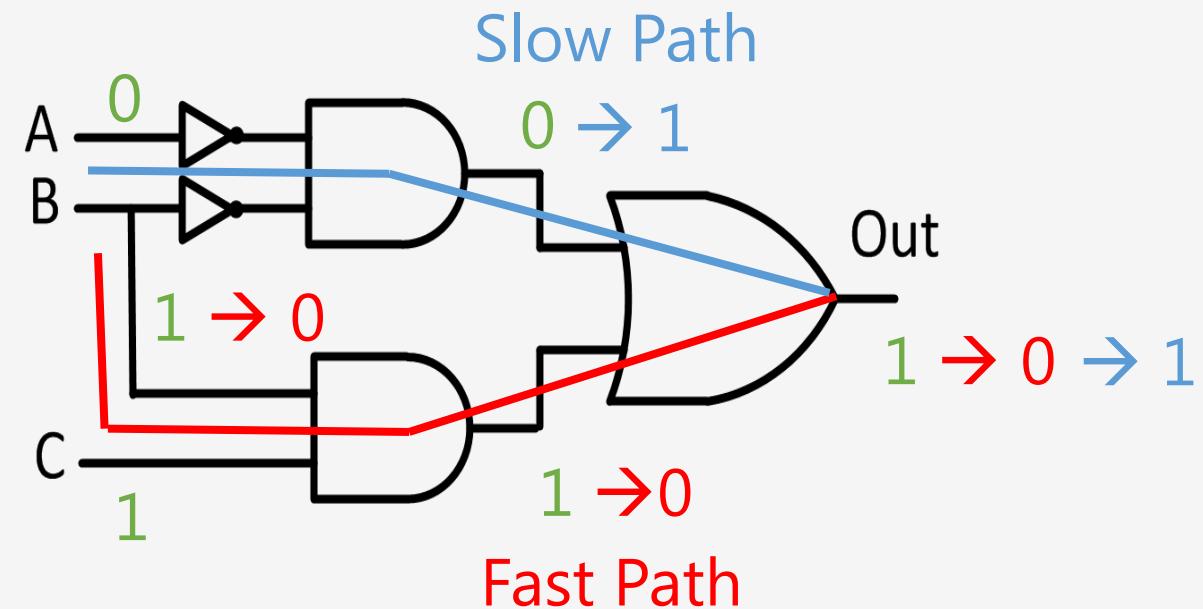
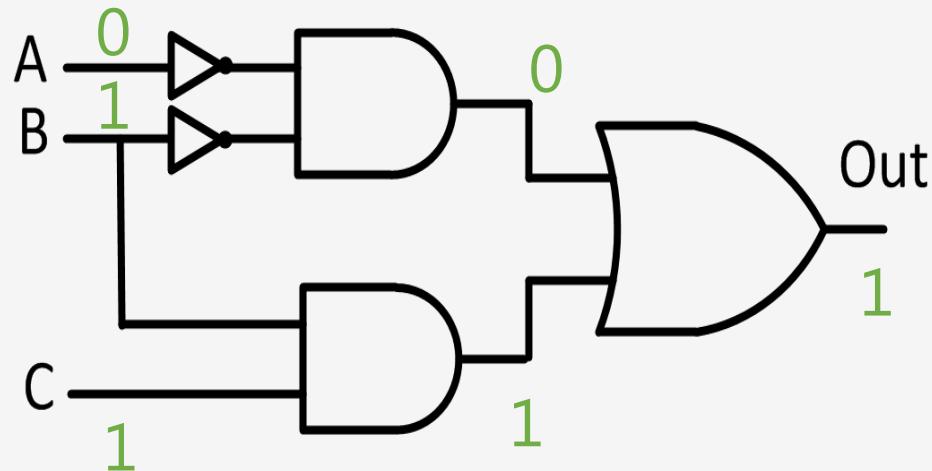


$$t_{cd} = t_{cd-AND}$$

$$t_{cd} = 60\text{ps}$$

Glitches

Glitch happens when one input transition causes multiple output transitions.



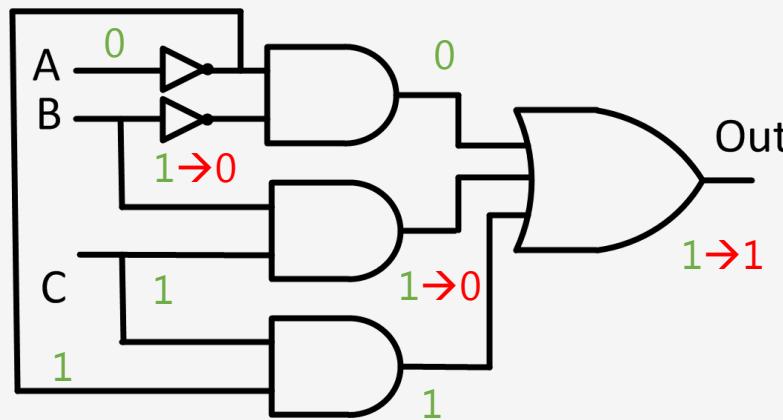
It is up to the designer to decide if glitches matter. If we only care about the long-term steady state output, we can safely ignore glitches.

Preventing Glitches in Karnaugh Map

Glitches are visible in K-map. A glitch occurs when moving between different groups.

	BC=00	BC=01	BC=11	BC=10
A=0	1	1	1	0
A=1	0	0	1	0

Glitches can be prevented by adding extra groups to the final formula in a way that avoids intergroup transitions.



	BC=00	BC=01	BC=11	BC=10
A=0	1	1	1	0
A=1	0	0	1	0

Topological Timing Analysis

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Topological Timing Analysis Usage

It answers the following question:

Will a circuit operate correctly at the intended clock frequency?



It tries to answer the following question:

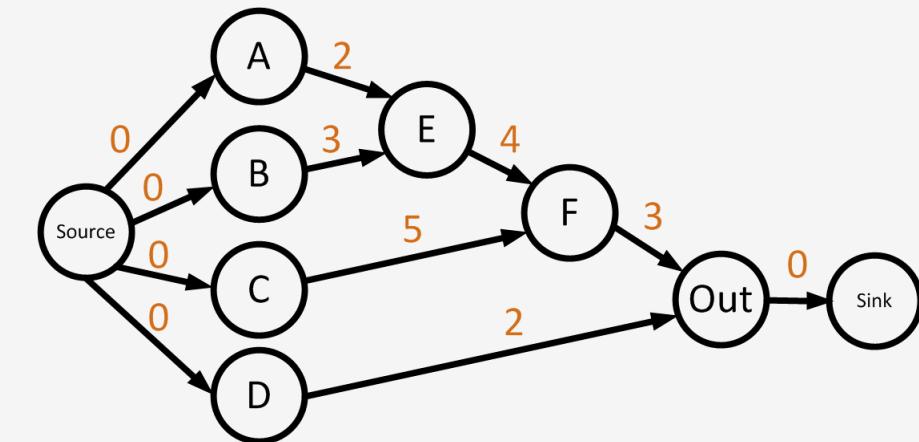
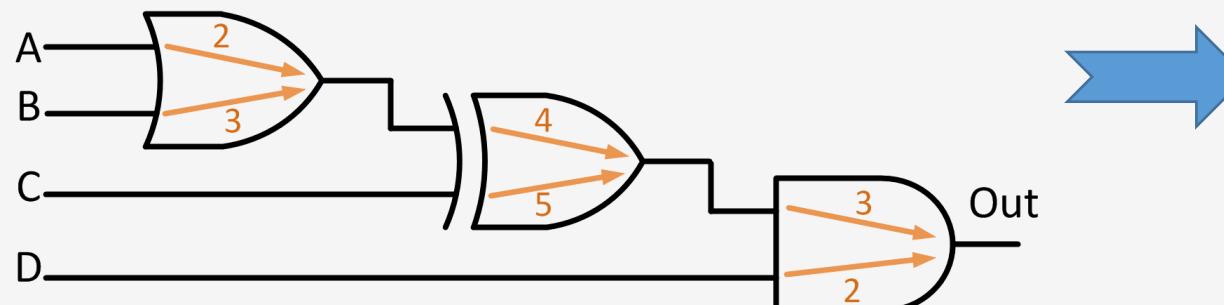
What can we do to improve the timing of the circuit?



Timing Graph

Each combinational circuit can be represented as Directed Acyclic Graph (DAG):

- ✓ Vertices denote wires (one for each input and one for each gate's output)
- ✓ Each edge denotes corresponding delay between two vertices.
- ✓ A source vertex is connected to the inputs, and the outputs are connected a sink vertex.



Arrival Times

Actual Arrival Time (AAT): AAT of vertex v is the longest path from source to v .

Required Arrival Time (RAT): RAT of vertex v is the latest time that the signal is allowed to leave v to make it to sink in time.

Slack: Slack at vertex v is the difference of RAT and AAT of v .

$$\text{Slack}(v) = \text{RAT}(v) - \text{AAT}(v)$$

Negative slack at any output means the circuit does not meet timing constraints.

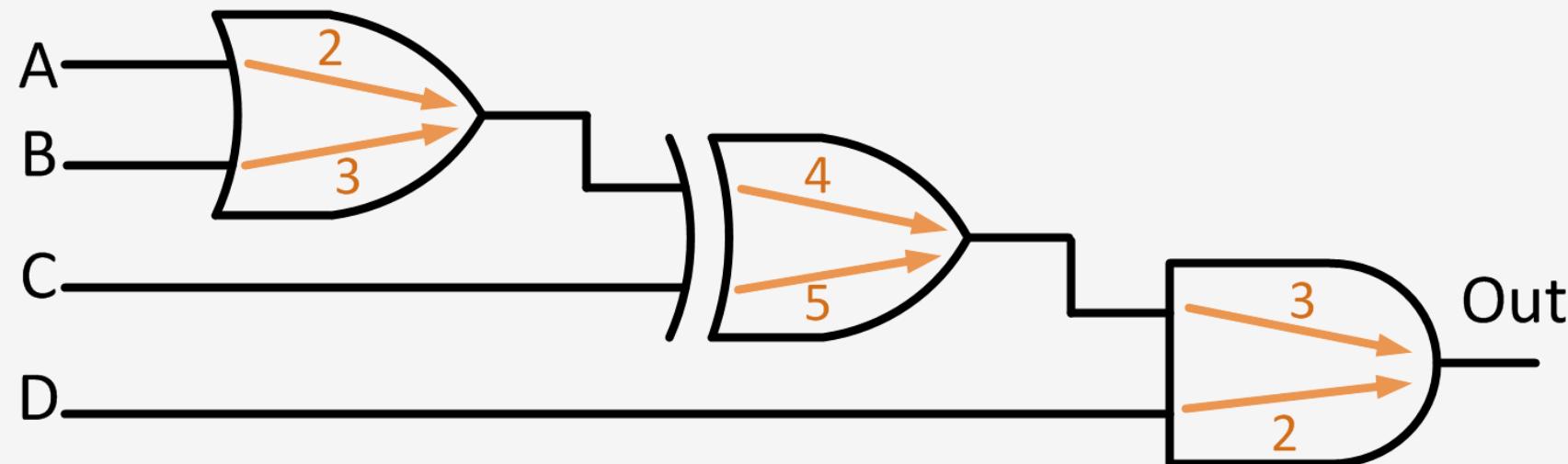
Positive slack at all outputs means the circuit meets timing constraints.

Computing Actual Arrival Time

- 1) Construct timing graph
- 2) Sort vertices in topological order (i.e., breadth-first order)
- 3) For each vertex v in sorted order, compute actual arrival time $\text{AAT}(v)$ as:
$$\forall w \in \text{inputs}(v): \text{MAX} \{ \text{AAT}(w) + D(w, v) \}$$

Actual Arrival Time - Sample Question

Compute actual arrival time of all the wires in the following design.

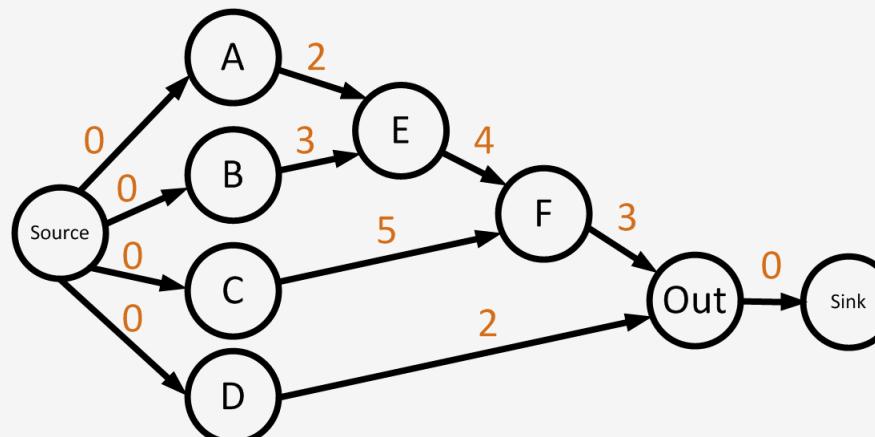
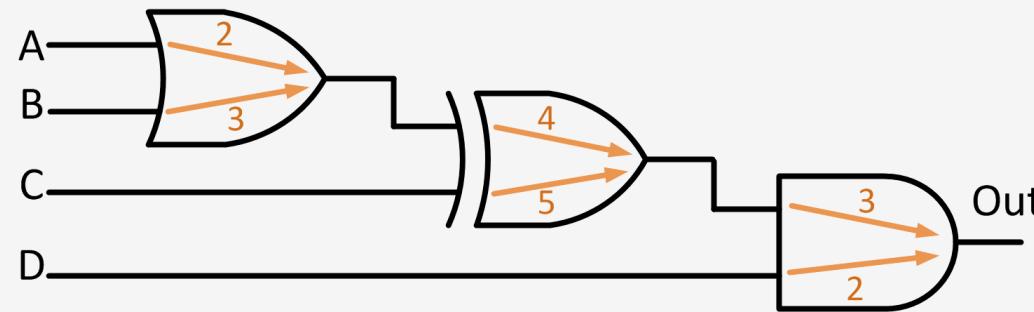


Actual Arrival Time - Sample Question - Solution

Compute actual arrival time of all the wires in the following design.

Vertex	AAT
A	0
B	0
C	0
D	0
E	3
F	7
Out	10

Order: A, B, C, D, E, F, Out



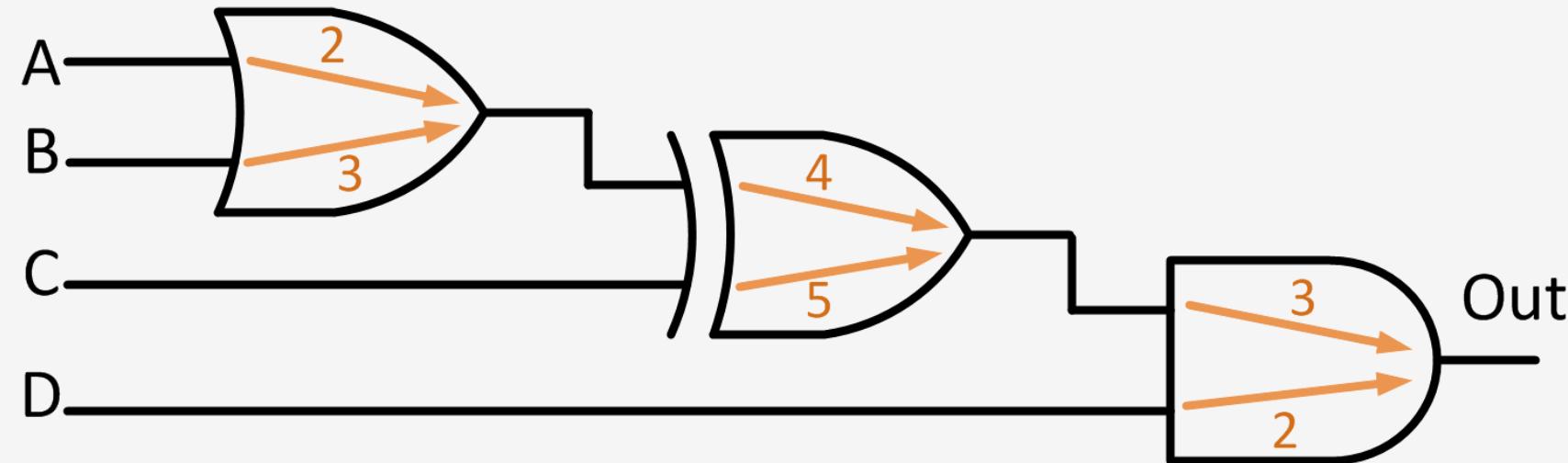
Computing Required Arrival Time

- 1) Construct timing graph
- 2) Sort vertices in topological order
- 3) Set required time at primary outputs
- 4) For each vertex v in reverse sorted order, compute required arrival time $\text{RAT}(v)$ as:

$$\forall w \in \text{fanouts}(v): \text{MIN } \{\text{RAT}(w) - D(w, v)\}$$

Required Arrival Time - Sample Question

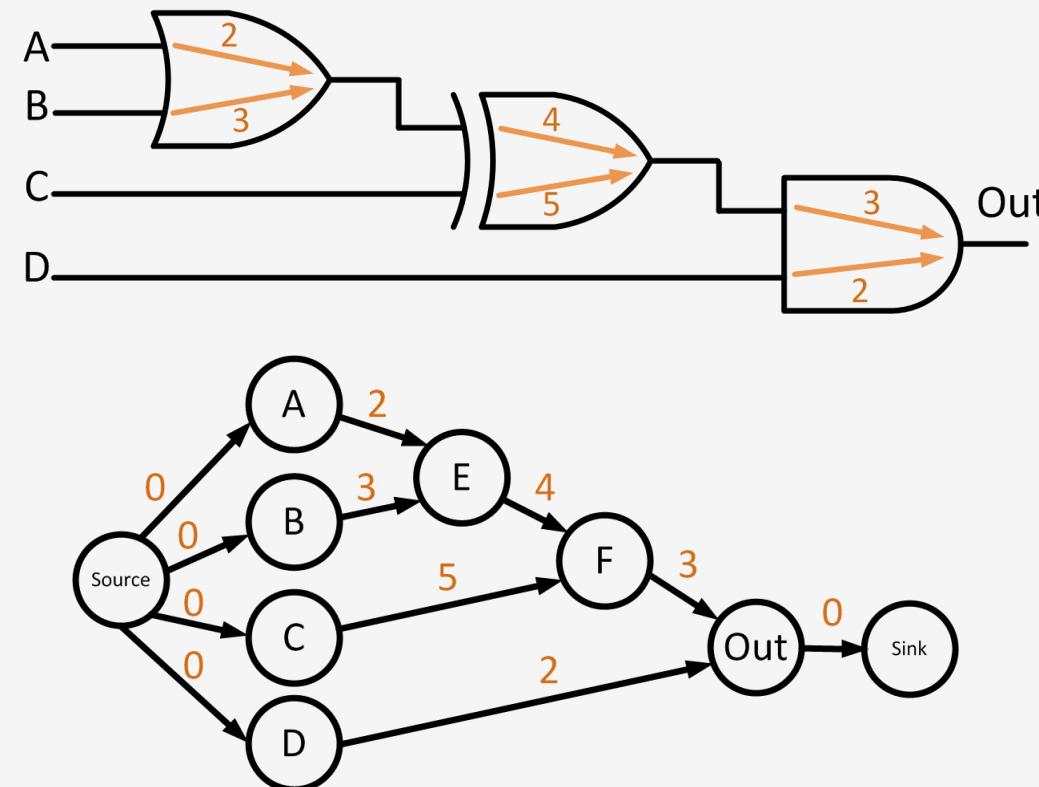
Compute required arrival time of all the wires in the following design. Suppose the clock cycle of $T_c = 9$.



Required Arrival Time - Sample Question - Solution

Compute required arrival time of all the wires in the following design. Suppose the clock cycle of $T_c = 9$.

Vertex	AAT	RAT
A	0	0
B	0	-1
C	0	1
D	0	7
E	3	2
F	7	6
Out	10	9



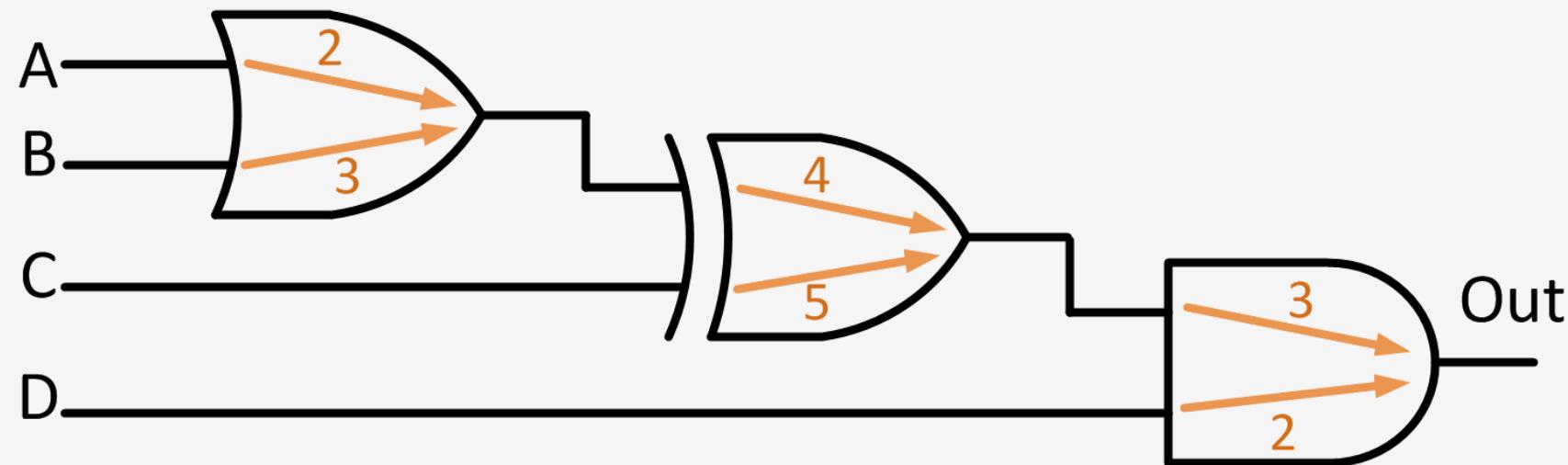
Reverse order: Out, F, E, D, C, B, A
19

Computing Critical Path

- 1) Construct timing graph
- 2) Sort nodes in topological order
- 3) Compute AAT, RAT, and Slack at each node
- 4) Compute Slack_{\min} as:
$$\forall \text{ vertices } v: \text{MIN} \{ \text{Slack}(v) \}$$
- 5) Trace path(s) through the network where each vertex v has $\text{Slack}(v) = \text{Slack}_{\min}$

Critical Path - Sample Question

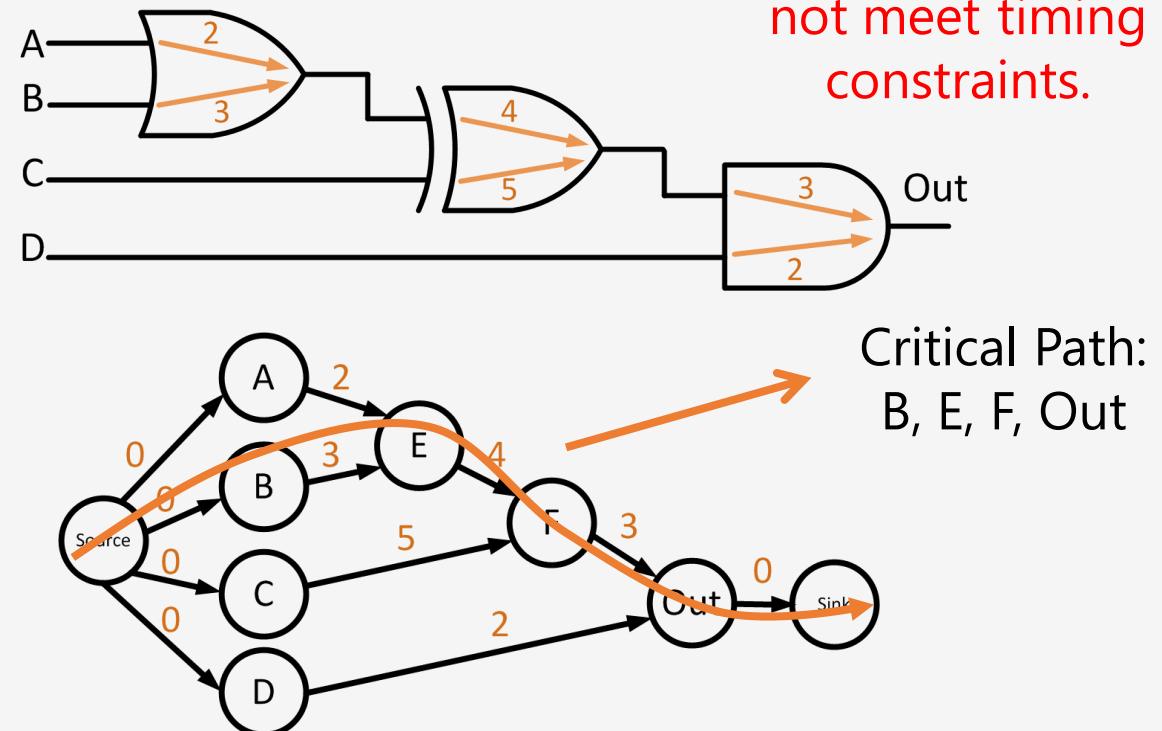
Compute slack of all the wires in the following design. Suppose the clock cycle of $T_c = 9$. What is the critical path? Does the design meet timing constraints?



Critical Path - Sample Question - Solution

Compute slack of all the wires in the following design. Suppose the clock cycle of $T_c = 9$. What is the critical path? Does the design meet timing constraints?

Vertex	AAT	RAT	Slack
A	0	0	0
B	0	-1	-1
C	0	1	1
D	0	7	7
E	3	2	-1
F	7	6	-1
Out	10	9	-1



ε -Critical Network

We are also interested in paths that are close-to-critical. These paths may become critical when we optimize the critical paths.

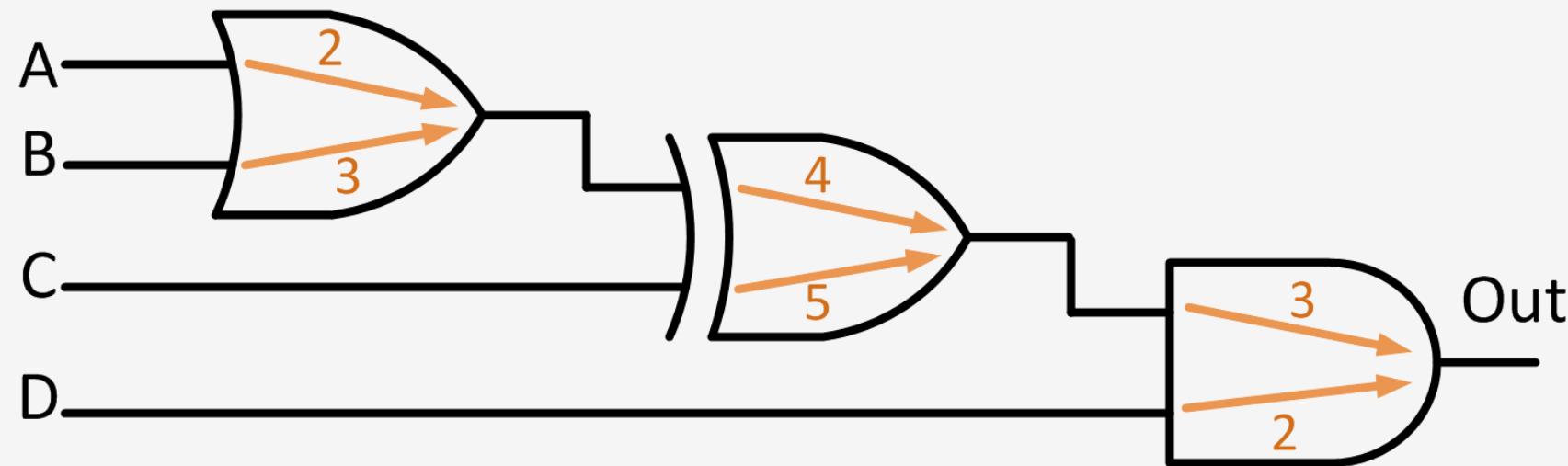
A vertex v is ε -critical if:

$$\text{Slack}(v) \leq \text{Slack}_{\min} + \varepsilon$$

The set of ε -critical vertices forms a connected network consisting of paths from sink to source called the **ε -critical network**.

ε -Critical Network - Sample Question

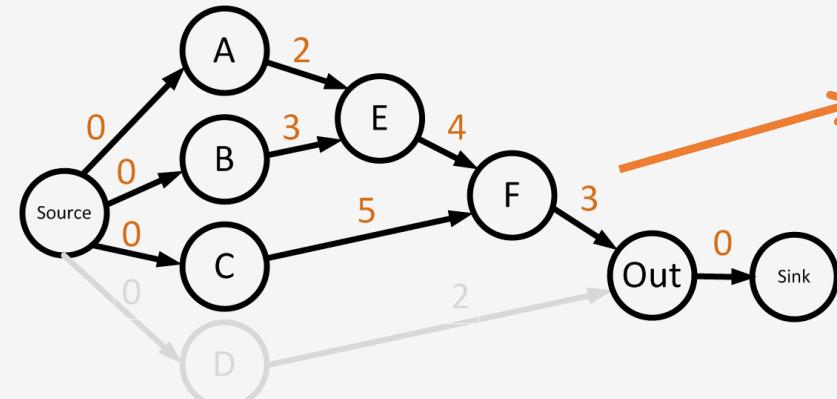
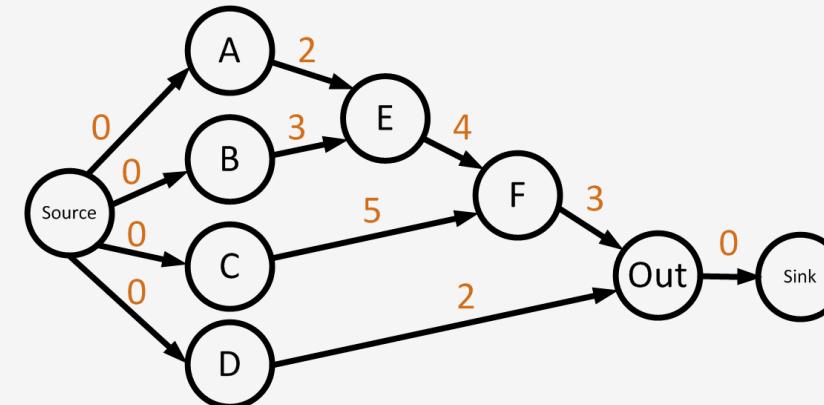
Suppose the clock cycle of $T_c = 9$. Find the 1-critical network in the following design.



ε -Critical Network - Sample Question - Solution

Suppose the clock cycle of $T_c = 9$. Find the 1-critical network in the following design.

Vertex	AAT	RAT	Slack
A	0	0	0
B	0	-1	-1
C	0	1	1
D	0	7	7
E	3	2	-1
F	7	6	-1
Out	10	9	-1



1-critical
network

Homework Assignment 3

Due Date: March 9, 11:30PM on BeachBoard Dropbox

Late Submission Policy:

One day delay (Second Due Date: Mar. 10, 11:30PM): 25% deduction

Two days delay (Third Due Date: Mar. 11, 11:30PM): 50% deduction

More than two days delay: No credit

Academic Integrity: There is zero tolerance for cheating, plagiarism, or any other act of violation of Academic Integrity.

Functional Timing Analysis

Amin Rezaei

CECS 301 - Computer Logic Design II

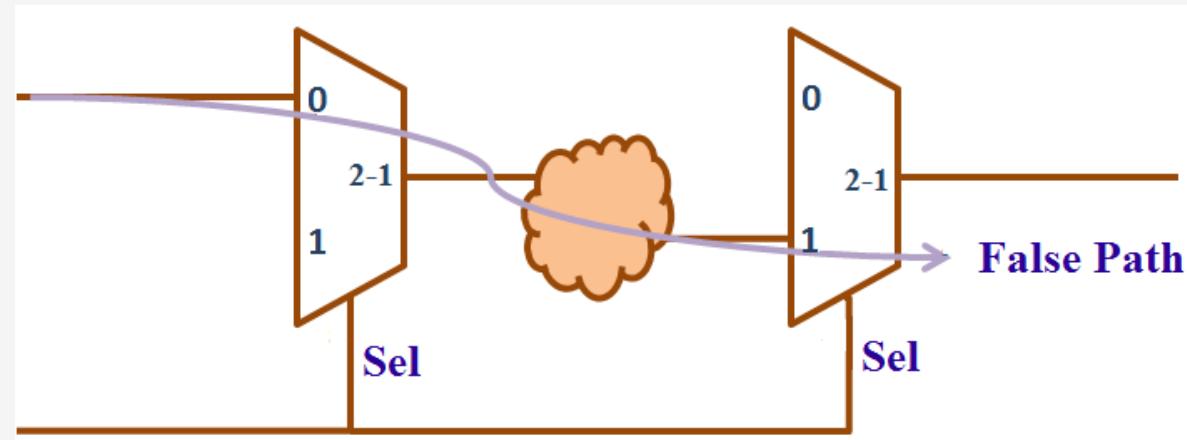
California State University, Long Beach

Spring 2021

False Path

In topological analysis, we only care about the delay in the timing graph not the logical functions of the vertices.

False Path: A path is a false path if no sequence of inputs can result in an event propagating along it.



A false path should not be responsible for the circuit delay.

Path Sensitization

A path is sensitized when it allows a logic signal to propagate along it.

The actual delay of the circuit should be defined as the length of the longest **sensitizable** path.

Controlling value at a gate input is the value that determines the output value of that gate irrespective of the other inputs.

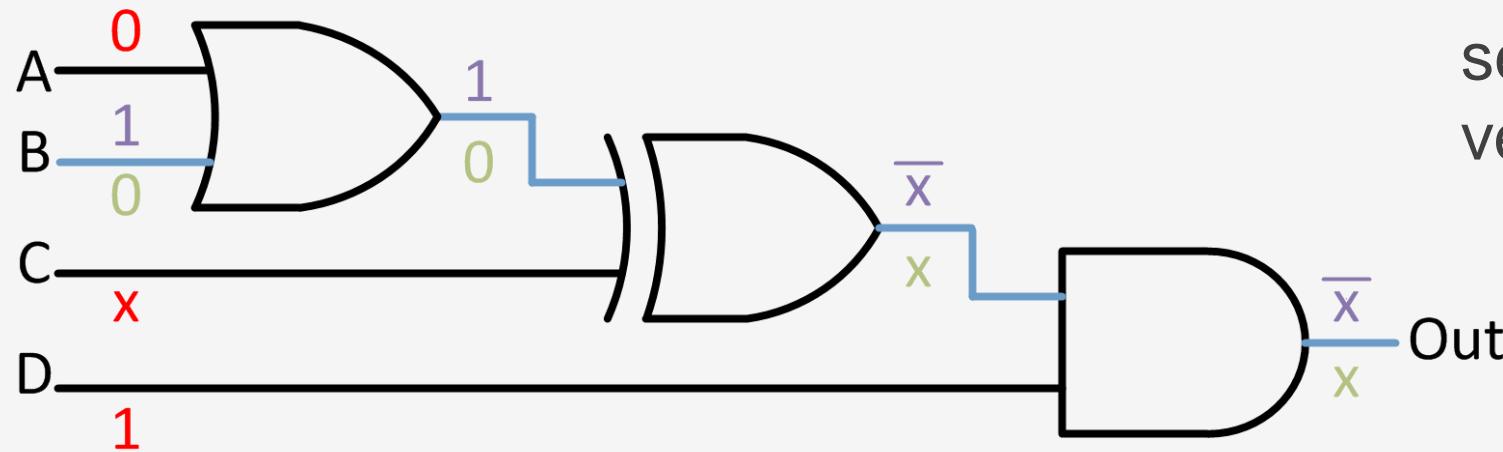
Controlling value
for AND gate is 0.

Controlling value
for OR gate is 1.

How about NAND
and NOR?

Static Sensitization

A path is statically sensitizable by an input vector, if the other incoming signals to the gates of that path have non-controlling values.



The blue path is statically sensitizable by input vector ACD=0X1.

Functional Analysis

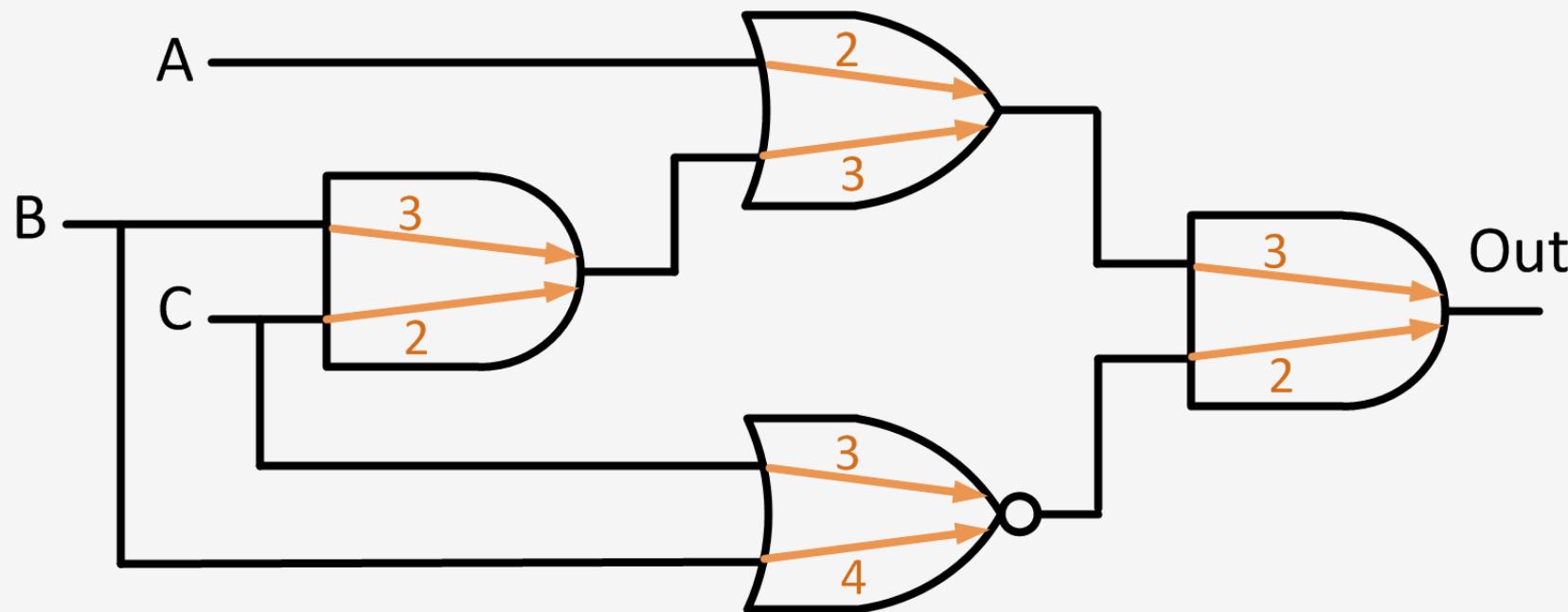
Once a critical path is detected using topological analysis, we need to make sure that the path is statically sensitizable.

Going back to the circuit, logic values that are necessary for the static sensitizability of the critical path should be determined.

Logic incompatibilities between the propagation conditions can be detected when opposite logic values are required at the same wire.

Functional Analysis - Sample Question

Find the critical path of the following circuit. Suppose the clock cycle of $T_c = 10$.



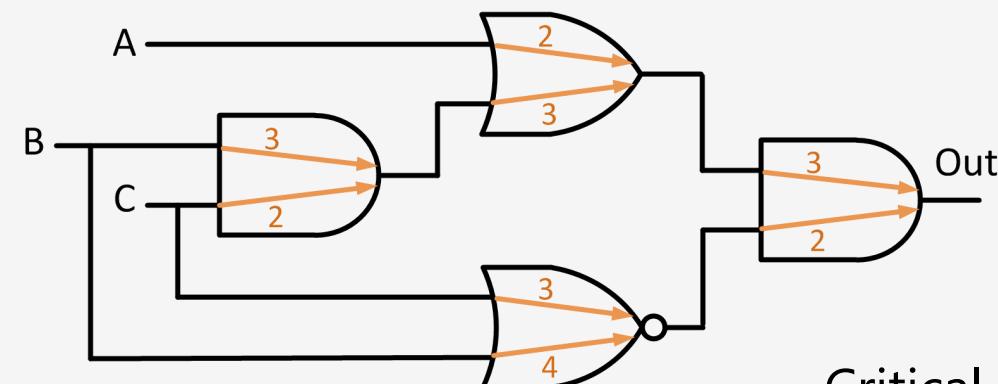
Functional Analysis - Sample Question - Solution

Find the critical path of the following circuit. Suppose the clock cycle of $T_c = 10$.

Vertex	AAT	RAT	Slack
A	0	5	5
B	0	1	1
C	0	2	2
D	3	4	1
E	6	7	1
F	4	8	4
Out	9	10	1

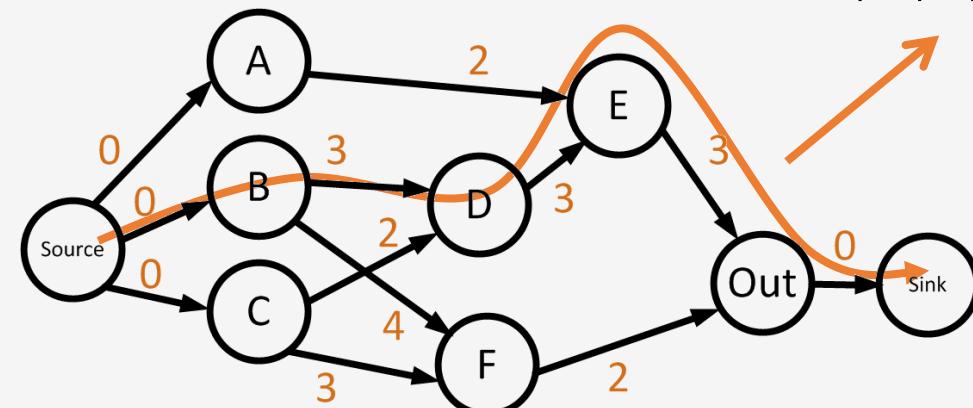
7

Order: A, B, C, D, E, F, Out



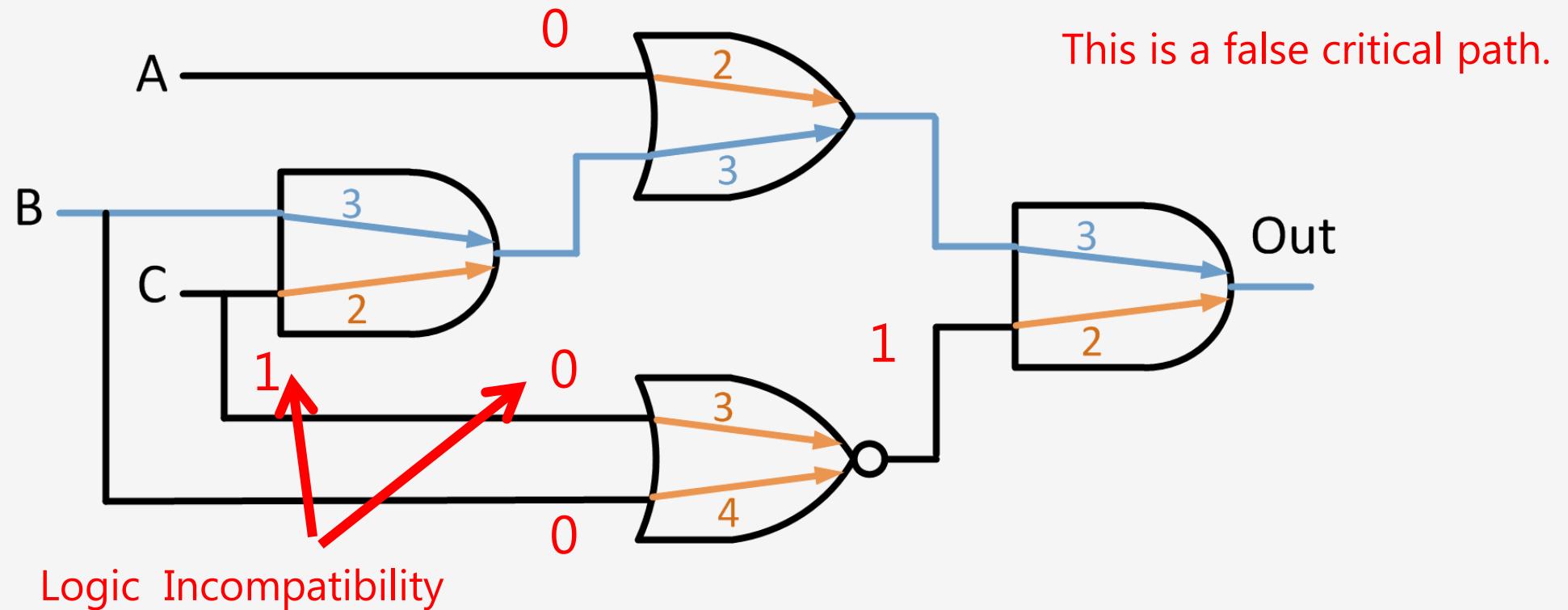
Critical Path:
B, D, E, Out

Is it?!



Functional Analysis - Sample Question - Solution

Find the critical path of the following circuit. Suppose the clock cycle of $T_c = 10$.



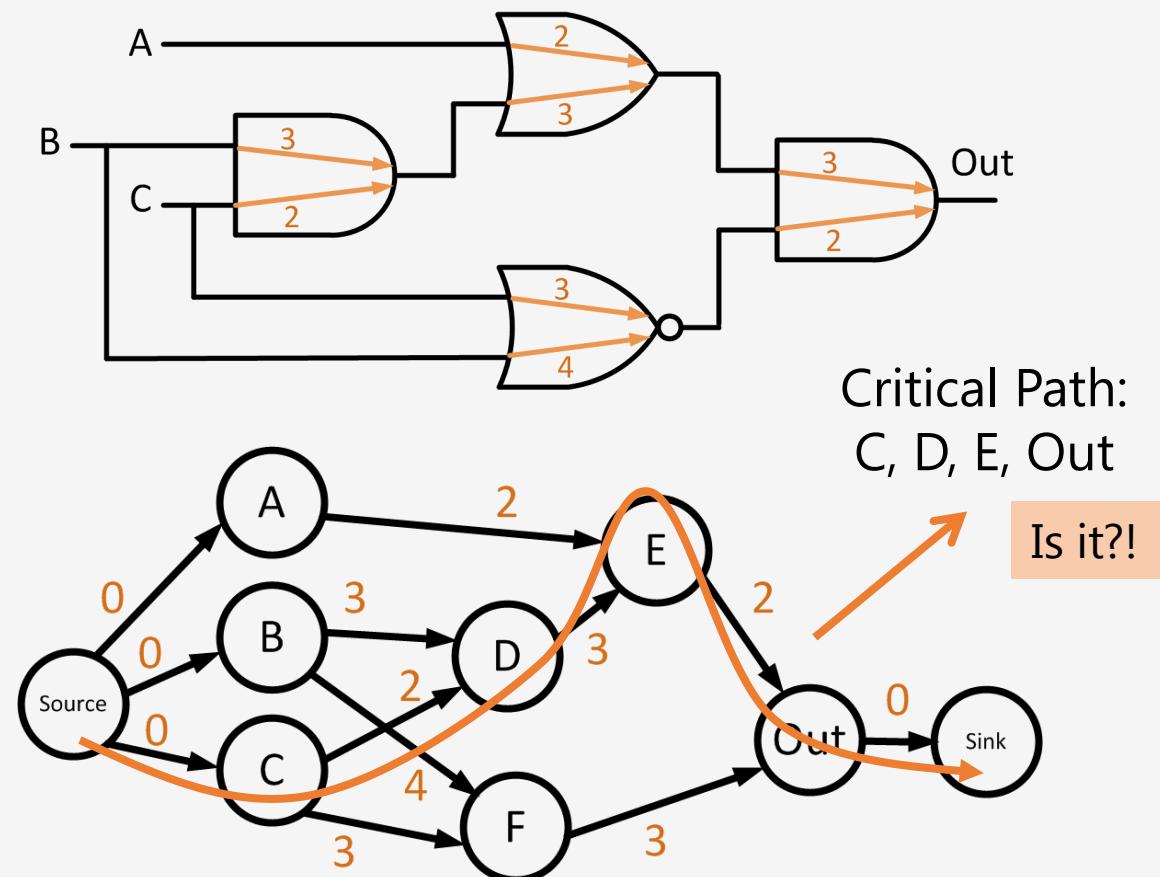
Functional Analysis - Sample Question - Solution

Find the critical path of the following circuit. Suppose the clock cycle of $T_c = 10$.

Vertex	AAT	RAT	Slack
A	0	5	5
B	0	1	1
C	0	2	2
D	3	4	1
E	6	7	1
F	4	8	4
Out	9	10	1

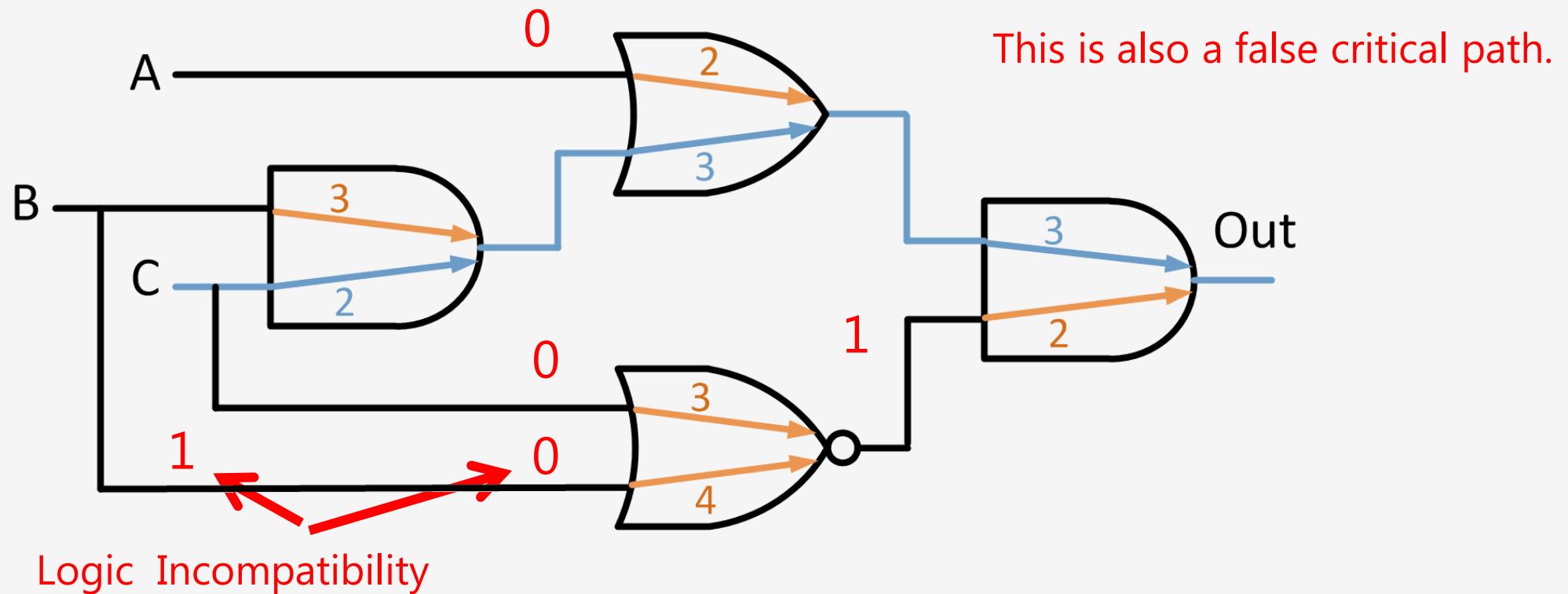
9

Order: A, B, C, D, E, F, Out



Functional Analysis - Sample Question - Solution

Find the critical path of the following circuit. Suppose the clock cycle of $T_c = 10$.

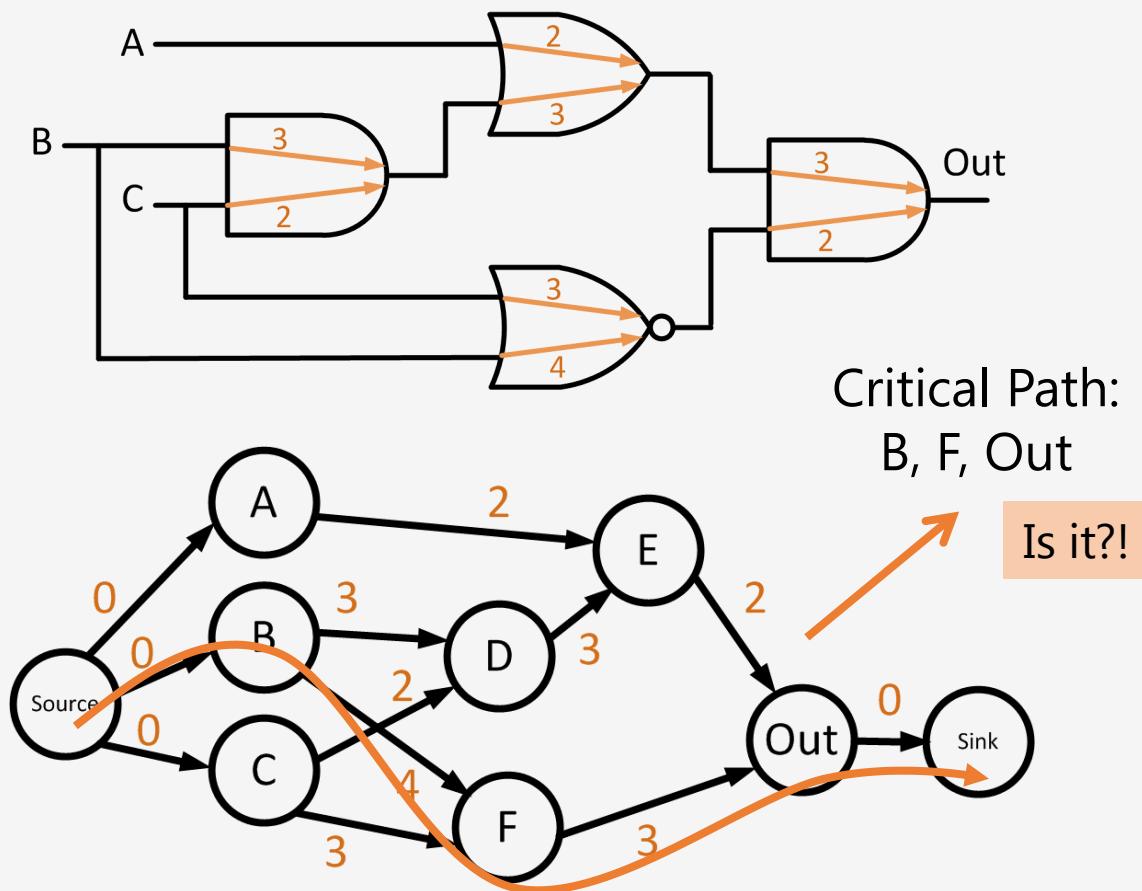


Functional Analysis - Sample Question - Solution

Find the critical path of the following circuit. Suppose the clock cycle of $T_c = 10$.

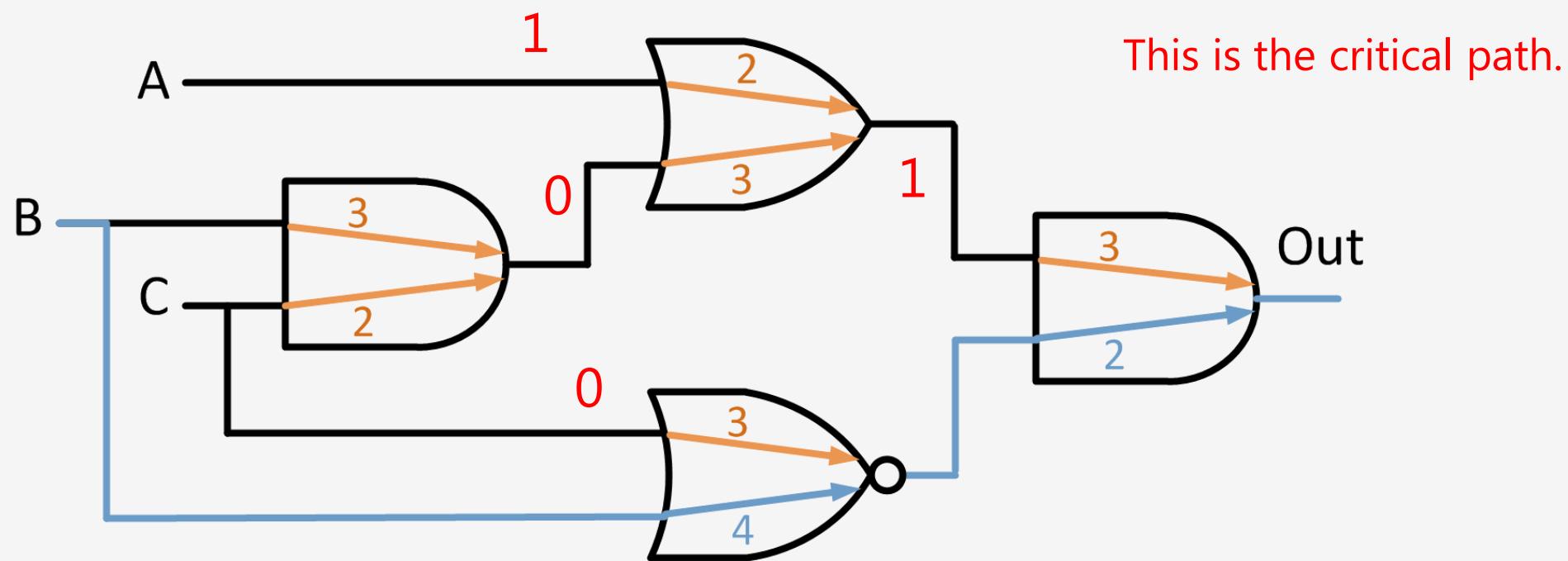
Vertex	AAT	RAT	Slack
A	0	5	5
B	0	1	1
C	0	2	2
D	3	4	1
E	6	7	1
F	4	8	4
Out	9	10	1

Order: A, B, C, D, E, F, Out



Functional Analysis - Sample Question - Solution

Find the critical path of the following circuit. Suppose the clock cycle of $T_c = 10$.



Sequential Timing Analysis

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Flip Flop Input/Output Timing Constraints

Setup Time (t_s): The minimum amount of time BEFORE the clock's active edge by which the input (i.e., D) must be stable.

Hold Time (t_h) : The minimum amount of time AFTER the clock's active edge by which the input (i.e., D) must be stable.

Metastability: If the flip flop input changes when sampled, the flip flop output may have an unstable equilibrium.

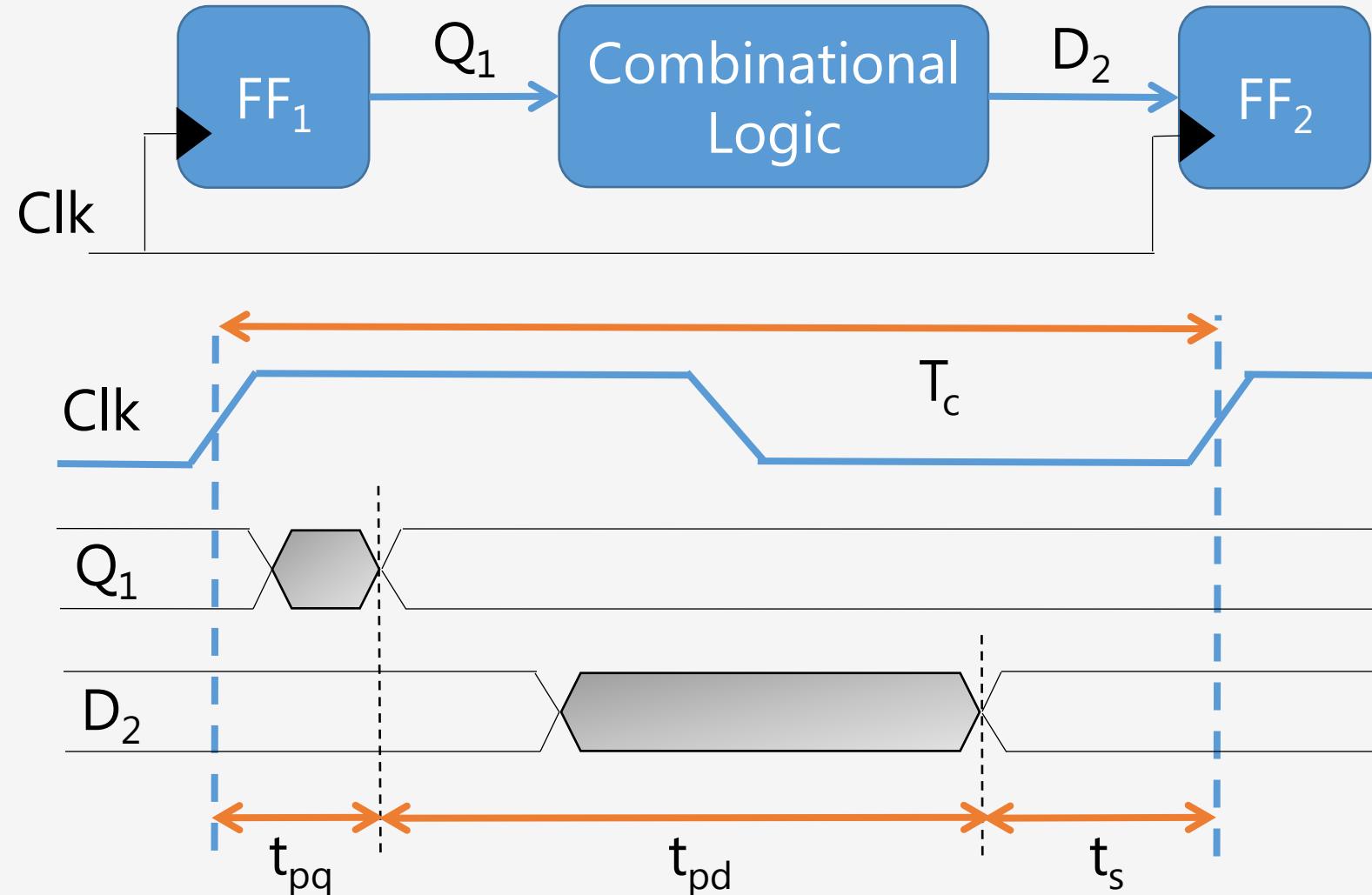
Contamination Delay Clock-to-Q (t_{cq}): The earliest time after the clock active edge that output (i.e., Q) starts to change.

Propagation Delay Clock-to-Q (t_{pq}): The latest time after the clock active edge that output (i.e., Q) stops changing.

Setup Time Constraint

Here we consider the **maximum** delay from FF_1 to FF_2 .

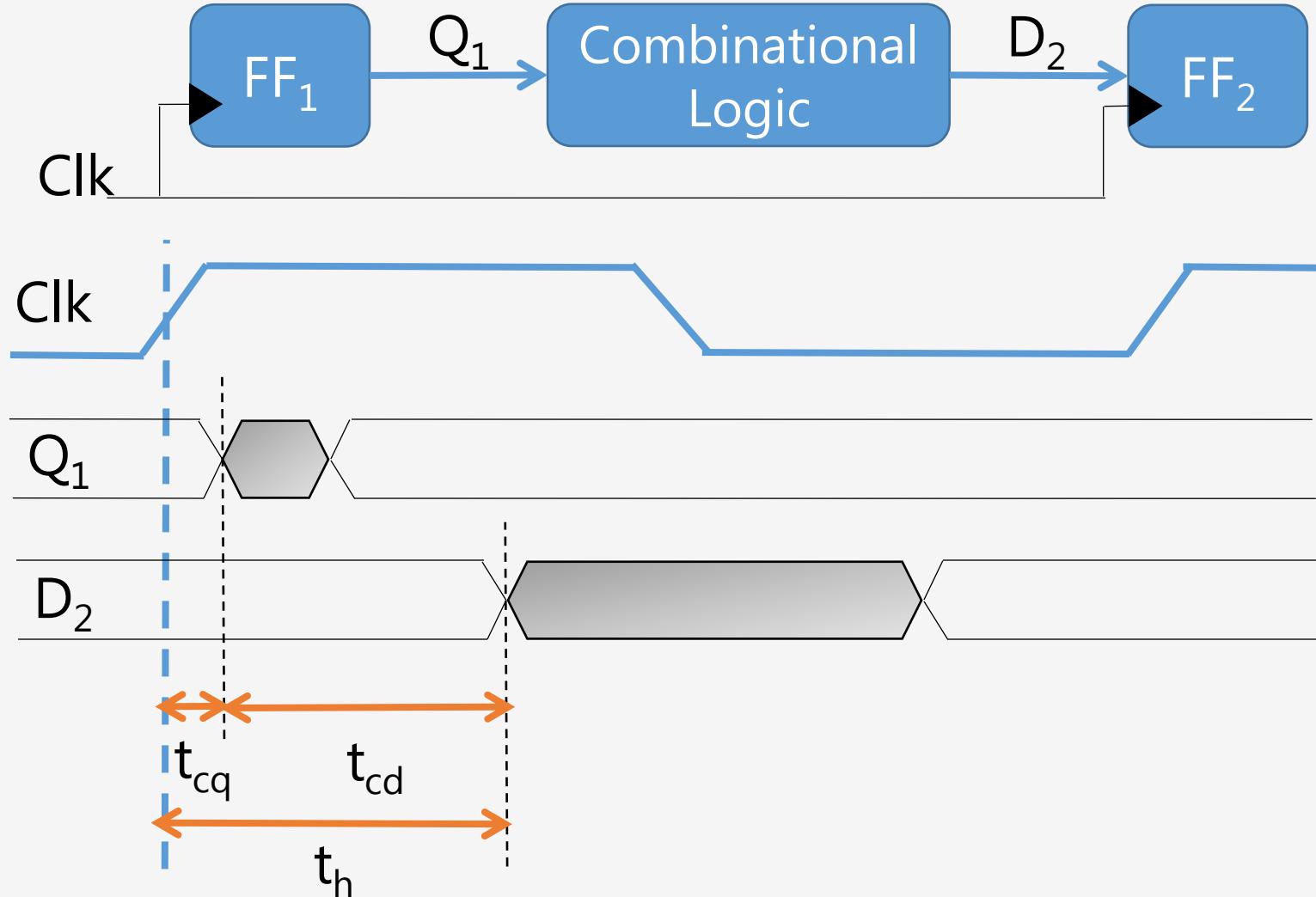
$$t_{pd} < T_c - t_{pq} - t_s$$



Hold Time Constraint

Here we consider the **minimum** delay from FF_1 to FF_2 .

$$t_{cd} > t_h - t_{cq}$$



Time Constraint Discussion

Setup Time Constraint:

$$t_{pd} < T_c - t_{pq} - t_s$$

Hold Time Constraint:

$$t_{cd} > t_h - t_{cq}$$

We may increase clock period to fix setup time after manufacturing.

It is hard to fix hold time after manufacturing and it requires circuit modification.

Design performance is determined by the critical path.

Sequential Delay - Sample Question

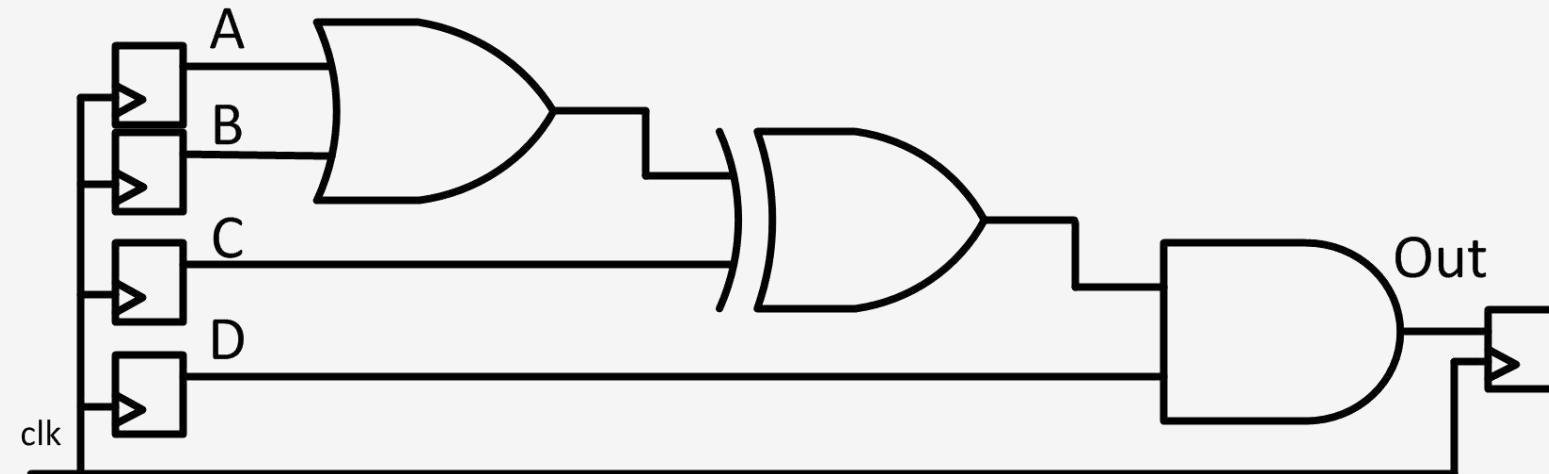
Suppose each gate has a propagation delay of 100ps and contamination delay of 60ps. Find the minimum clock period for the given circuit based on the following timing information. Does it have any timing violation?

$$t_{cq} = 10\text{ps}$$

$$t_{pq} = 40\text{ps}$$

$$t_s = 50\text{ps}$$

$$t_h = 75\text{ps}$$



Sequential Delay - Sample Question - Solution

Suppose each gate has a propagation delay of 100ps and contamination delay of 60ps. Find the minimum clock period for the given circuit based on the following timing information. Does it have any timing violation?

$$t_{cq} = 10\text{ps}$$

$$t_{pd} < T_c - t_{pq} - t_s$$

$$t_{pq} = 40\text{ps}$$

$$t_{pd} = 3 \times 100 = 300\text{ps}$$

$$t_s = 50\text{ps}$$

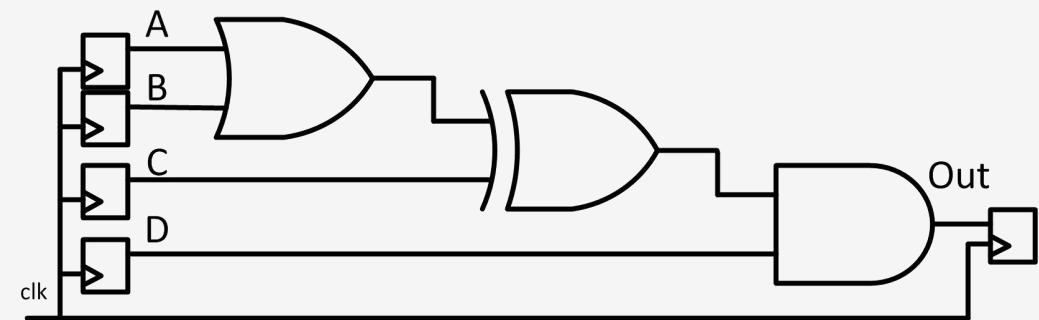
$$T_c > 300 + 40 + 50 = 390$$

$$t_h = 75\text{ps}$$

$$t_{cd} > t_h - t_{cq}$$

$$t_{cd} = 1 \times 60 = 60\text{ps}$$

$$60 >? 75 - 10 = 65 \rightarrow \text{Fail}$$



How can we modify the circuit in order to pass the hold time constraint?

Sequential Delay - Sample Question - Solution

Suppose each gate has a propagation delay of 100ps and contamination delay of 60ps. Find the minimum clock period for the given circuit based on the following timing information. Does it have any timing violation?

$$t_{cq} = 10\text{ps}$$

$$t_{pq} = 40\text{ps}$$

$$t_s = 50\text{ps}$$

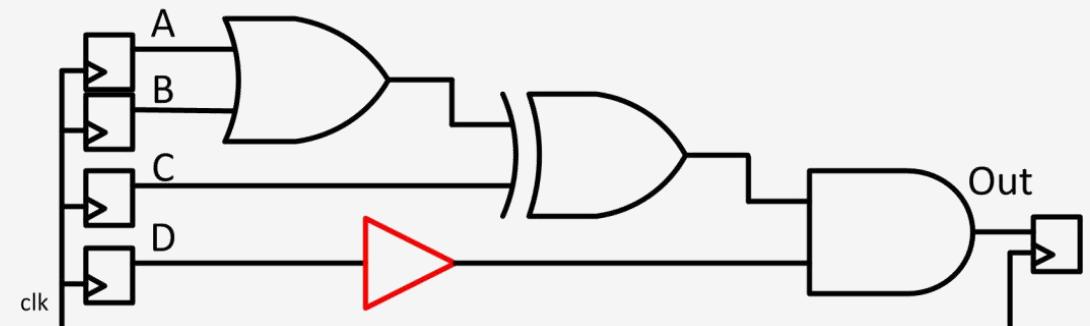
$$t_h = 75\text{ps}$$

We can add a buffer to the shortest path to increase t_{cd} :

$$t_{cd} > t_h - t_{cq}$$

$$t_{cd} = 2 \times 60 = 120\text{ps}$$

$$120 >? 75 - 10 = 65 \rightarrow \text{Pass}$$



In this example, there will be no change on the minimum clock period after adding the buffer.

Clock Skew

The clock does not reach all parts of the chip at the same time.
Clock Skew is the time difference between two clock edges.

Considering the worst-case skew (i.e., t_{sk}), here are the revisited timing constraints:

Setup time constraint: $t_{pd} < T_c - t_{pq} - t_s - t_{sk}$

Hold time constraint: $t_{cd} > t_h - t_{cq} + t_{sk}$

Laboratory Assignment 3

Due Date: March 16th, 11:30PM on BeachBoard Dropbox

Late Submission Policy:

One day delay (Second Due Date: Mar. 17, 11:30PM): 25% deduction

Two days delay (Third Due Date: Mar. 18, 11:30PM): 50% deduction

More than two days delay: No credit

Demo Time:

Mar. 19, 8:00AM - 10:45AM

Mar. 19, 1:00PM - 3:45PM

Academic Integrity: There is zero tolerance for cheating, plagiarism, or any other act of violation of Academic Integrity.

Master-Slave System Bus 1

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Computer System

A computer system is made up of three things:

Central Processing Unit (CPU)

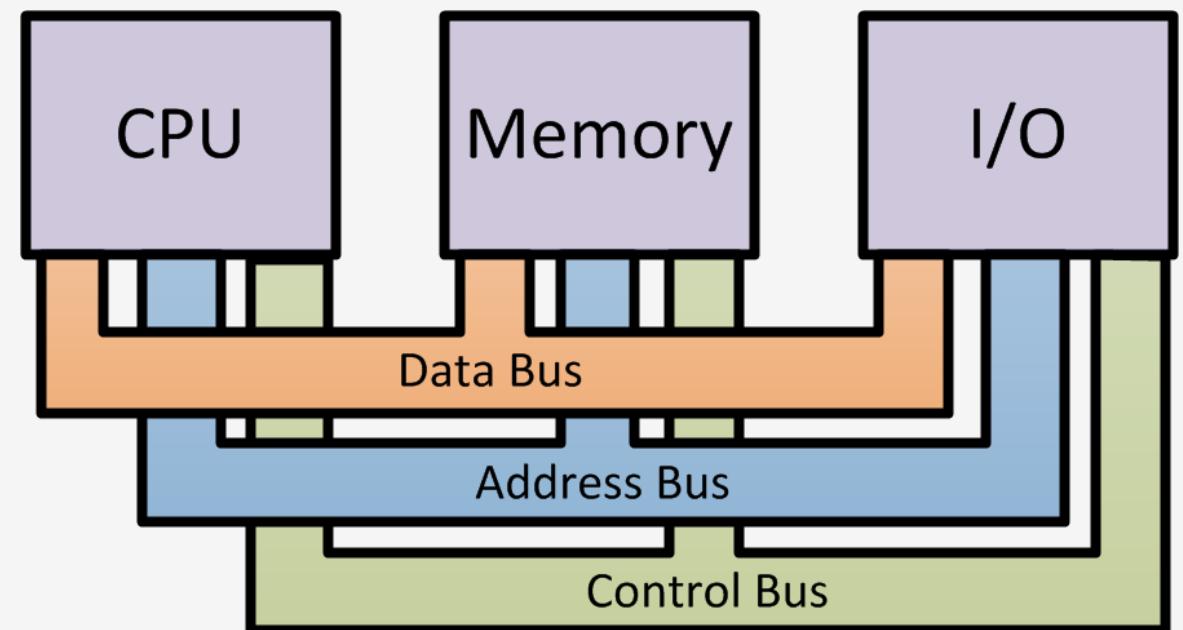
Job: Data Processing

Memory

Job: Storing Data

Input/Output (I/O) Devices

Job: Interacting with Outside



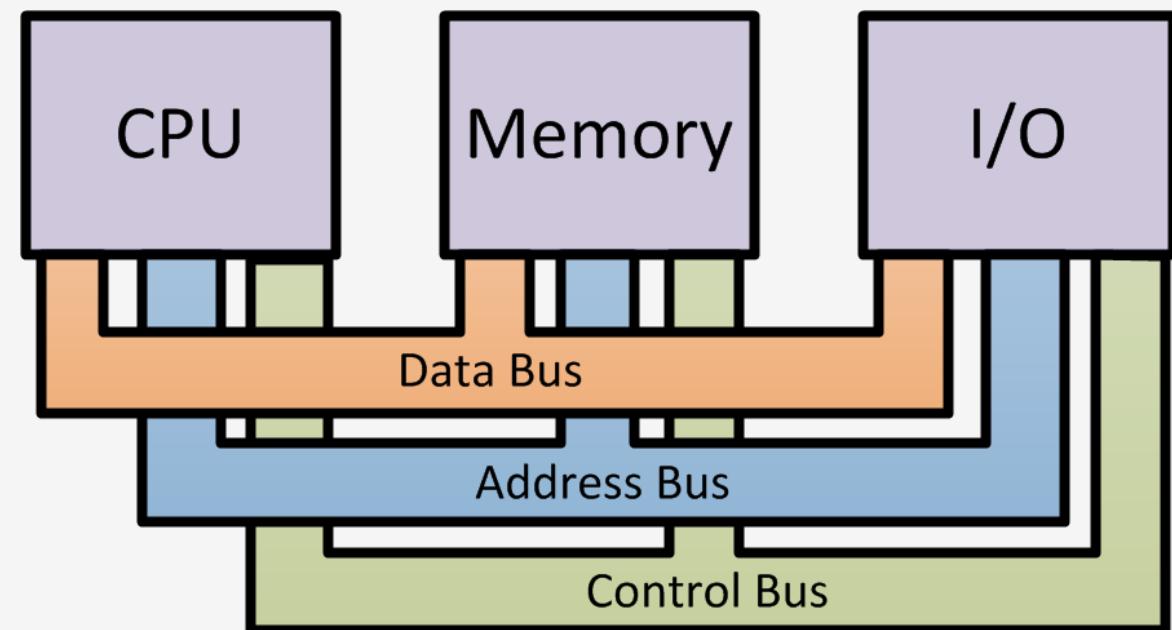
System Bus

The classical way to connect the CPU, memory, and I/O devices is to use a shared set of lines called a system bus.

Data Bus: It contains actual digital pieces of information.

Address Bus: It describes where the data is located and where it needs to go.

Control Bus: It manages the flow of data & address information.



Bus Protocols

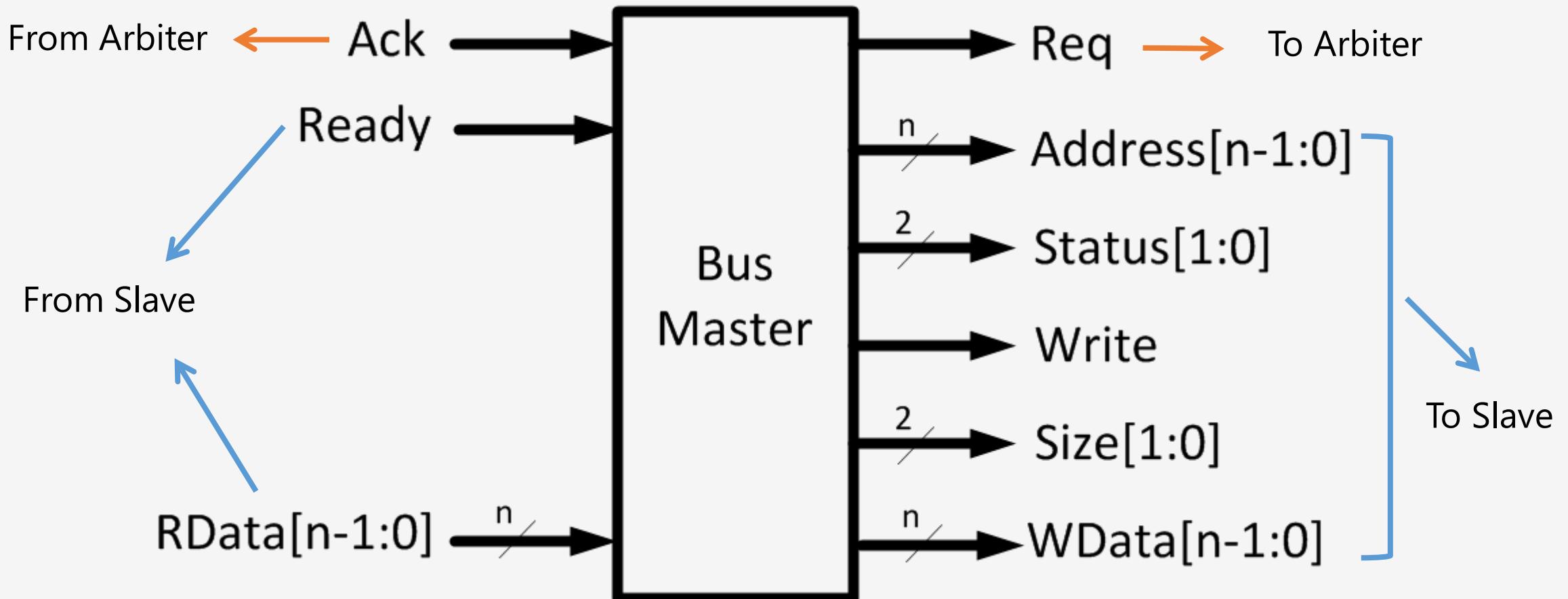
The system bus operates with different bus protocols to exchange data between a bus master and a bus slave.

Bus master initiates the data transfer, and sends or receives data from a slave device or a system memory.

Bus slave, on the other hand, does not have the capability to start data transfer.

Bus protocol ensures to isolate all other system devices from interfering the bus while the bus master exchanges data with a bus slave.

Bus Master Interface



Bus Master Interface

Write	Action
0	Read
1	Write

Bus Master
Write Control

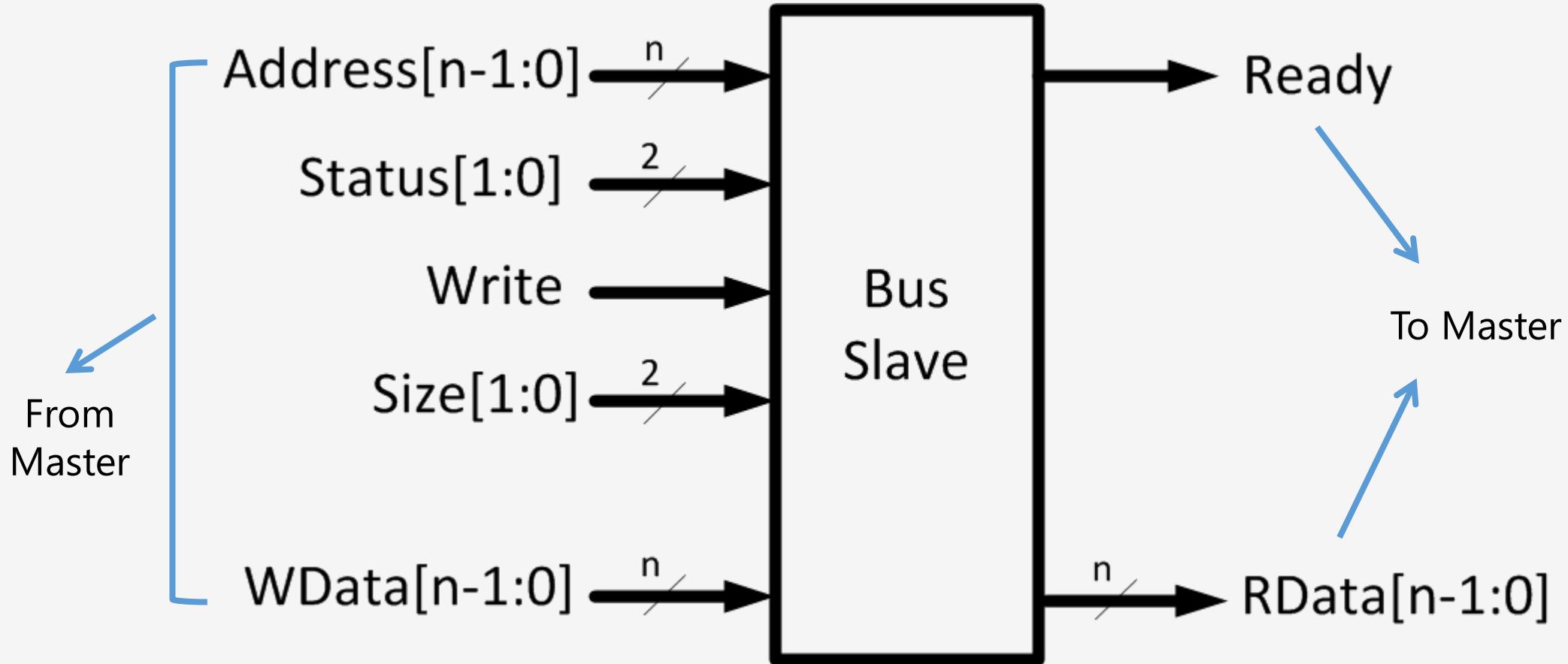
Size	#Bits
0	8
1	16
2	32
3	64

Bus Master
Size Control

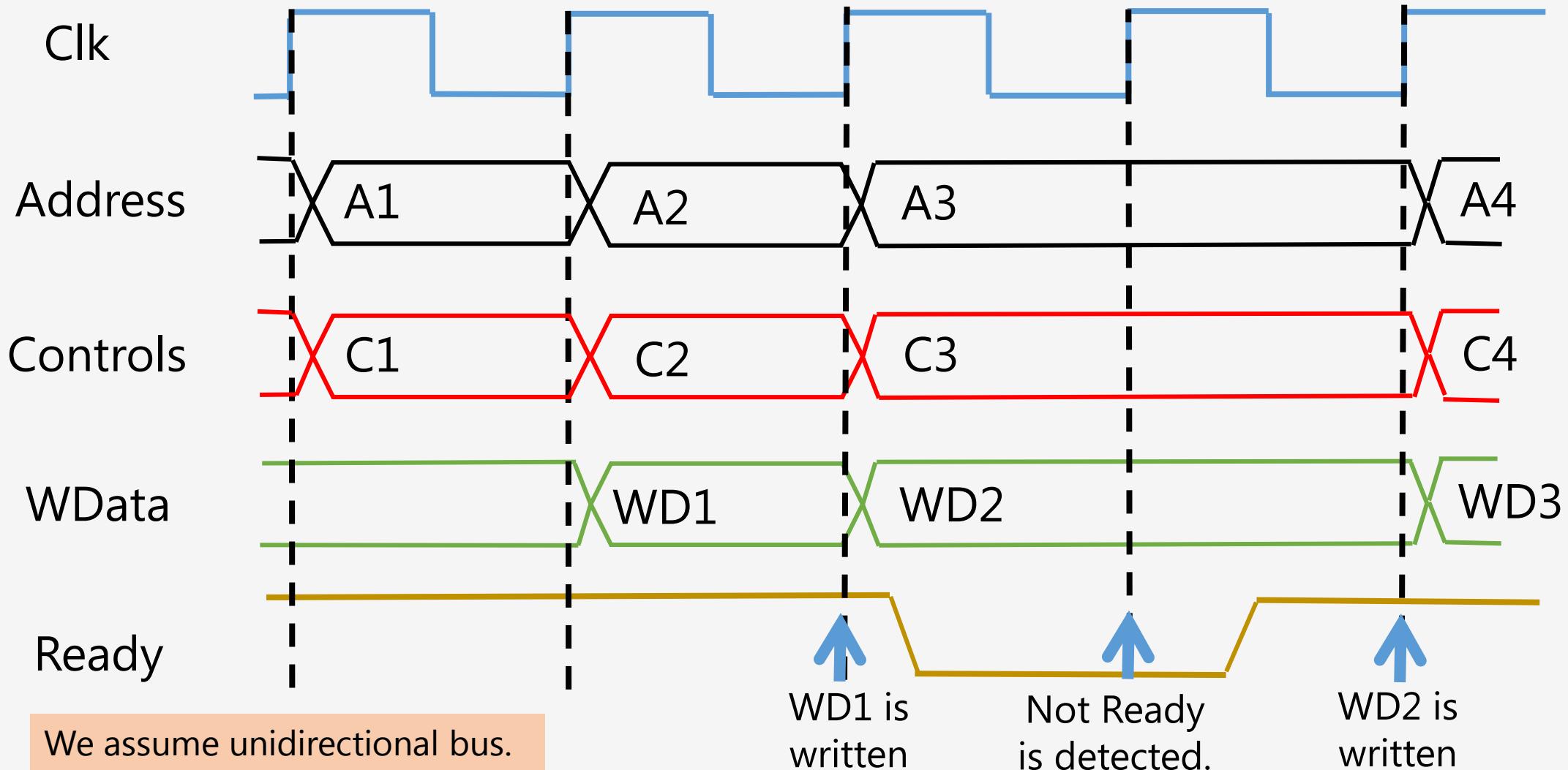
Status	Description
0	Start
1	Continue
2	Idle
3	Busy

Bus Master
Status Control

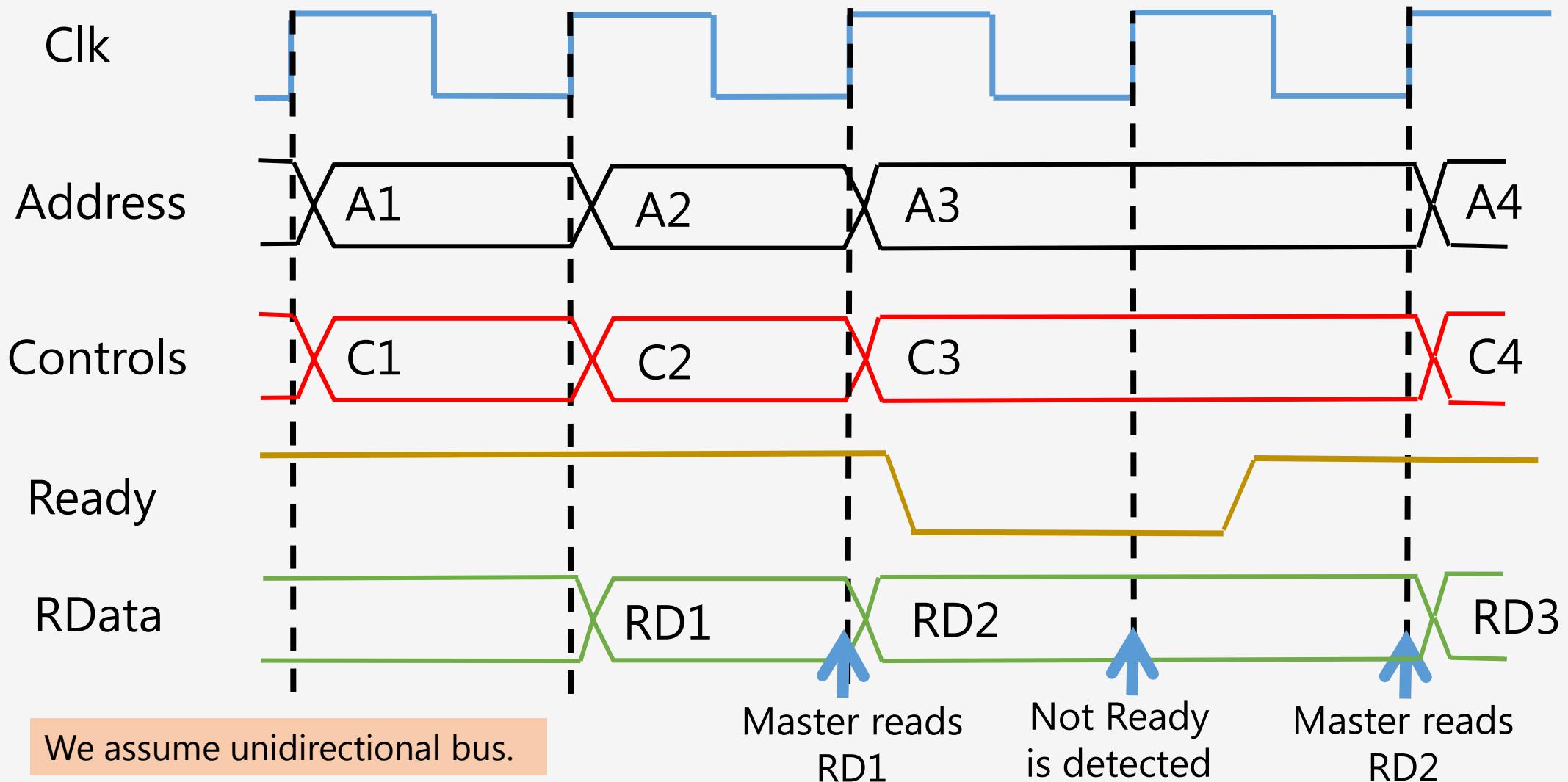
Bus Slave Interface



Write Transfer (Master Writes to Slave) Example



Read Transfer (Master Reads from Slave) Example



Master-Slave System Bus 2

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

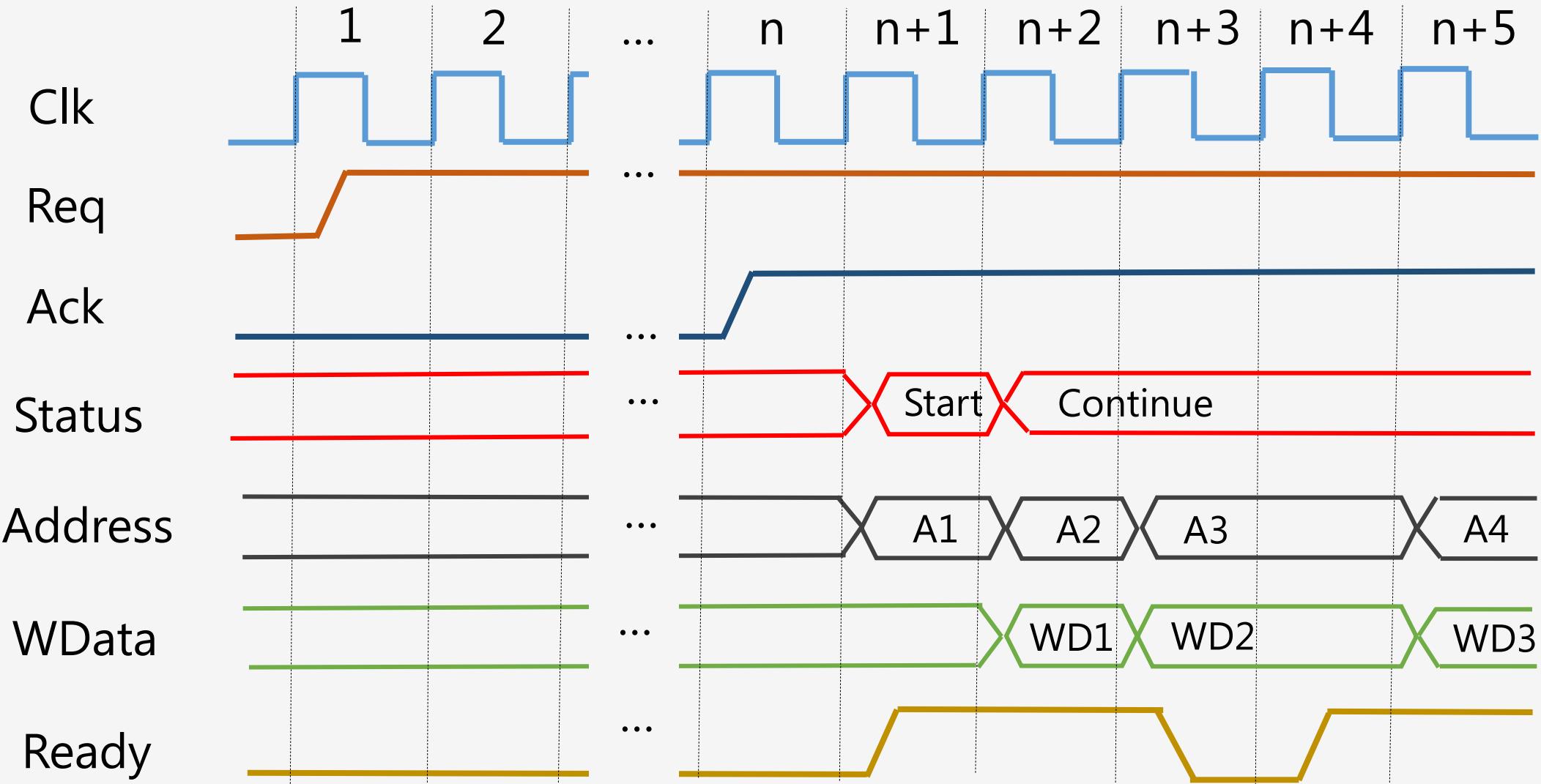
Bus Master Handshake

The bus master needing a data transfer issues a request signal, **Req**, requesting the ownership of the bus from the arbiter.

The arbiter grants this request by an acknowledgement signal, **Ack**.

If there is no ongoing data transfer, the Ack signal is issued in the following cycle after the master generates the Req signal. However, the Ack signal may not be generated many cycles after the Req signal is issued due to an existing data transfer.

Bus Master Handshake Example

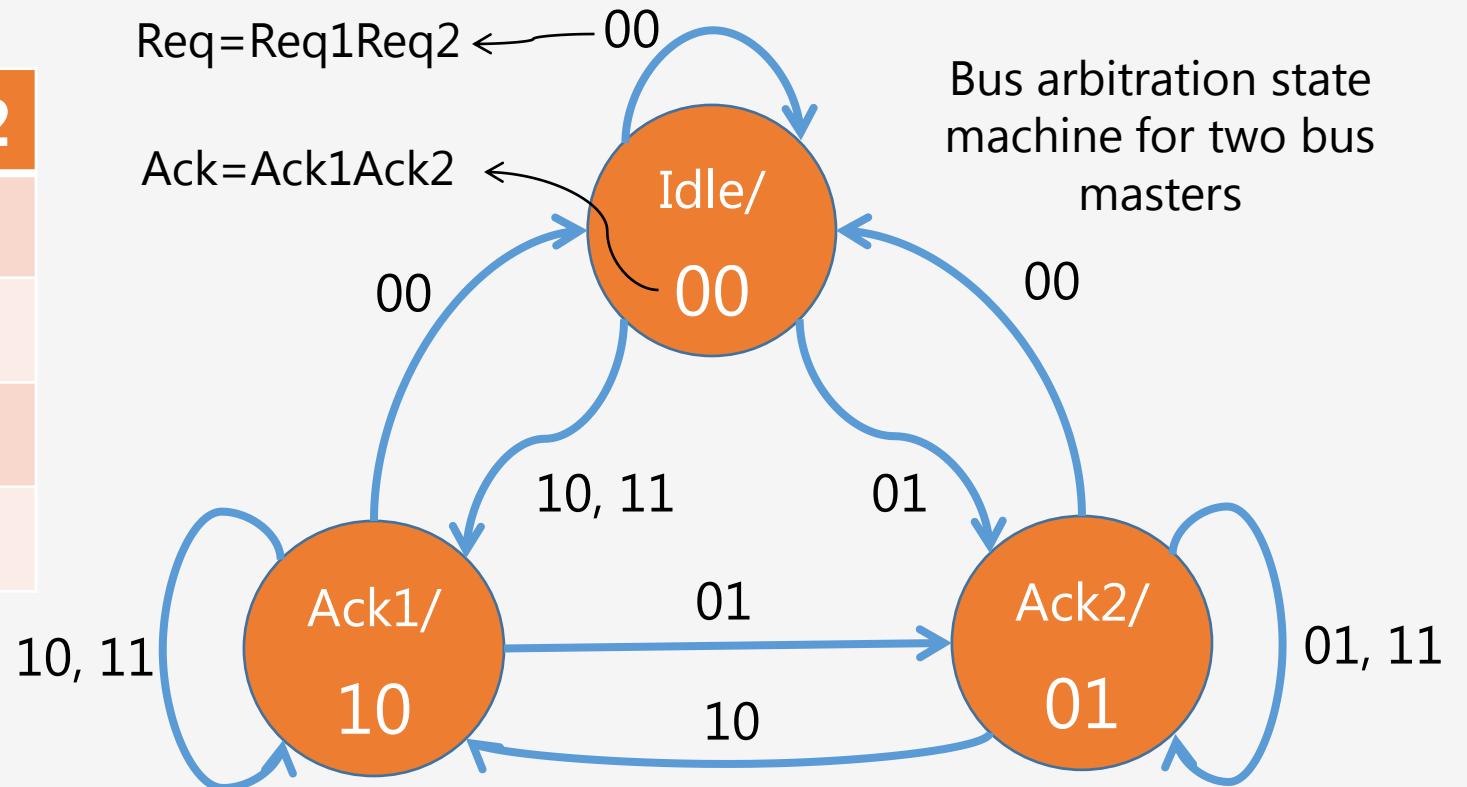


Arbiter

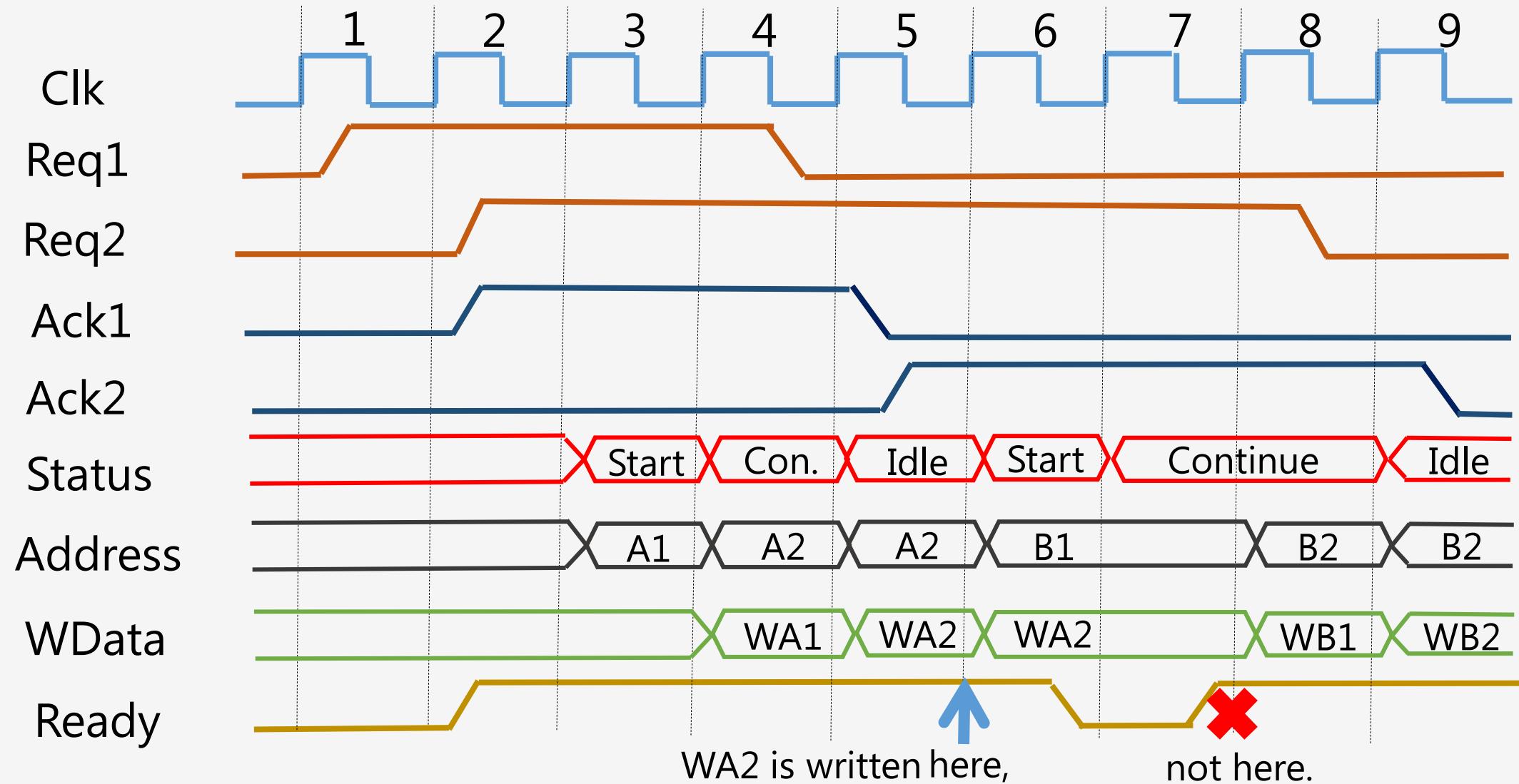
Bus arbitration is an essential part of bus management if there are more than one bus master requesting the bus.

Req1	Req2	Ack1	Ack2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	0

Bus arbitration table for two bus masters



Bus Master Handover Example



Homework Assignment 4

Due Date: March 23, 11:30PM on BeachBoard Dropbox

Late Submission Policy:

One day delay (Second Due Date: Mar. 24, 11:30PM): 25% deduction

Two days delay (Third Due Date: Mar. 25, 11:30PM): 50% deduction

More than two days delay: No credit

Academic Integrity: There is zero tolerance for cheating, plagiarism, or any other act of violation of Academic Integrity.

Interconnection Networks 1

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Interconnection Networks: Basics

An interconnection network is a backbone of a computer system that connects different nodes (i.e., CPUs, memories, and I/O devices) together.

Topology: It specifies the connection pattern of different nodes. Parameters → Cost, Speed, Contention, etc.

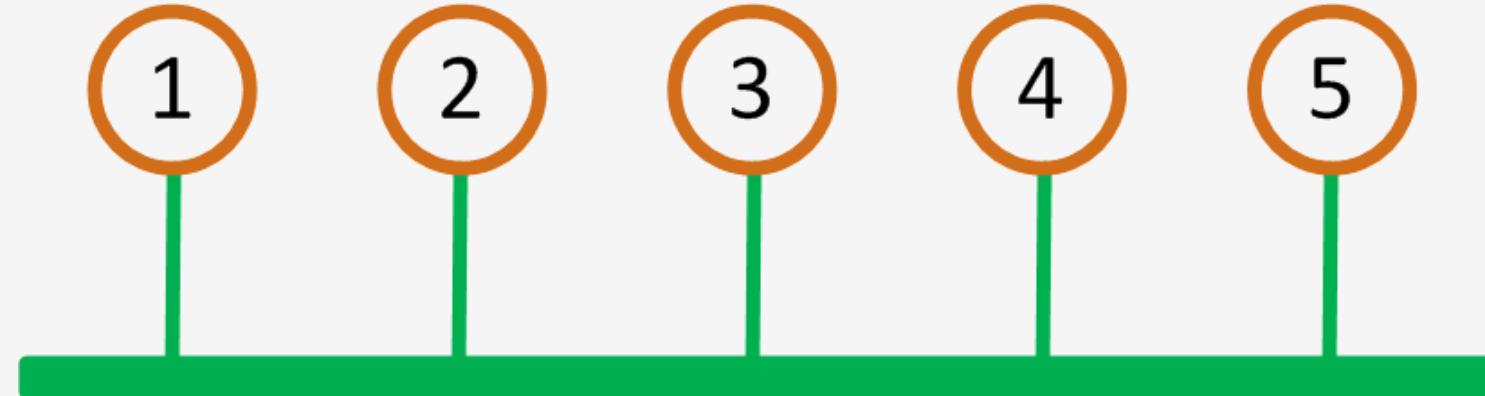
Routing: It determines which path a message takes from source node to destination node.

Topology: Bus

Cost: Lowest cost since sets of wires are shared in multiple ways.

Speed: Low speed since at each time only one node can utilize the shared bus.

Contention: Highest contention since it can easily become saturated.

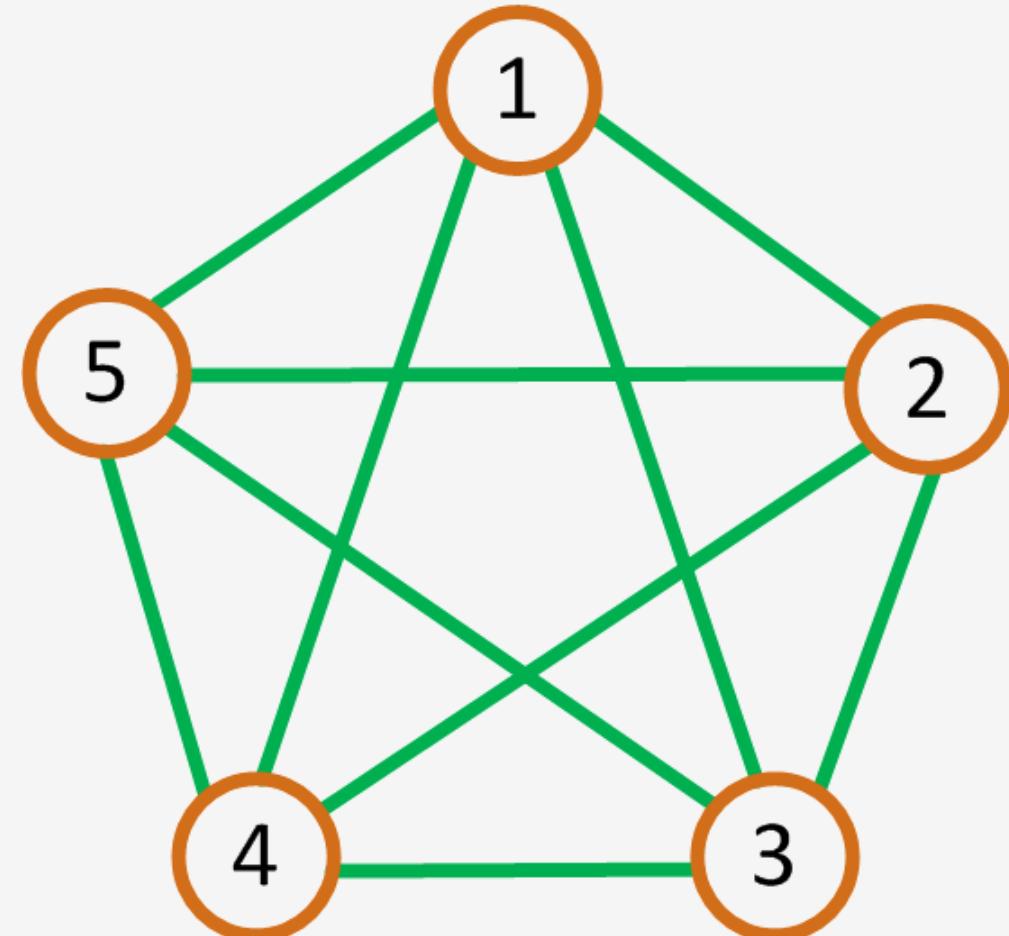


Topology: Point-to-Point (P2P)

Cost: Highest cost since each node requires $n-1$ (n is number of nodes) connections and ports.

Speed: Highest speed since there is a direct link between each two nodes.

Contention: Lowest contention since there is a direct link between each two nodes.

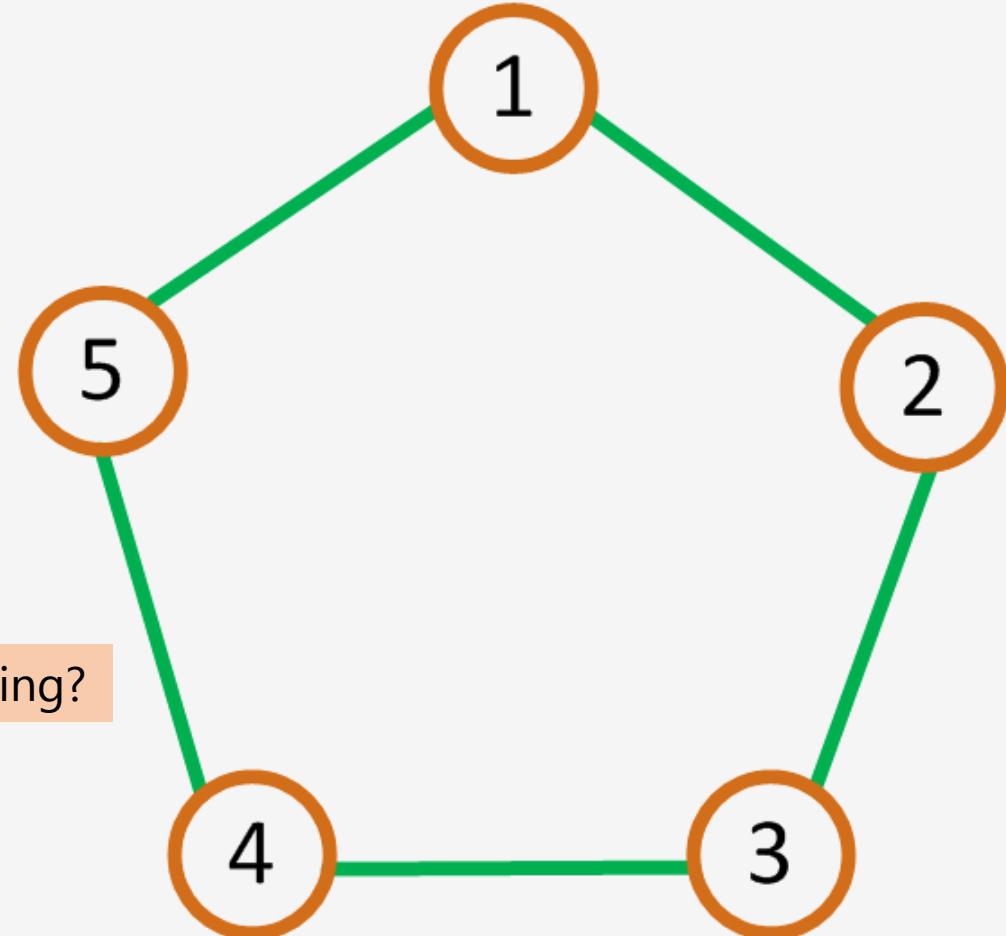


Topology: Ring

Cost: Low cost since each node requires only two ports and connections.

Speed: Low speed since on average a message requires $n/2$ hops to reach to its destination for unidirectional ring. How about bidirectional ring?

Contention: Medium contention.

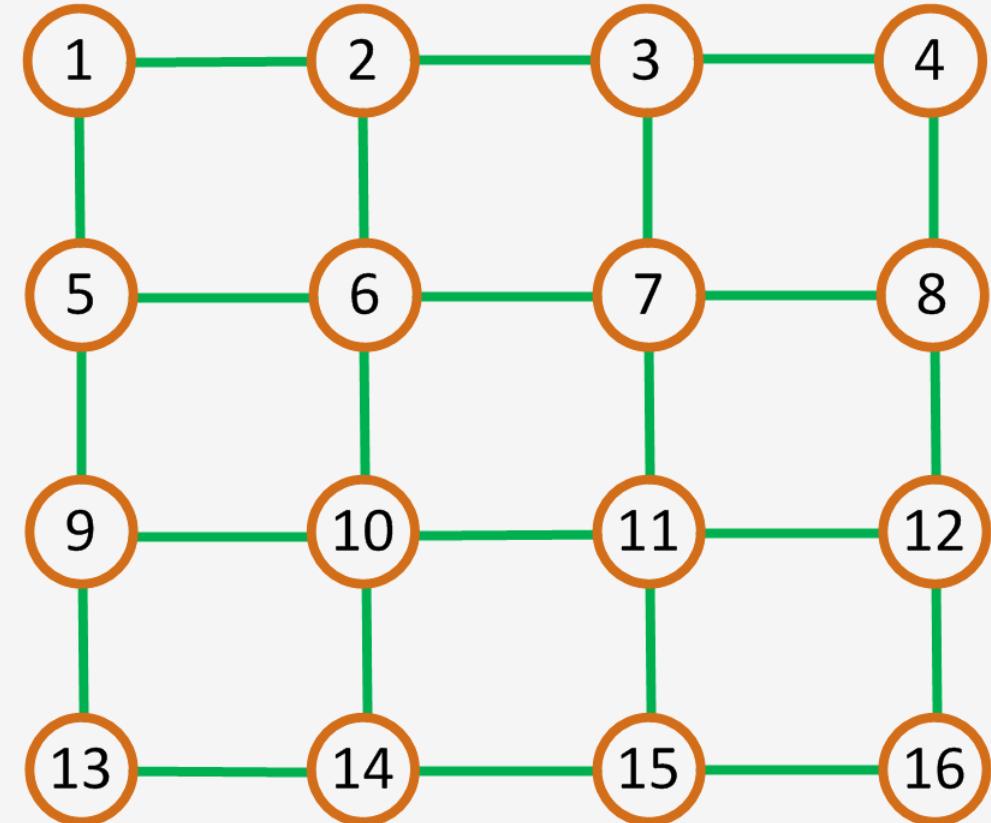


Topology: Mesh

Cost: Low cost for large systems since each node requires at most four ports and connections.

Speed: Medium speed since on average a message requires \sqrt{n} hops to reach to its destination.

Contention: Low contention.

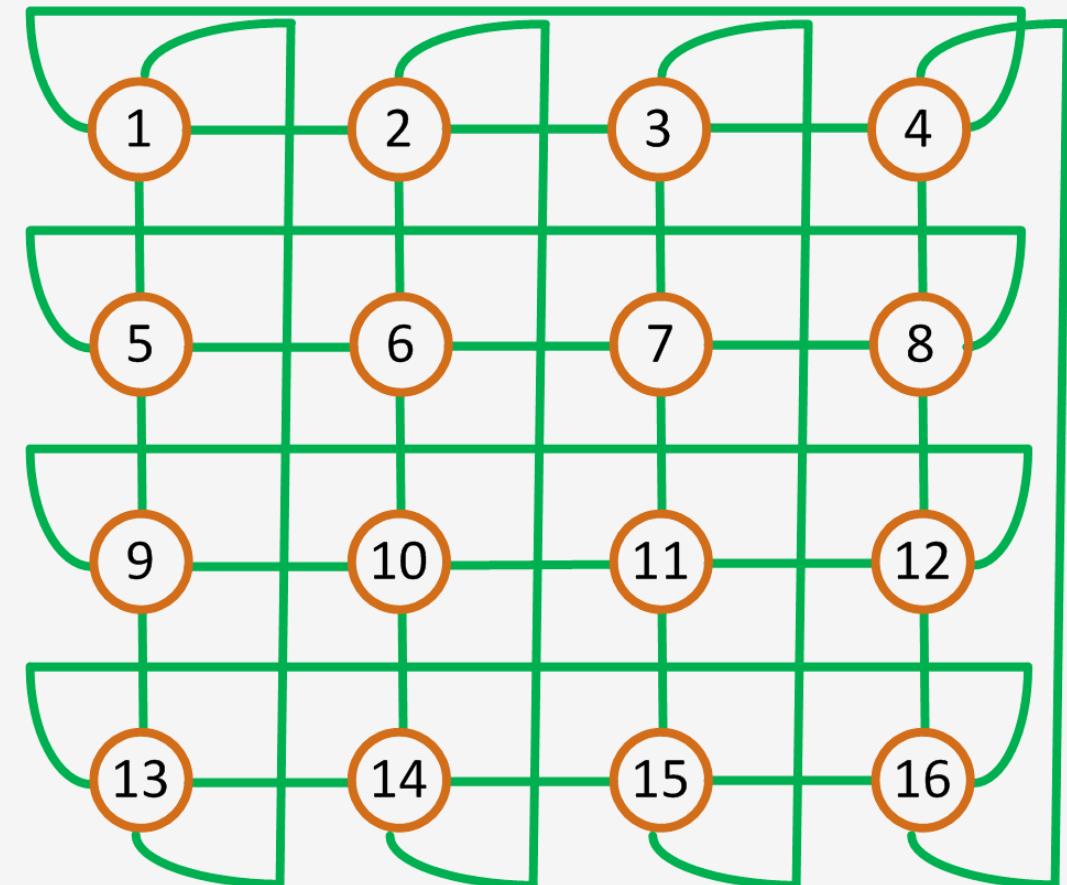


Topology: Torus

Cost: Low-cost for large systems since each node requires four ports and connections.

Speed: Medium to High speed since on average a message requires $\sqrt{n}/2$ hops to reach to its destination.

Contention: Low contention.

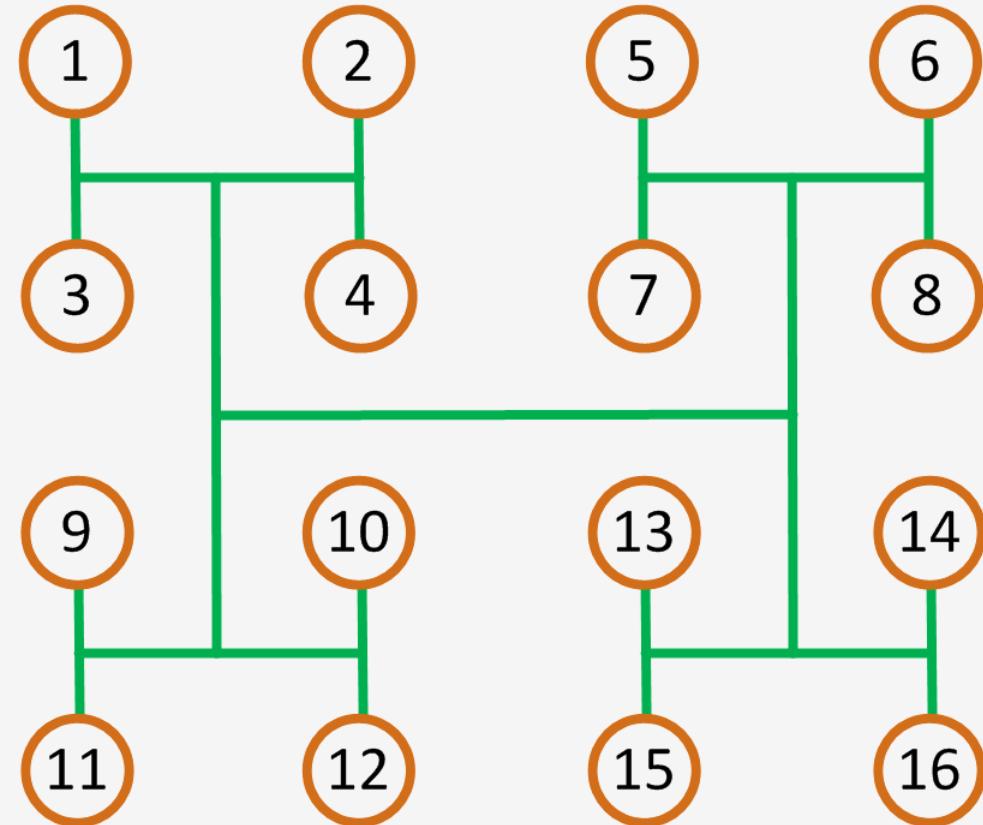


Topology: H-Tree

Cost: Low cost.

Speed: Medium to high speed.

Contention: Low contention.



Topology: Summary

Topology	Cost	Speed	Contention
Bus	Lowest	Low	Highest
P2P	Highest	Highest	Lowest
Ring	Low	Low	Medium
Mesh	Low	Medium	Low
Torus	Low	Medium to High	Low
H-Tree	Low	Medium to High	Low

Interconnection Networks 2

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Routing Algorithm

Routing algorithms establish the path followed by each message.

Deterministic: It always chooses the same path for communication between every pair of nodes.

Oblivious: It may choose different paths without considering network state.

Adaptive : It chooses different paths adapting to the state of the network. → Adaptive routing is beyond the scope of the course.

Deterministic algorithms are also oblivious but the reverse is not necessarily true.

Deterministic Routing

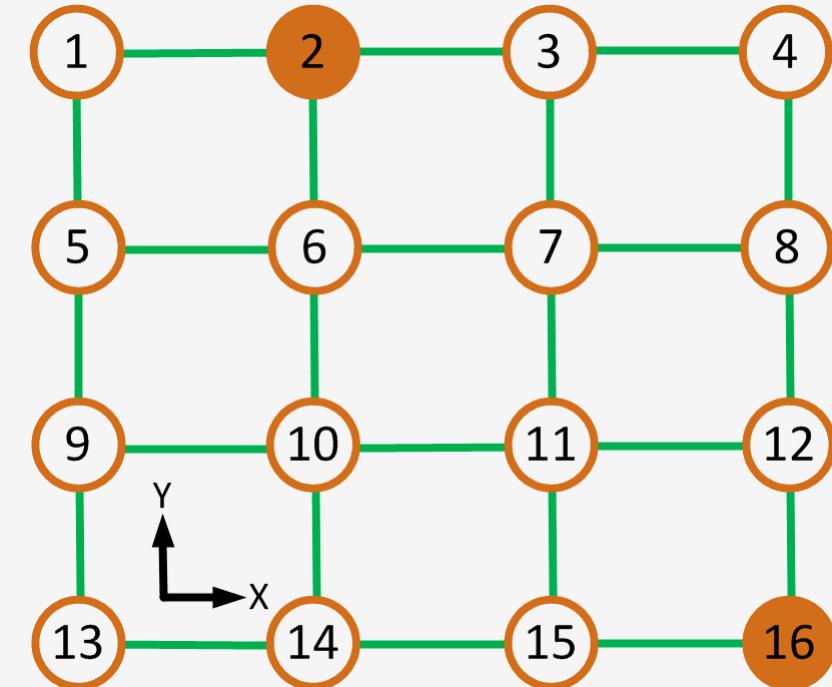
Some topologies can be decomposed into several orthogonal dimensions (i.e., mesh & torus.)

In these topologies, it is easy to compute the distance between current and destination nodes as the sum of the offsets in all the dimensions.

Dimension-Order Routing (DOR) routes messages by crossing dimensions in strictly increasing (or decreasing) order, reducing to zero the offset in one dimension before routing in the next one.

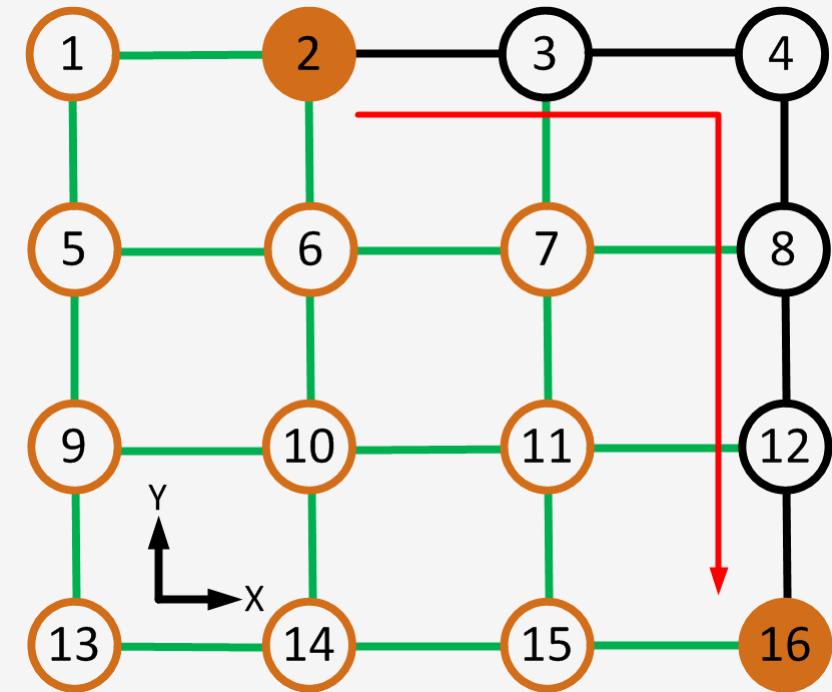
Deterministic Routing - Sample Question 1

In the given mesh topology, the DOR algorithm routes first in X dimension and then in Y dimension. Highlight the path that a message takes from source node 2 to destination node 16.



Deterministic Routing - Sample Question 1 - Solution

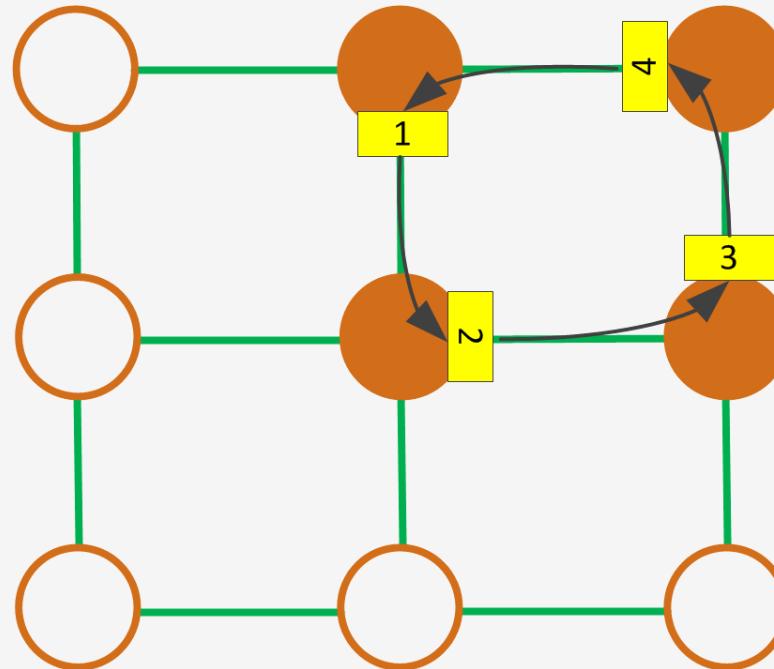
In the given mesh topology, the DOR algorithm routes first in X dimension and then in Y dimension. Highlight the path that a message takes from source node 2 to destination node 16.



Does the algorithm choose the minimal path?

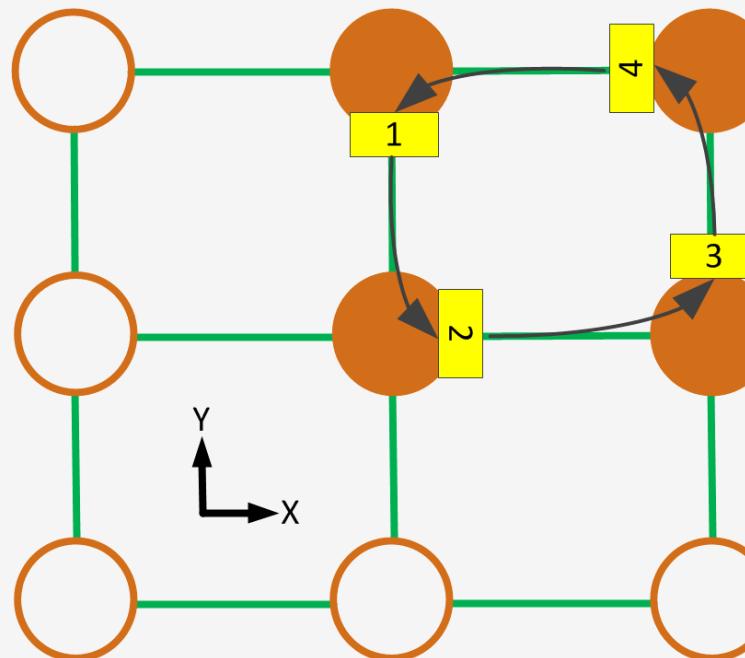
Deadlock

Deadlock is a situation in which no further transportation of messages can take place due to the circular dependencies on resources.



Deterministic Routing - Sample Question 2

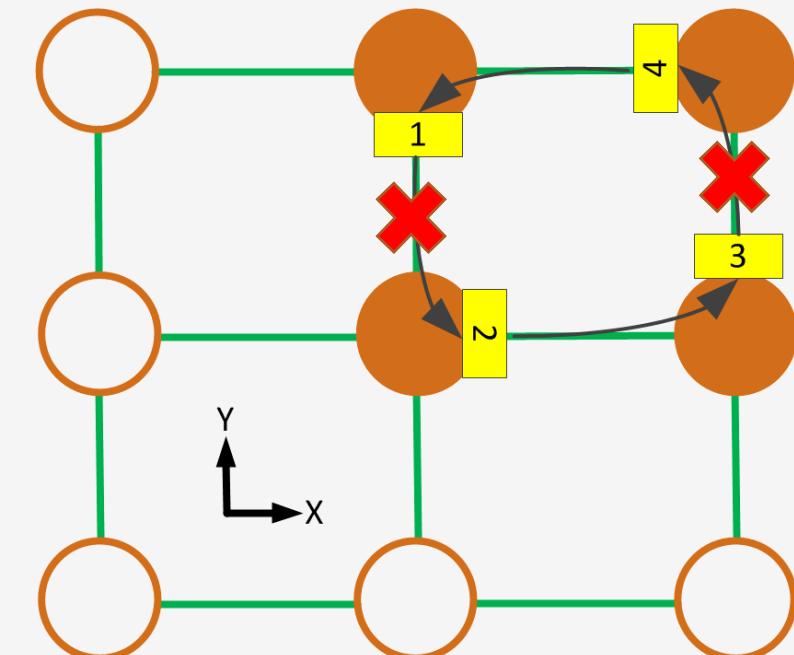
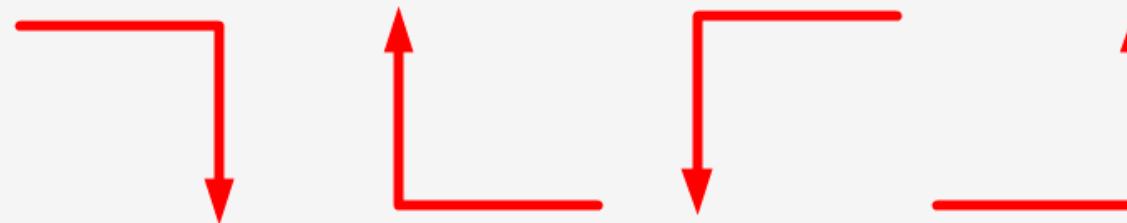
In the given mesh topology, the DOR algorithm routes first in X dimension and then in Y dimension. Can the given deadlock situation happen?



Deterministic Routing - Sample Question 2 - Solution

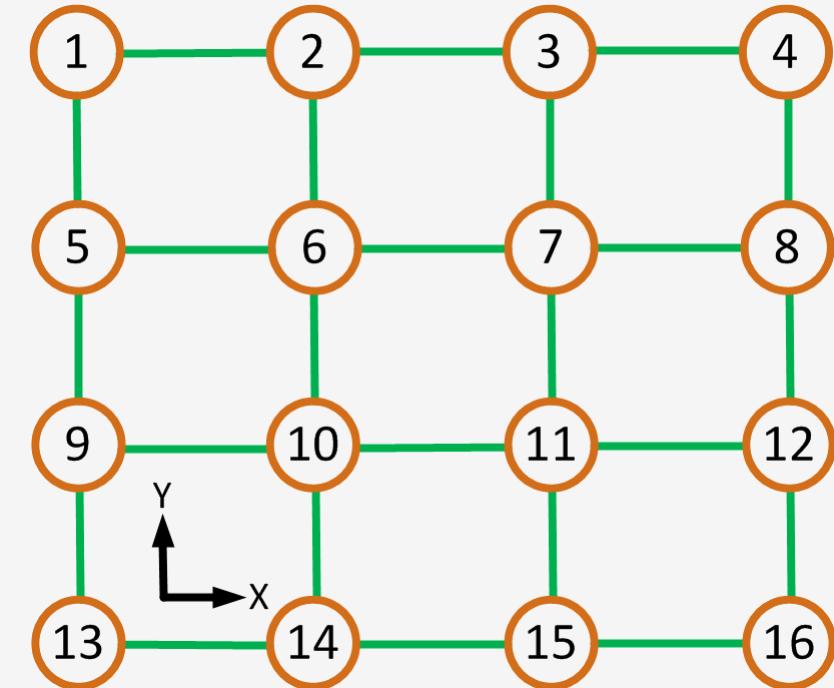
In the given mesh topology, the DOR algorithm routes first in X dimension and then in Y dimension. Can the given deadlock situation happen?

No. DOR is deadlock-free in mesh topology since only the following turns can be taken:



Deterministic Routing - Sample Question 3

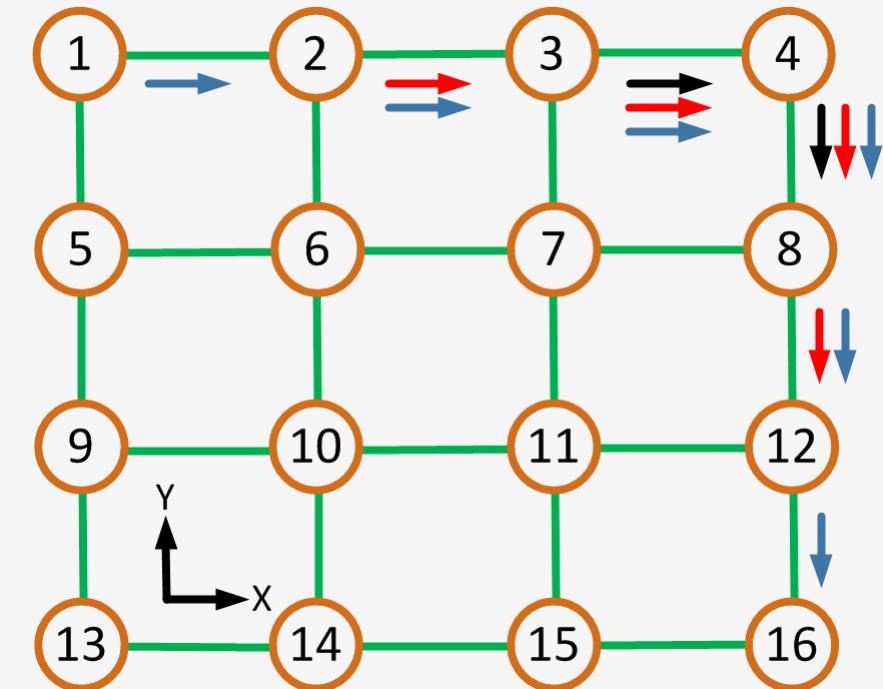
In the given mesh topology, the DOR algorithm routes first in X dimension and then in Y dimension. Suppose nodes 1, 2, and 3 frequently send messages to nodes 16, 12, and 8 respectively. What will be the problem?



Deterministic Routing - Sample Question 3 - Solution

In the given mesh topology, the DOR algorithm routes first in X dimension and then in Y dimension. Suppose nodes 1, 2, and 3 frequently send messages to nodes 16, 12, and 8 respectively. What will be the problem?

There will be edge contention on node 4.



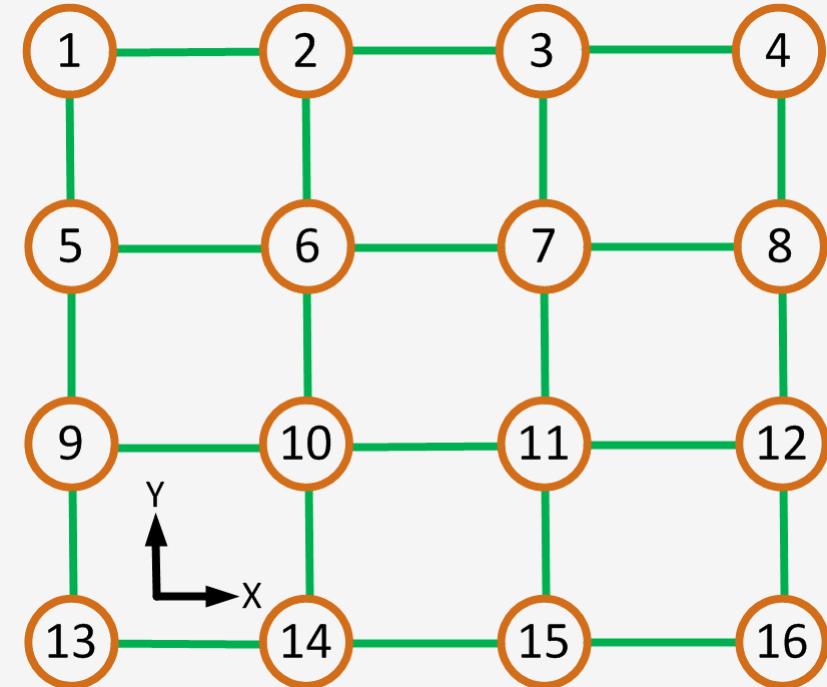
Oblivious Routing

Valiant's Trick: For every message with source-destination pair (s, t) , choose a random intermediate node v and send the message first from s to v , and then from v to t .

If we apply Valiant's trick on DOR algorithm, we will have an oblivious DOR algorithm.

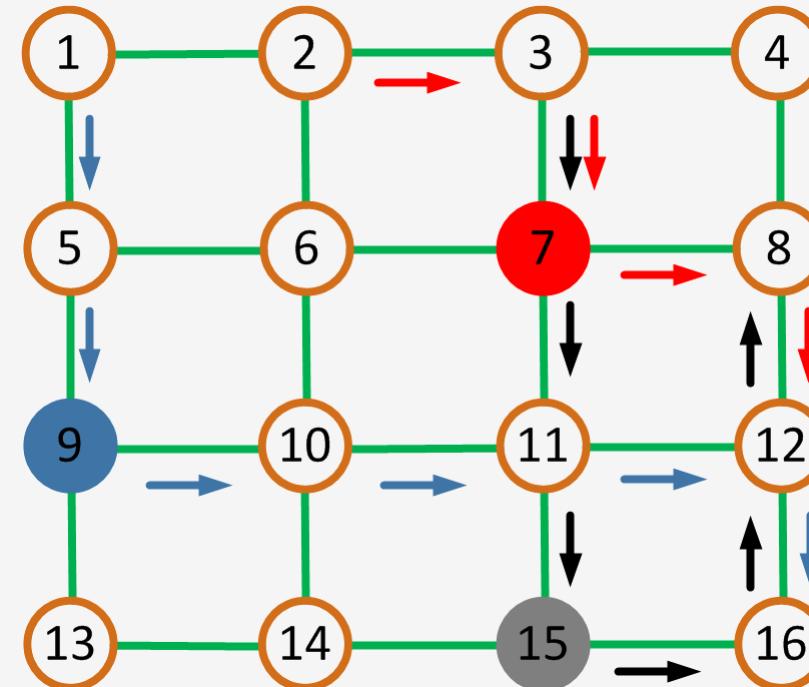
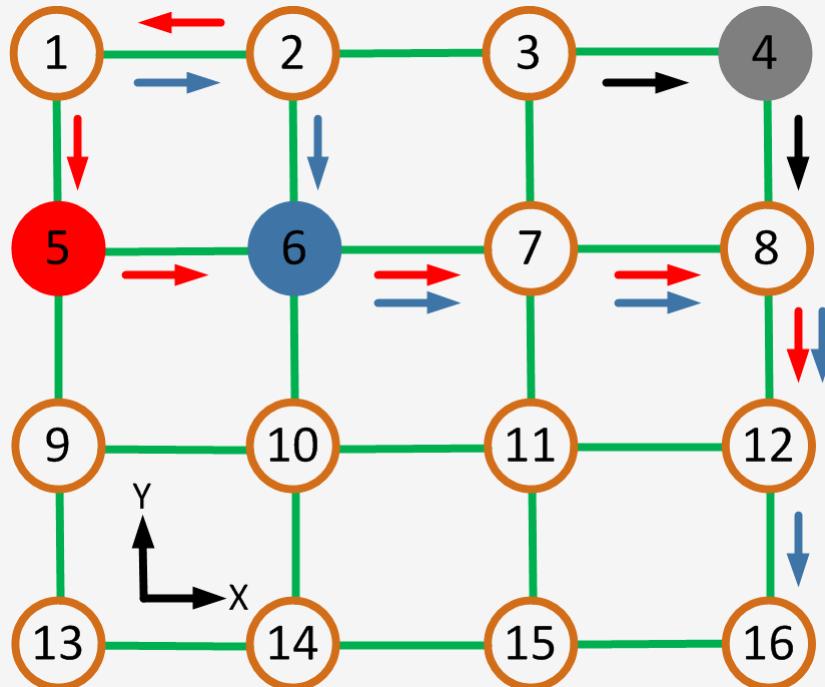
Oblivious Routing - Sample Question

Suppose an oblivious DOR algorithm for the given mesh topology. Suppose nodes 1, 2, and 3 frequently send messages to nodes 16, 12, and 8 respectively. Does the system have contention problem?



Oblivious Routing - Sample Question - Solution

No. Since a new intermediate node will be chosen for each transfer, the load will be distributed.



Does the algorithm choose the minimal path?

Laboratory Assignment 4

Due Date: April 13, 11:30PM on BeachBoard Dropbox

Late Submission Policy:

One day delay (Second Due Date: Apr. 14, 11:30PM): 25% deduction

Two days delay (Third Due Date: Apr. 15, 11:30PM): 50% deduction

More than two days delay: No credit

Demo Time:

Apr. 16, 8:00AM - 10:45AM

Apr. 16, 1:00PM - 3:45PM

Academic Integrity: There is zero tolerance for cheating, plagiarism, or any other act of violation of Academic Integrity.

Midterm 2 Review

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

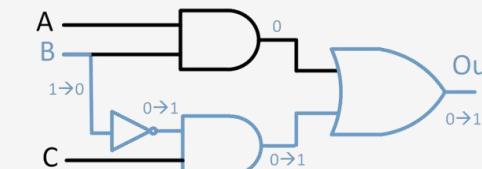
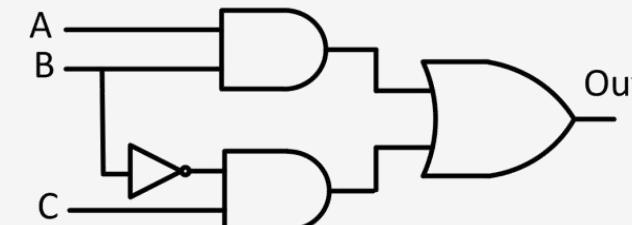
Combinational Timing Analysis

Suppose each gate has propagation delay of 100ps and contamination delay of 60ps. What is the propagation delay and the contamination delay for the simplified combinational circuit implemented based on the given truth table.

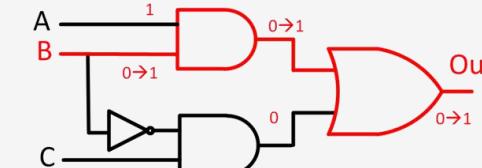
A	B	C	Out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

	BC=00	BC=01	BC=11	BC=10
A=0	0	1	0	0
A=1	0	1	1	1

$\text{Out} = \bar{B}C + AB$



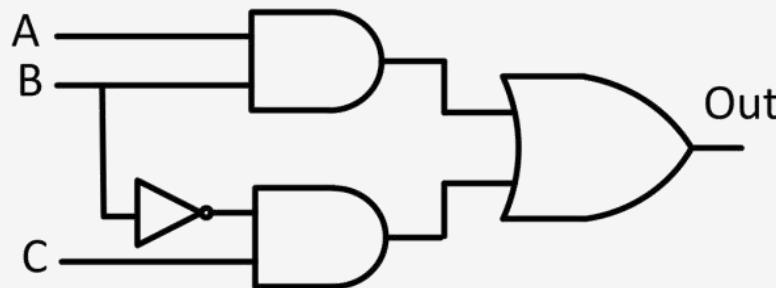
$$t_{pd} = t_{pd_{INV}} + t_{pd_{AND}} + t_{pd_{OR}}$$
$$t_{pd} = 3 \times 100 = 300\text{ps}$$



$$t_{cd} = t_{cd_{AND}} + t_{cd_{OR}}$$
$$t_{cd} = 2 \times 60 = 120\text{ps}$$

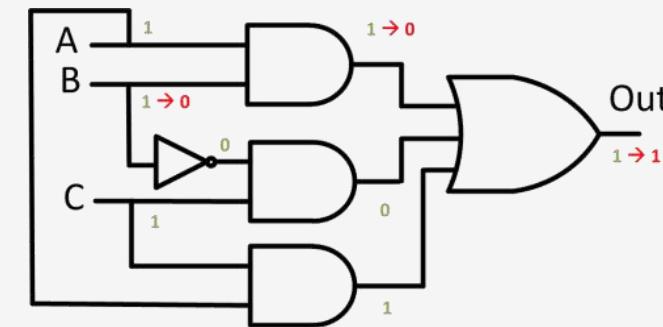
Glitches

For the given circuit, avoid all the glitches by modifying the circuit.



	BC=00	BC=01	BC=11	BC=10
A=0	0	1	0	0
A=1	0	1	1	1

$$\text{Out} = \bar{B}C + AB$$



	BC=00	BC=01	BC=11	BC=10
A=0	0	1	0	0
A=1	0	1	1	1

$$\text{Out} = \bar{B}C + AB + AC$$

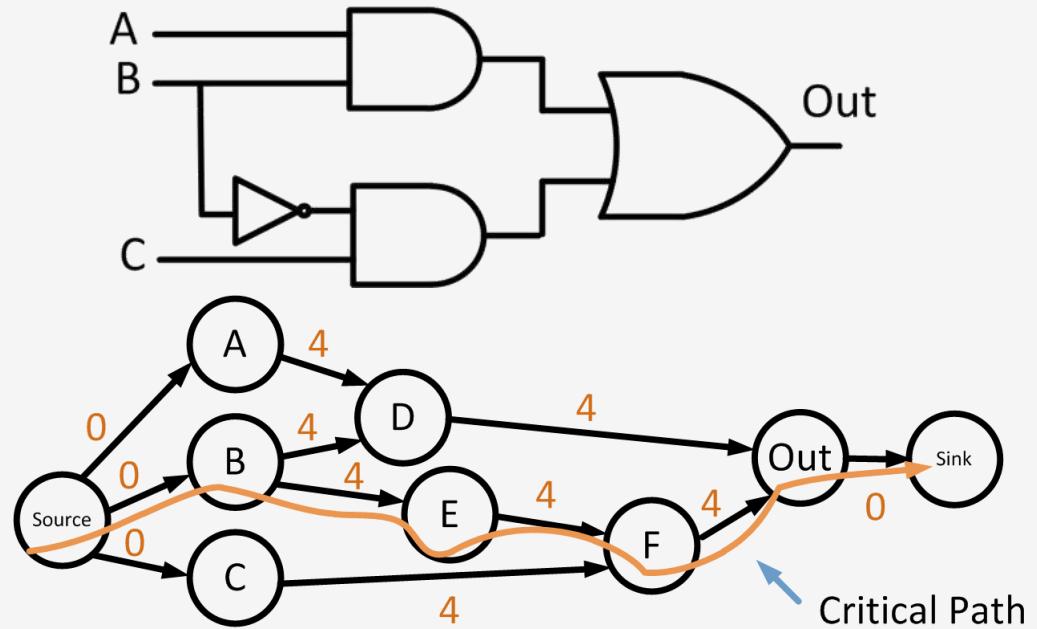
The circuit has glitch in this transition.

In order to avoid the glitch, we can add an extra group to the final formula.

Topological Timing Analysis

Suppose the corresponding delay between two vertices is $D=4$, construct timing graph for the given circuit. Then, suppose the clock cycle is $T_c=10$, compute AAT, RAT, and Slack for each vertex of the timing graph. What is the topological critical path? Does the circuit meet timing constraints?

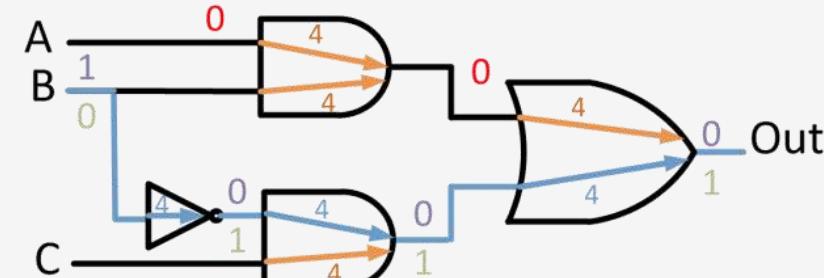
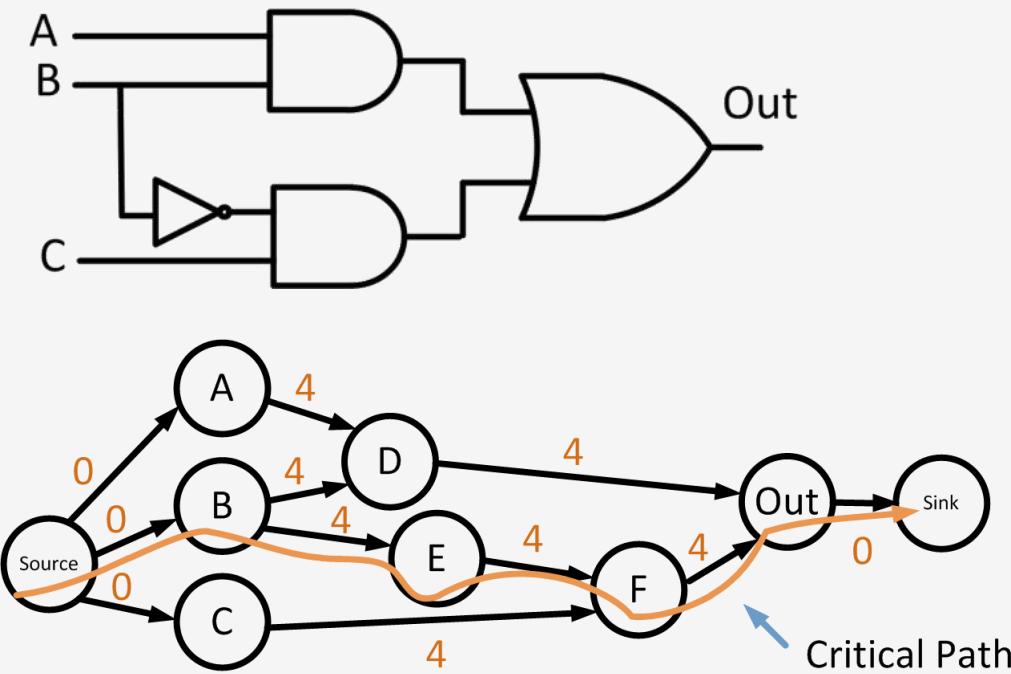
The design does not meet timing constraints.



Vertex	AAT	RAT	Slack
A	0	2	2
B	0	-2	-2
C	0	2	2
D	4	6	2
E	4	2	-2
F	8	6	-2
Out	12	10	-2

Functional Timing Analysis

Does the marked critical path in the given circuit meet the sensitization requirements? If yes, show the input vector for it to be statically sensitizable. If no, find the true critical path.

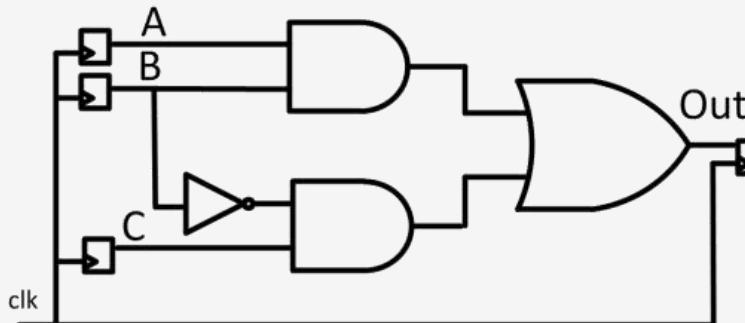


Yes. This is a statically sensitizable critical path by input vector AC=01.

Sequential Timing Analysis

Suppose each gate has propagation delay of 100ps and contamination delay of 60ps. Find the minimum clock period based on the following timing information for the given circuit. Does the design have any timing violation? If yes, modify the circuit to pass the timing constraints.

Contamination delay clock-to-q	$t_{cq}=10\text{ps}$
Propagation delay clock-to-q	$t_{pq}=30\text{ps}$
Setup time	$t_s=40\text{ps}$
Hold time	$t_h=60\text{ps}$
Clock skew	$t_{sk}=5\text{ps}$



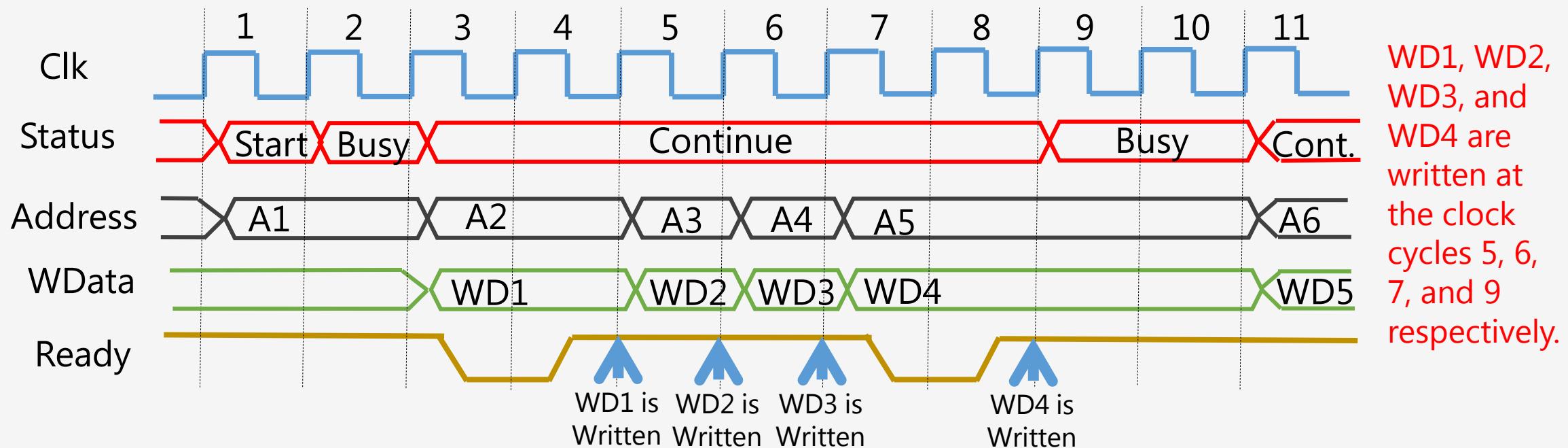
$$t_{pd} < T_c - t_{pq} - t_s - t_{sk}$$
$$T_c > 300 + 30 + 40 + 5 = 375\text{ps}$$

$$t_{cd} > t_h - t_{cq} + t_{sk}$$
$$120 > ? 60 - 10 + 5 = 55 \Rightarrow \text{Pass}$$

The design does not have timing violation.

System Bus - Write Transfer

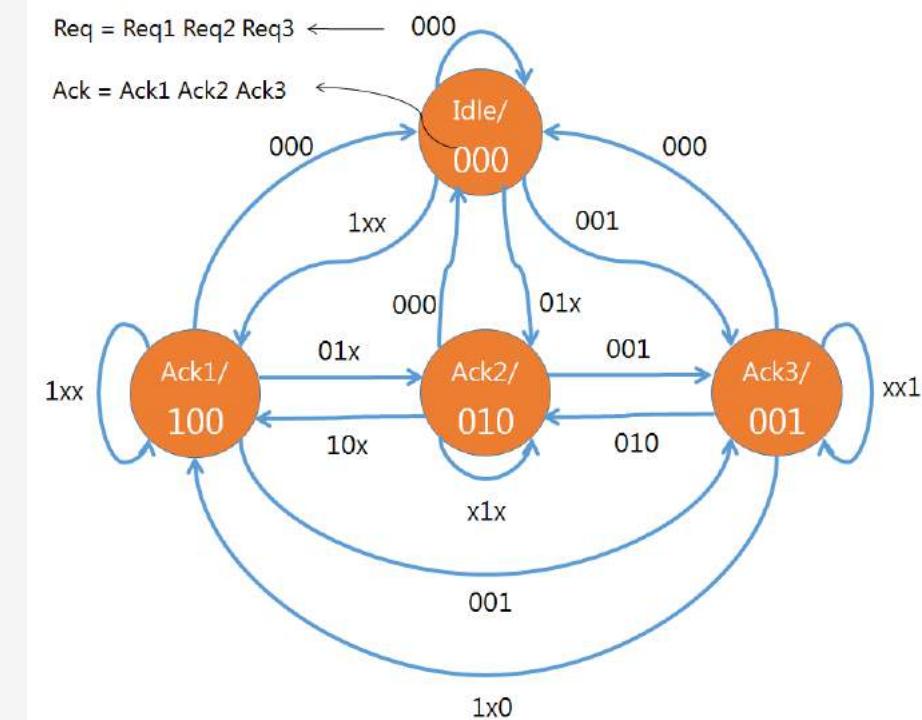
The following figure illustrates an example where the bus master becomes busy in clock cycles 2, 9 and 10 while writing data into a slave. Draw corresponding Address and Write Data (WData) rows. At which clock cycle each WData is written?



System Bus - Arbiter

Draw bus arbitration table and state machine of an arbiter with three bus masters where bus master 1 has the highest priority followed by bus masters 2 and 3.

Req1	Req2	Req3	Ack1	Ack2	Ack3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0



Interconnection Networks - Topology

Complete the given topology comparison table.

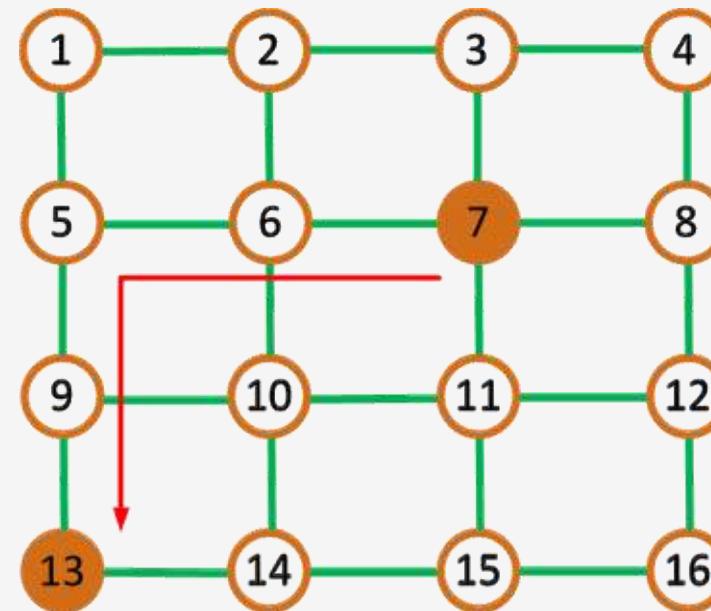
Topology	Cost	Speed	Contention	Reliability	Planarity
Bus	Lowest	Low	Highest	Low	Planar
P2P	Highest	Highest	Lowest	Highest	Not Planar
Ring	Low	Low	Medium	Low	Planar
Mesh	Low	Medium	Low	High	Planar
Torus	Low	Medium to High	Low	High	Not Planar
H-Tree	Low	Medium to High	Low	Medium	Planar
Star	Medium	High	Medium	Low	Planar

In star topology, hub is costly for large systems. Also, hub is the hot-spot and may become saturated. In addition, hub is a single point of failure.

Interconnection Networks - Routing

A suggested DOR algorithm for mesh topology restricts all turns to the west direction. This means west direction should be taken first if needed in the proposed route.

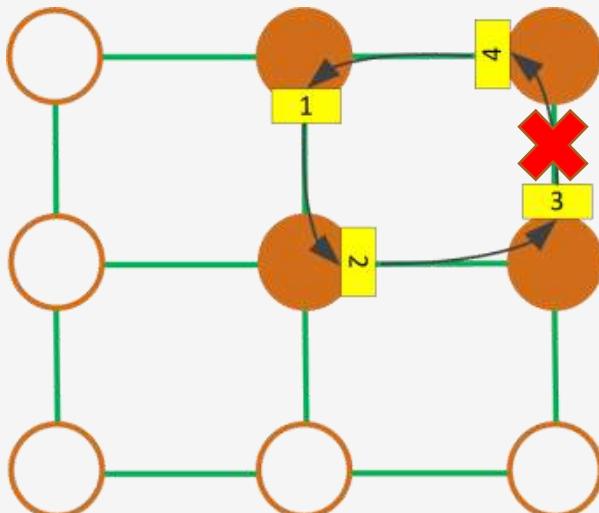
- (a) For the given topology, highlight the path that a message takes from source node 7 to destination node 13.



Interconnection Networks - Routing

A suggested DOR algorithm for mesh topology restricts all turns to the west direction. This means west direction should be taken first if needed in the proposed route.

(b) Can the given deadlock situation happen in this system?



No. Since there should not be any turn to the west, message 3 cannot be waited for message 4 to proceed.

Static Random Access Memory

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Memory

The master device may have an internal memory that stores user data, and the slave device may have a large capacity system memory or a small buffer memory.

Memory	Speed	Volatility
Static Random Access Memory (SRAM)	Fastest	Volatile
Dynamic Random Access Memory (DRAM)	Fast	Volatile
Electrically Erasable Programmable Read Only Memory (E ² PROM)	Slowest	Non-Volatile
Flash Memory	Slow	Non-Volatile

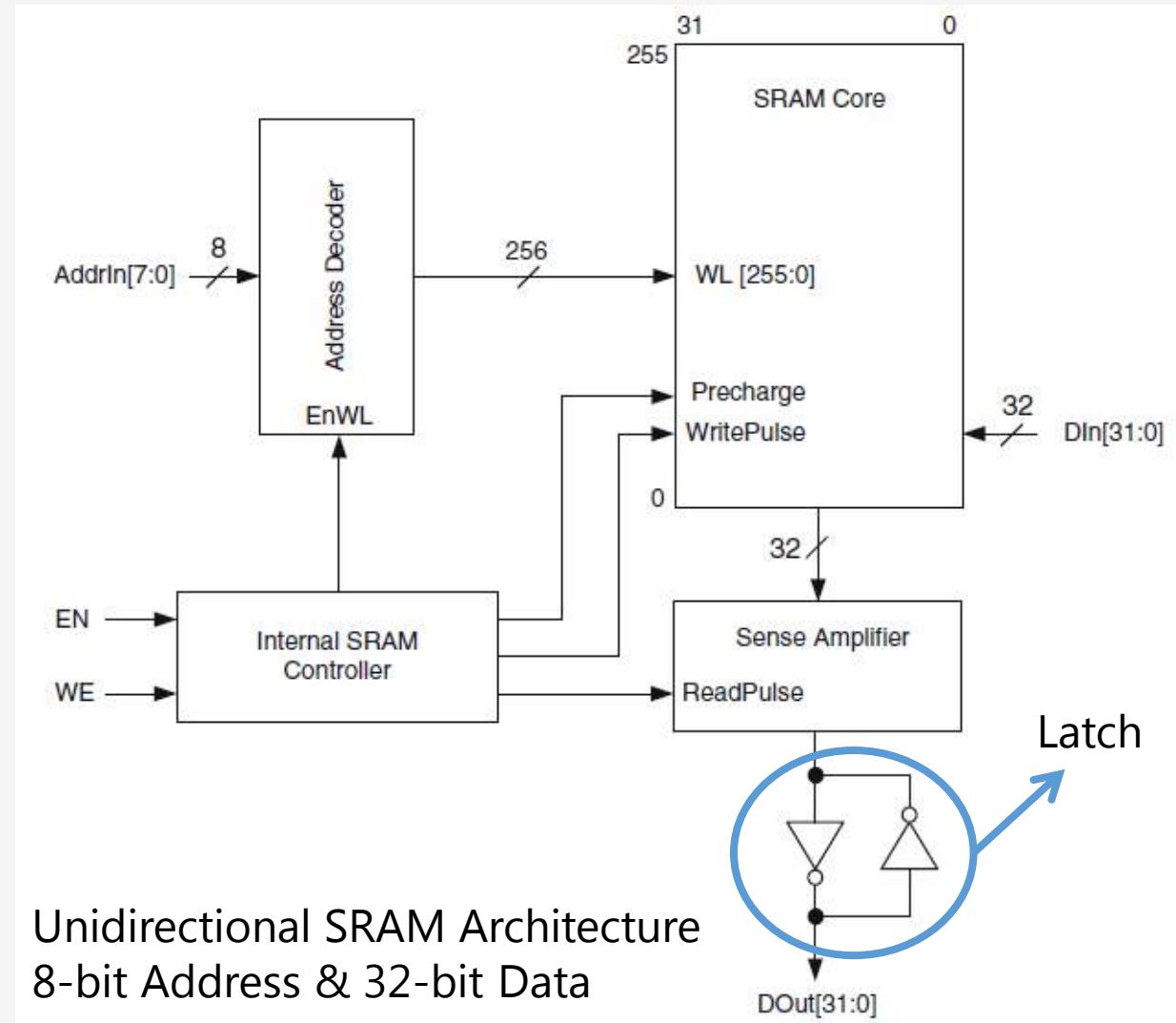
SRAM Architecture

SRAM Core: It retains all immediate data.

Address Decoder: It generates all 2^n Word Lines (WL) from an n-bit wide address.

Sense Amplifier: It amplifies the cell voltage to full logic levels during read.

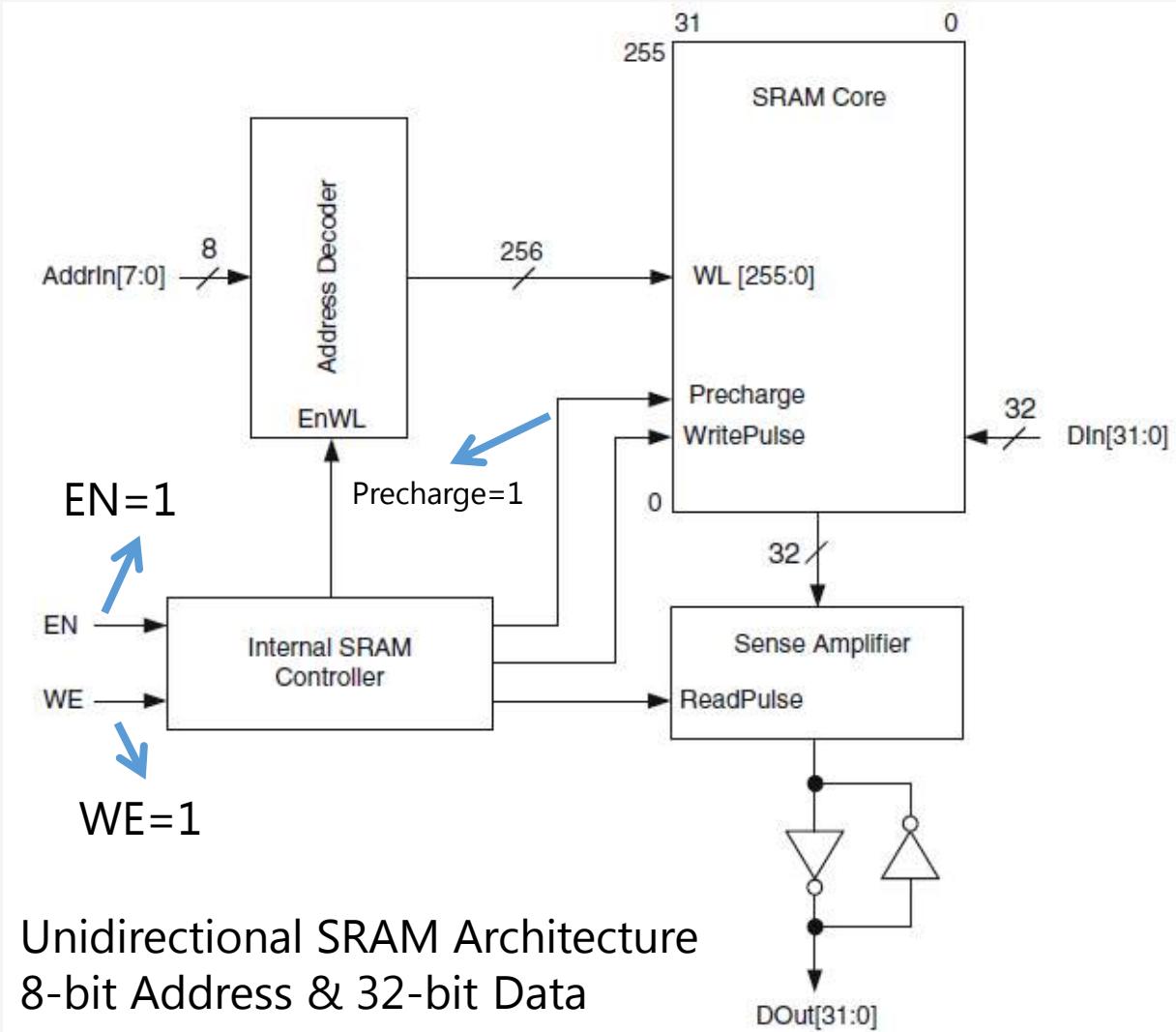
Internal SRAM Controller: It generates self-timed pulses required during a read or write cycle.



SRAM - Data Write Sequence

The data write sequence starts with Enable (EN) = 1 and Write Enable (WE) = 1.

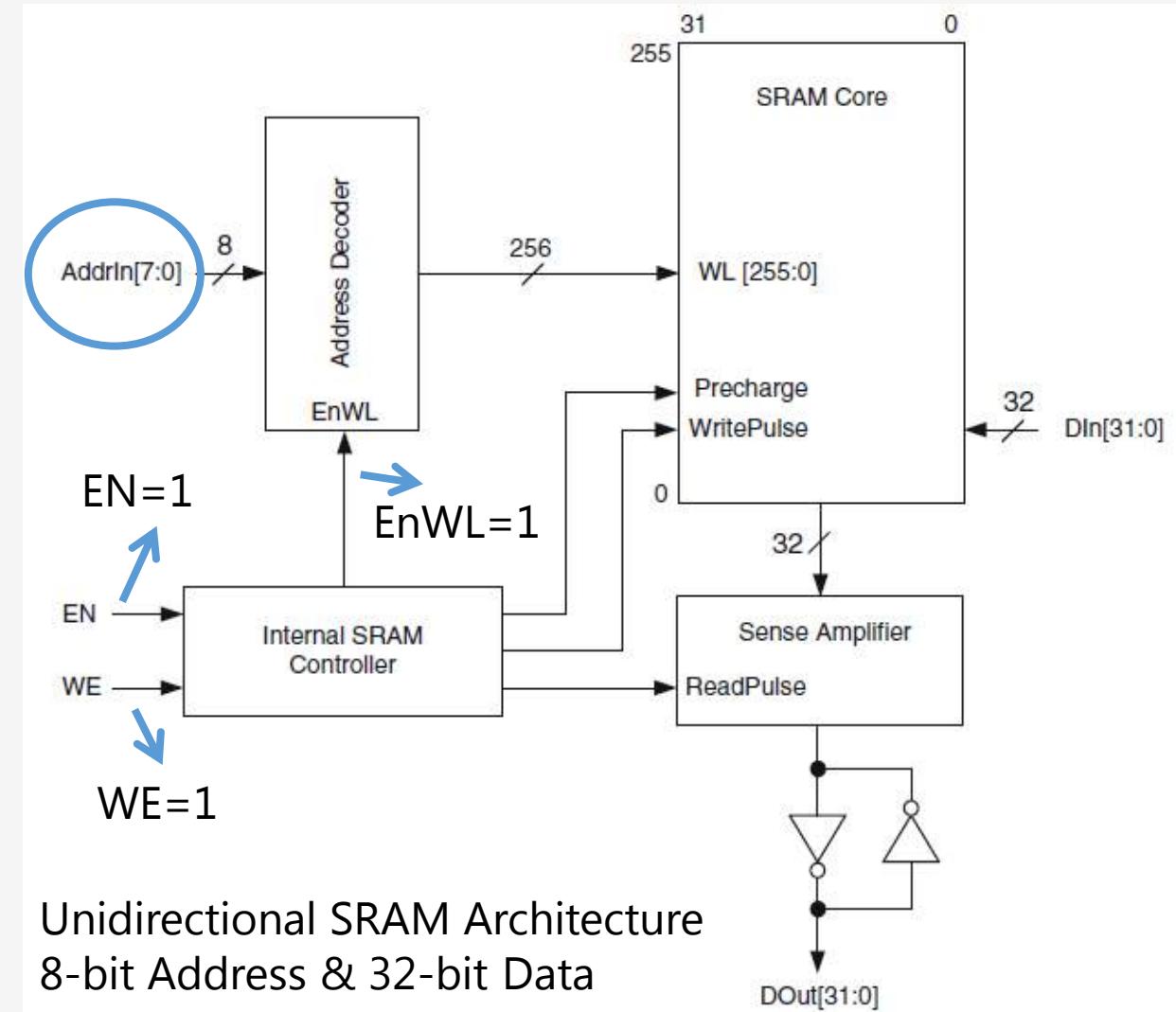
This combination precharges the SRAM core to a preset voltage and prepares the memory for a write.



SRAM - Data Write Sequence

The controller enables the address decoder by $\text{EnWL} = 1$.

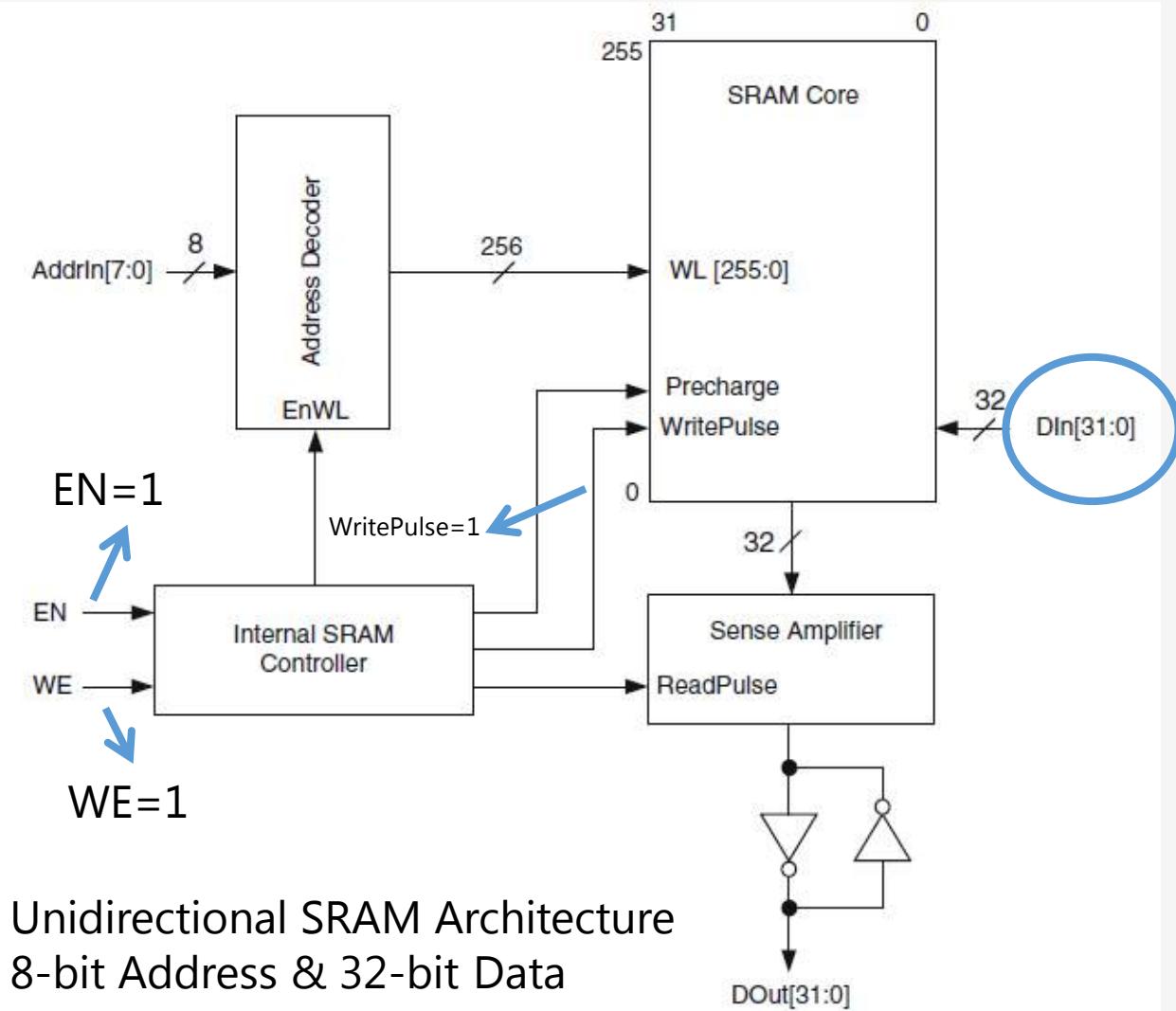
The decoder activates a single WL input out of 256 WLs according to the value provided at AddrIn .



SRAM - Data Write Sequence

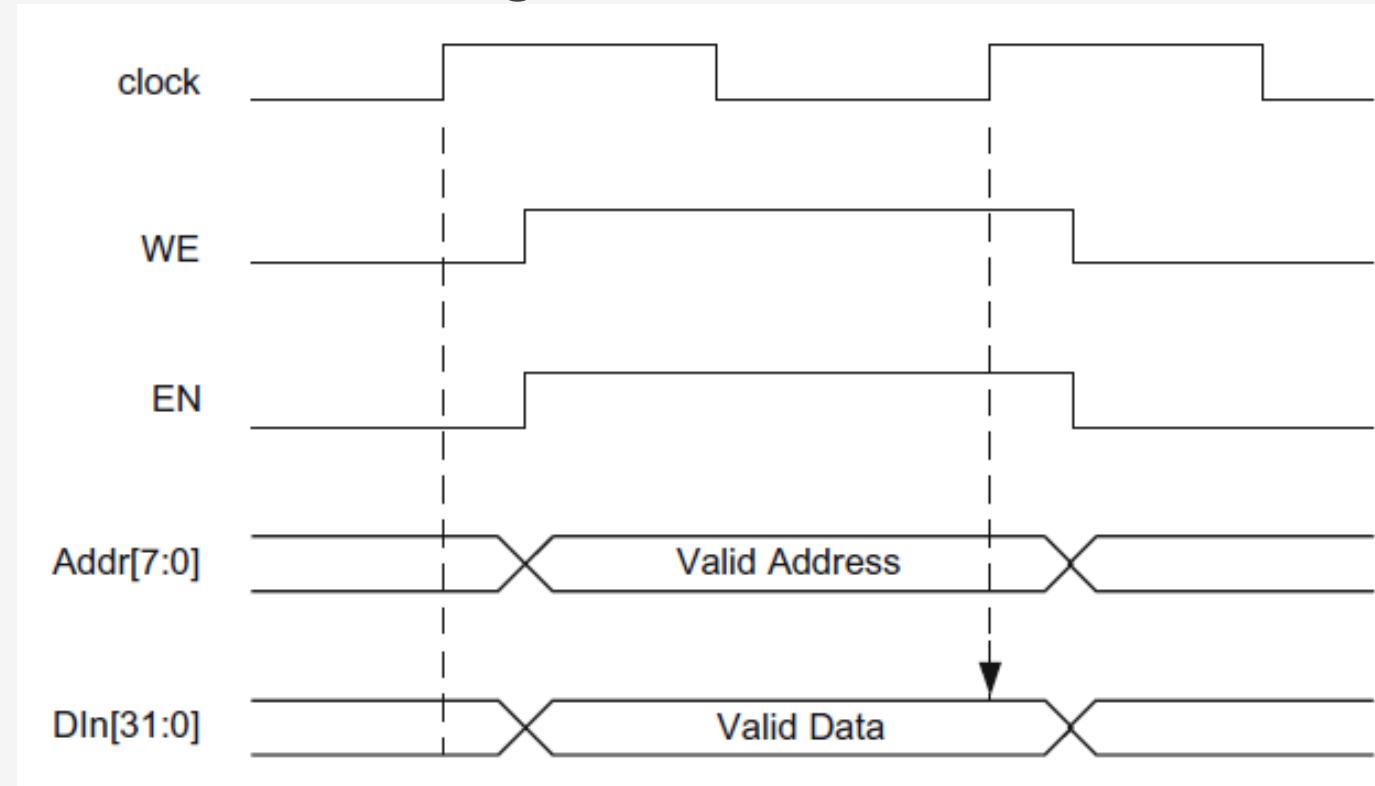
The controller produces
 $\text{WritePulse} = 1$

It allows the valid data at D_{In} to be written into the specified address.



SRAM - Data Write Timing Diagram

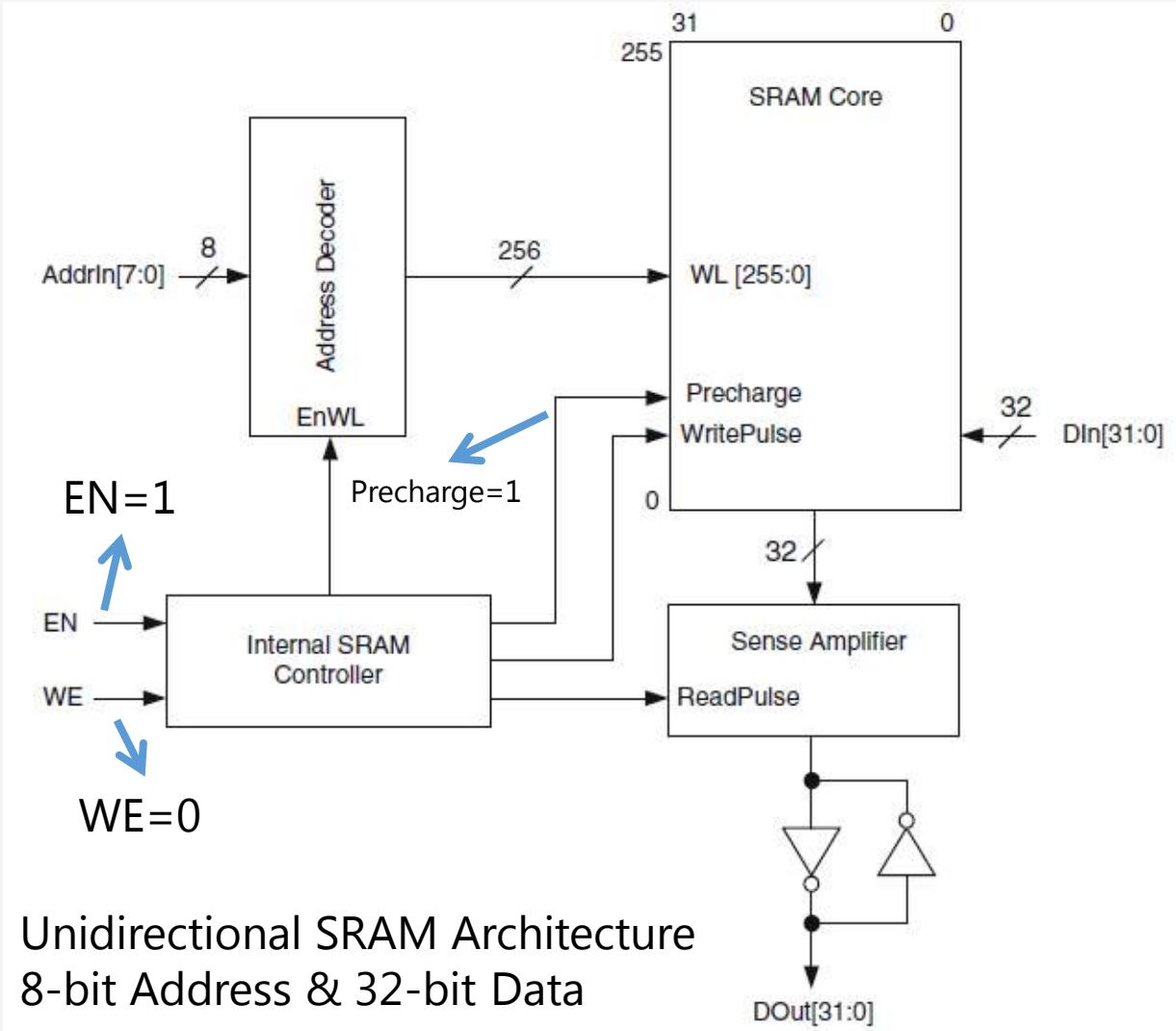
When EN and WE inputs are raised to logic 1, SRAM goes into the write mode, and the valid data is written to a specified address at the next positive clock edge.



SRAM - Data Read Sequence

The data read sequence starts with Enable (EN) = 1 and Write Enable (WE) = 0.

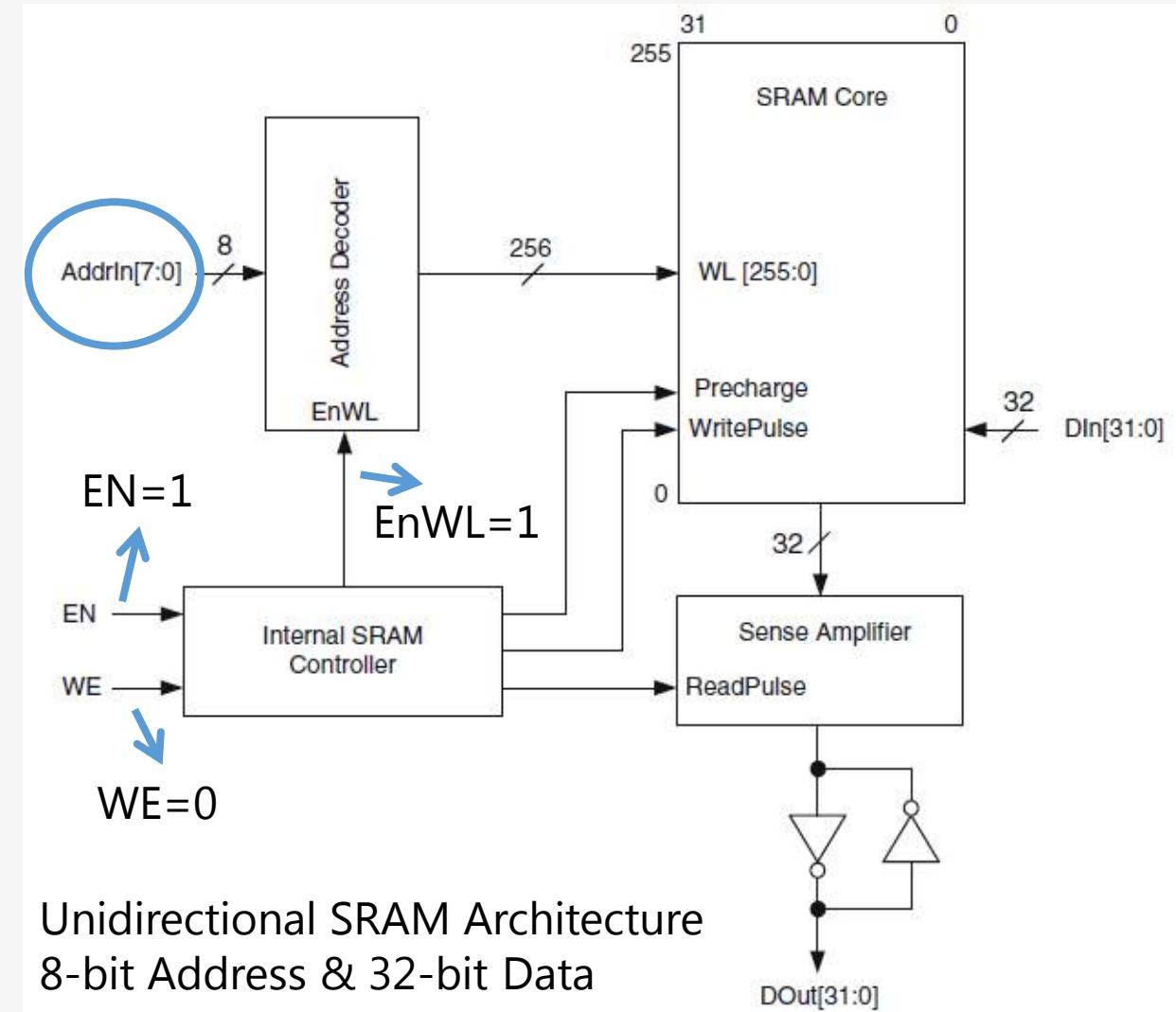
This combination precharges the SRAM core to a preset voltage and prepares the memory for a read.



SRAM - Data Read Sequence

The controller enables the address decoder by $\text{EnWL} = 1$.

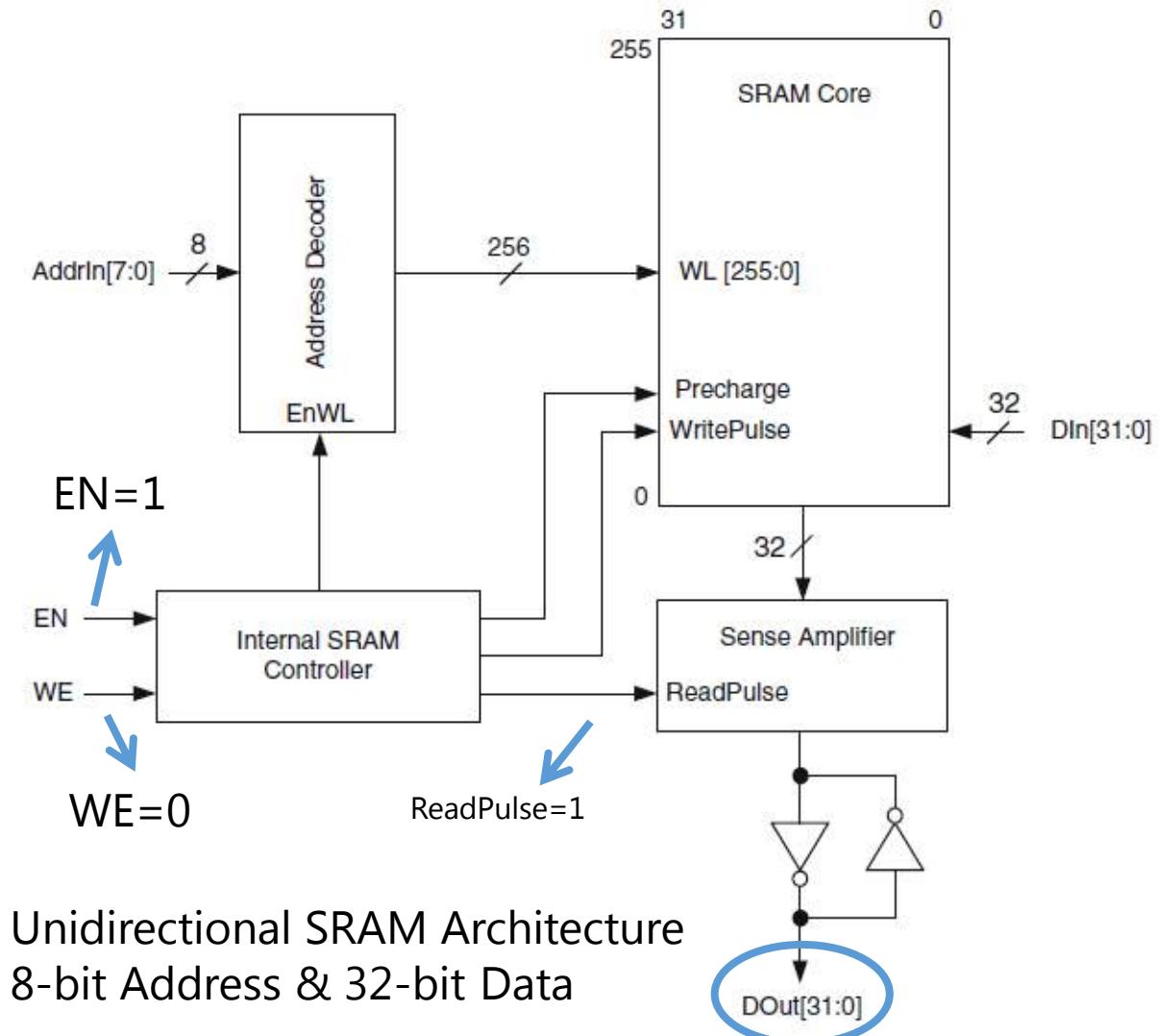
The decoder activates a single WL input out of 256 WLs according to the value provided at AddrIn .



SRAM - Data Read Sequence

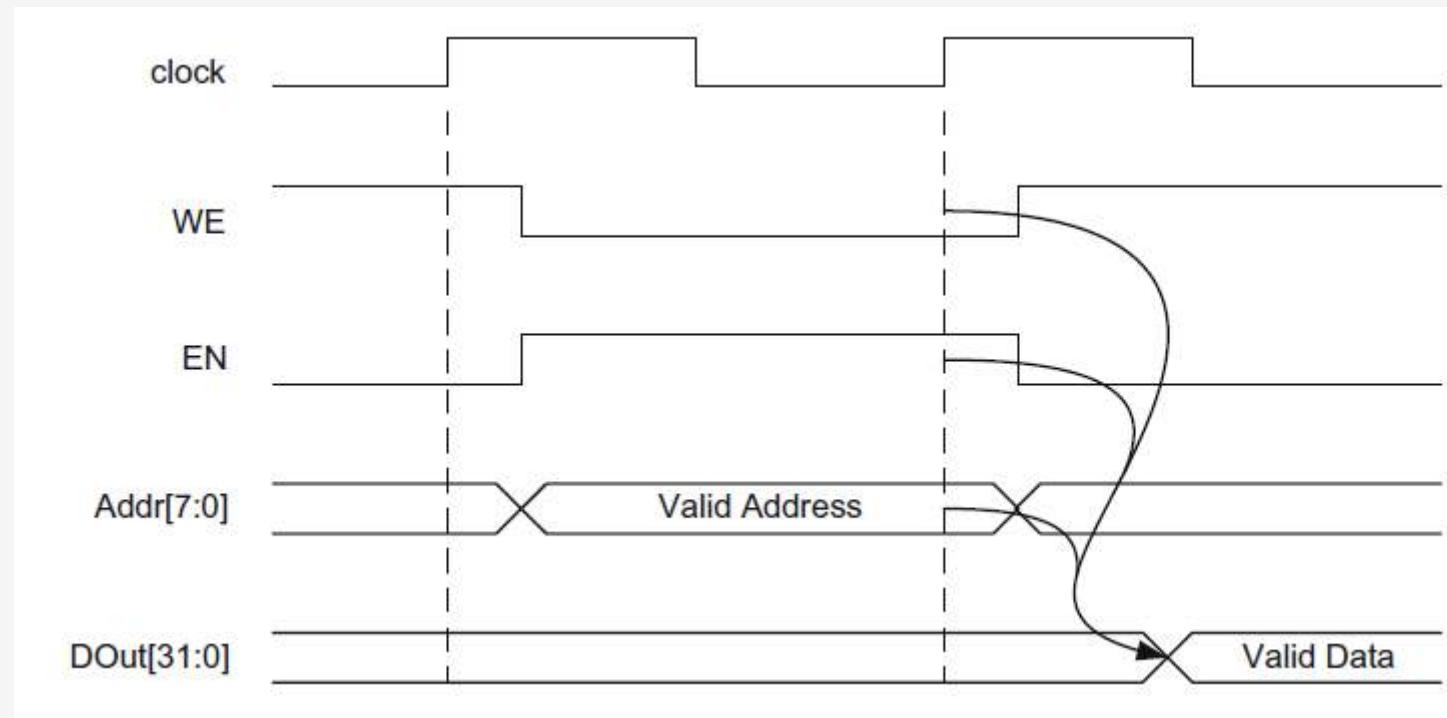
The controller produces
ReadPulse (RP) = 1

The sense amplifier amplifies
the cell voltage to full logic
levels and delivers the data
to the DOut port.



SRAM - Data Read Timing Diagram

When $EN = 1$ and $WE = 0$, SRAM is enabled and operates in the read mode. The core delivers the valid data sometime after the next positive edge of the clock.

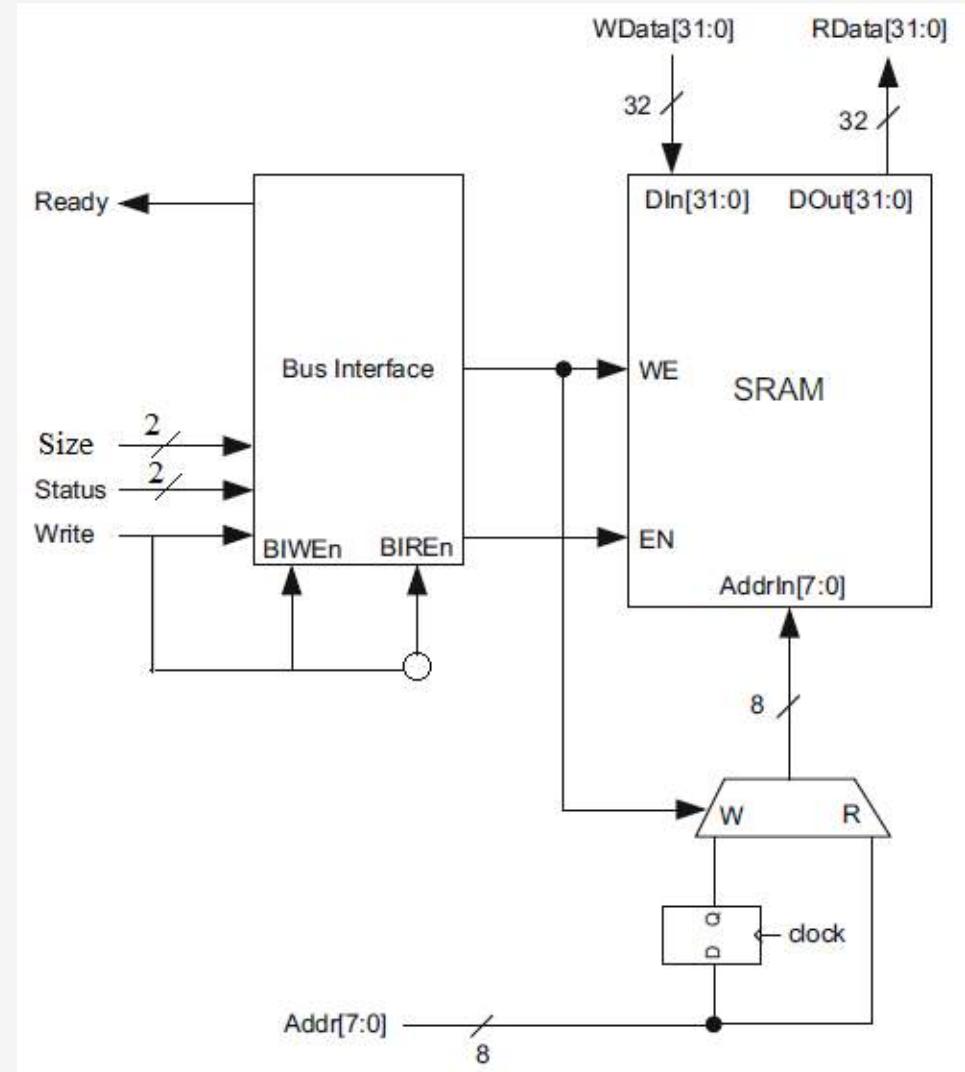


SRAM - Bus Interface

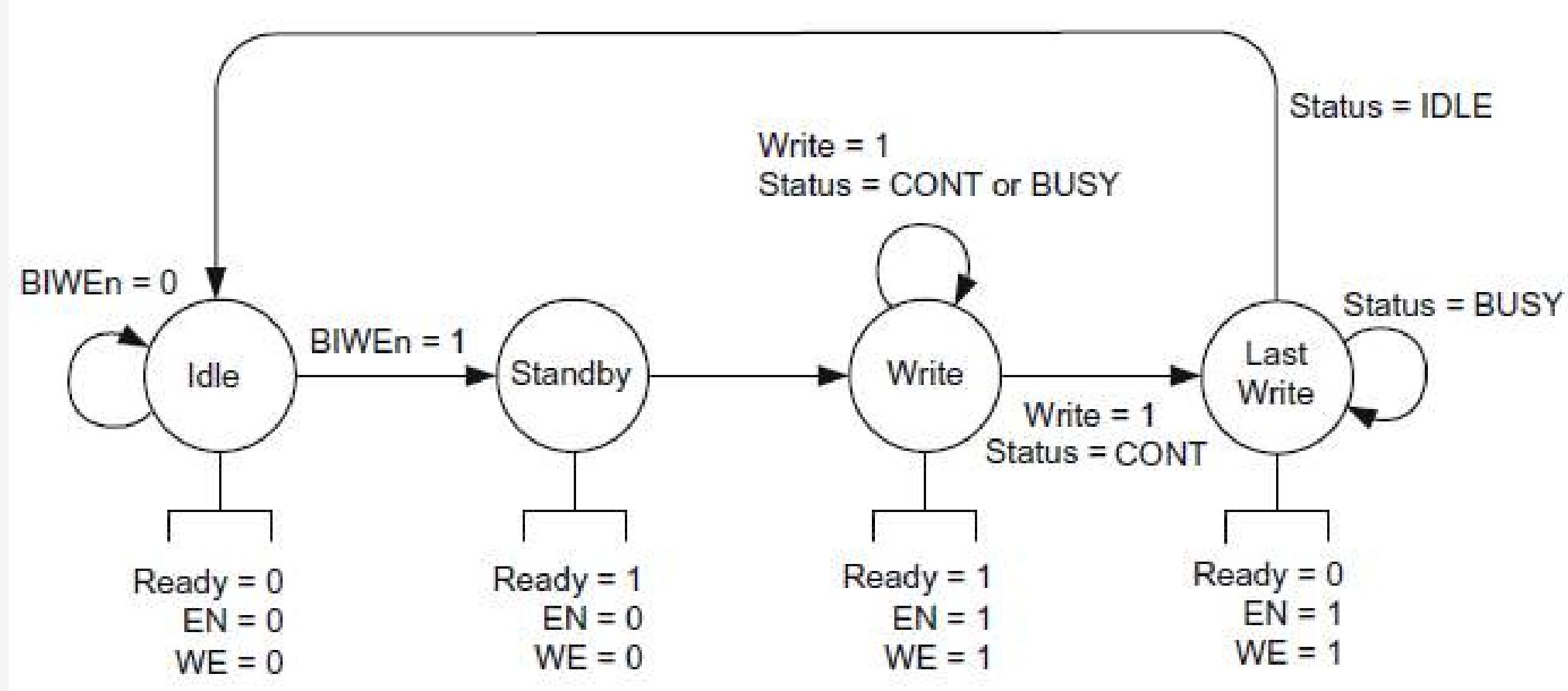
SRAM is considered to be a bus slave that exchanges data with the bus master on the basis of a Ready signal.

The bus interface translates all bus control signals (Size, Status, and Write) to SRAM control signals.

Check previous lectures to refresh your memory on system bus.

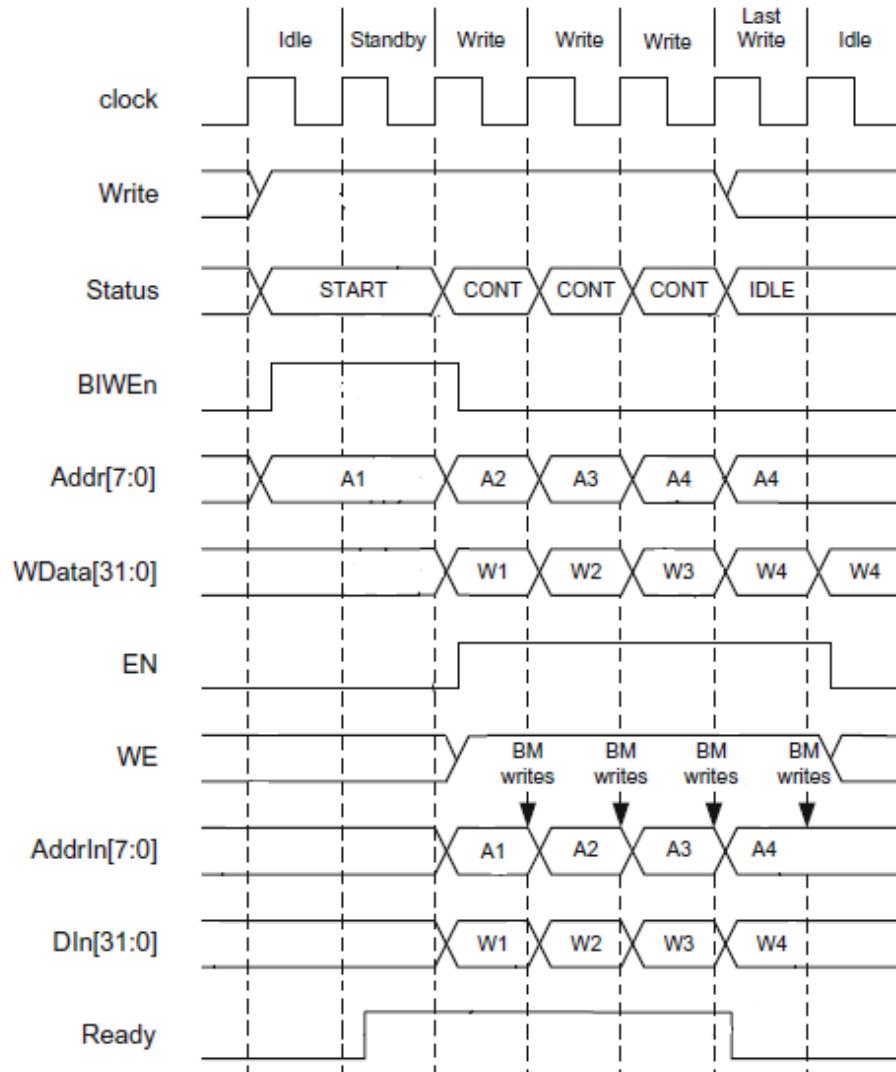


SRAM - Bus Interface State Machine for Write



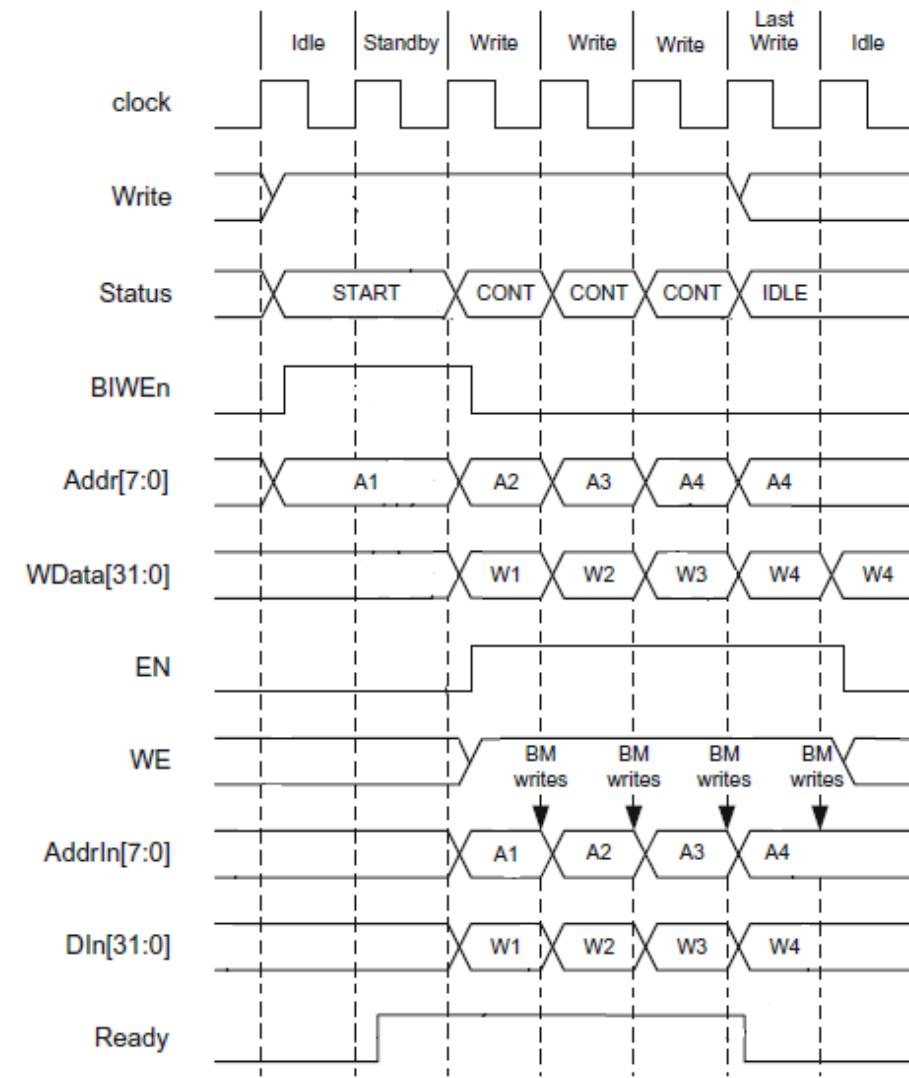
SRAM - Bus Interface Timing Diagram for Write

To initiate a write sequence, the bus master issues a valid address, Status = START and Write = 1 in the first clock cycle, and enables the bus interface for a write by producing an active-high Bus Interface Write Enable (BIWEn) signal.



SRAM - Bus Interface Timing Diagram for Write

Upon receiving the BIWEn = 1, the bus interface produces Ready = 1 in the second cycle, and prompts the bus master to change the address and control signals in the third cycle.

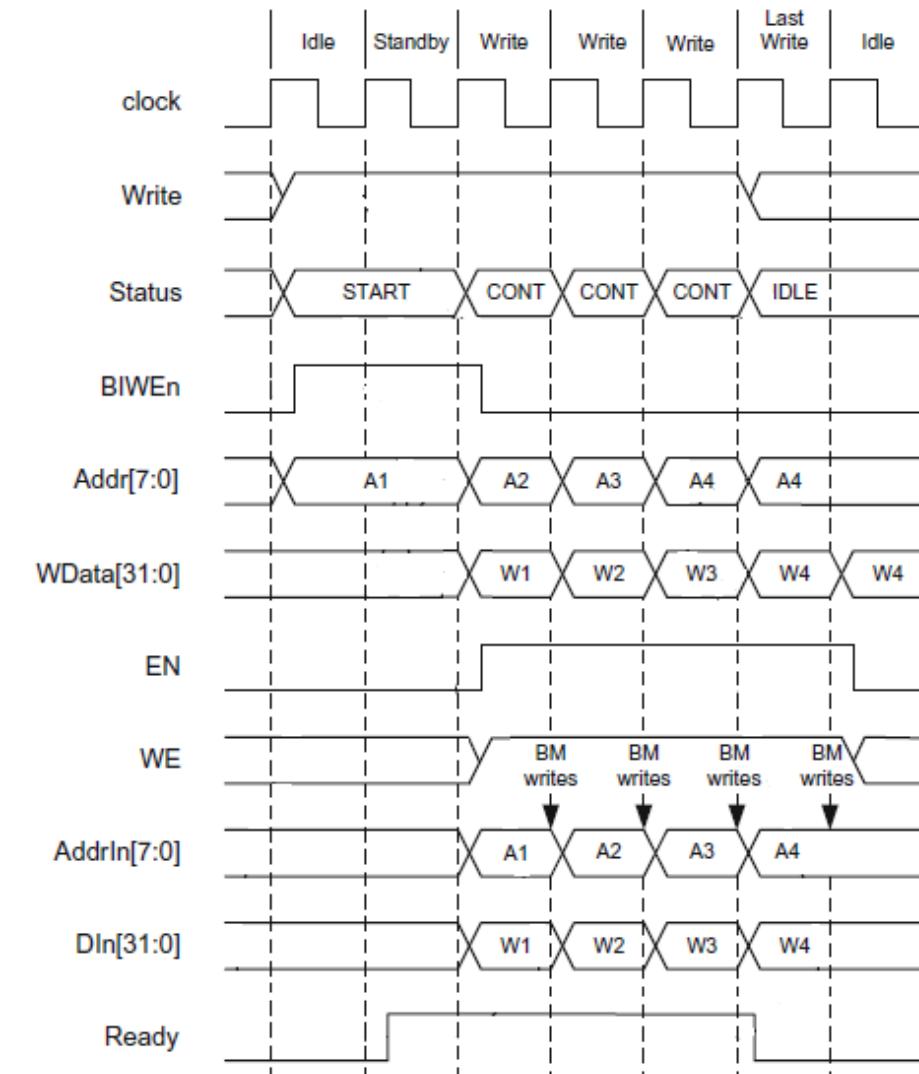


SRAM - Bus Interface Timing Diagram for Write

As the bus master changes its address from A1 to A2, it also sends its first data message, W1.

However, in order to write data into an SRAM address, a valid data must be available within the same cycle as the valid address.

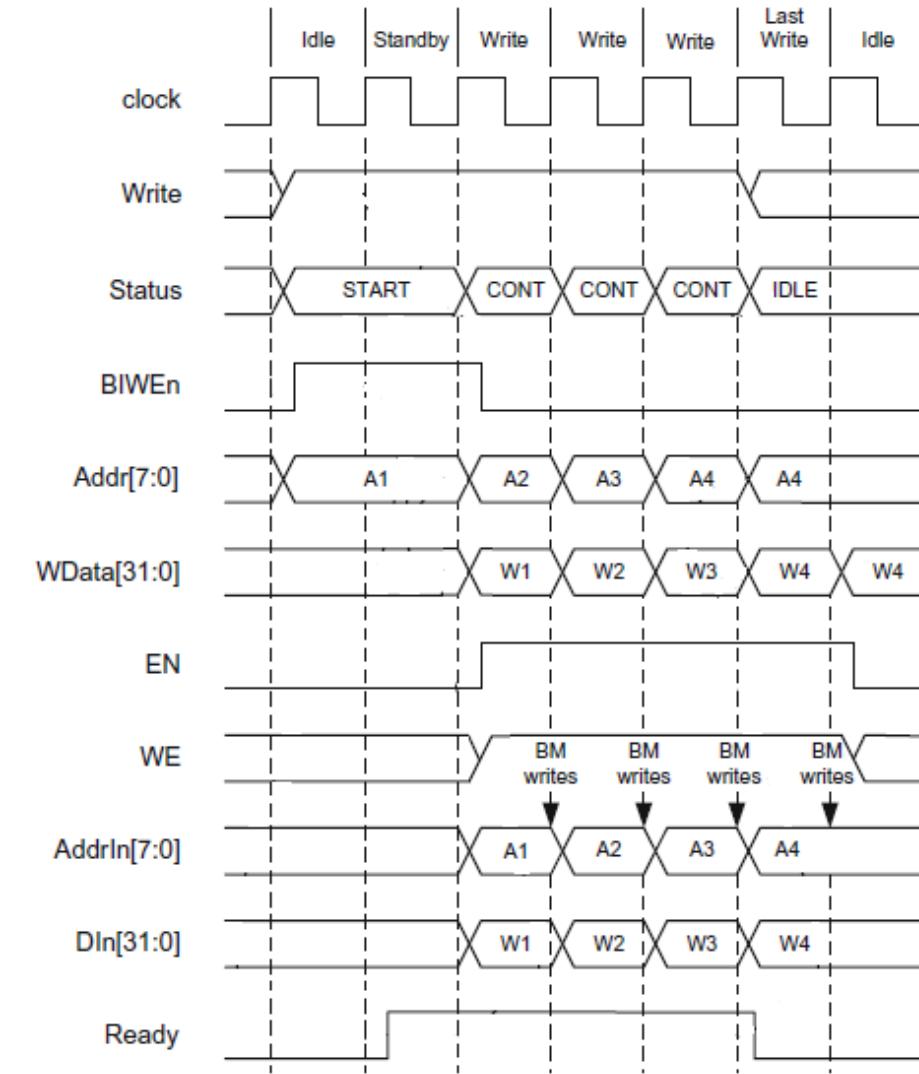
Therefore, a set of eight flip-flops are added to the write path prior to the AddrIn port so that the address, A1, is delayed for one clock cycle, and aligned with the current data, W1.



SRAM - Bus Interface Timing Diagram for Write

The bus interface also produces $EN = WE = 1$ in the third cycle so that $W1$ is written to $A1$ at the positive edge of the fourth clock cycle.

The next write is accomplished in the same way: the SRAM address is delayed for one cycle in order to write $W2$ into $A2$ at the positive edge of the fifth cycle.

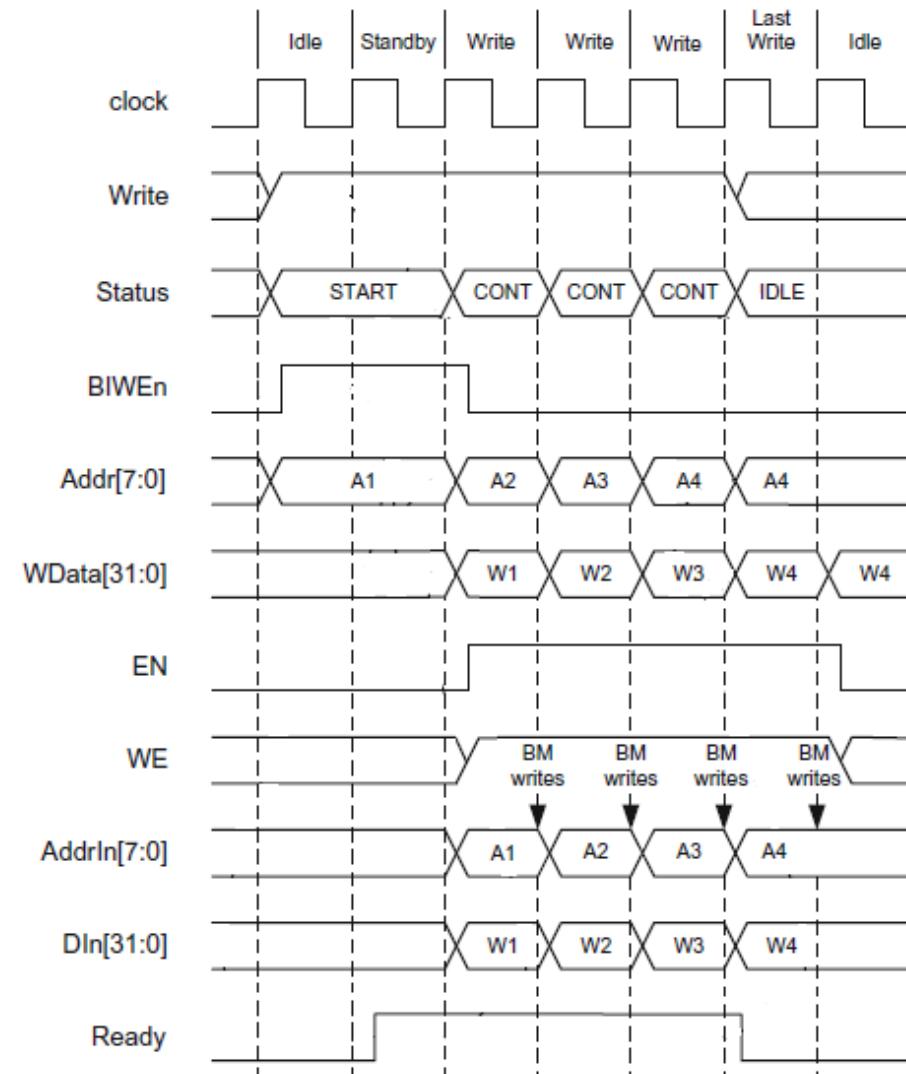


SRAM - Bus Interface Timing Diagram for Write

In the sixth cycle, the bus interface keeps $EN = WE = 1$ to be able to write $W3$ to $A3$, but lowers the Ready signal to logic 0 so that the bus master stops incrementing the slave address and suspends the controls in the next cycle.

In the seventh cycle, the bus interface lowers both EN and WE to logic 0 but allows the last data, $W4$, to be written to $A4$ at the positive edge of the clock.

You will draw SRAM bus interface state machine and timing diagram for read in the homework.



Dynamic Random Access Memory

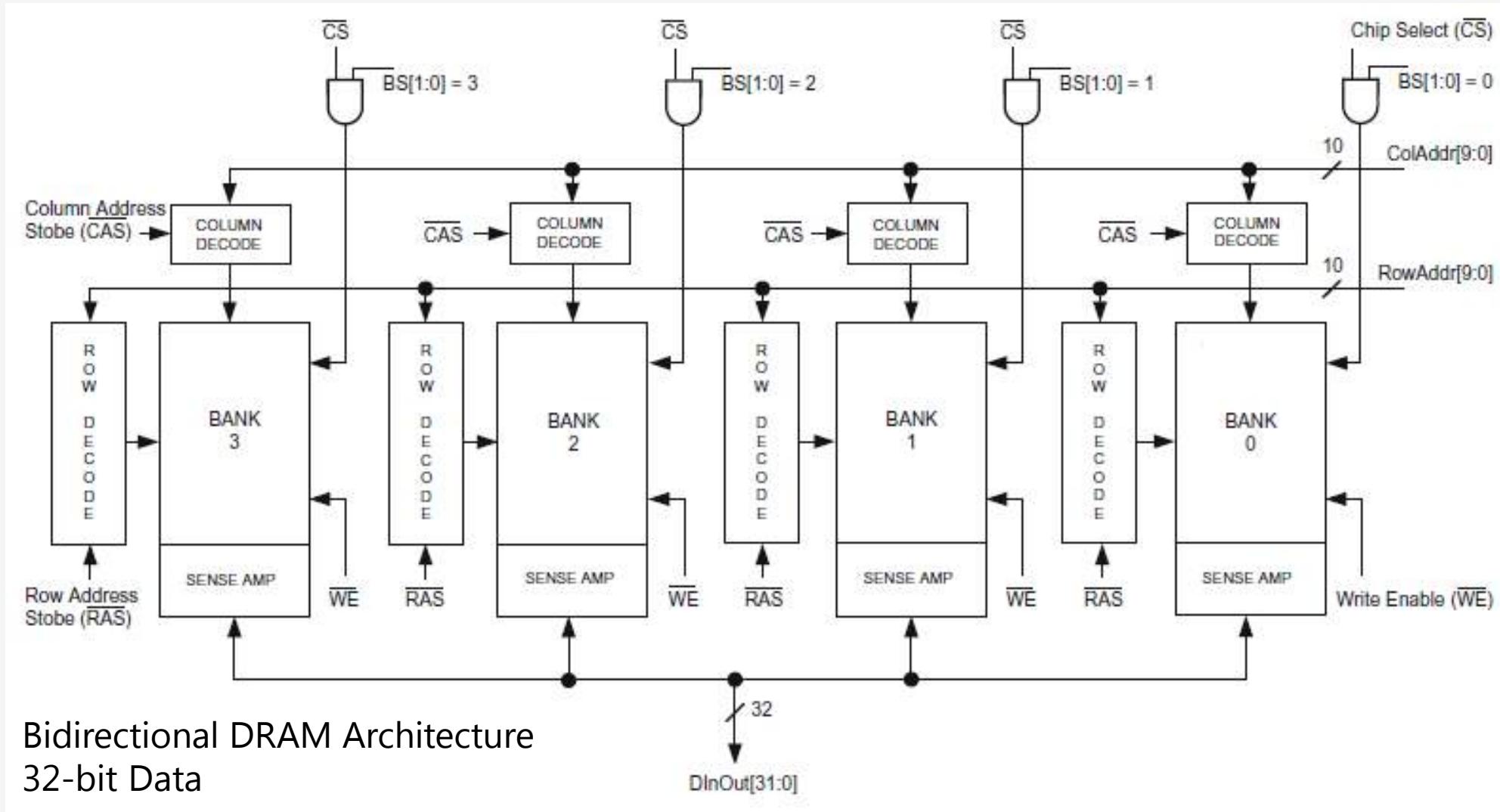
Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

DRAM Architecture



DRAM Architecture - Notes

Our DRAM consists of 4 memory cores (i.e., banks) accessible by a bidirectional input/output port.

The active-low Row Address Strobe (RAS), Column Address Strobe (CAS), Write Enable (WE), and Chip Select (CS) signals determine the functionality of the memory.

The charge on the DRAM cell slowly leaks, resulting in a reduced cell voltage. Thus, an automatic or manual cell refresh cycle becomes mandatory during DRAM operation to preserve the bit value in the cell.

DRAM Operation

$\overline{\text{CS}}$	$\overline{\text{RAS}}$	$\overline{\text{CAS}}$	$\overline{\text{WE}}$	OPERATION
0	0	0	1	Self Refresh
0	0	1	0	Precharge a Bank with BS[1:0]
0	0	1	1	Activate a Bank with BS[1:0]
0	1	0	0	Write into a Bank with BS[1:0]
0	1	0	1	Read from a Bank with BS[1:0]
0	1	1	0	Stop
1	X	X	X	DRAM Deselect

DRAM Operation - Notes

Prior to a read or a write, all the rows and columns of a bank must be precharged to a certain voltage level for one clock cycle.

The time interval between the precharge and the activation cycles is called the precharge time period, t_{PRE} .

In the activation cycle, row address is defined. Column address takes a time period called column access strobe, t_{CAS} .

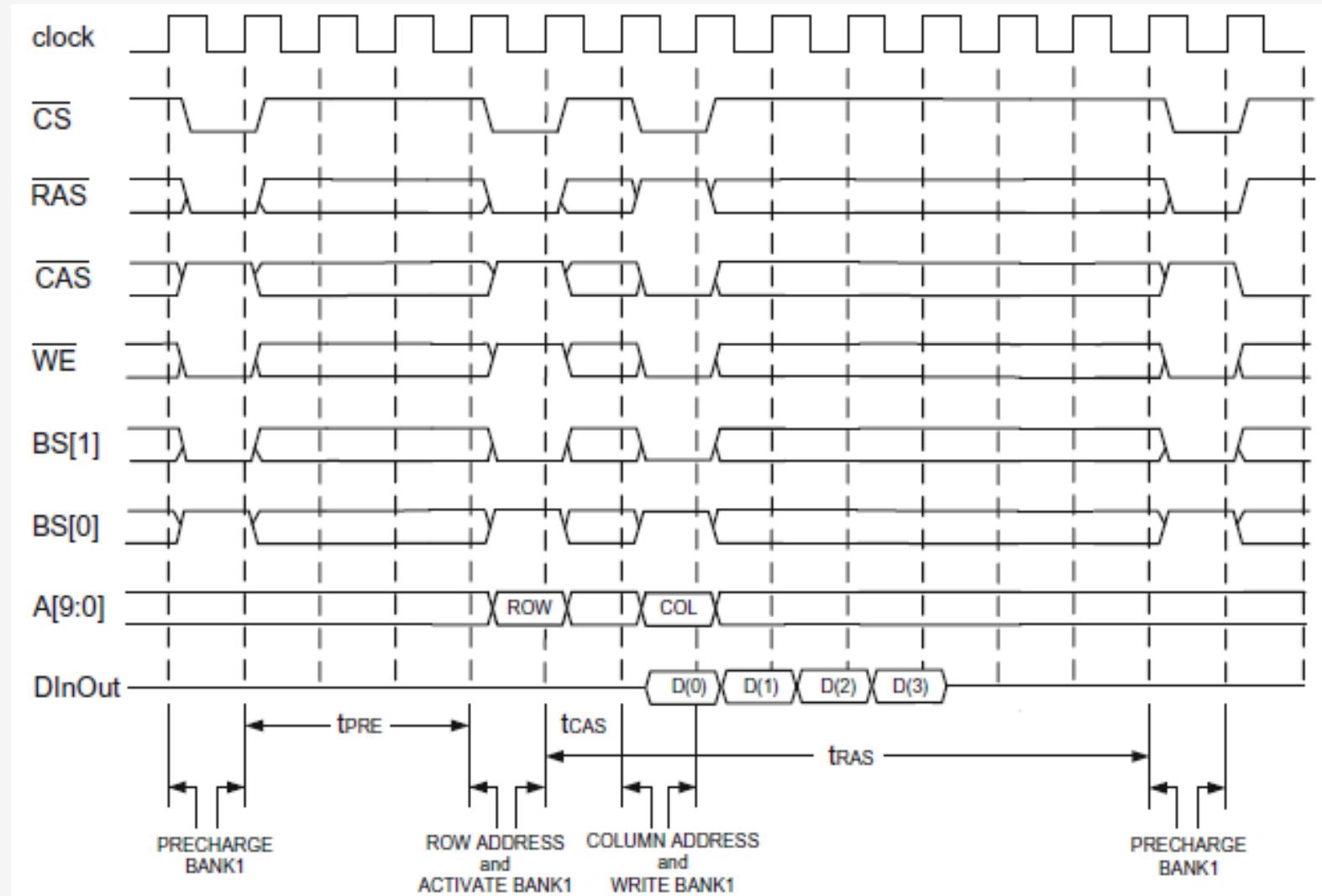
The next precharge period must not start until after a certain time period has elapsed for the same bank called the row access strobe time period, t_{RAS} .

DRAM Timing Diagram for Write - Example 1

The write starts with precharging Bank 1.

After $t=t_{PRE}$, the activation period starts and the row address is defined.

When the column address is generated after $t=t_{CAS}$, four data messages, D(0) to D(3), are written to Bank 1.



DRAM Write - Sample Question 1

A bus master writes 3 messages in consecutive addresses on Bank 2 of a DRAM, and 5 messages in another consecutive addresses of the same bank. Considering the following time periods, compute the total number of clock cycles, the bus master requires to finish its job.

$$t_{\text{PRE}} = 3 \text{ cycles}$$

$$t_{\text{CAS}} = 2 \text{ Cycles}$$

$$t_{\text{RAS}} = 7 \text{ cycles}$$

DRAM Write - Sample Question 1 - Solution

A bus master writes 3 messages in consecutive addresses on Bank 2 of a DRAM, and 5 messages in another consecutive addresses of the same bank. Considering the following time periods, compute the total number of clock cycles, the bus master requires to finish its job.

$$t_{PRE} = 3 \text{ cycles}$$

$$t_{CAS} = 2 \text{ Cycles}$$

$$t_{RAS} = 7 \text{ cycles}$$

$$\begin{aligned}\text{First Write} &= \text{Precharge Code} + t_{PRE} + \text{Row Address} + t_{CAS} + \text{Write Period 1} \\ &= 1 + 3 + 1 + 2 + 3 = 10\end{aligned}$$

$$\begin{aligned}\text{Wait Time} &= t_{RAS} - t_{CAS} - \text{Write Period 1} \\ &= 7 - 2 - 3 = 2\end{aligned}$$

$$\begin{aligned}\text{Second Write} &= \text{Precharge Code} + t_{PRE} + \text{Row Address} + t_{CAS} + \text{Write Period 2} \\ &= 1 + 3 + 1 + 2 + 5 = 12\end{aligned}$$

$$\begin{aligned}\text{Total} &= \text{First Write} + \text{Wait Time} + \text{Second Time} \\ &= 10 + 2 + 12 = 24 \text{ cycles}\end{aligned}$$

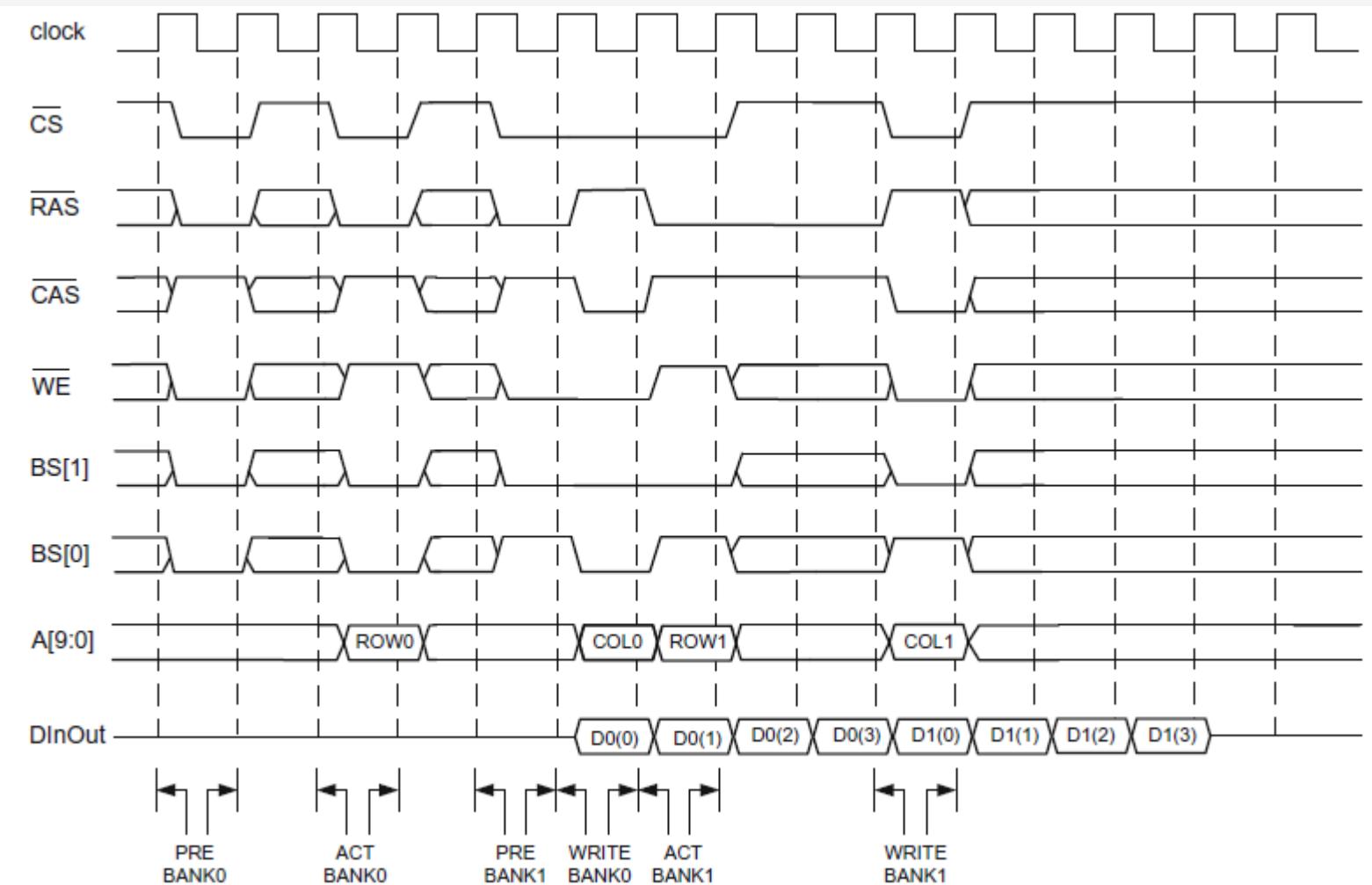
DRAM Timing Diagram for Write - Example 2

Write to multiple banks may result in a time-saving scenario.

Here four messages are written to bank 0 and four messages to bank 1 without any cycle loss.

What are t_{PRE} and t_{CAS} here?

How about DRAM timing diagram for read?



DRAM Write - Sample Question 2

A bus master writes 3 messages in consecutive addresses on Bank 1 of a DRAM, and another 5 messages in consecutive addresses of Bank 2 of the same DRAM. Considering the following time periods, compute the total number of clock cycles, the bus master requires to finish its job.

$$t_{\text{PRE}} = 3 \text{ cycles}$$

$$t_{\text{CAS}} = 2 \text{ Cycles}$$

$$t_{\text{RAS}} = 7 \text{ cycles}$$

DRAM Write - Sample Question 2 - Solution

A bus master writes 3 messages in consecutive addresses on Bank 1 of a DRAM, and another 5 messages in consecutive addresses of Bank 2 of the same DRAM. Considering the following time periods, compute the total number of clock cycles, the bus master requires to finish its job.

$$t_{\text{PRE}} = 3 \text{ cycles}$$

$$t_{\text{CAS}} = 2 \text{ Cycles}$$

$$t_{\text{RAS}} = 7 \text{ cycles}$$

Pre. Code	t_{PRE}	Row Ad.	t_{CAS}		Write 1			
1	3	1	2		3			
				Pre. Code	t_{PRE}	Row Ad.	t_{CAS}	Write 2
1	3	1	1	1	3	1	2	5

$$\text{Total} = 1 + 3 + 1 + 1 + 1 + 3 + 1 + 2 + 5 = 18 \text{ cycles}$$

Homework Assignment 5

Due Date: April 20, 11:30PM on BeachBoard Dropbox

Late Submission Policy:

One day delay (Second Due Date: Apr. 21, 11:30PM): 25% deduction

Two days delay (Third Due Date: Apr. 22, 11:30PM): 50% deduction

More than two days delay: No credit

Academic Integrity: There is zero tolerance for cheating, plagiarism, or any other act of violation of Academic Integrity.

Electrically Erasable Programmable Read Only Memory

Amin Rezaei

CECS 301 - Computer Logic Design II

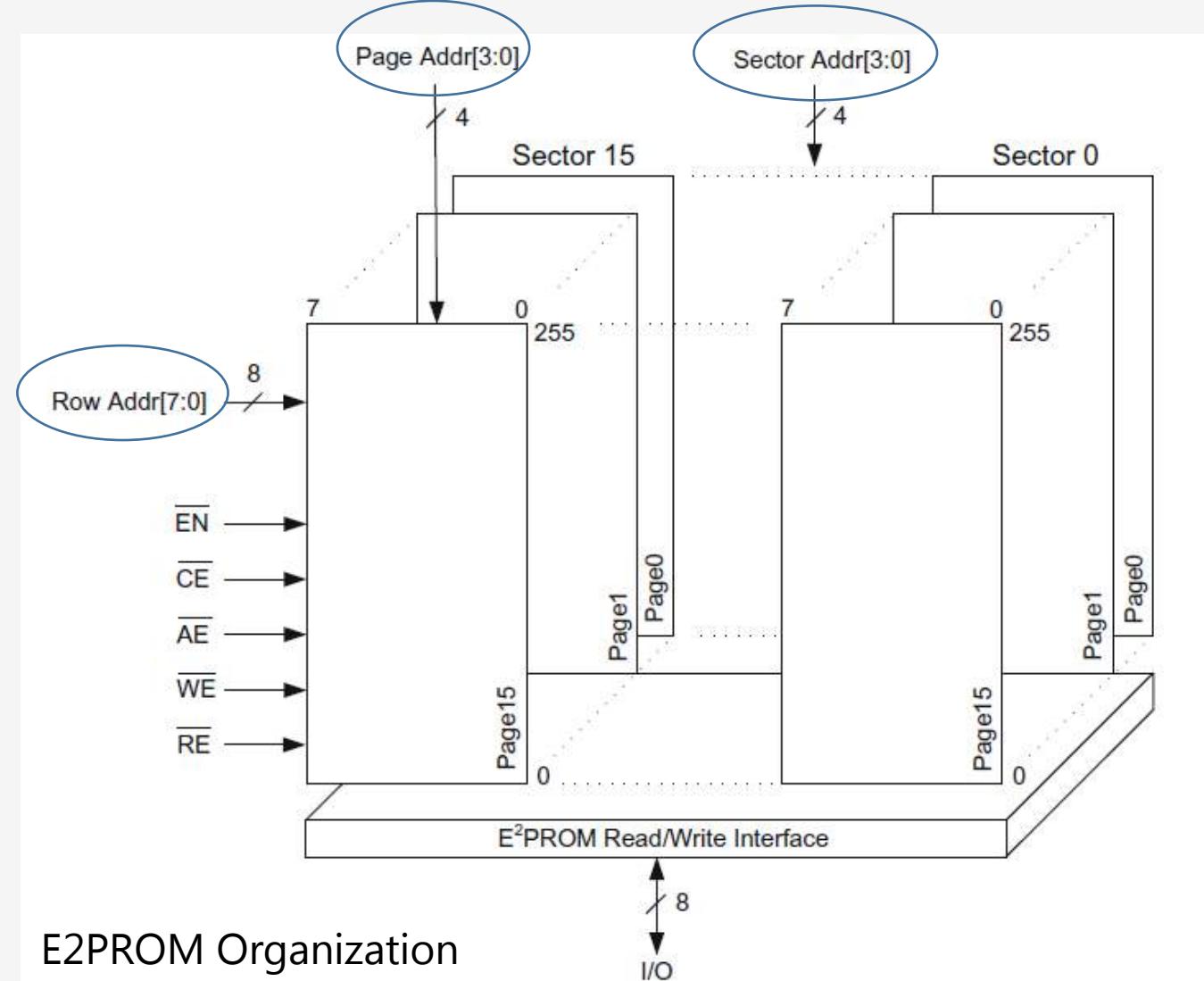
California State University, Long Beach

Spring 2021

E²PROM Organization

E²PROM is composed of multiple **sectors**, each of which contains multiple **pages**.

A single word in E²PROM can be located by specifying its sector address, page address and row address.



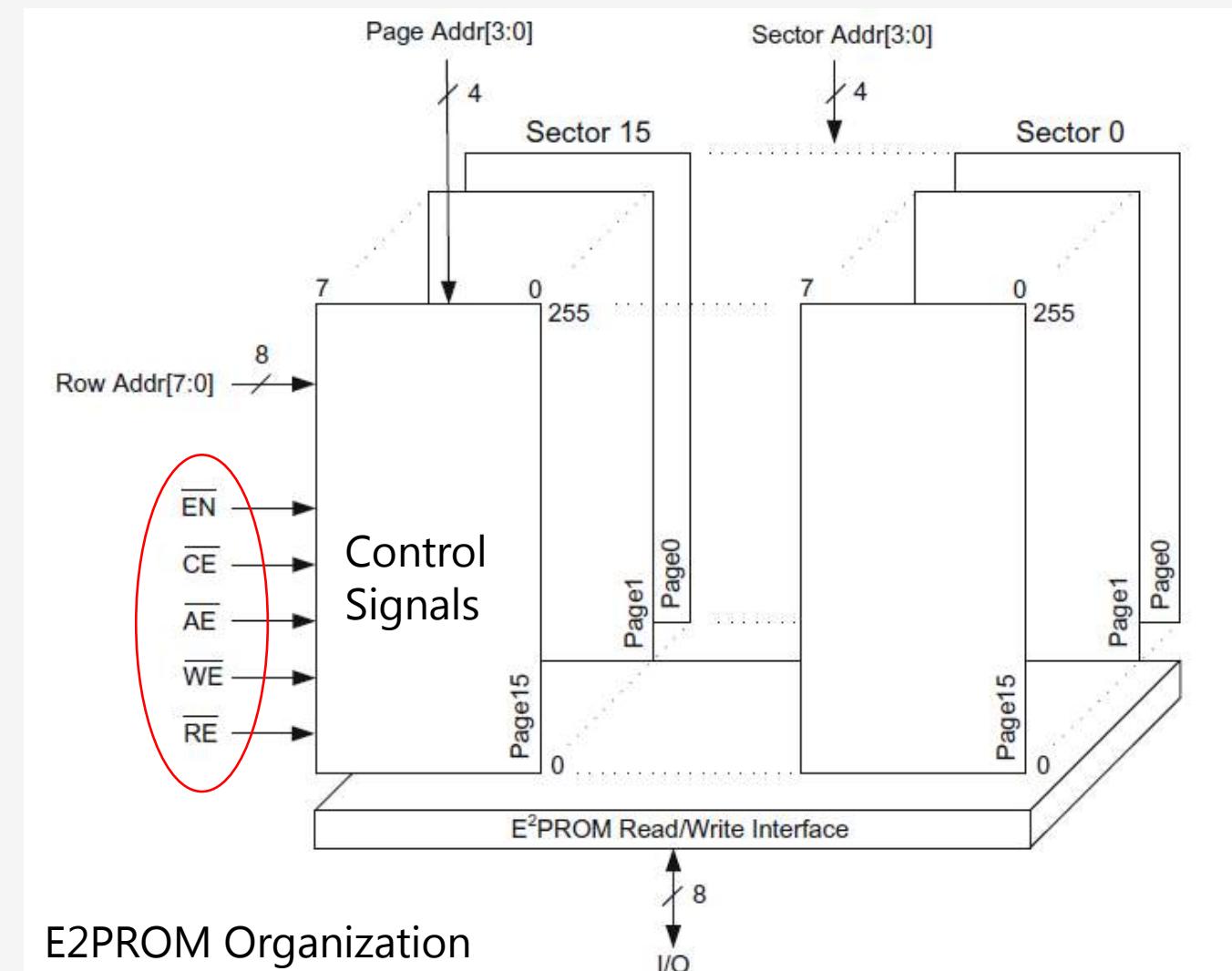
E²PROM Organization

The active-low **Enable signal (EN)**, places a particular page in standby mode and prepares it for an upcoming operation.

The active-low **Command Enable signal (CE)** is issued with a command code, such as read, write or erase.

The active-low **Address Enable signal (AE)** is issued when an address is provided.

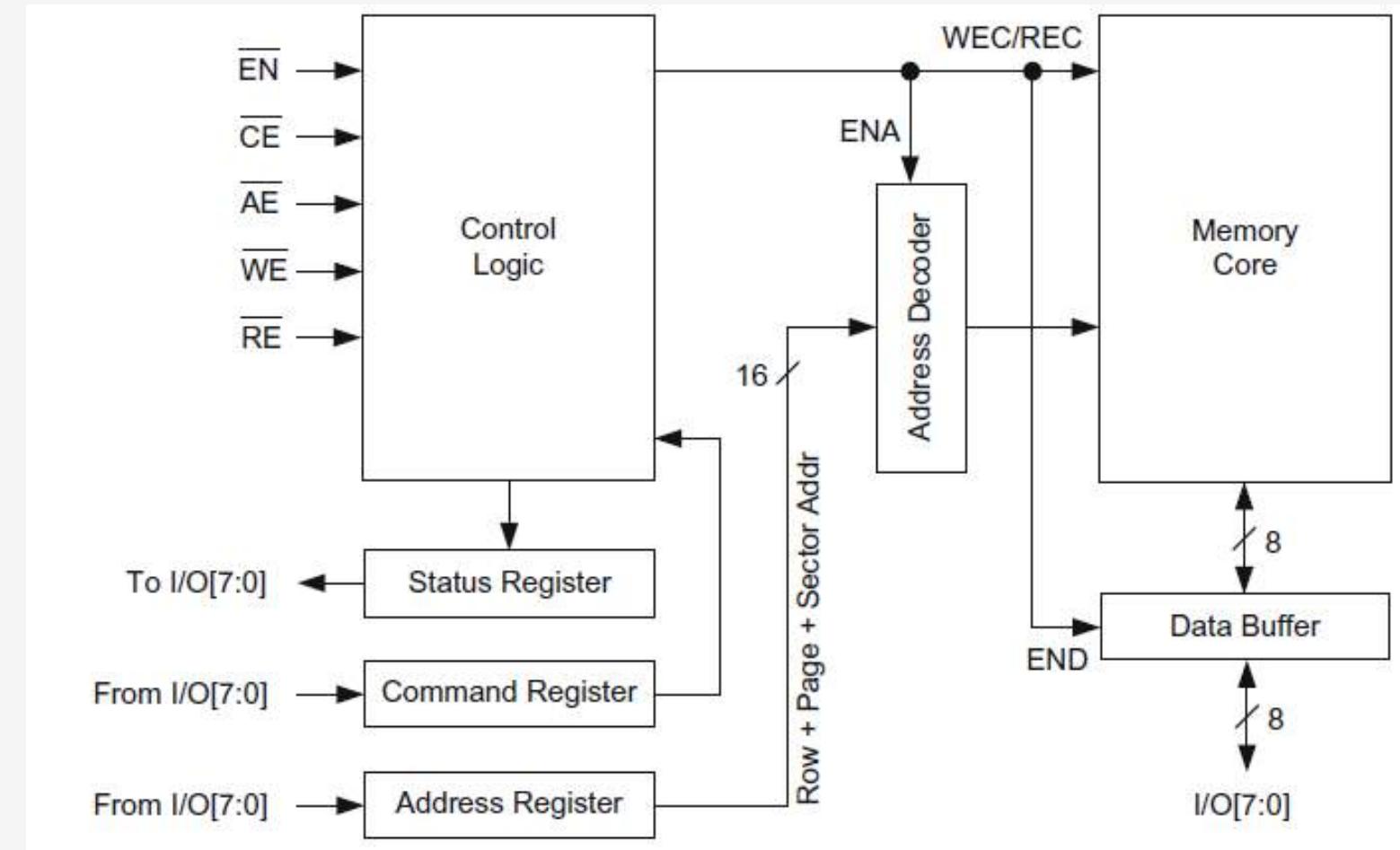
The active-low **Write Enable (WE)** and **Read Enable (RE)** signals are issued for writing and reading data.



E²PROM Architecture

First, command address registers and are programmed.

Second, the control logic enables the address decoder, the data buffer, and the memory core using the active-high **Enable Address (ENA)**, **Enable Data Buffer (END)**, and **Write Enable Core (WEC)** or **Read Enable Core (REC)** signals depending on the operation.



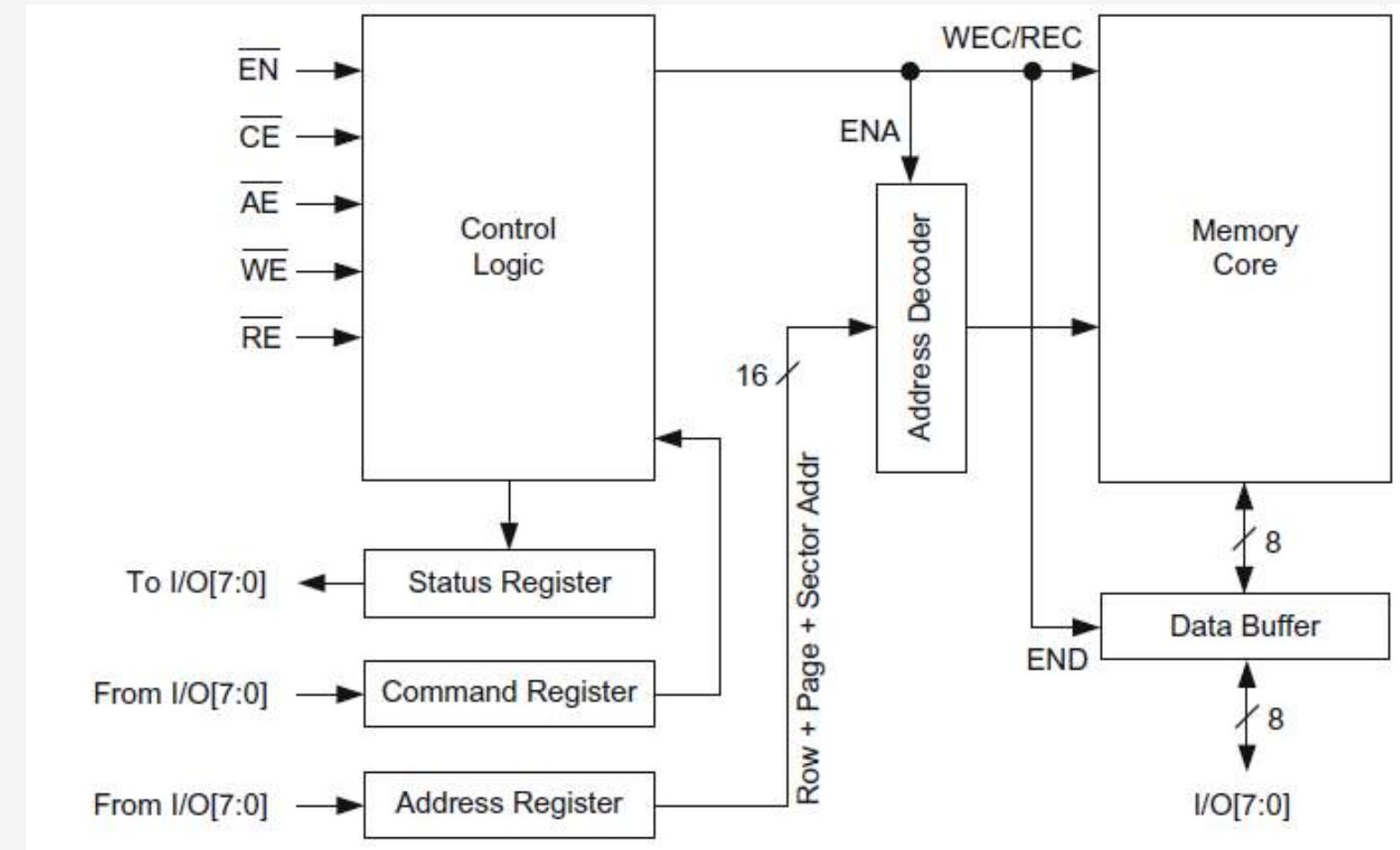
Bidirectional E2PROM Architecture

E²PROM Architecture

Third, the address stored in the address register is decoded to point the location of data.

Read operation → The required data is retrieved from the E²PROM core and stored in the data buffer before it is delivered to the I/O bus.

Write operation → The data is stored in the data buffer first before it is uploaded to the designated E²PROM address.



Bidirectional E2PROM Architecture

E²PROM Operation

$\overline{\text{EN}}$	$\overline{\text{WE}}$	$\overline{\text{RE}}$	
0	1	1	Standby
0	0	1	Write
0	1	0	Read
1	X	X	Hibernate

Hibernate mode disables the address decoder, memory core and data buffer to reduce power dissipation, and puts the device into sleep.

E²PROM - Command Input Timing Diagram

The operation sequence always starts with the command input.

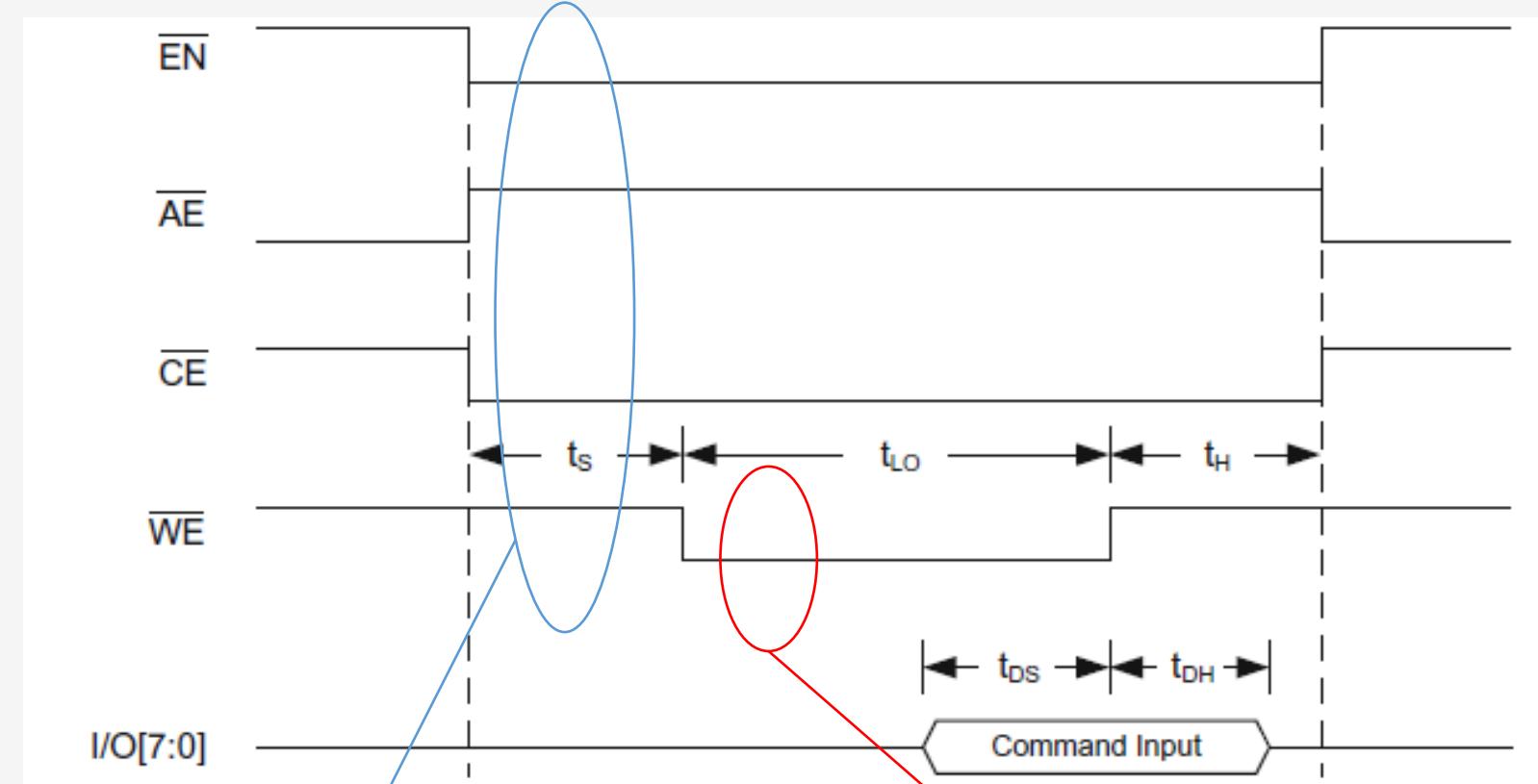
Setup time (t_s)

Low phase time (t_{LO})

Hold time (t_H)

Data setup time (t_{DS})

Data hold time (t_{DH})



Issue
command input.

Command input is written into
command register.

E²PROM - Address Input Timing Diagram

After command input,
the operation sequence
continues by address
entry.

Setup time (t_S)

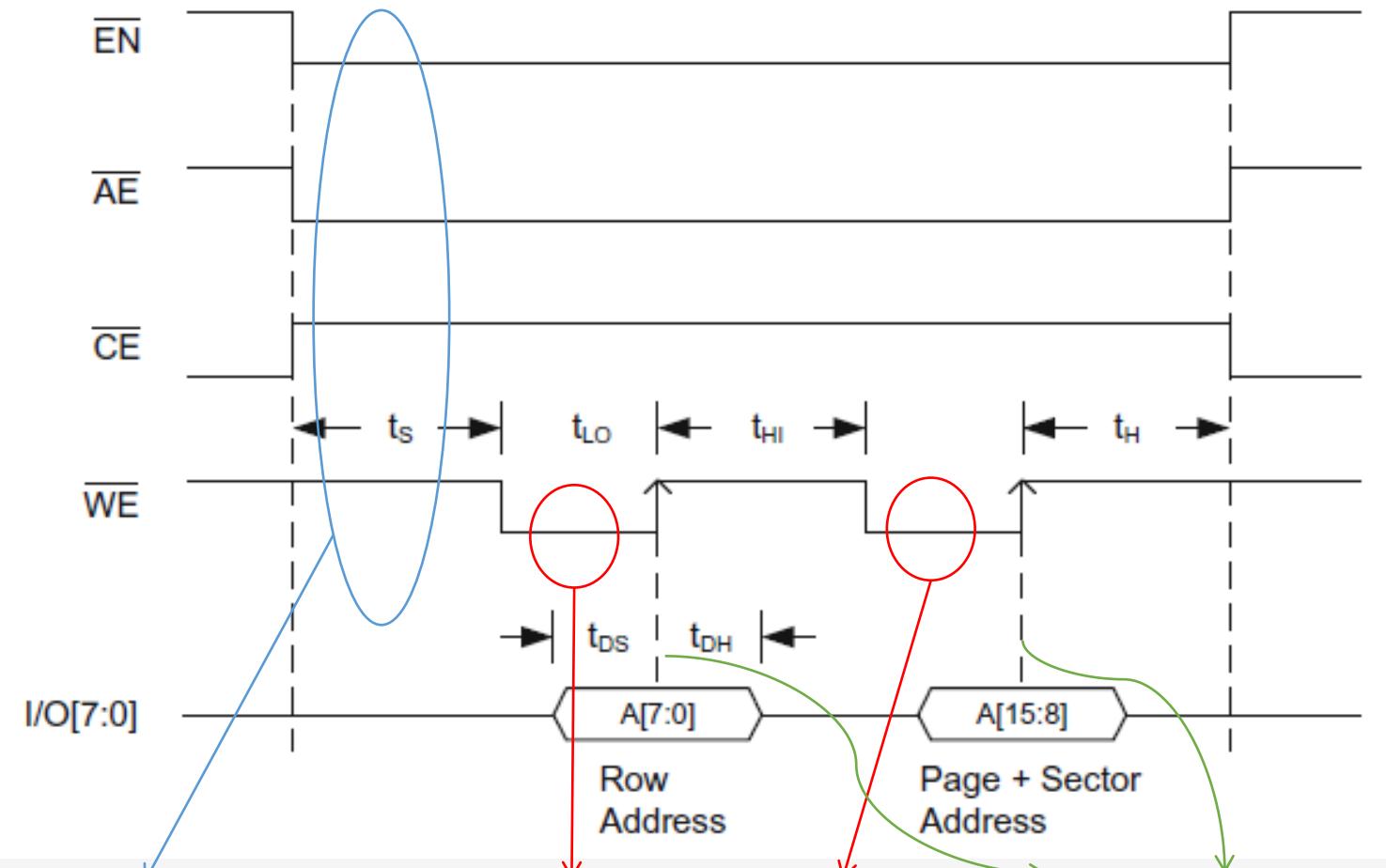
Hold time (t_H)

Low phase time (t_{LO})

High phase time (t_{HI})

Data setup time (t_{DS})

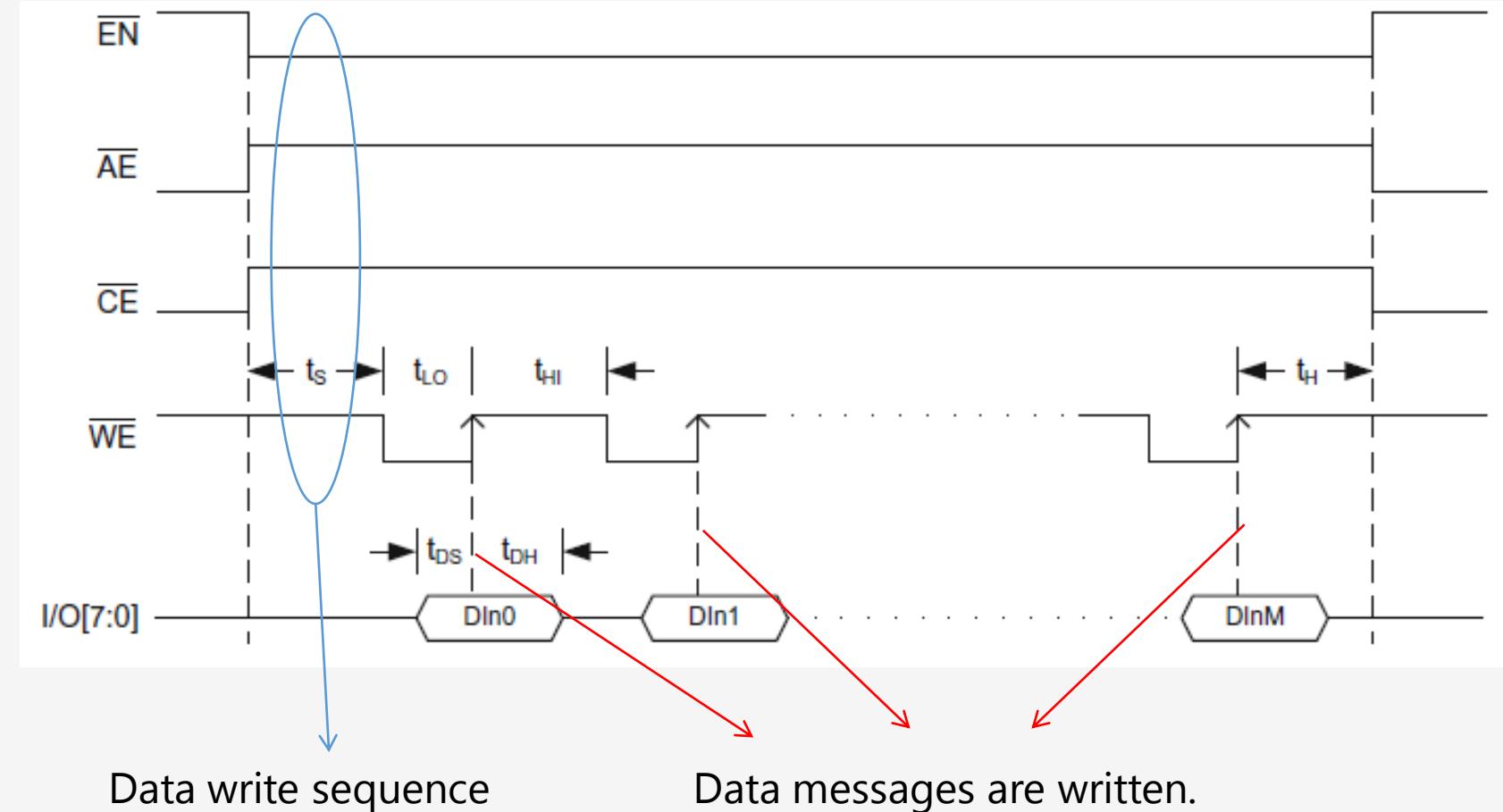
Data hold time (t_{DH})



E²PROM - Data Write Timing Diagram

After command input and address entry, the operation sequence continues by data write or read entries.

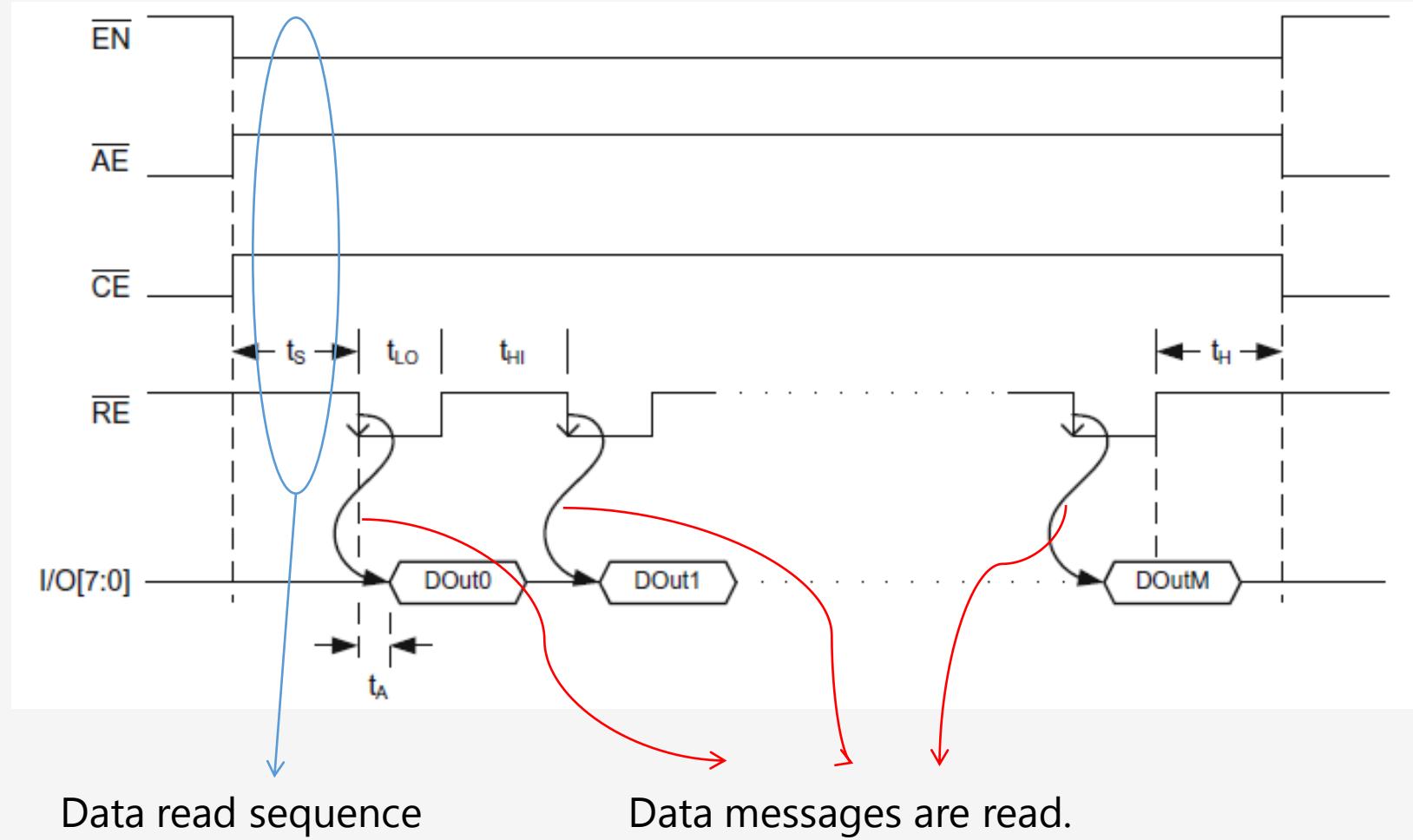
Here, M + 1 number of data messages are written to the E²PROM.



E²PROM - Data Read Timing Diagram

After command input and address entry, the operation sequence continues by data write or read entries.

Access time (t_A)

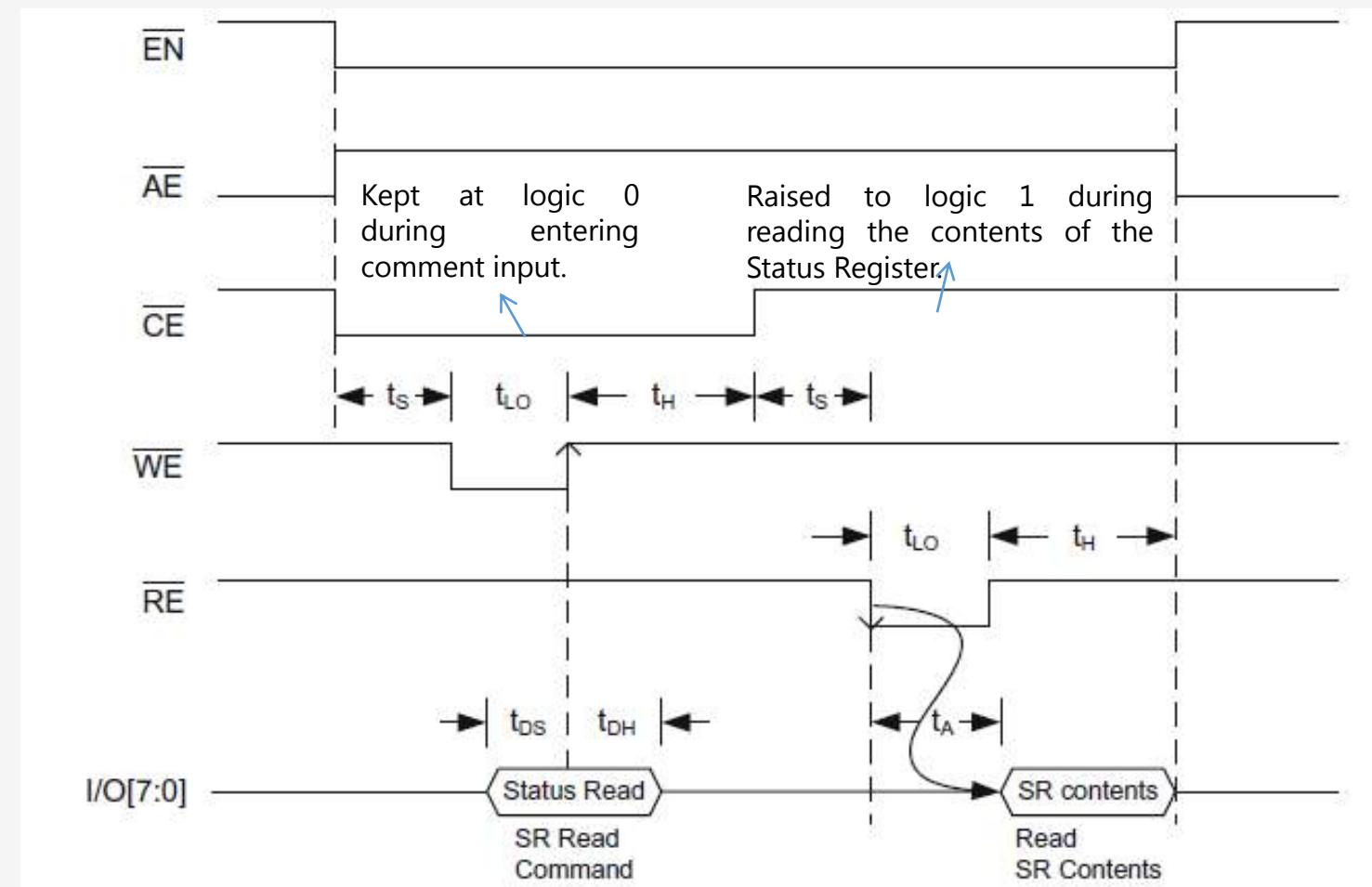


E²PROM - Status Register Timing Diagram

Reading data from the Status Register is a two-step process:

Entering Command Input

Reading Status Register



E²PROM - Full Page Write Timing Diagram

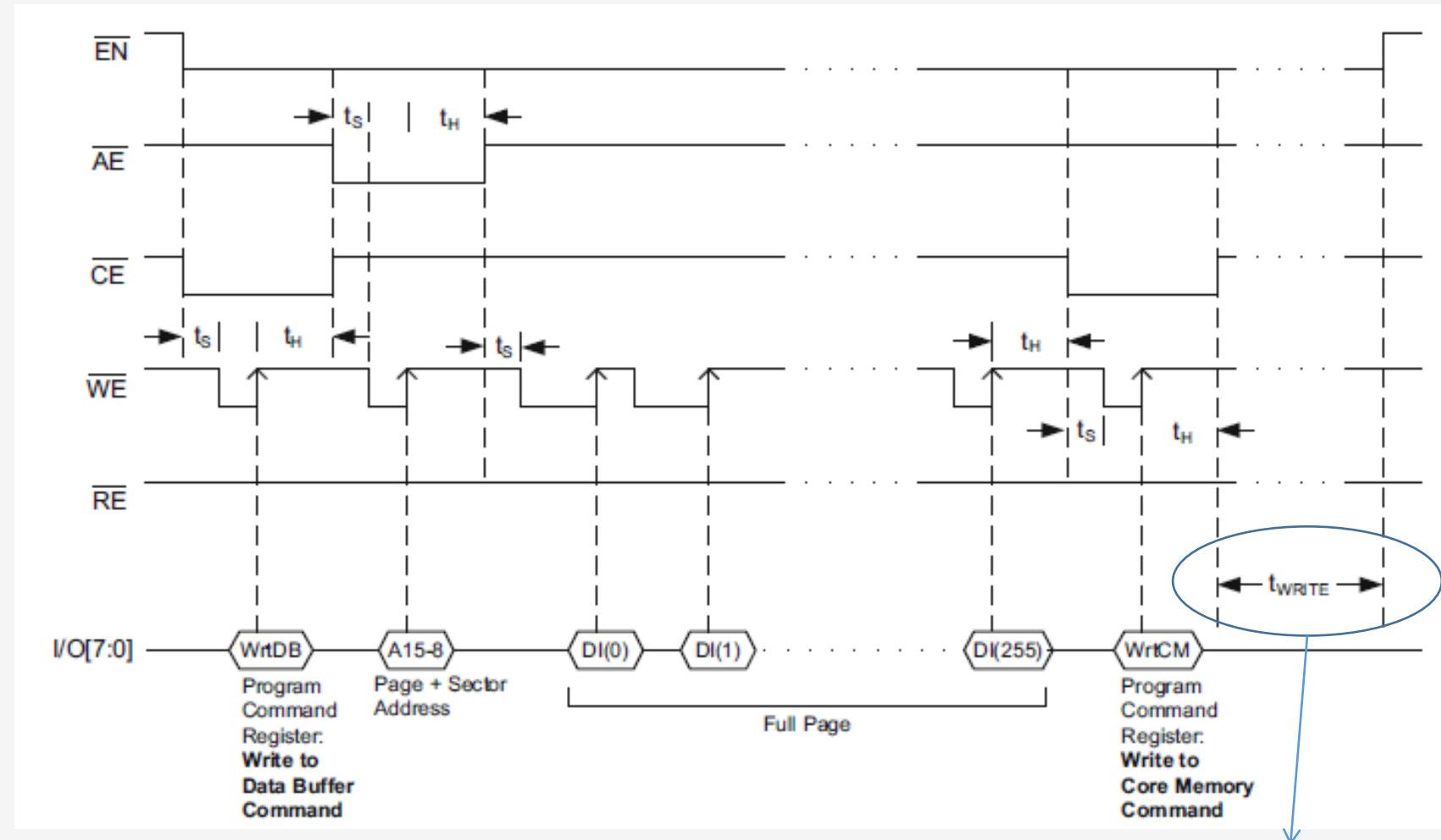
Four tasks:

Entering the Write to
Data Buffer
command

Entering the page
and sector addresses

Entering the full-page
of data into the data
buffer

Entering the Write to
Core Memory
command



The last cycle needs a relatively longer time period, t_{WRITE} .

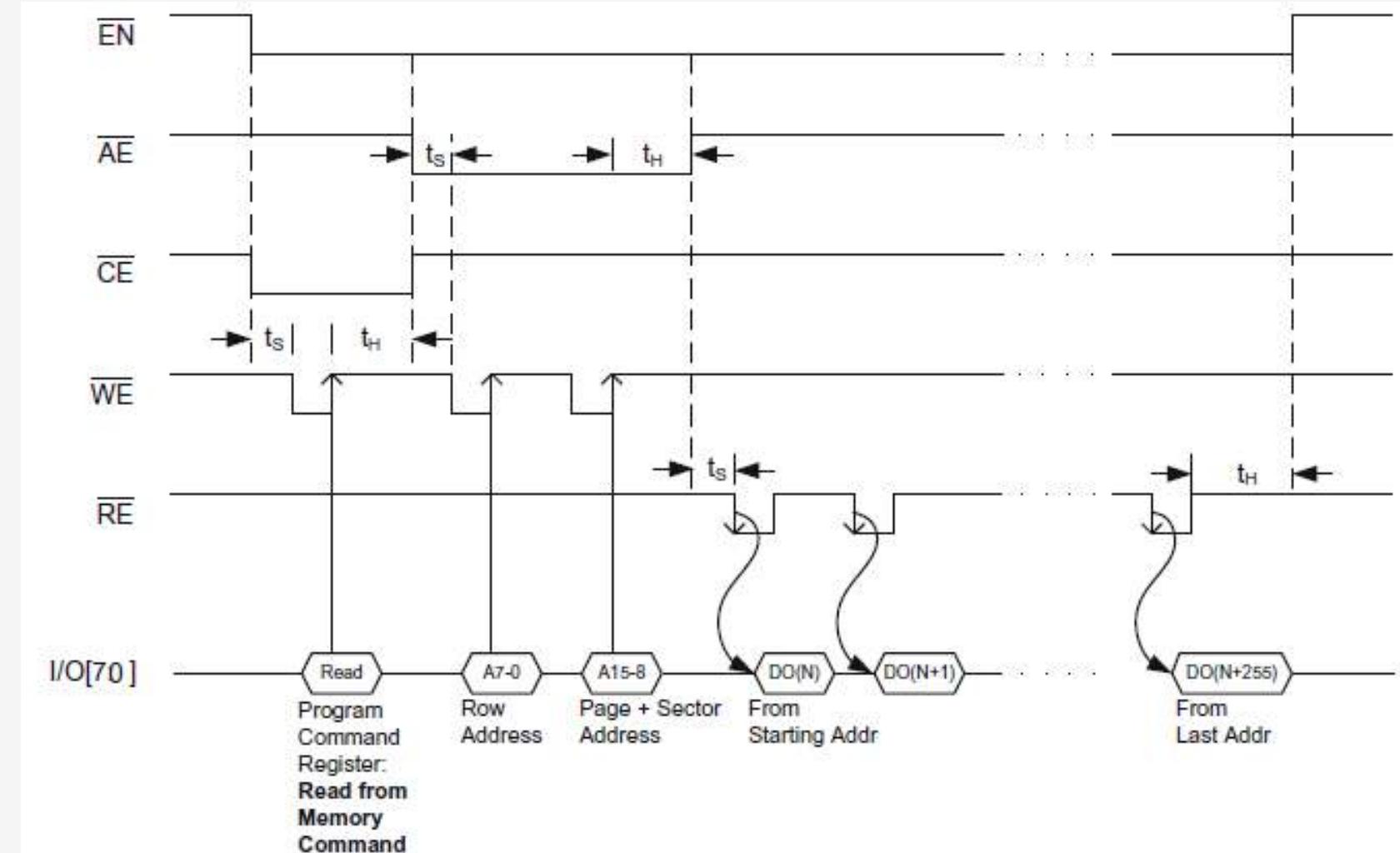
E²PROM - Full Page Read Timing Diagram

Three tasks:

Entering the Read from Memory command

Entering the starting address by specifying the row, page and sector address values

Reading data from the memory core



E²PROM - Full Page Erase Timing Diagram

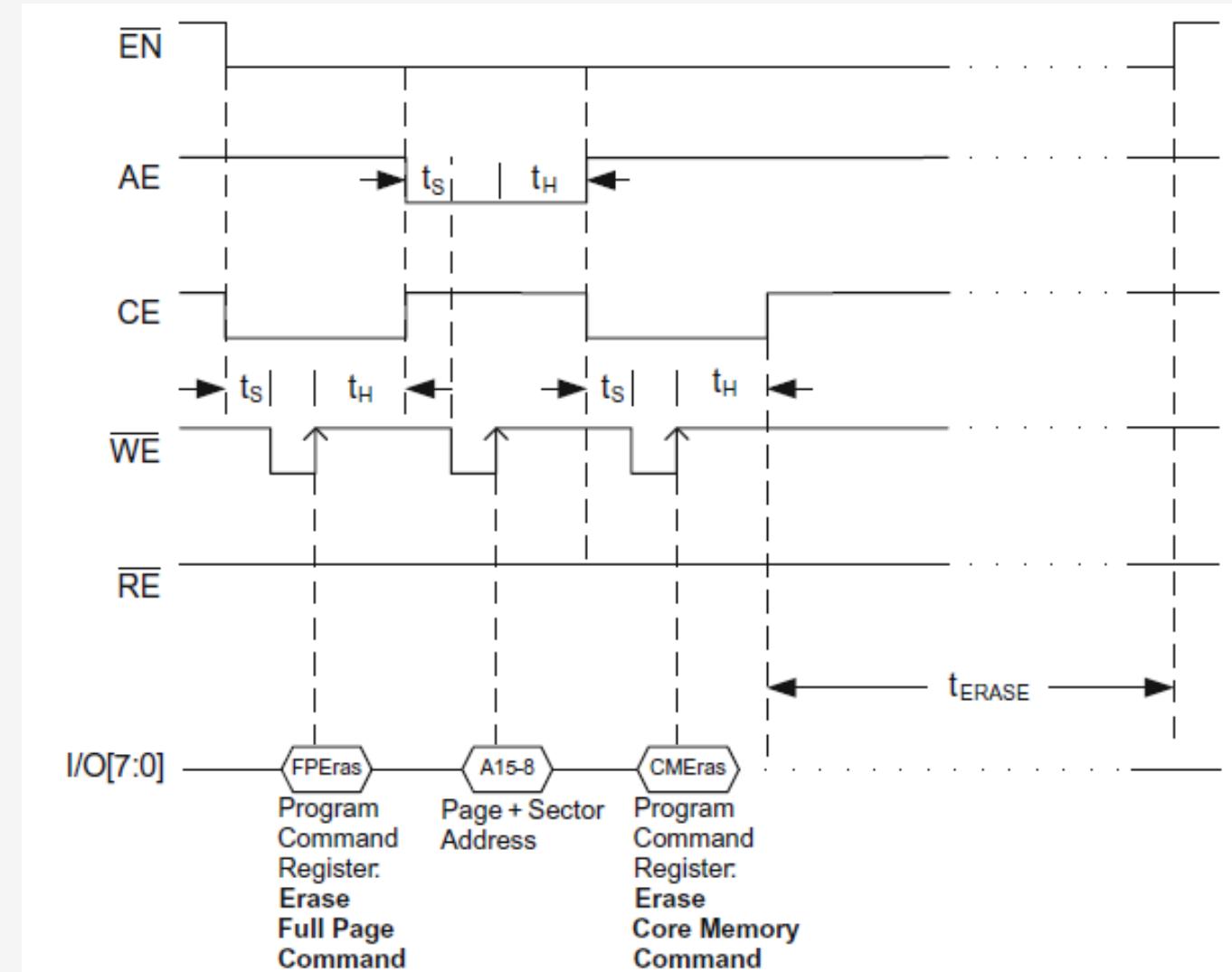
Three tasks:

Entering the Erase Full Page command

Entering the memory address by specifying page and sector address values

Entering the Erase Core Memory command

Why do we need erase for E²PROM?



Flash Memory

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

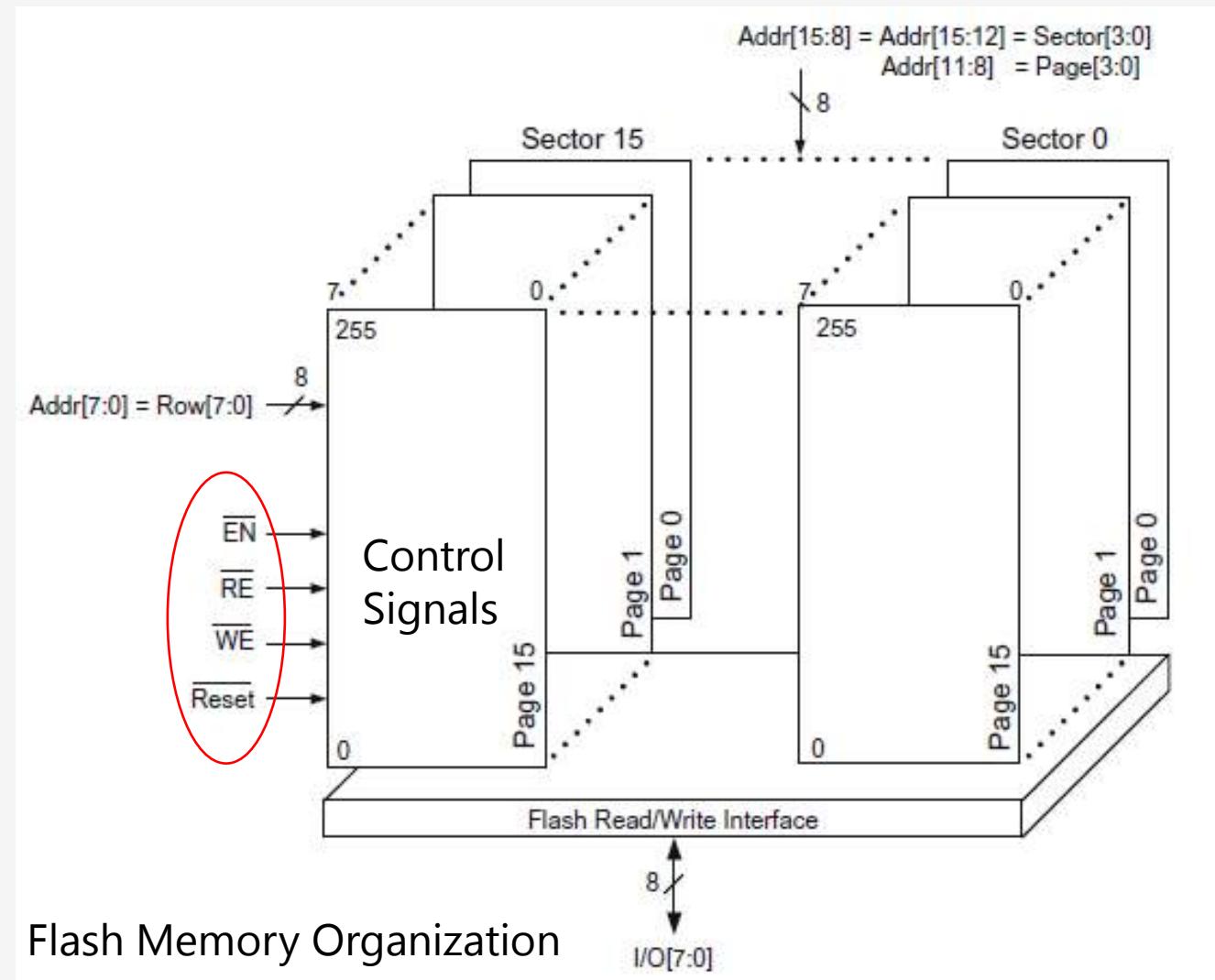
Flash Memory Organization

The active-low **Enable signal (EN)**, activates a particular page in the Flash memory to prepare it for an upcoming operation.

The active-low **Read Enable signal (RE)**, activates the Read/Write interface to read data from the memory.

The active-low **Write Enable signal (WE)**, enables to write data to the memory.

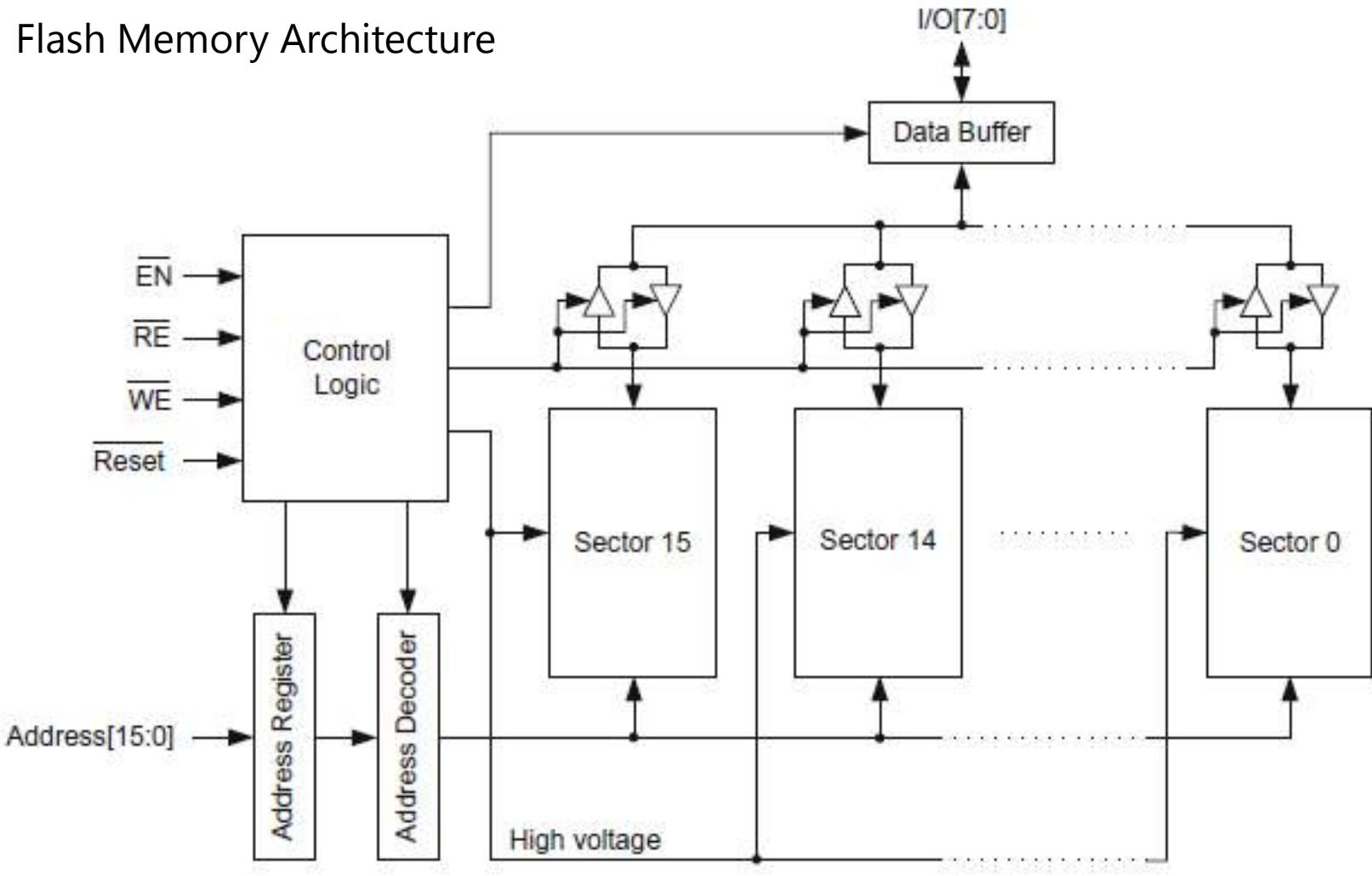
The active-low **Reset signal** is used for resetting the hardware. → After this command, Flash memory automatically goes into the read mode.



Flash Memory Architecture

When a memory operation starts, the control logic enables the **address decoder**, the **address register**, and the appropriate **data buffers** in order to activate the read or the write data-path.

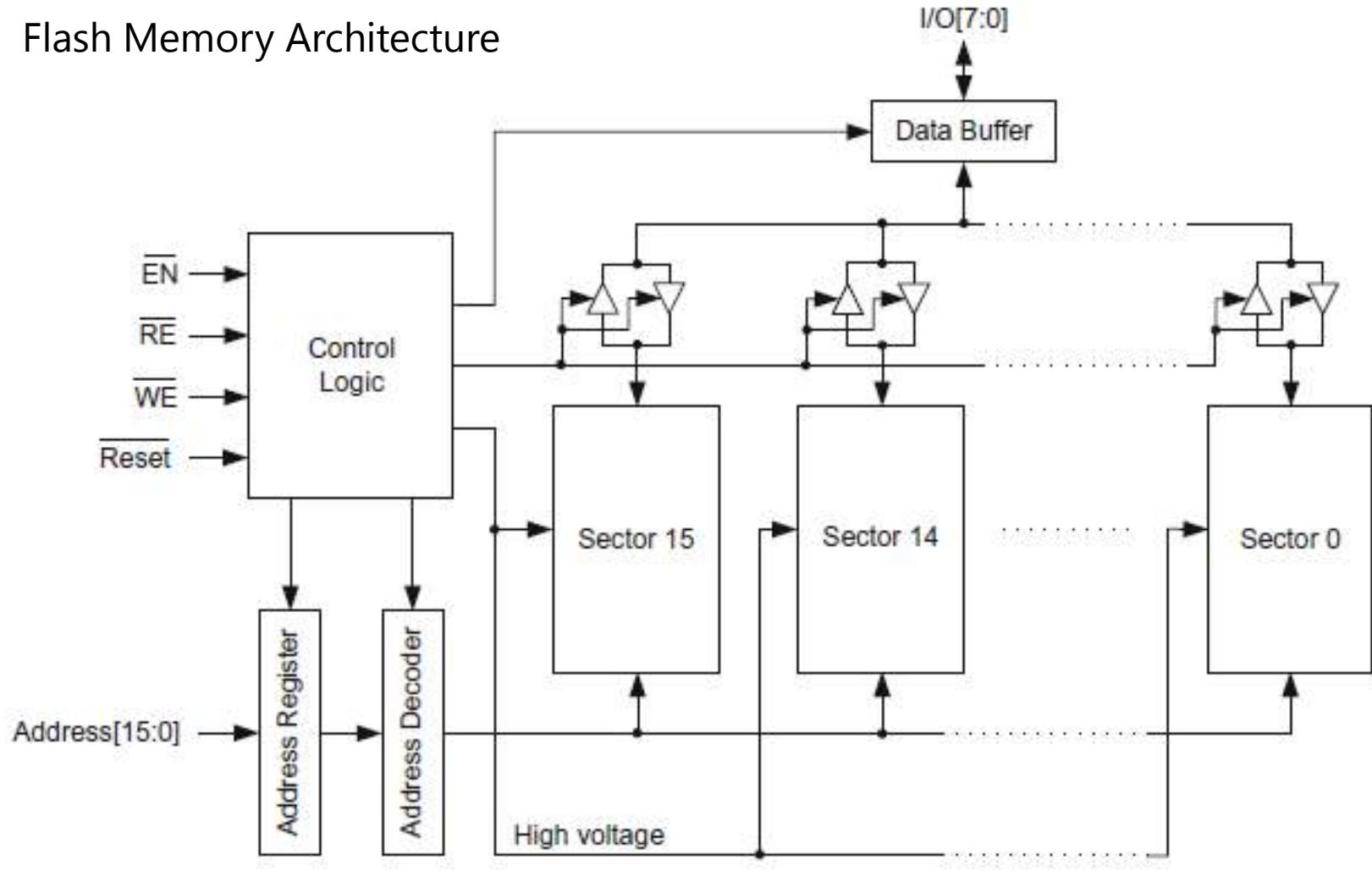
Flash Memory Architecture



Flash Memory Architecture

The address in the address register is decoded to point the location of data in the memory core.

Flash Memory Architecture

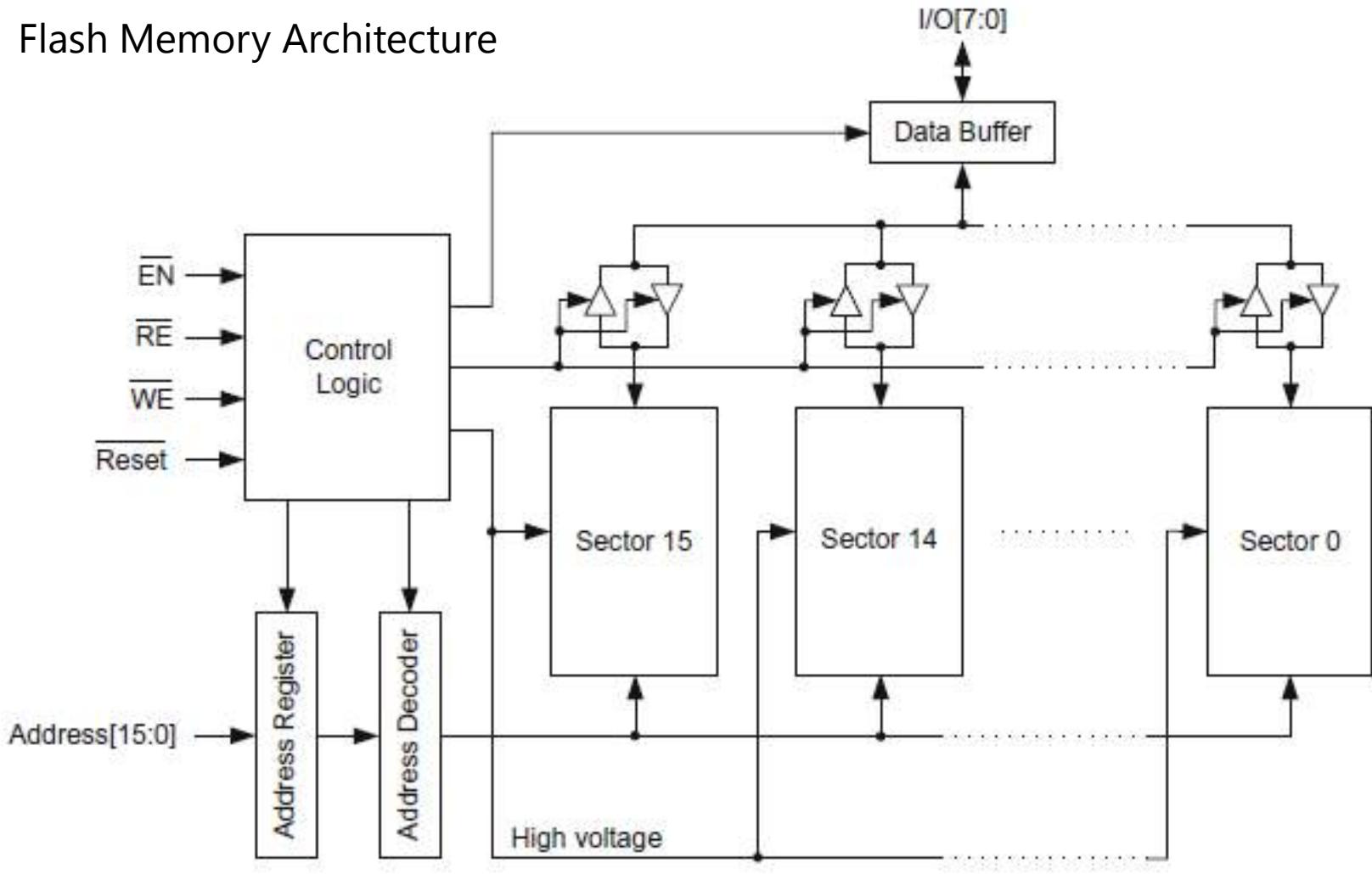


Flash Memory Architecture

If a read operation needs to be performed, the retrieved data is first stored in the data buffer, and then released to the bus.

If the operation calls for a write, the data is stored in the data buffer first, and then directed to the designated address in the memory core.

Flash Memory Architecture



Flash Memory Operation

$\overline{\text{EN}}$	$\overline{\text{RE}}$	$\overline{\text{WE}}$	reset	MODE
0	0	1	1	Read
0	1	0	1	Write
0	1	1	1	Standby
1	x	x	1	Hibernation
x	x	x	0	Hardware reset

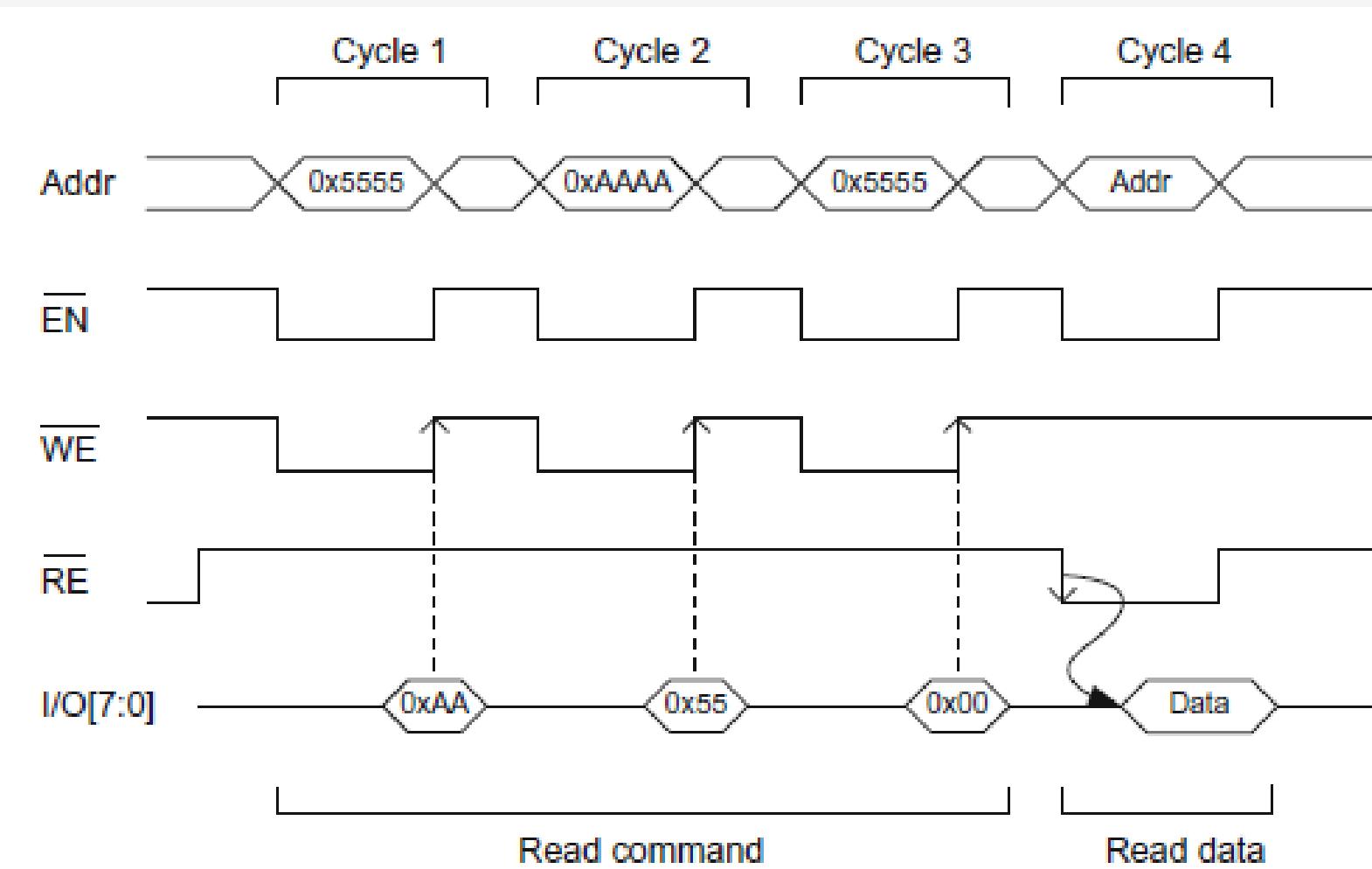
The standby mode neither writes to the memory nor reads from it.

The hibernation mode disables the address decoder, memory core and data buffer to reduce power dissipation.

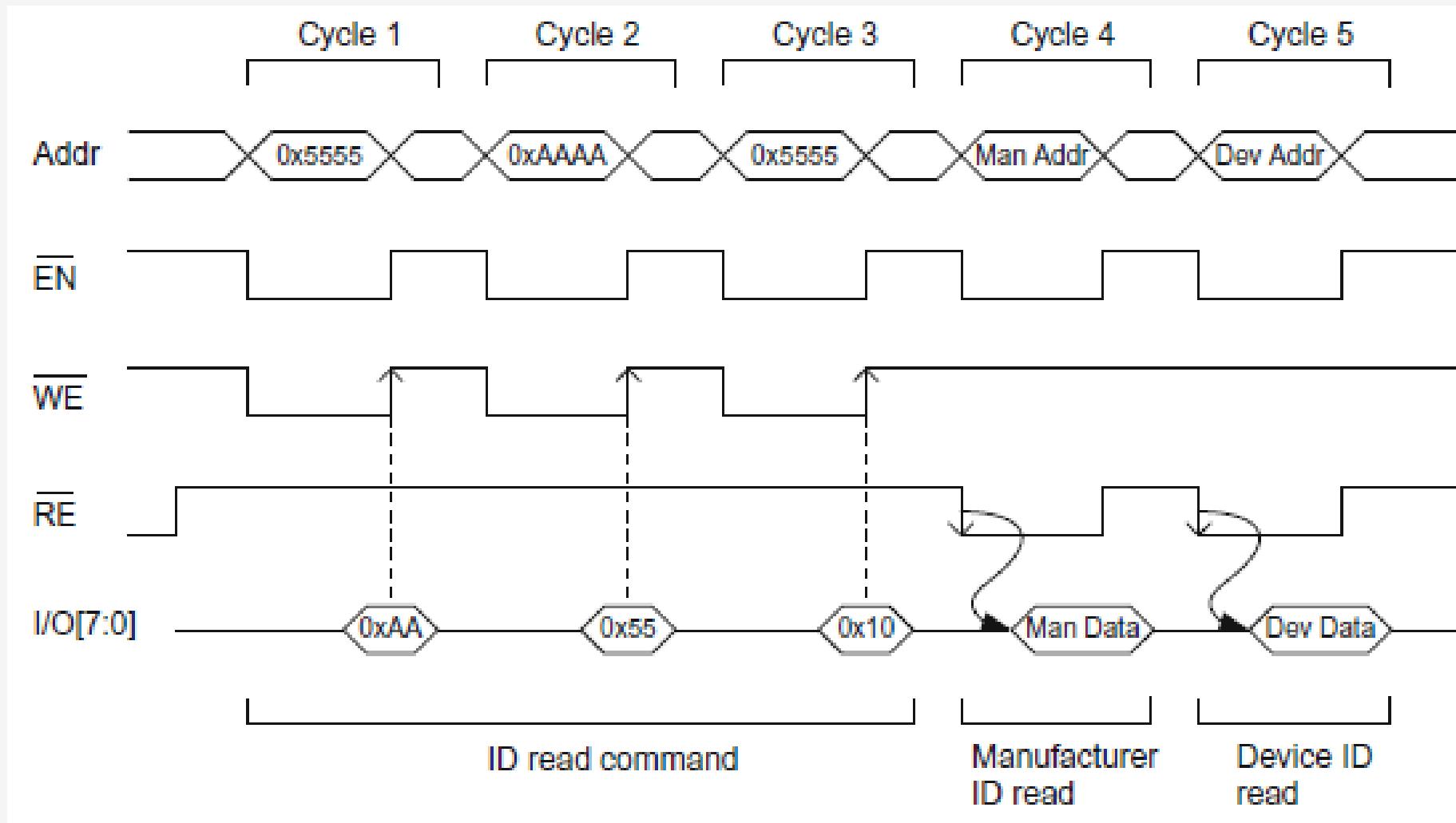
Flash Memory Commands

MAIN COMMANDS	CYCLE 1		CYCLE 2		CYCLE 3		CYCLE 4		CYCLE 5		CYCLE 6	
	Addr	Data	Addr	Data	Addr	Data	Addr	Data	Device Addr	Device Data		
Read	0x5555	0xAA	0xAAAA	0x55	0x5555	0x00	Read Addr	Read Data				
ID Read	0x5555	0xAA	0xAAAA	0x55	0x5555	0x10	Manuf. Addr	Manuf. Data	Device Addr	Device Data		
Write	0x5555	0xAA	0xAAAA	0x55	0x5555	0x20	Write Addr	Write Data				
Write suspend	Page Addr	0x30										
Write resume	Page Addr	0x40										
Chip erase	0x5555	0xAA	0xAAAA	0x55	0x5555	0x50	0x5555	0xAA	0xAAAA	0x55	0x5555	0x60
Page erase	0x5555	0xAA	0xAAAA	0x55	0x5555	0x50	0x5555	0xAA	0xAAAA	0x55	Page Addr	0x70
Page protect	Page Addr	0xA0	Page Addr	0xA0	Page Addr	0xA0	Page Addr	Verif. Code				
Reset	0x5555	0xAA	0xAAAA	0x55	0x5555	0xF0						

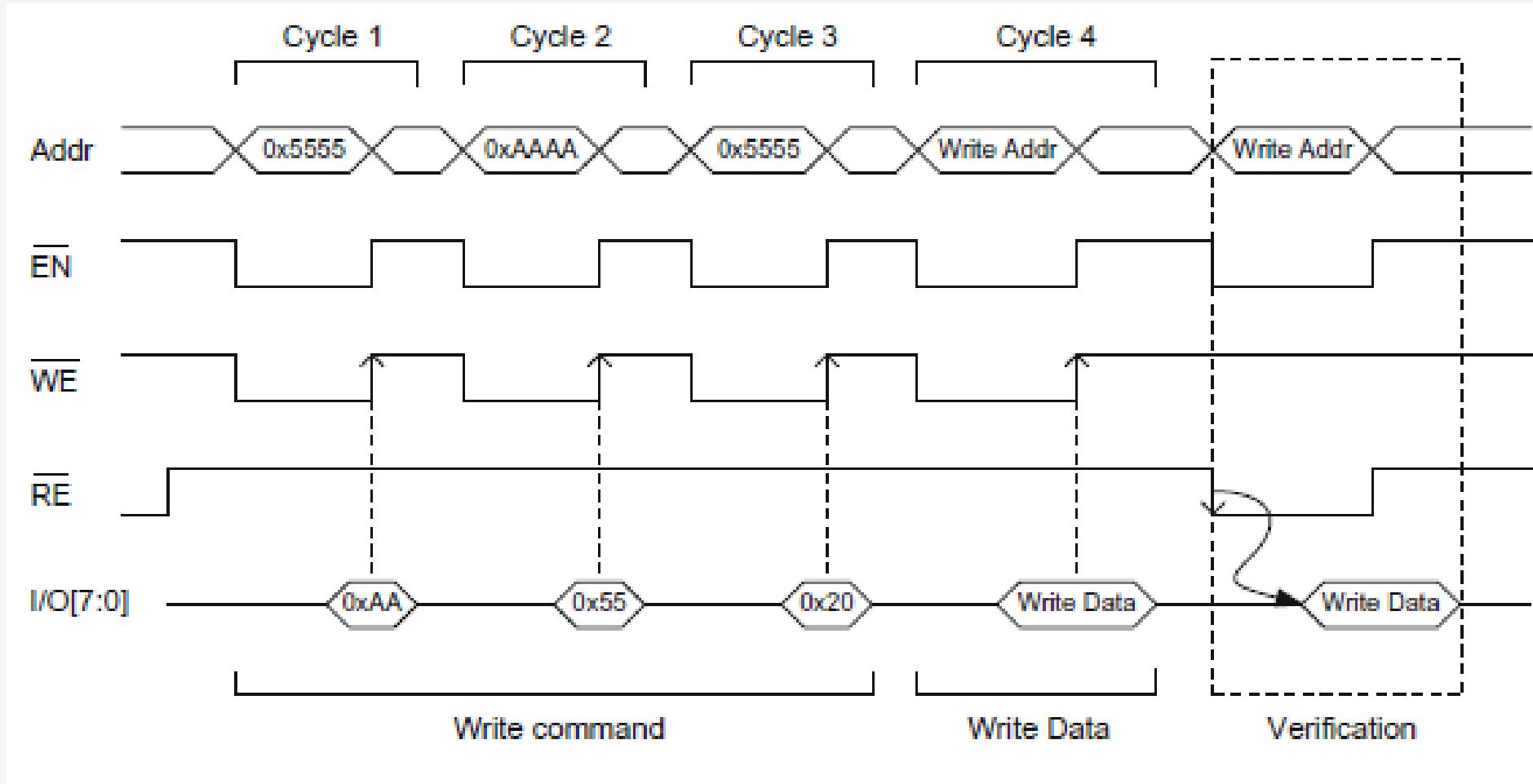
Flash Memory - Read Timing Diagram



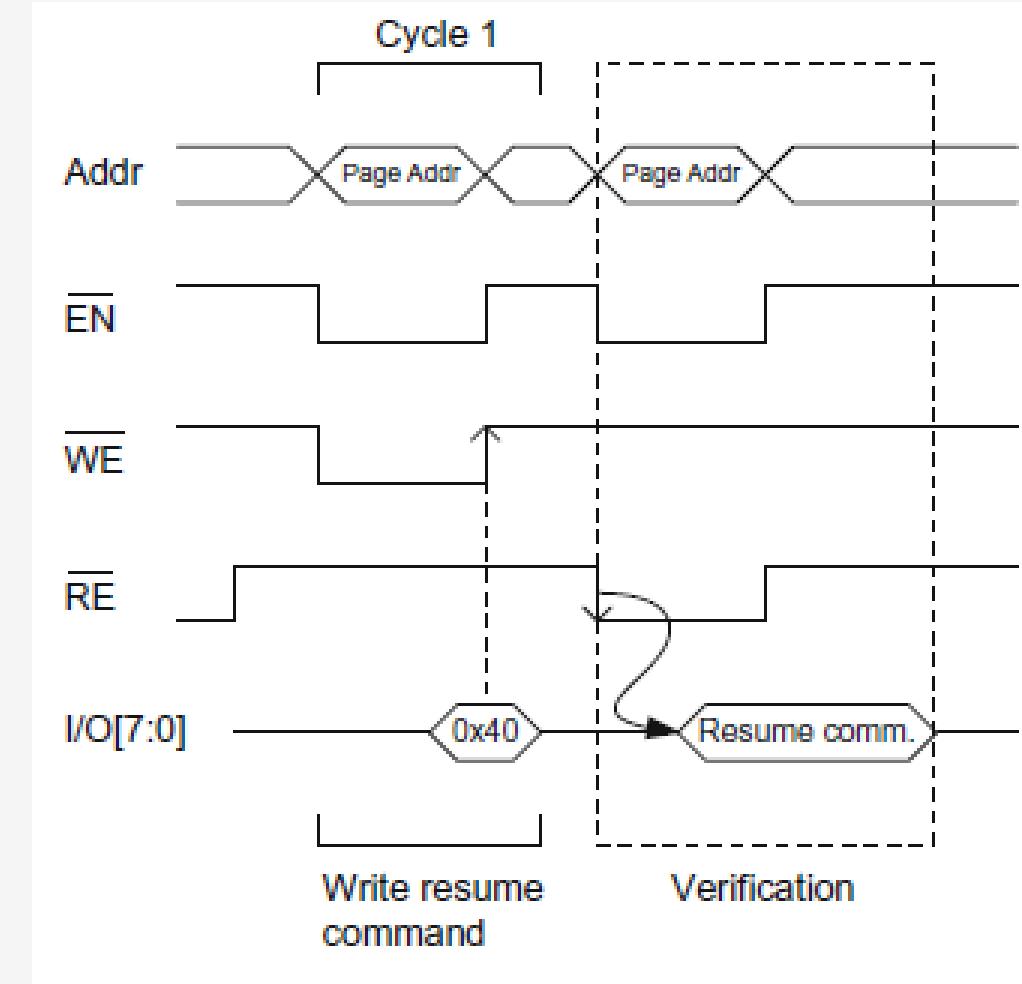
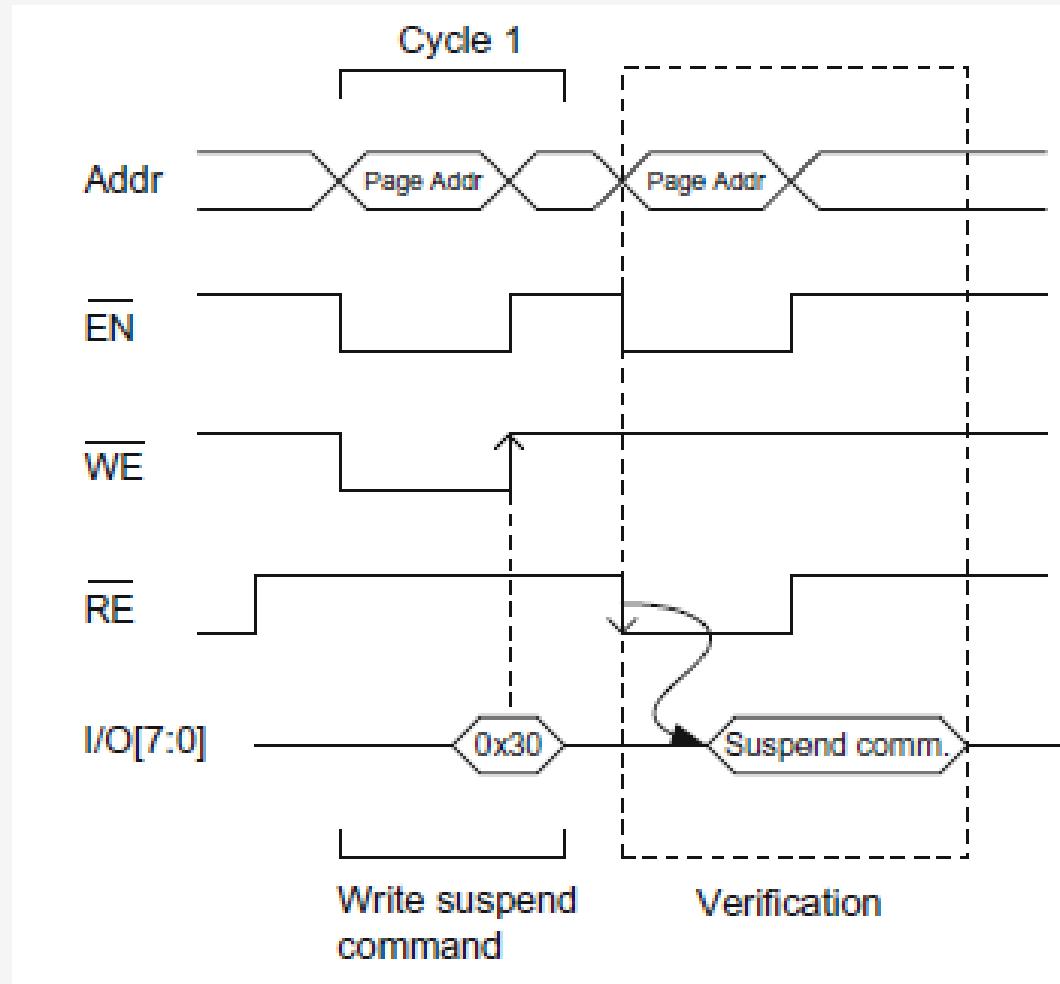
Flash Memory - ID Read Timing Diagram



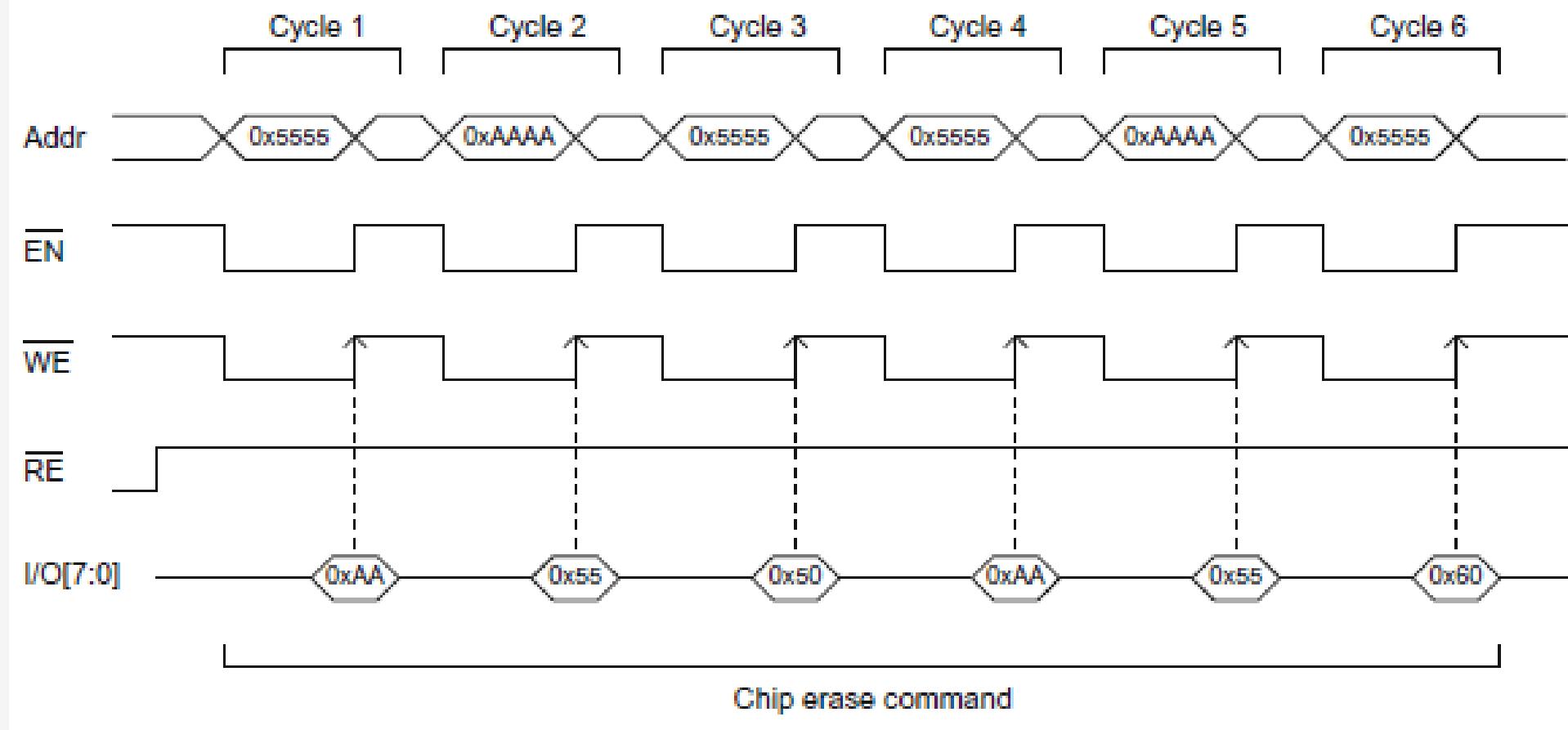
Flash Memory - Write Timing Diagram



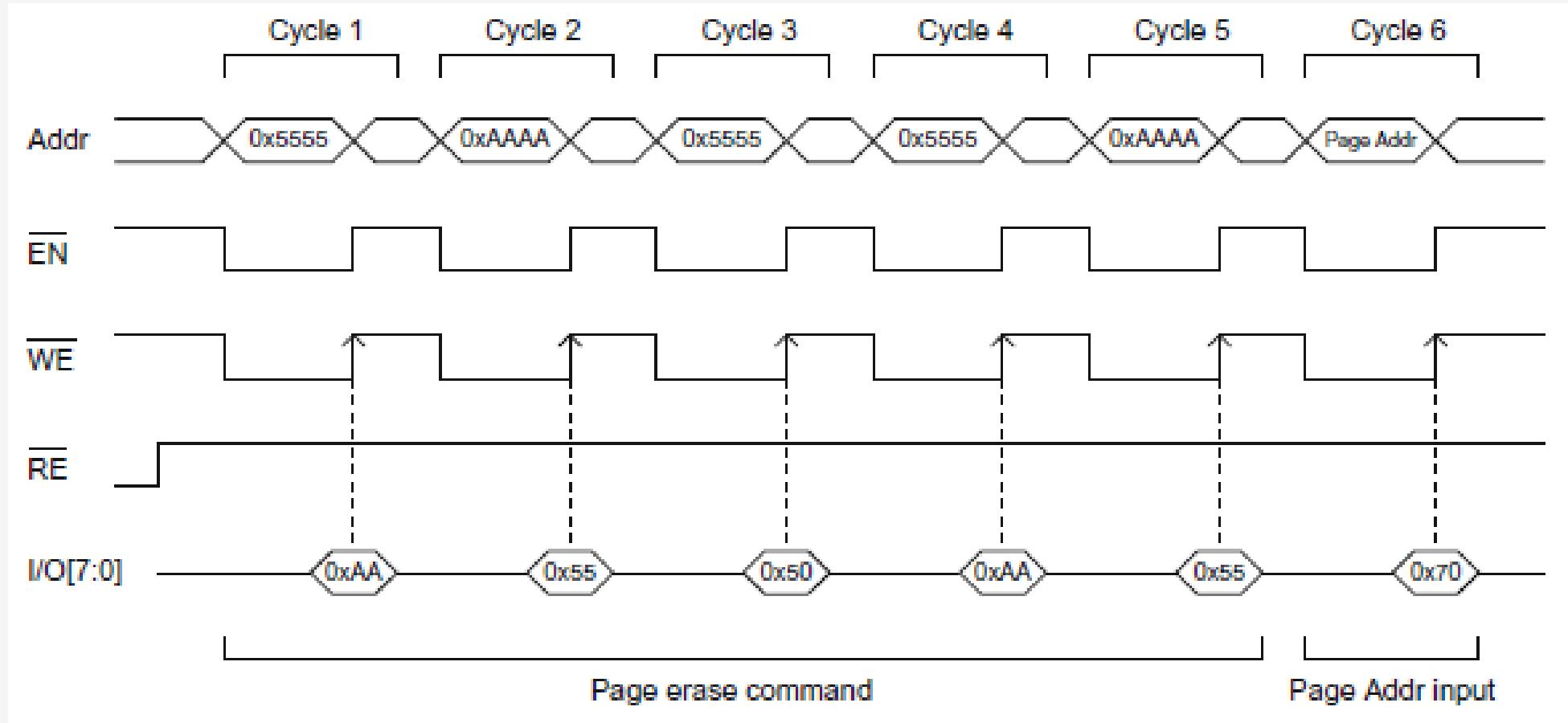
Flash Memory - Write Suspend & Resume Timing Diagrams



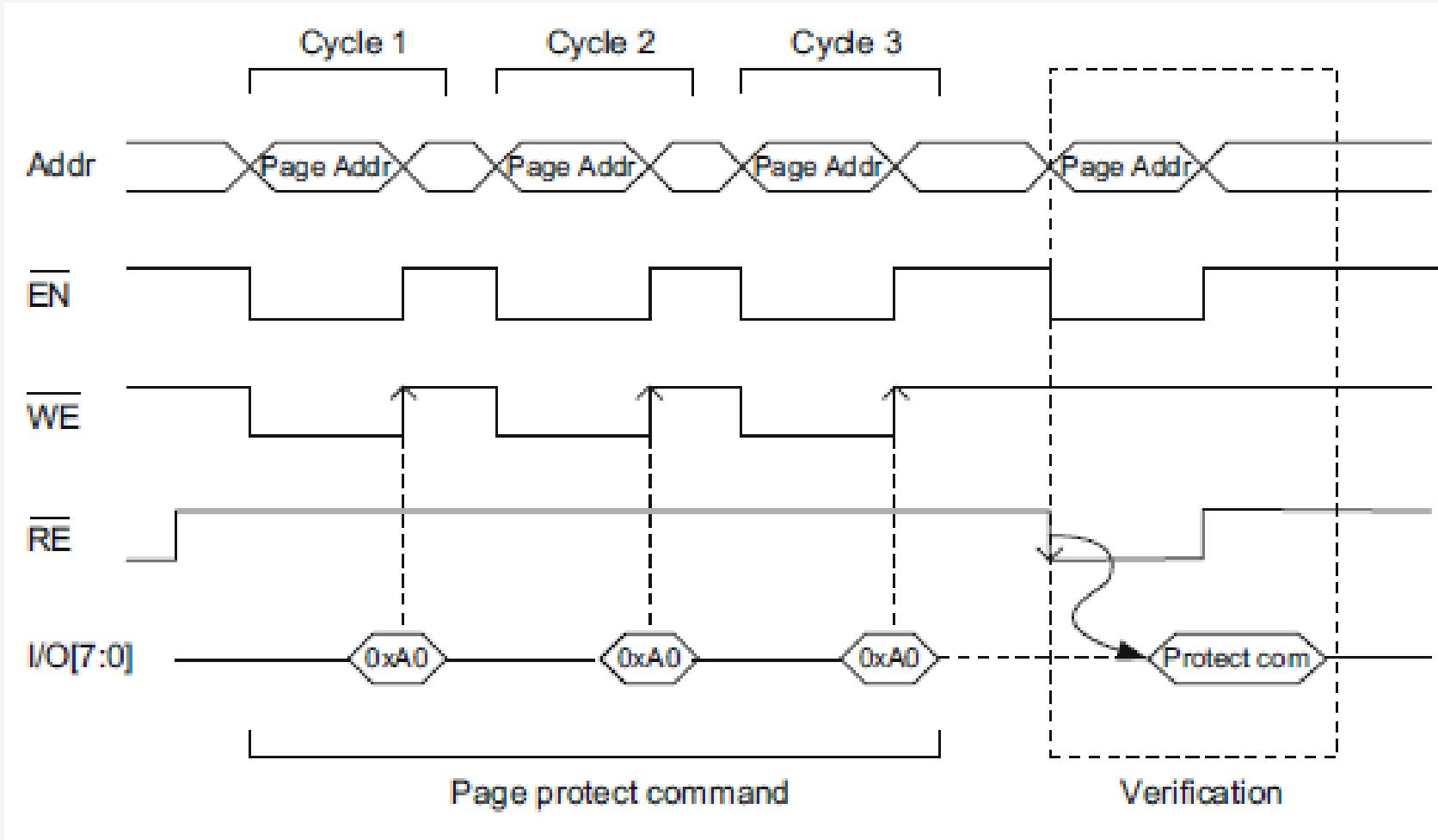
Flash Memory - Chip Erase Timing Diagram



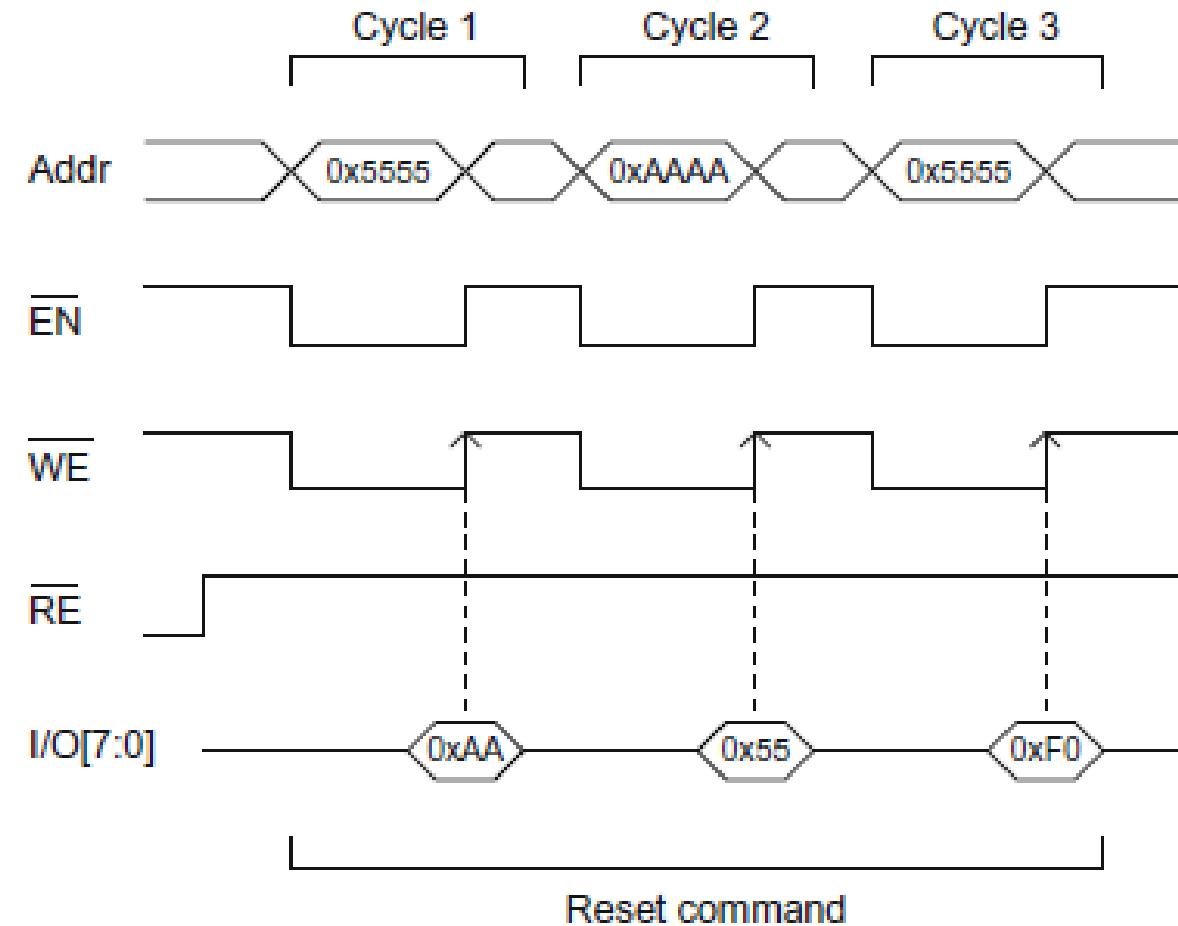
Flash Memory - Page Erase Timing Diagram



Flash Memory - Page Protect Timing Diagram



Flash Memory - Reset Timing Diagram



Laboratory Assignment 5

Due Date: May 4, 11:30PM on BeachBoard Dropbox

Late Submission Policy:

One day delay (Second Due Date: Apr. 28 , 11:30PM): 25% deduction

Two days delay (Third Due Date: Apr. 29, 11:30PM): 50% deduction

More than two days delay: No credit

Demo Time:

May. 7, 8:00AM - 10:45AM

May. 7, 1:00PM - 3:45PM

Academic Integrity: There is zero tolerance for cheating, plagiarism, or any other act of violation of Academic Integrity.

Programmable Logic Arrays

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Logic Circuit Design

Basic Components → Using basic universal building gates/blocks (NAND, NOR, MUX, etc) to design a logic circuit.

Programmable Logic Devices (PLD) → Using special integrated circuits that can be programmed to implement wide variety of designs.

Examples: Programmable Logic Arrays (PLA), Programmable Array Logic (PAL), Complex Programmable Logic Devices (CPLD), and Field Programmable Gate Arrays (FPGA)

Application Specific Integrated Circuits (ASIC) → Designing custom integrated circuits to target a specific market.

Examples: Full Custom, Standard Cells, and Gate Array

Basic Logic Array

The simplest way to implement an **n-input, m-output** logic circuit is to form the minterms using AND gates and then OR the appropriate minterms for formation of the outputs.

The circuit will have an array of AND gates and an array of OR gates, that are referred to as the AND-plane and the OR-plane respectively.

In the AND-plane, all the minterms are generated → 2^n AND gates.

The OR plane uses only the minterms that are needed for each output of the circuit → m OR gates.

Basic Logic Array - Sample Question

Design a basic logic array circuit for the given truth table.

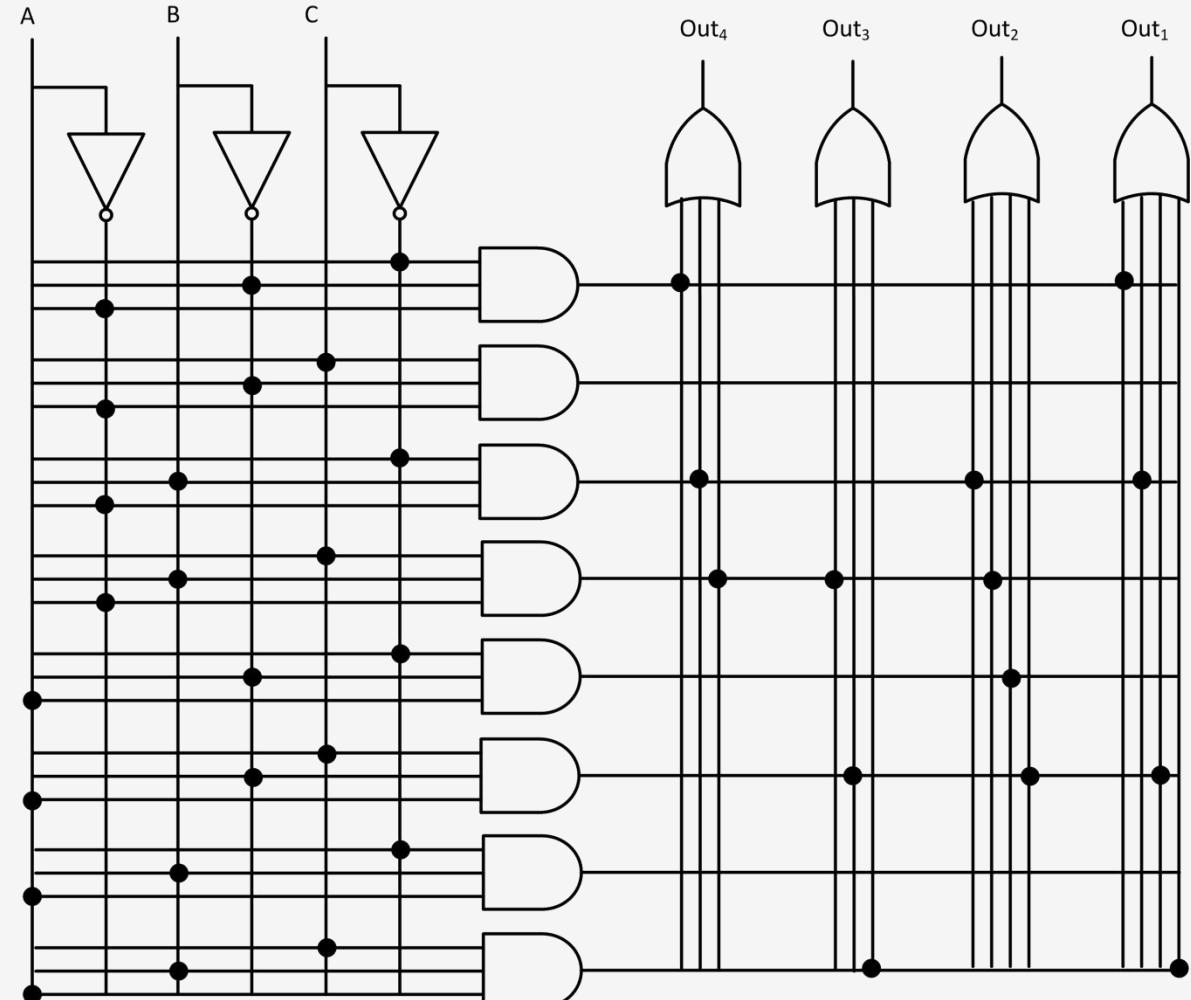
A	B	C	Out ₄	Out ₃	Out ₂	Out ₁
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	1	0	1	0	1	1
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	1	0	1	1	1
1	1	0	0	0	0	0
1	1	1	0	1	0	1

Basic Logic Array - Sample Question - Solution

Design a basic logic array circuit for the given truth table.

A	B	C	Out ₄	Out ₃	Out ₂	Out ₁
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	1	0	1	0	1	1
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	1	0	1	1	1
1	1	0	0	0	0	0
1	1	1	0	1	0	1

What is the impact of adding another input on basic logic array?



Programmable Logic Array (PLA)

PLA is a logic array with **programmable AND-planes** and **programmable OR-planes**.

PLA can be programmed to implement a small logic circuit.

PLA uses one-time programmable mask.

Product terms can be shared in PLA.

If we minimize our output functions (using K-map) and only implement the regained product terms, we will be able to save some rows of PLA.

Can we design sequential circuits with PLA?

PLA - Sample Question 1

Design a simplified PLA circuit for the given truth table.

A	B	C	Out ₄	Out ₃	Out ₂	Out ₁
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	1	0	1	0	1	1
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	1	0	1	1	1
1	1	0	0	0	0	0
1	1	1	0	1	0	1

PLA - Sample Question 1 - Solution

Design a simplified PLA circuit for the given truth table.

	BC=00	BC=01	BC=11	BC=10
A=0	1	0	1	1
A=1	0	0	0	0

$$\text{Out}_4 = \bar{A}B + \bar{A}\bar{C}$$

	BC=00	BC=01	BC=11	BC=10
A=0	0	0	1	1
A=1	1	1	0	0

$$\text{Out}_2 = A\bar{B} + \bar{A}B$$

	BC=00	BC=01	BC=11	BC=10
A=0	0	0	1	0
A=1	0	1	1	0

$$\text{Out}_3 = AC + BC$$

	BC=00	BC=01	BC=11	BC=10
A=0	1	0	0	1
A=1	0	1	1	0

$$\text{Out}_1 = AC + \bar{A}\bar{C}$$

PLA - Sample Question 1 - Solution

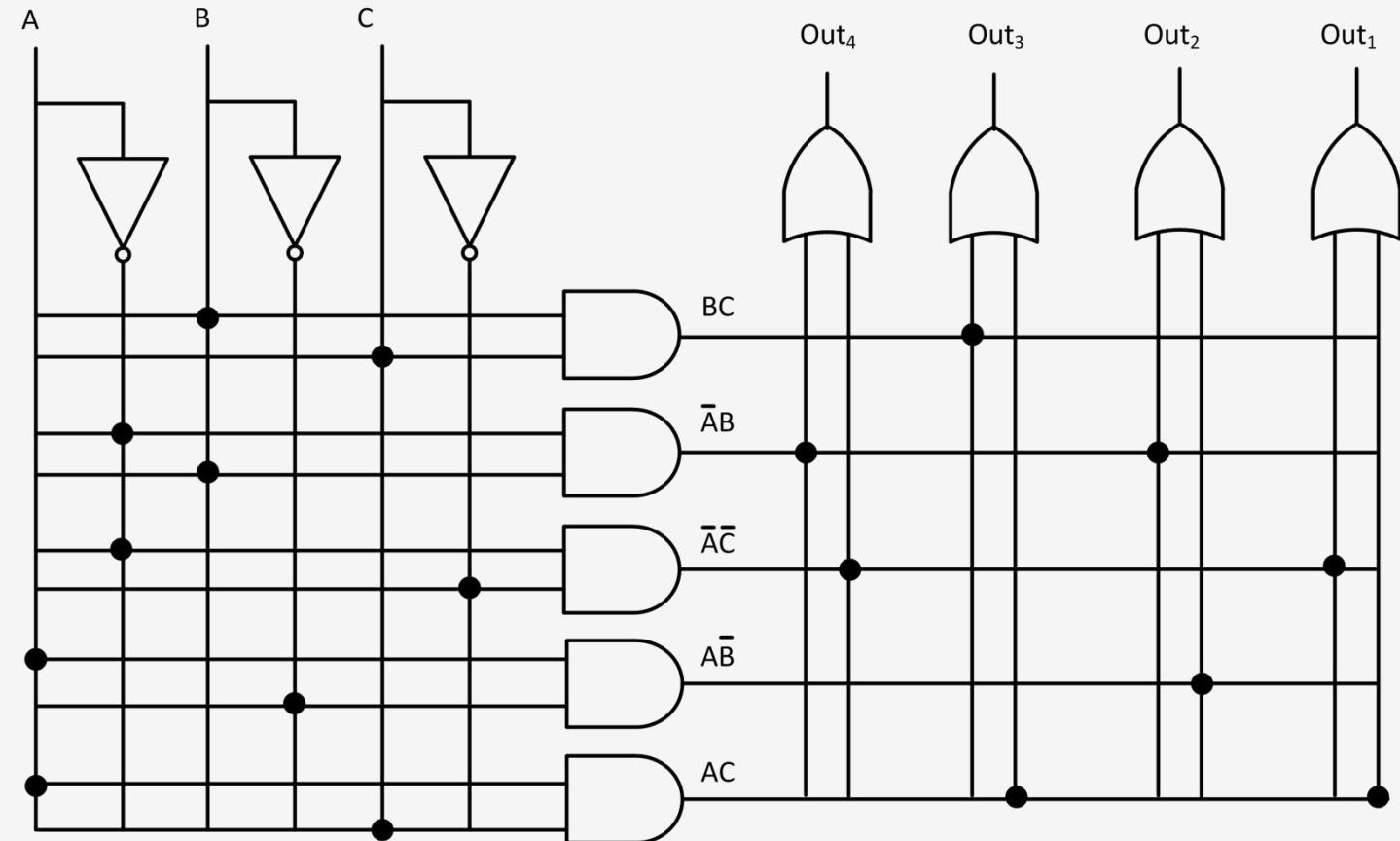
Design a simplified PLA circuit for the given truth table.

$$\text{Out}_4 = \bar{A}B + \bar{A}\bar{C}$$

$$\text{Out}_3 = AC + BC$$

$$\text{Out}_2 = A\bar{B} + \bar{A}B$$

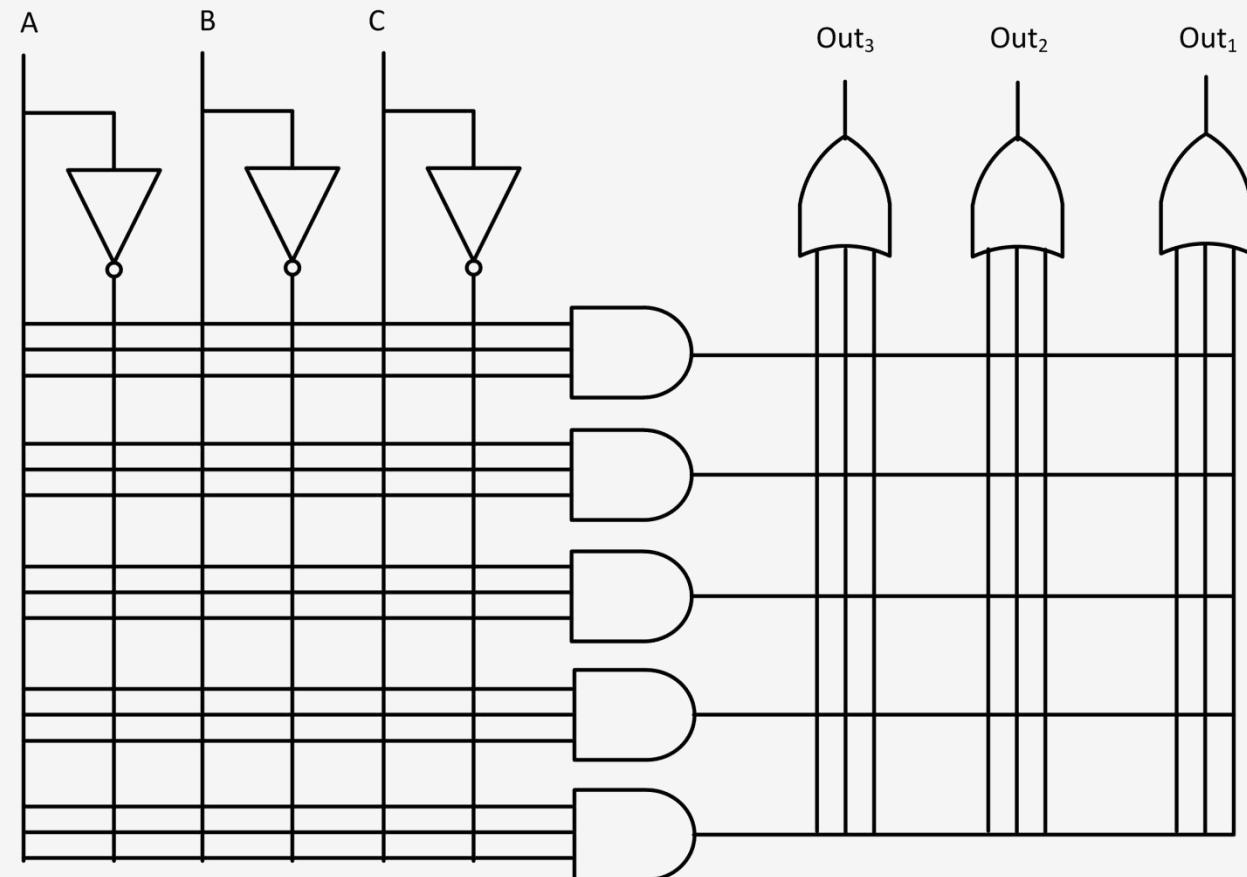
$$\text{Out}_1 = AC + \bar{A}\bar{C}$$



PLA - Sample Question 2

Program the following PLA based on the given truth table.

A	B	C	Out ₃	Out ₂	Out ₁
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	0	0	0



PLA - Sample Question 2 - Solution

Program the following PLA based on the given truth table.

A	B	C	Out ₃	Out ₂	Out ₁
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	0	0	0

	BC=00	BC=01	BC=11	BC=10
A=0	1	1	1	1
A=1	1	1	0	0

$$\text{Out}_3 = \bar{A} + \bar{B}$$

	BC=00	BC=01	BC=11	BC=10
A=0	1	1	0	0
A=1	1	1	0	1

$$\text{Out}_2 = \bar{B} + A\bar{C}$$

	BC=00	BC=01	BC=11	BC=10
A=0	0	1	1	0
A=1	1	0	0	1

$$\text{Out}_1 = \bar{A}\bar{C} + A\bar{C}$$

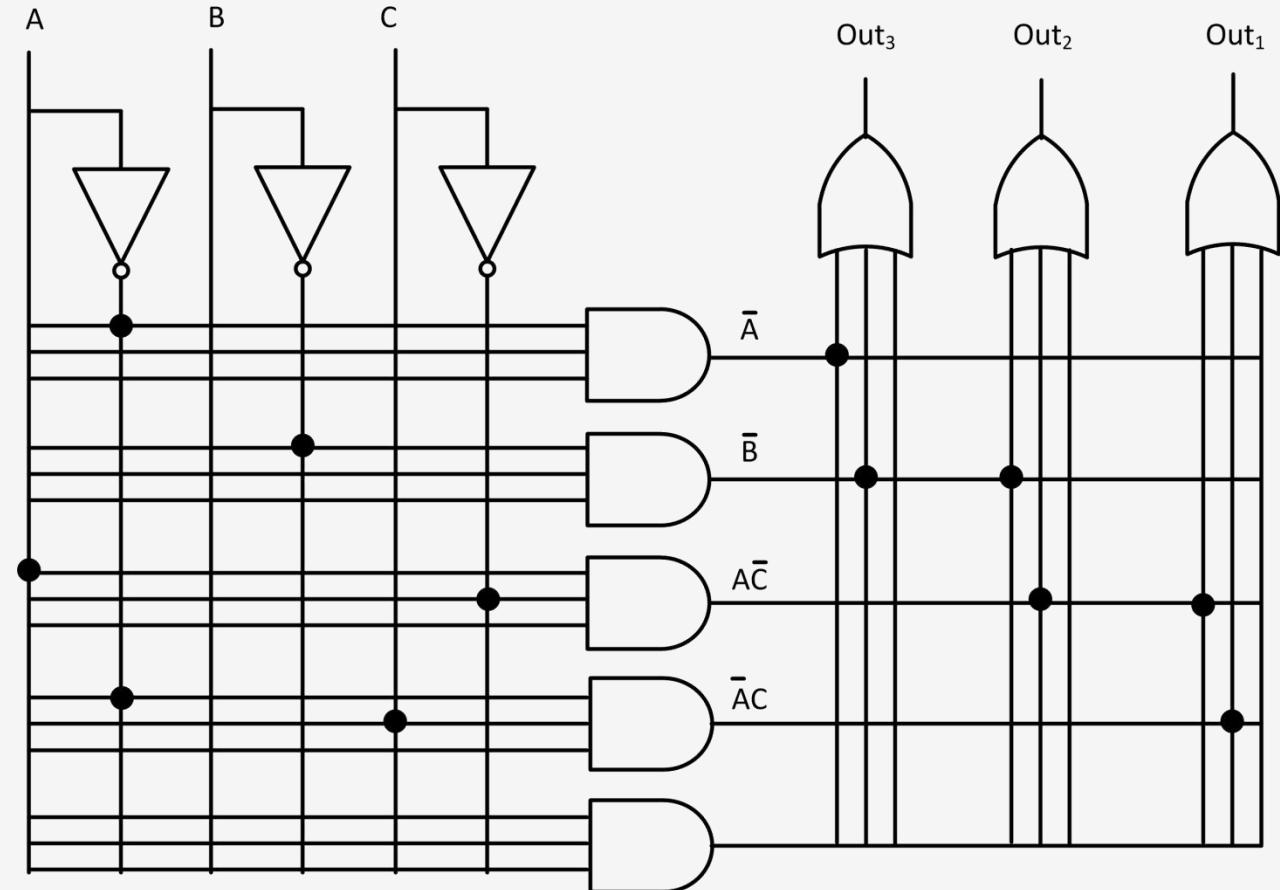
PLA - Sample Question 2 - Solution

Program the following PLA based on the given truth table.

$$\text{Out}_3 = \bar{A} + \bar{B}$$

$$\text{Out}_2 = \bar{B} + A\bar{C}$$

$$\text{Out}_1 = \bar{A}C + A\bar{C}$$



Programmable Array Logic

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Programmable Array Logic (PAL)

PAL is a logic array with **programmable AND-planes** and **fixed OR-planes**.

PAL can be programmed to implement a small logic circuit.

PAL uses one-time programmable fuse links.

There is no product term sharing in PAL.

Can we design sequential circuits with PAL?

PAL - Sample Question 1

Design a simplified PAL circuit for the given truth table.

A	B	C	Out ₄	Out ₃	Out ₂	Out ₁
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	1	0	1	0	1	1
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	1	0	1	1	1
1	1	0	0	0	0	0
1	1	1	0	1	0	1

PAL - Sample Question 1 - Solution

Design a simplified PAL circuit for the given truth table.

	BC=00	BC=01	BC=11	BC=10
A=0	1	0	1	1
A=1	0	0	0	0

$$\text{Out}_4 = \bar{A}B + \bar{A}\bar{C}$$

	BC=00	BC=01	BC=11	BC=10
A=0	0	0	1	1
A=1	1	1	0	0

$$\text{Out}_2 = A\bar{B} + \bar{A}B$$

	BC=00	BC=01	BC=11	BC=10
A=0	0	0	1	0
A=1	0	1	1	0

$$\text{Out}_3 = AC + BC$$

	BC=00	BC=01	BC=11	BC=10
A=0	1	0	0	1
A=1	0	1	1	0

$$\text{Out}_1 = AC + \bar{A}\bar{C}$$

PAL - Sample Question 1 - Solution

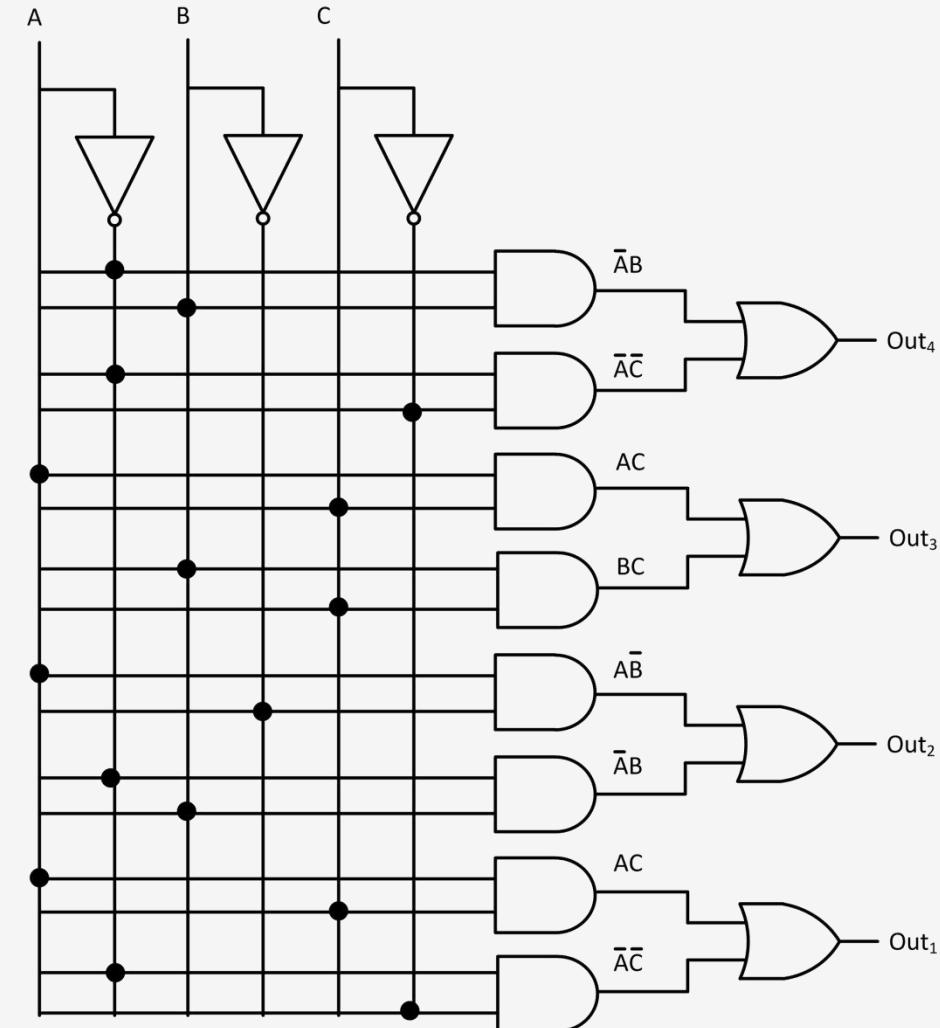
Design a simplified PAL circuit for the given truth table.

$$Out_4 = \bar{A}B + \bar{A}\bar{C}$$

$$Out_3 = AC + BC$$

$$Out_2 = A\bar{B} + \bar{A}B$$

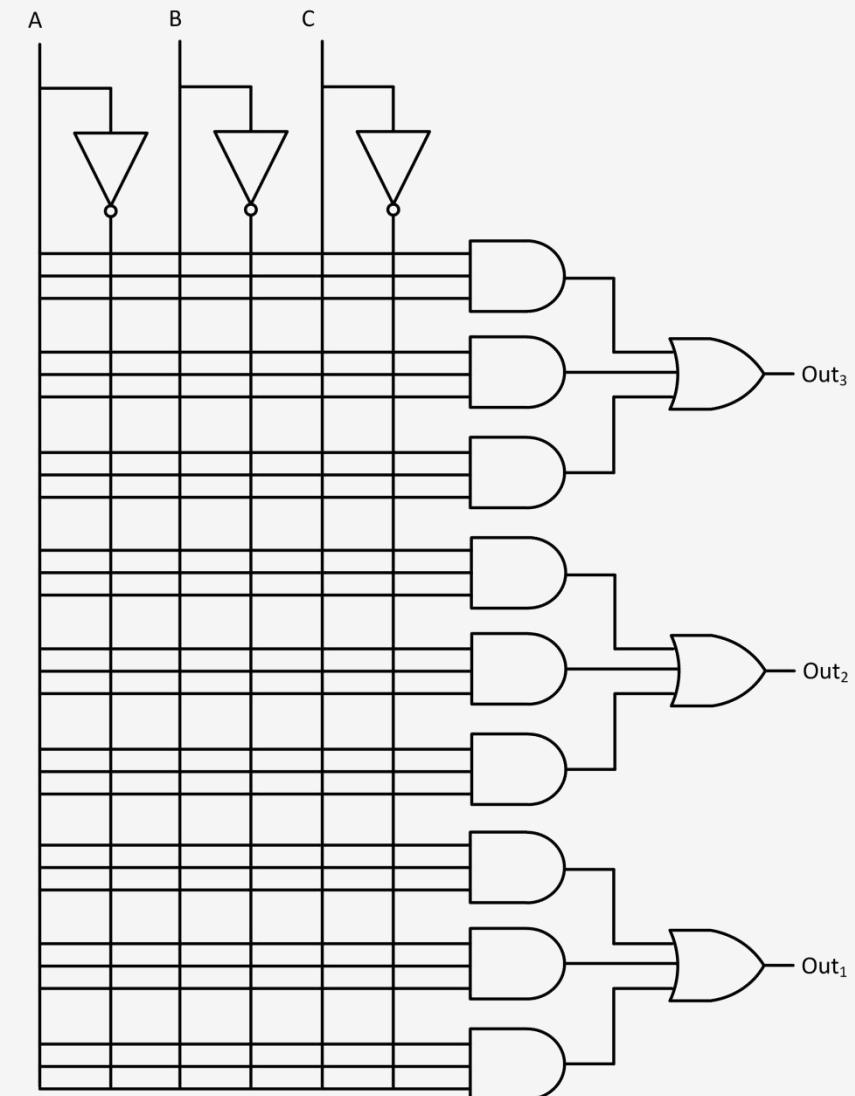
$$Out_1 = AC + \bar{A}\bar{C}$$



PAL - Sample Question 2

Program the following PAL based on the given truth table.

A	B	C	Out ₃	Out ₂	Out ₁
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	0	0	0



PAL - Sample Question 2 - Solution

Program the following PAL based on the given truth table.

A	B	C	Out ₃	Out ₂	Out ₁
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	0	0	0

	BC=00	BC=01	BC=11	BC=10
A=0	1	1	1	1
A=1	1	1	0	0

$$\text{Out}_3 = \bar{A} + \bar{B}$$

	BC=00	BC=01	BC=11	BC=10
A=0	1	1	0	0
A=1	1	1	0	1

$$\text{Out}_2 = \bar{B} + A\bar{C}$$

	BC=00	BC=01	BC=11	BC=10
A=0	0	1	1	0
A=1	1	0	0	1

$$\text{Out}_1 = \bar{A}\bar{C} + A\bar{C}$$

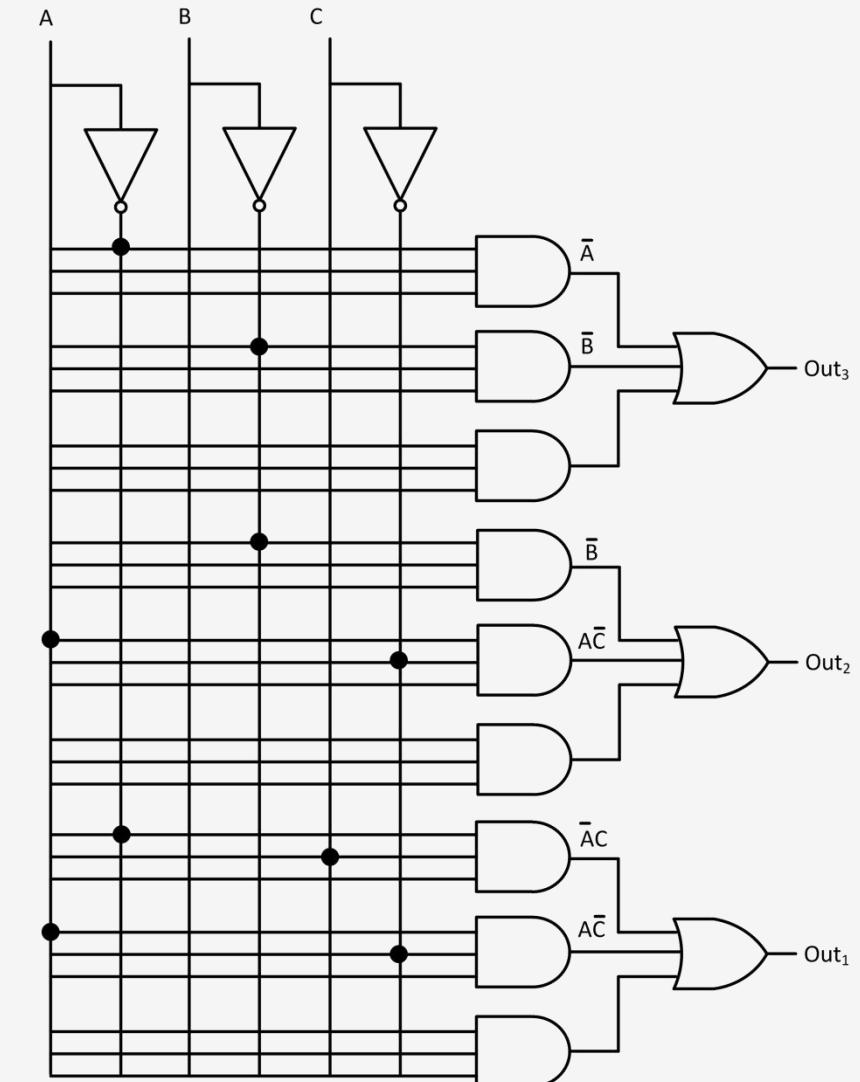
PAL - Sample Question 2 - Solution

Program the following PAL based on the given truth table.

$$\text{Out}_3 = \bar{A} + \bar{B}$$

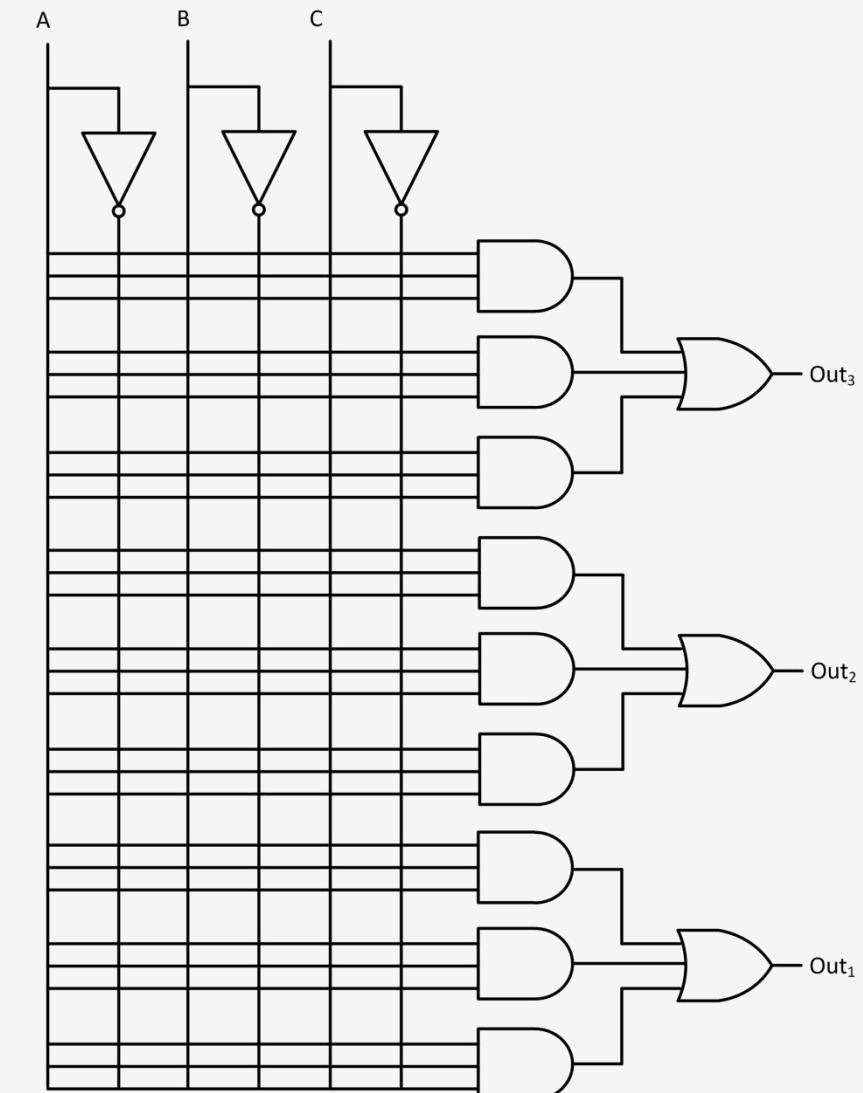
$$\text{Out}_2 = \bar{B} + A\bar{C}$$

$$\text{Out}_1 = \bar{A}C + A\bar{C}$$



PAL - Sample Question 3

Can the following PAL implement all the logic circuits with at most three inputs and at most three outputs?



PAL - Sample Question 3 - Solution

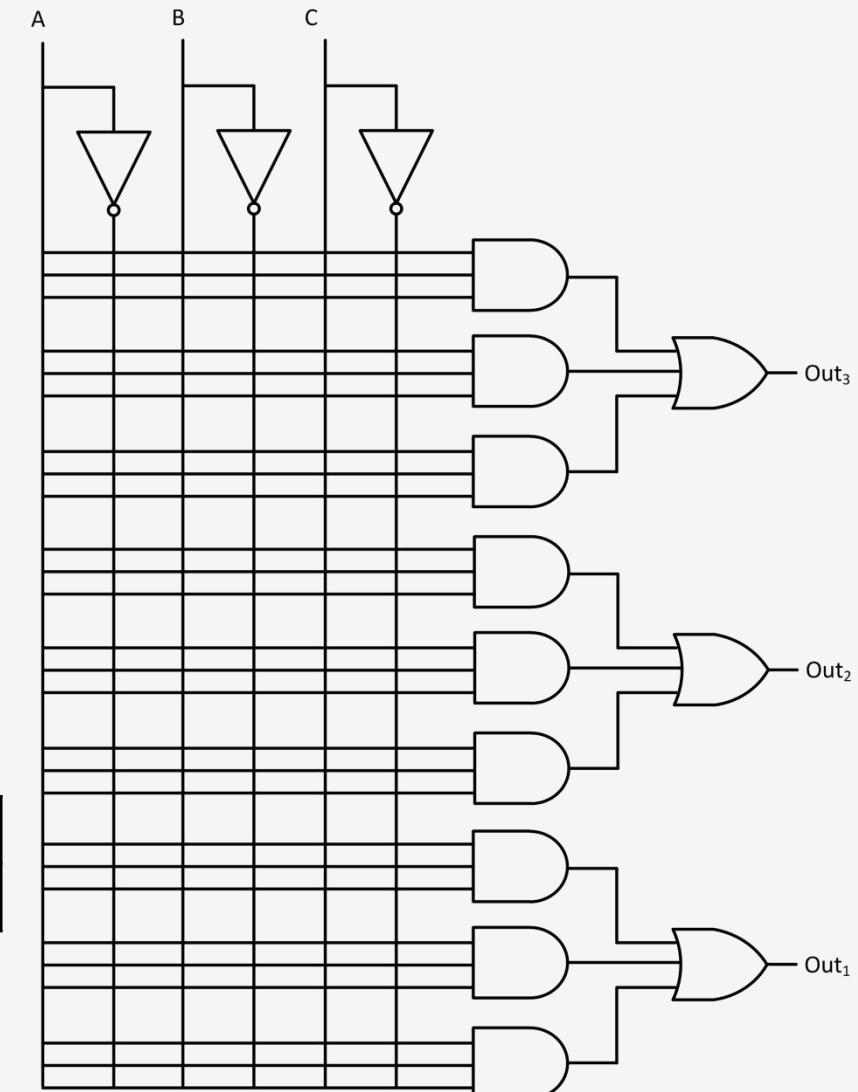
Can the following PAL implement all the logic circuits with at most three inputs and at most three outputs?

A	B	C	Out
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

No. Here is a counter example that requires 4-input OR-planes to be implemented.

	BC=00	BC=01	BC=11	BC=10
A=0	0	1	0	1
A=1	1	0	1	0

$$\text{Out} = A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$$



Programmable Logic Devices - Sample Question

Implement a logic circuit that implements a 2-input OR gate, a 2-input NAND gate, and a 2-input XOR gate using the following architectures:

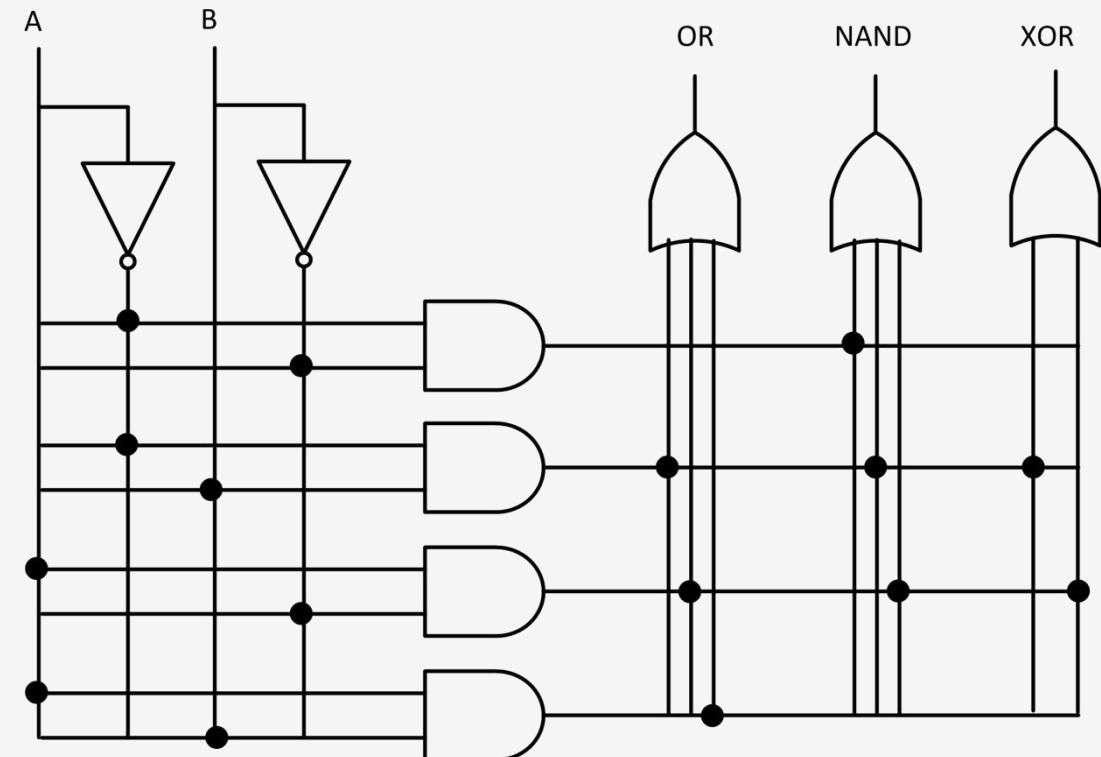
- 1) Basic logic array
- 2) PLA with 2-input AND-planes and 2-input OR-planes
- 3) PAL with 2-input AND-planes and 2-input OR-planes

Programmable Logic Devices - Sample Question - Solution

Implement a logic circuit that implements a 2-input OR gate, a 2-input NAND gate, and a 2-input XOR gate using the following architectures:

1) Basic logic array architecture

A	B	OR	NAND	XOR
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0



Programmable Logic Devices - Sample Question - Solution

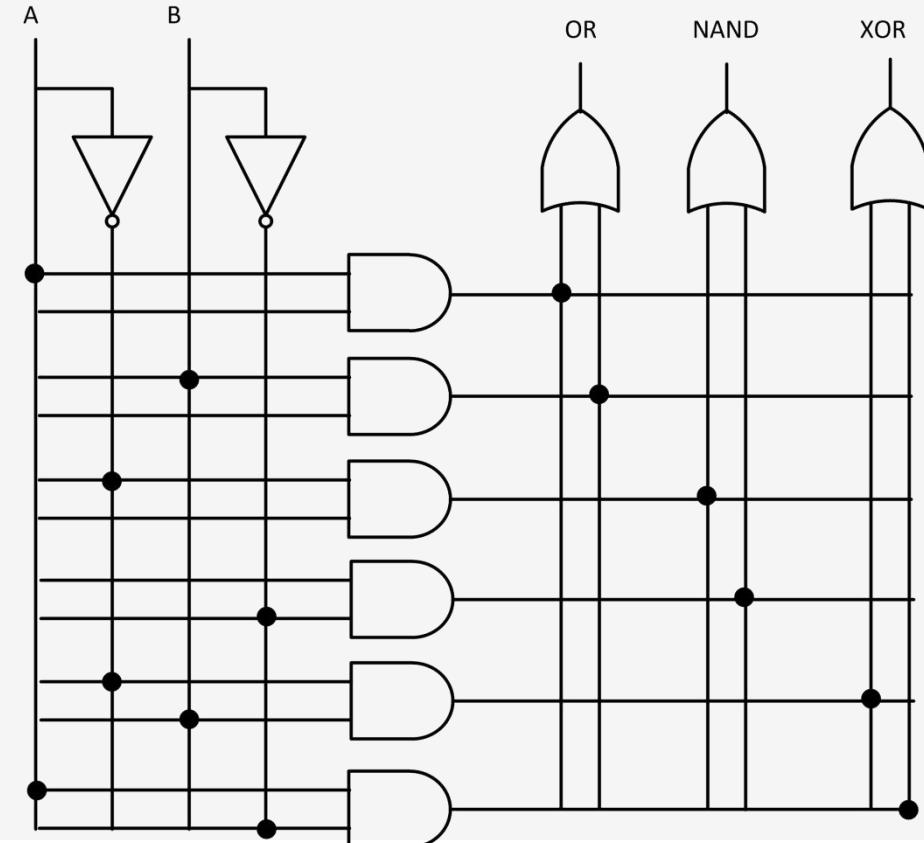
Implement a logic circuit that implements a 2-input OR gate, a 2-input NAND gate, and a 2-input XOR gate using the following architectures:

2) PLA architecture

$$\text{OR} \rightarrow A+B$$

$$\text{NAND} \rightarrow \bar{A}+\bar{B}$$

$$\text{XOR} \rightarrow \bar{A}B+A\bar{B}$$



Programmable Logic Devices - Sample Question - Solution

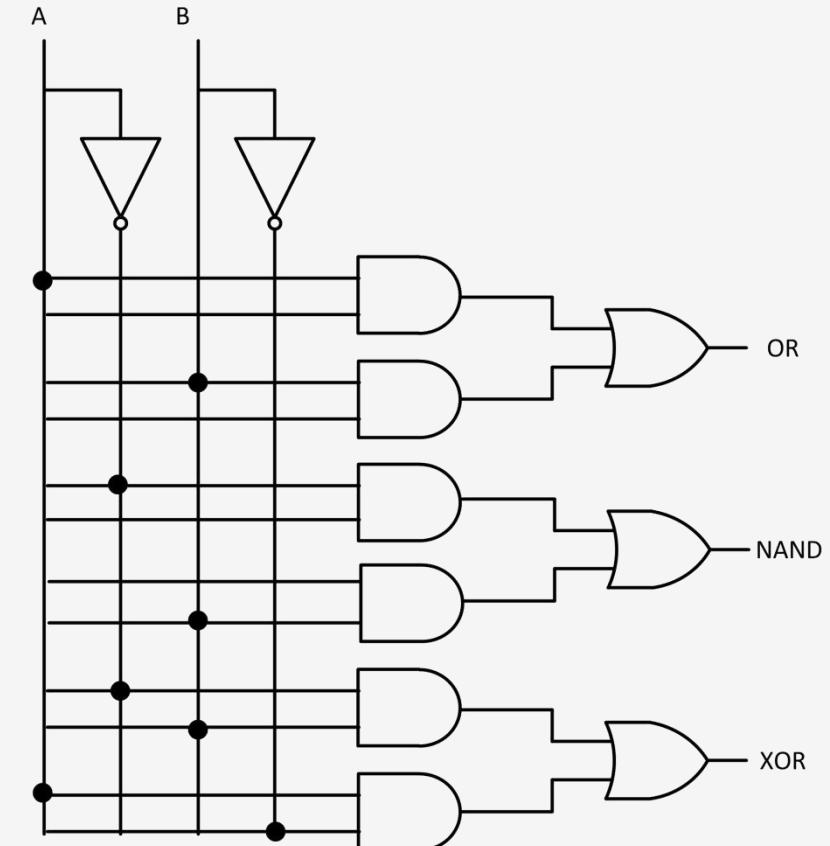
Implement a logic circuit that implements a 2-input OR gate, a 2-input NAND gate, and a 2-input XOR gate using the following architectures:

3) PAL architecture

$$\text{OR} \rightarrow A+B$$

$$\text{NAND} \rightarrow \bar{A}+\bar{B}$$

$$\text{XOR} \rightarrow \bar{A}B+A\bar{B}$$



Advanced Programmable Logic Devices

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Complex Programmable Logic Devices (CPLD)

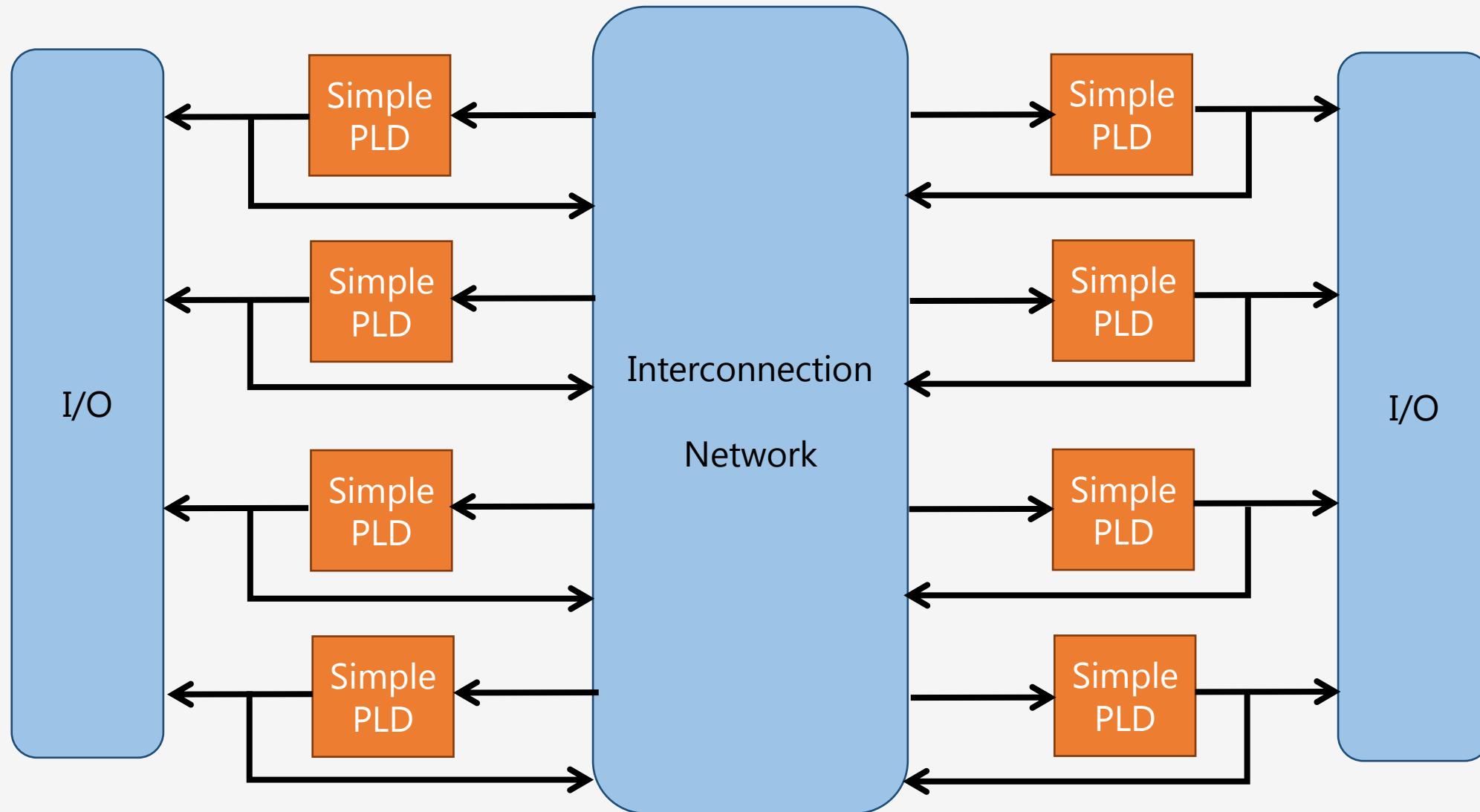
CPLD consists of an array of simple PLDs (like PLAs and PALs,) an interconnection network, and Input/Outputs (I/O.)

CPLD can be programmed to implement a medium-size logic circuit.

CPLD uses re-programmable and non-volatile E²PROM or Flash Memory cells.

CPLD requires global routing for resource connection.

CPLD Architecture



Field Programmable Gate Arrays (FPGA)

FPGA consists of an array Look-Up Tables (LUTs,) memory blocks, an interconnection network, and Input/Outputs (I/O.)

FPGA can be programmed to implement a large-size logic circuit.

FPGA uses re-programmable and volatile SRAM cells.

FPGA requires global routing for resource connection.

With technology scaling, the distinction between CPLDs and FPGAs has become blurred.

LUT - Sample Question

Design a 3-input LUT using an 8-1 MUX and an 8-bit SRAM.
Then, implement the following function using the created LUT.

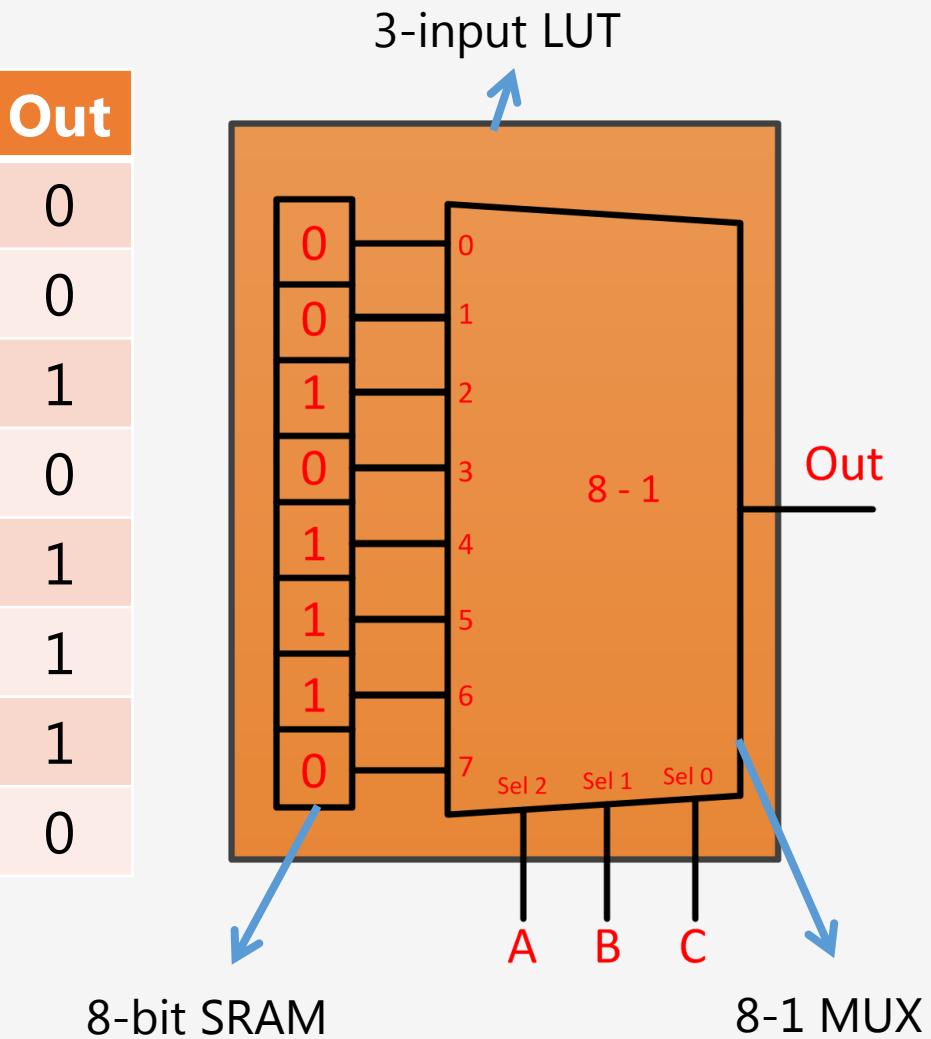
$$\text{Out} = (A \cdot \bar{B}) + (B \cdot \bar{C})$$

LUT - Sample Question - Solution

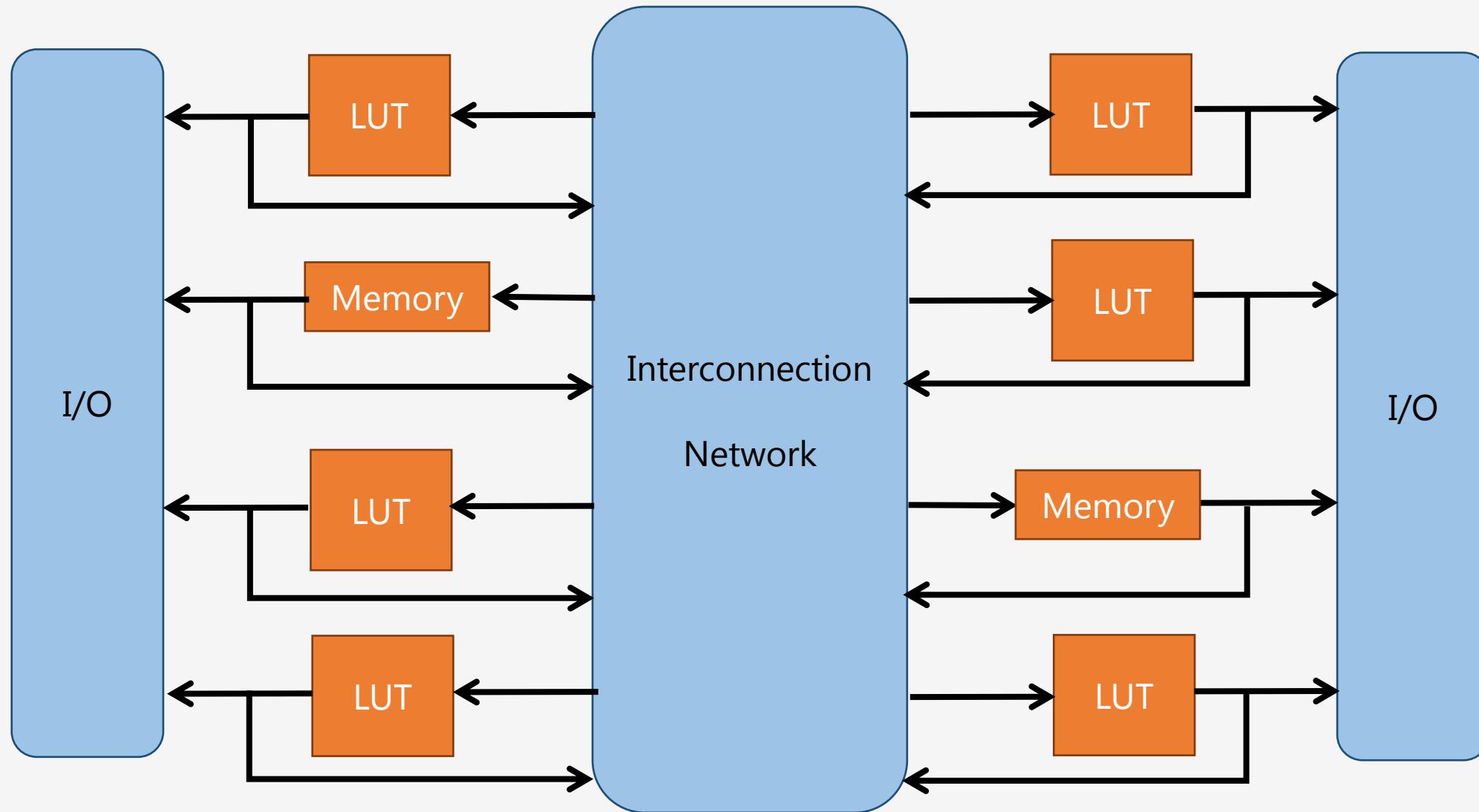
Design a 3-input LUT using an 8-1 MUX and an 8-bit SRAM. Then, implement the following function using the created LUT.

$$\text{Out} = (A \cdot \bar{B}) + (B \cdot \bar{C})$$

A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



FPGA Architecture



Choosing CPLDs/FPGAs

Each CPLD or FPGA device fits a particular need together with tool interface. There are many alternatives and choices in selecting a perfect device for the design.

Performance

Testability

Cost

Reliability

Design Tools

Technology

Power Consumption

Architecture

I/O Pins

Documentation

Digilent Nexys A7 FPGA Board

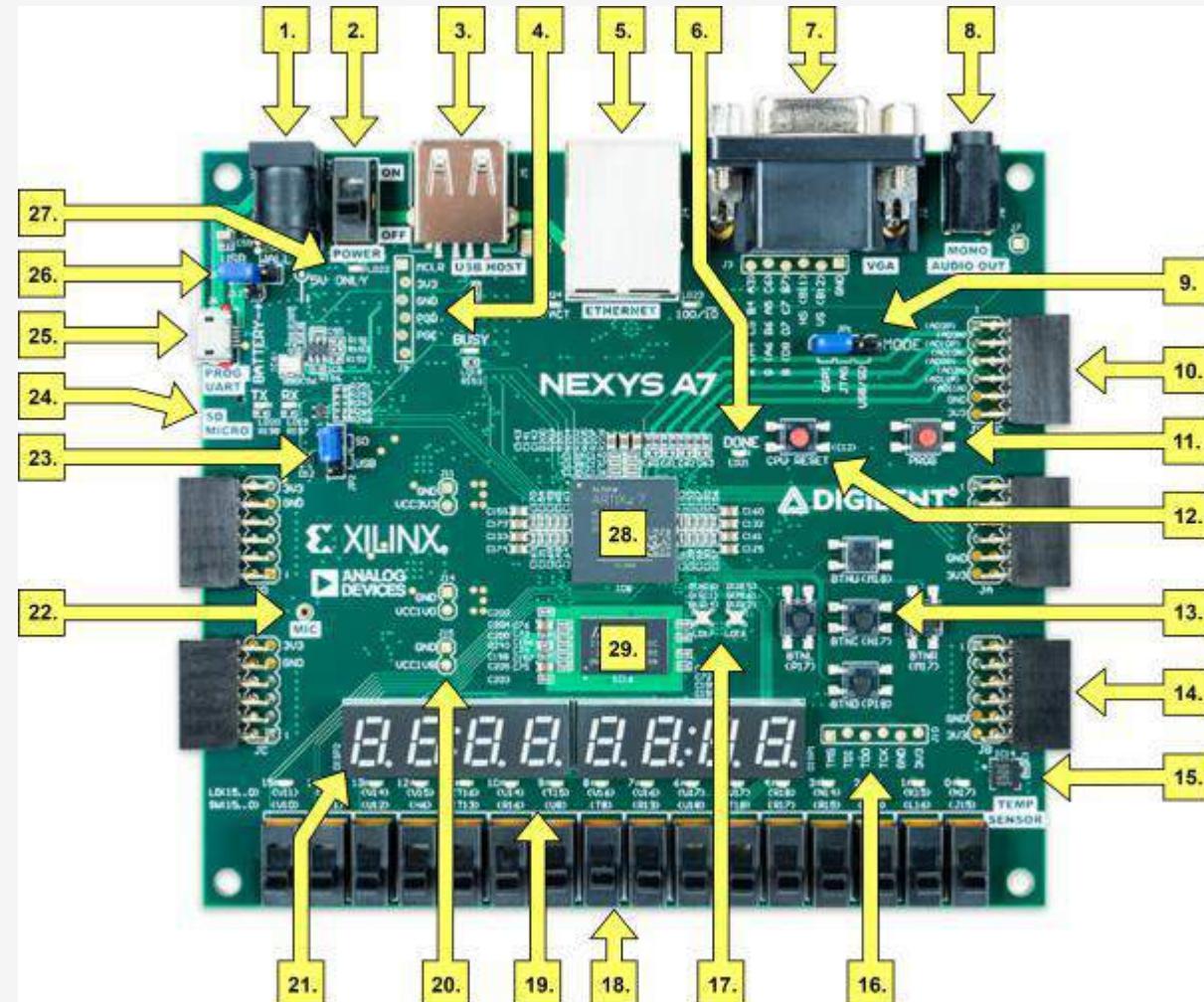
Nexys A7 is a ready-to-use digital circuit development platform based on the Artix-7™ FPGA from Xilinx.

Nexys A7 is compatible with Xilinx's Vivado® Design Suite. Xilinx offers free WebPACK versions of Vivado, so designs can be implemented at no additional cost.

The Nexys A7 can be purchased with either a XC7A100T or XC7A50T FPGA loaded. The difference between Nexys A7-100T and Nexys A7-50T is the size of the Artix-7 part. The Artix-7 FPGA in XC7100T has about a two times larger internal FPGA than the XC750T.

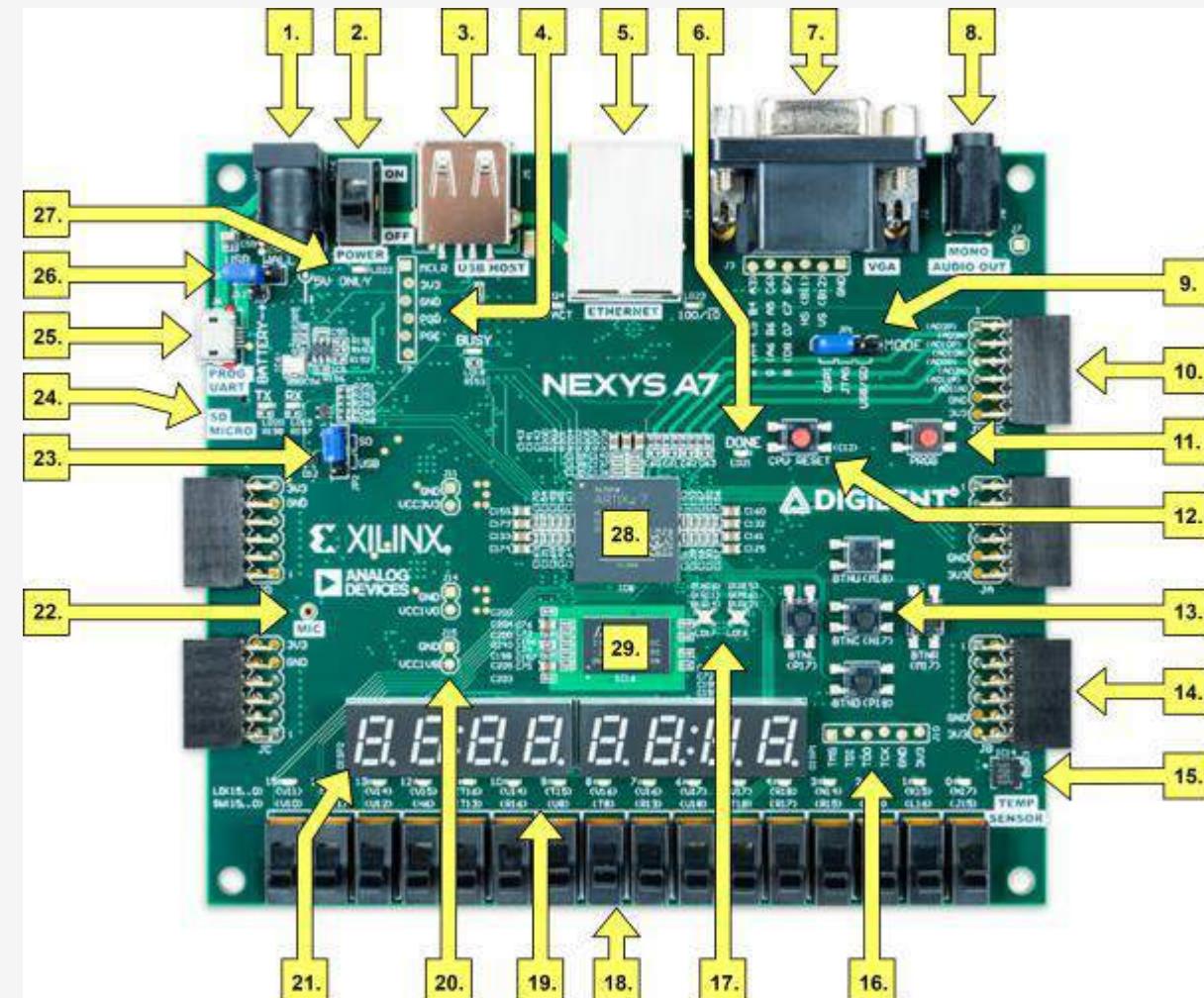
Nexys A7 Features

#	Component
1	Power jack
2	Power switch
3	USB host connector
4	PIC24 programming port (factory use)
5	Ethernet connector
6	FPGA programming done LED
7	VGA connector
8	Audio connector
9	Programming mode jumper
10	Analog signal Pmod port (XADC)



Nexys A7 Features

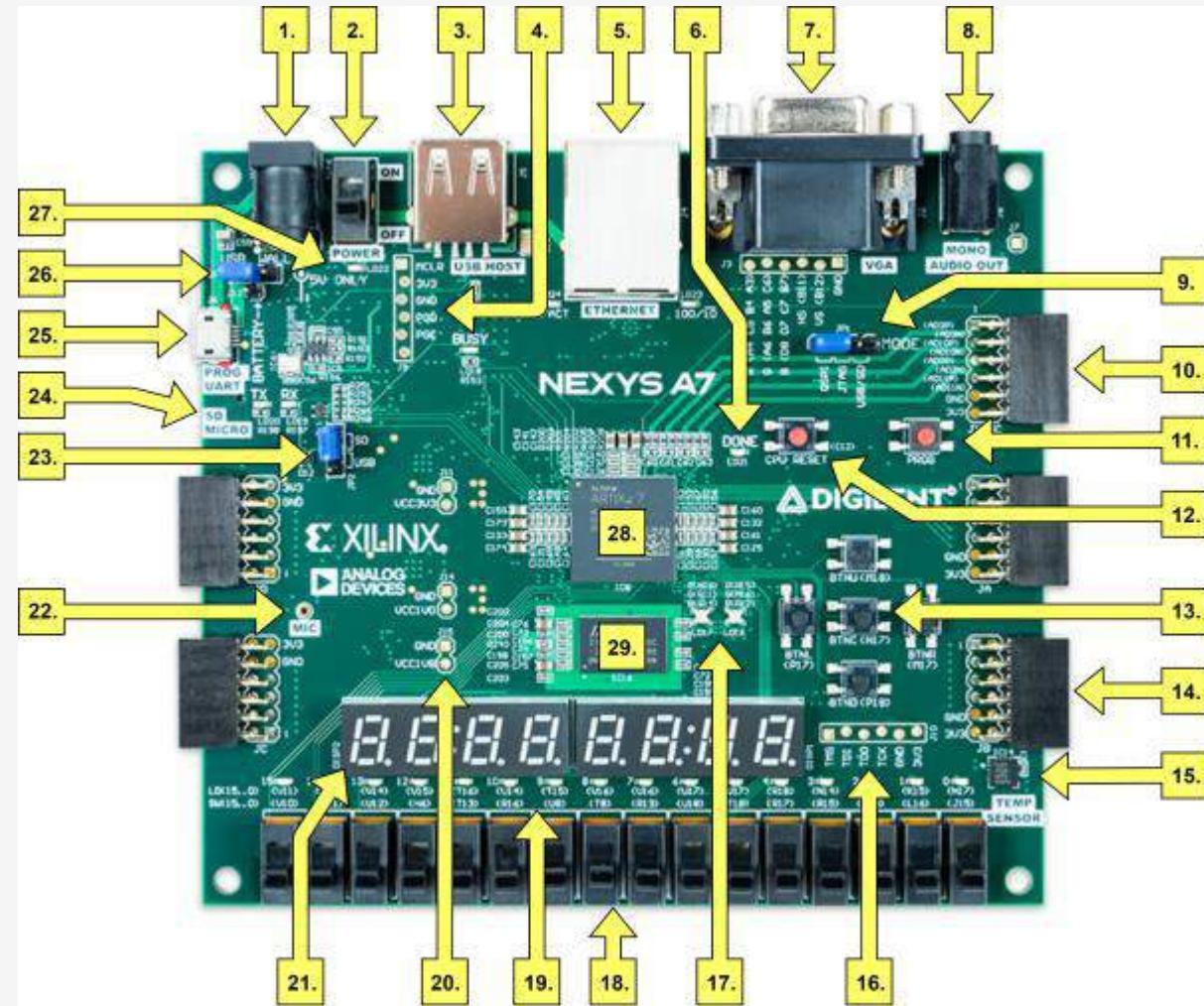
#	Component
11	FPGA configuration reset button
12	CPU reset button
13	Pushbuttons (5)
14	Pmod port
15	Temperature sensor
16	JTAG port for external cable
17	Tri-Color (RGB) LEDs
18	Side Switches (16)
19	LEDs (16)
20	Power supply test point



Nexys A7 Features

#	Component
21	7-segment display (eight digits)
22	Microphone
23	External configuration jumper
24	MicroSD card slot
25	Shared UART/JTAG USB port
26	Power select jumper and battery header
27	Power-good LED
28	Xilinx Artix-7 FPGA
29	DDR2 memory

You can check Nexys A7 FPGA Board Reference Manual for detailed information of each component.



Application Specific Integrated Circuits

Amin Rezaei

CECS 301 - Computer Logic Design II

California State University, Long Beach

Spring 2021

Application Specific Integrated Circuits (ASIC)

ASIC is a custom integrated circuit that is designed for a specific purpose or application.

Integrated circuits are made on a silicon wafer. Circuits are built up with successive mask layers.

The number of masks used to customize the interconnect and other layers is different between different ASIC approaches.

Generally, an ASIC design will be undertaken for a product that will have a large production run.

Full Custom ASIC

In full custom ASIC, **both logic cells and mask layers are customized.**

It only makes sense to design a full-custom integrated circuit when no suitable library exist or existing library cells are not fast enough (e.g., mixed digital/analog design.)

Full-custom ASIC offers the highest performance, while increases design time, complexity, design expense, and highest risk.

Standard Cells-based ASIC

In standard cells-based ASIC, **logic cells are predesigned and mask layers are customized.**

Standard cells are custom designed and then inserted into a library. These cells are then used in the design by being placed in rows and wired together using place and route tools.

Standard cells-based ASIC saves time, money, and reduces risk by using a predesigned standard library, while does not guarantee the best performance.

Gate Array-based ASIC

In gate array-based ASIC, the transistors are predefined (called base array) and **only the interconnect mask layers is customized.**

Gate array-based ASIC is slower than cell-based one, but the implementation time is faster as less time is spent in factory.

Channeled gate arrays → The interconnect uses predefined spaces between rows of base cells.

Channel-less gate arrays → Only some (the top few) mask layers are customized the interconnect.

Logic Circuit Design Comparison

Criterion	Basic Components	PLD	ASIC
Time to Market	Medium	Short	Long
Availability	High	High	Medium
Performance	Low	Medium to High	High
Density	Low	Medium	High
Design Change Cost	Low	Medium	High

Review Question

Implement the given truth table using the following architectures:

- (a) Basic logic array.
- (b) PLA with double 2-input AND-plane and a single 2-input OR-plane.
- (c) FPGA with 3-input LUT.
- (d) FPGA with 2-input LUT.

A	B	C	Out
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1