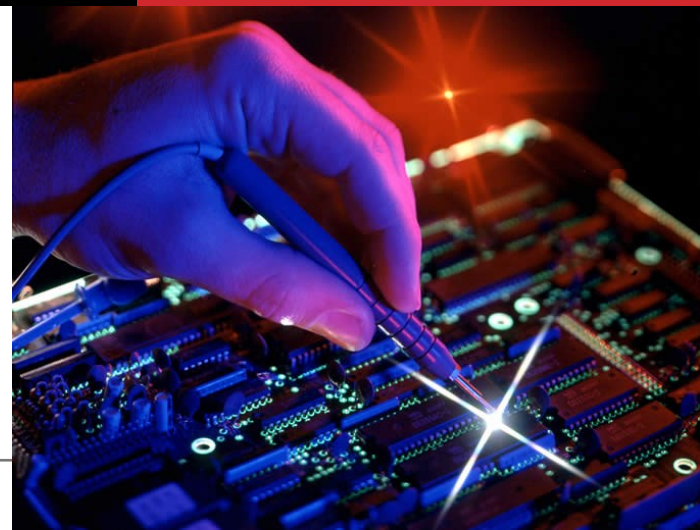




Advanced Computer Architecture

COMBINATIONAL LOGIC DESIGN

Dennis A. N. Gookyi





CONTENTS

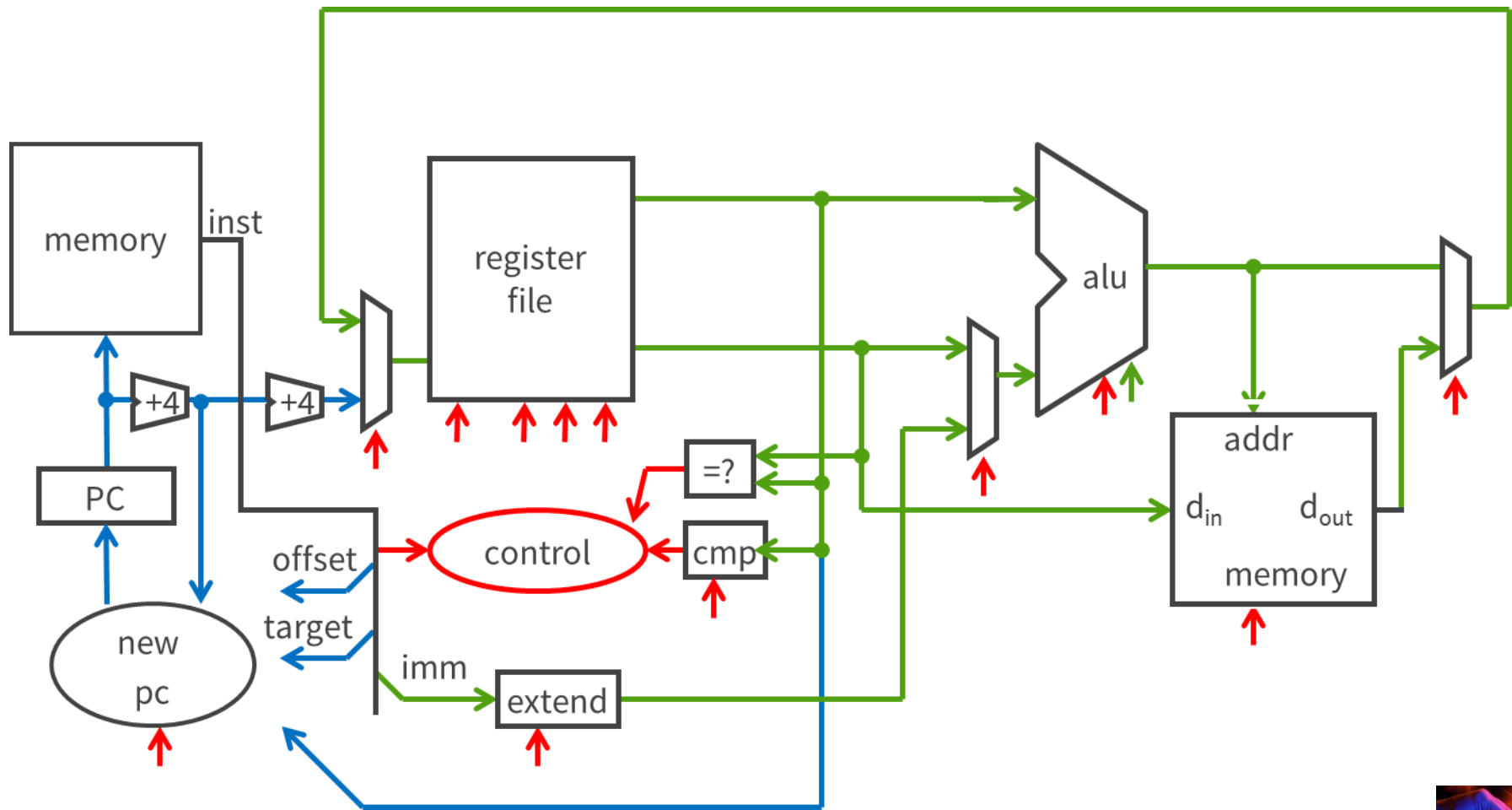
❖ Combinational Logic Design





BIG PICTURE: BUILDING A PROCESSOR

❖ Single cycle processor



COMMON LOGIC GATES

❖ Basic Logic gates

Buffer



A	Z
0	0
1	1

AND



A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR



A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

XOR



A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Inverter



A	Z
0	1
1	0

NAND



A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

NOR



A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

XNOR



A	B	Z
0	0	1
0	1	0
1	0	0
1	1	1



COMBINATIONAL BUILDING BLOCKS

- ❖ Combinational logic is often grouped into larger building blocks to build more complex systems
- ❖ Hides the unnecessary gate-level details to emphasize the function of the building block
- ❖ We now examine:
 - ☐ Decoder
 - ☐ Multiplexer
 - ☐ Full adder
 - ☐ PLA (Programmable Logic Array)

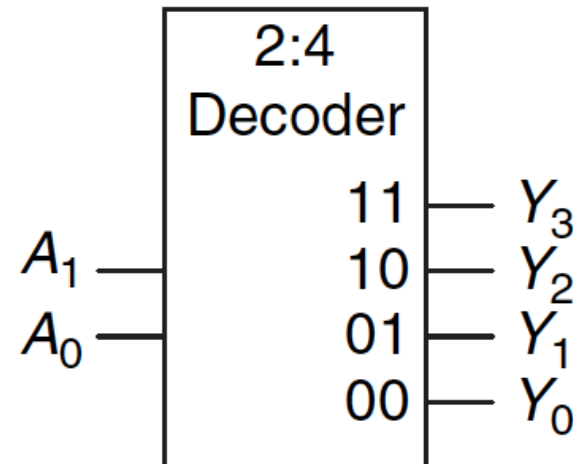




DECODER

- ❖ “Input pattern detector”
- ❖ n inputs and 2^n outputs
- ❖ Exactly one of the outputs is 1 and all the rest are 0s
- ❖ The output that is logically 1 is the output corresponding to the input pattern that the logic circuit is expected to detect
- ❖ Example: 2-to-4 decoder

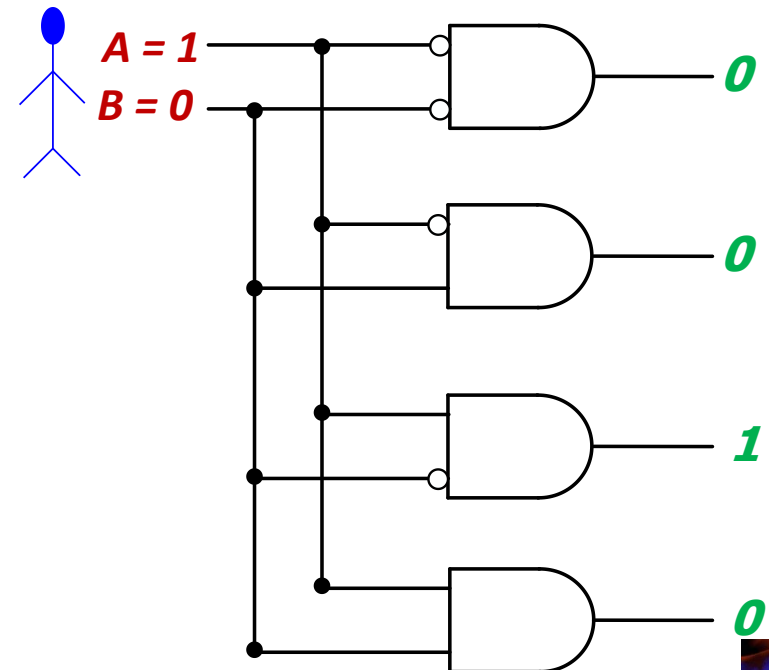
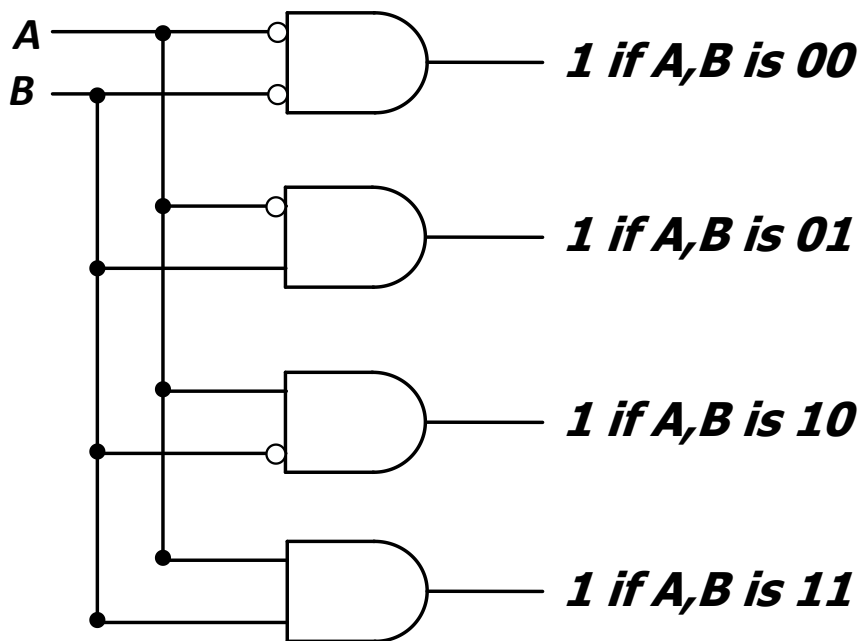
A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0





DECODER

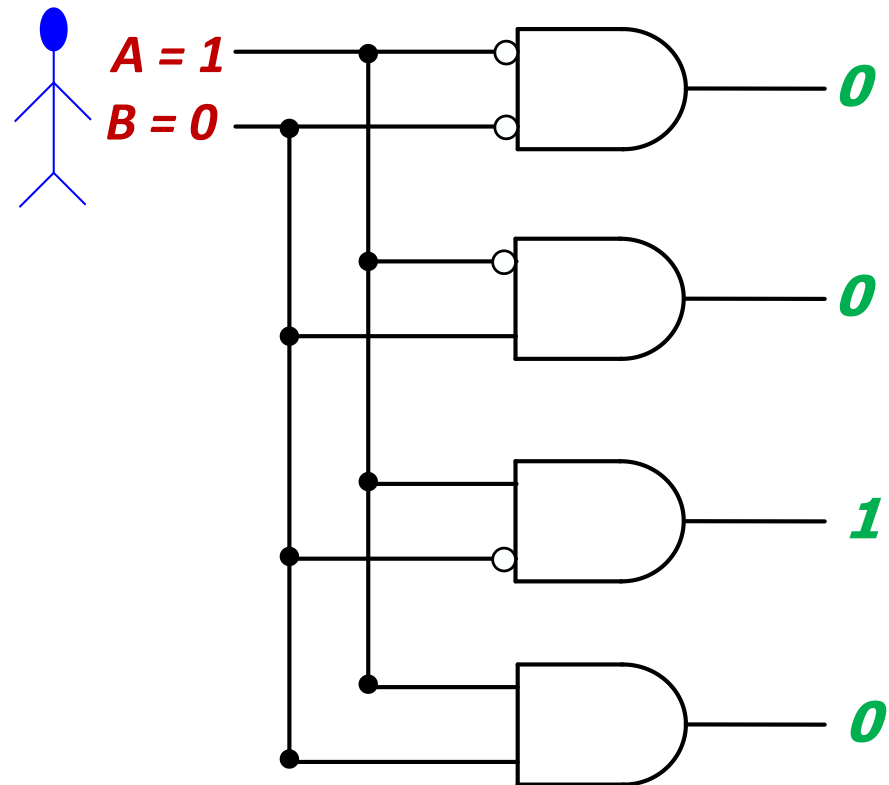
- ❖ n inputs and 2^n outputs
- ❖ Exactly one of the outputs is 1 and all the rest are 0s
- ❖ The output that is logically 1 is the output corresponding to the input pattern that the logic circuit is expected to detect





DECODER

- ❖ The decoder is useful in determining how to interpret a bit pattern
 - ❑ **It could be the address of a location in memory, that the processor intends to read from**
 - ❑ **It could be an instruction in the program and the processor needs to decide what action to take (based on *instruction opcode*)**

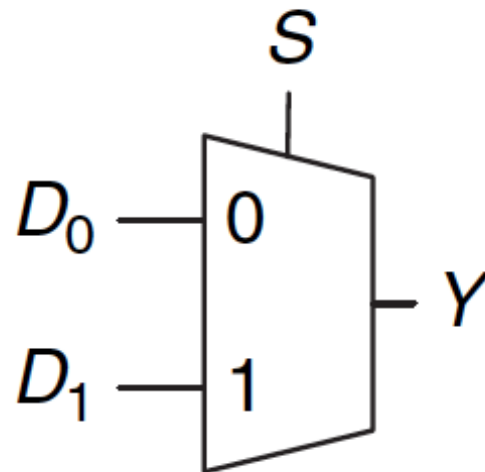




MULTIPLEXER

- ❖ Selects one of the N inputs to connect it to the output
 - Based on the value of a $\log_2 N$ -bit control input called select
- ❖ Example: 2-to-1 MUX

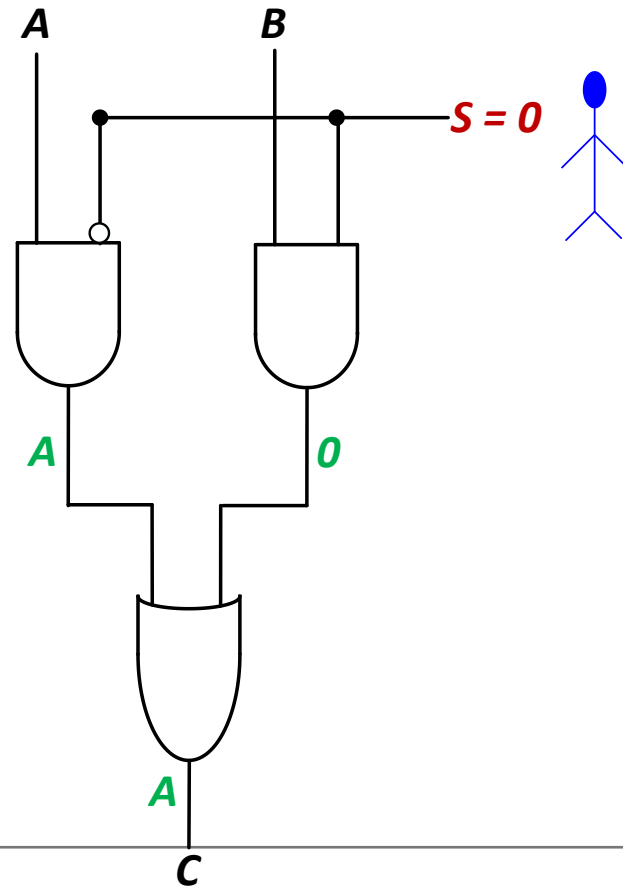
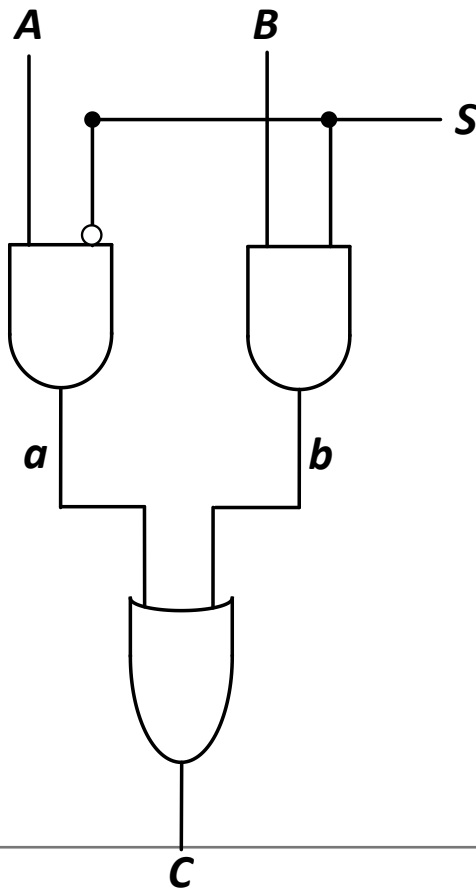
S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1





MULTIPLEXER

- ❖ Selects one of the N inputs to connect it to the output
 - Based on the value of a $\log_2 N$ -bit control input called select
- ❖ Example: 2-to-1 MUX



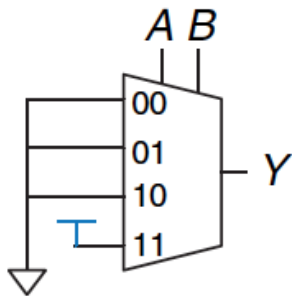


MULTIPLEXER

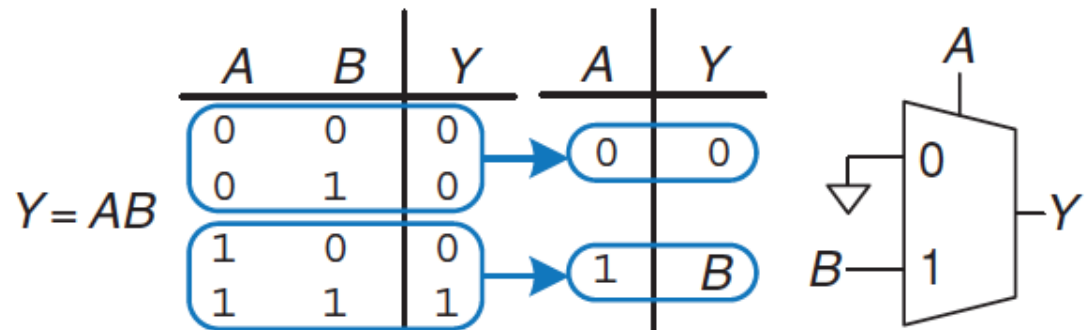
- ❖ Multiplexers can be used as lookup tables to perform logic functions

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = AB$



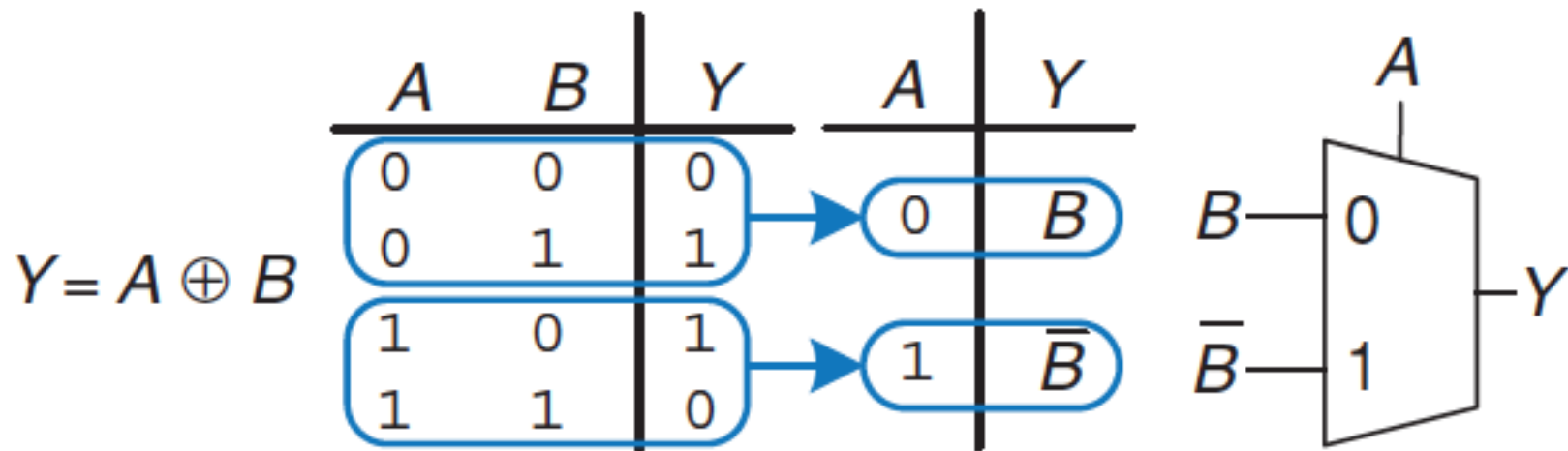
4:1 multiplexer
implementation of two-
input AND function





MULTIPLEXER

- ❖ Multiplexers can be used as lookup tables to perform logic functions



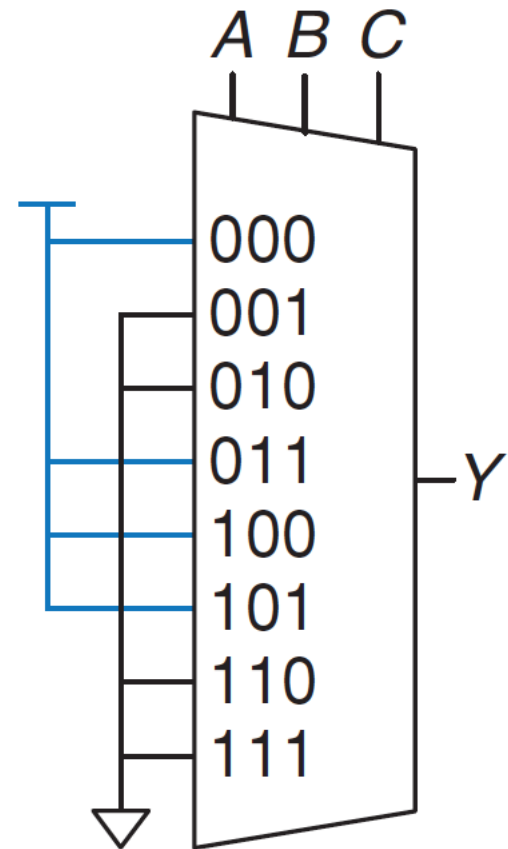


MULTIPLEXER

- ❖ Multiplexers can be used as lookup tables to perform logic functions

<i>A</i>	<i>B</i>	<i>C</i>	<i>Y</i>
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$$Y = A\bar{B} + \bar{B}\bar{C} + \bar{A}BC$$

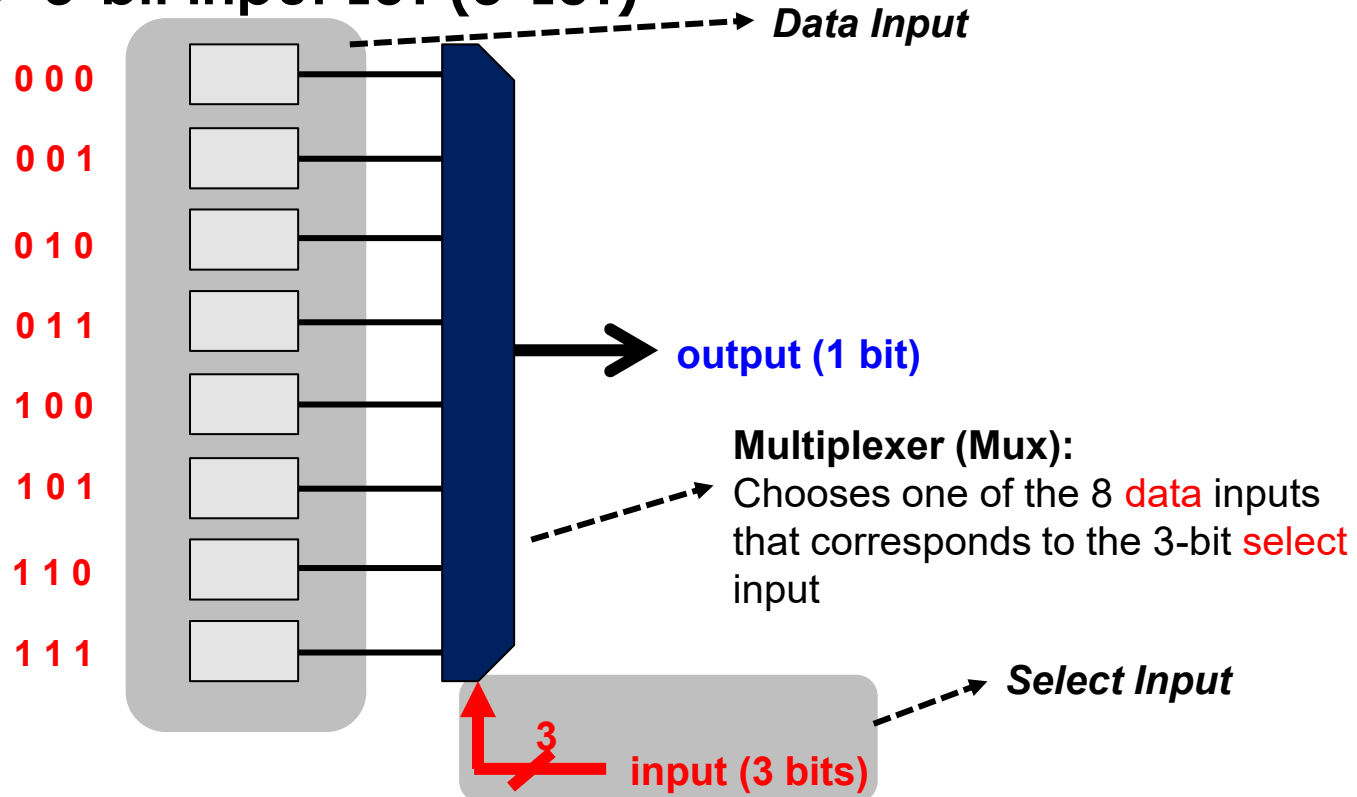




MULTIPLEXER

- ❖ Multiplexers can be used as lookup tables to perform logic functions: 8-input Lookup Table (LUT)

- ❖ **3-bit input LUT (3-LUT)**



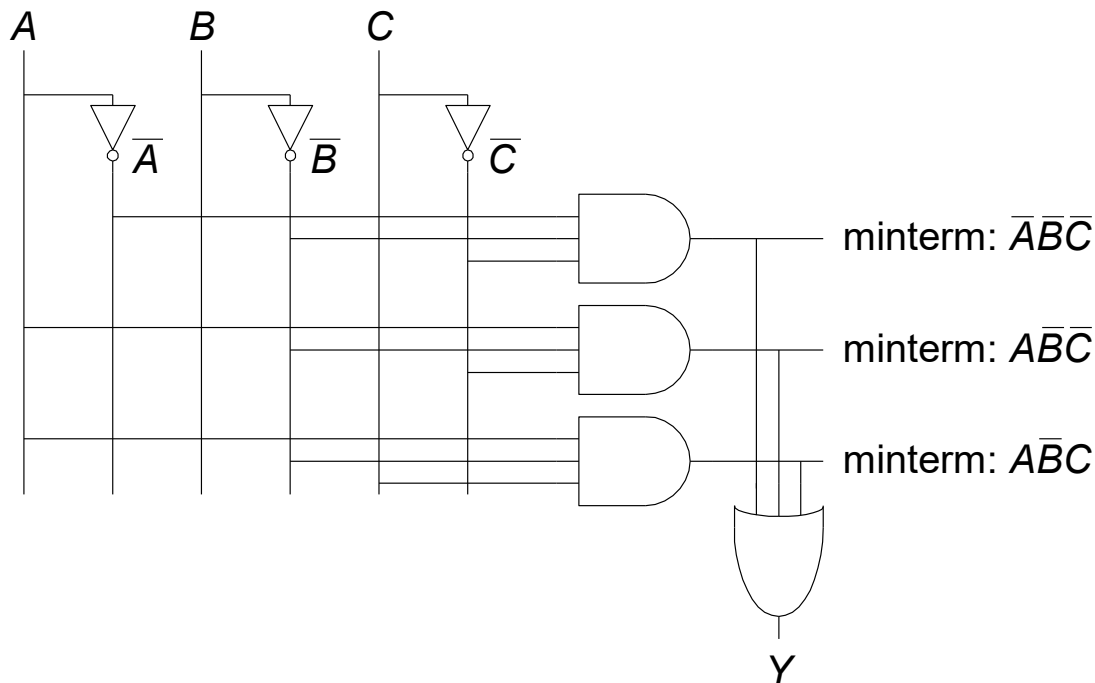
3-LUT can implement
any 3-bit input function





PROGRAMMABLE LOGIC ARRAY (PLA)

- ❖ SOP (sum-of-products) leads to two-level logic
- ❖ Example: $Y = (\bar{A} \cdot \bar{B} \cdot \bar{C}) + (A \cdot \bar{B} \cdot \bar{C}) + (A \cdot \bar{B} \cdot C)$



- ❖ A PLA enables the two-level SOP implementation of any N-input M-output function





PROGRAMMABLE LOGIC ARRAY (PLA)

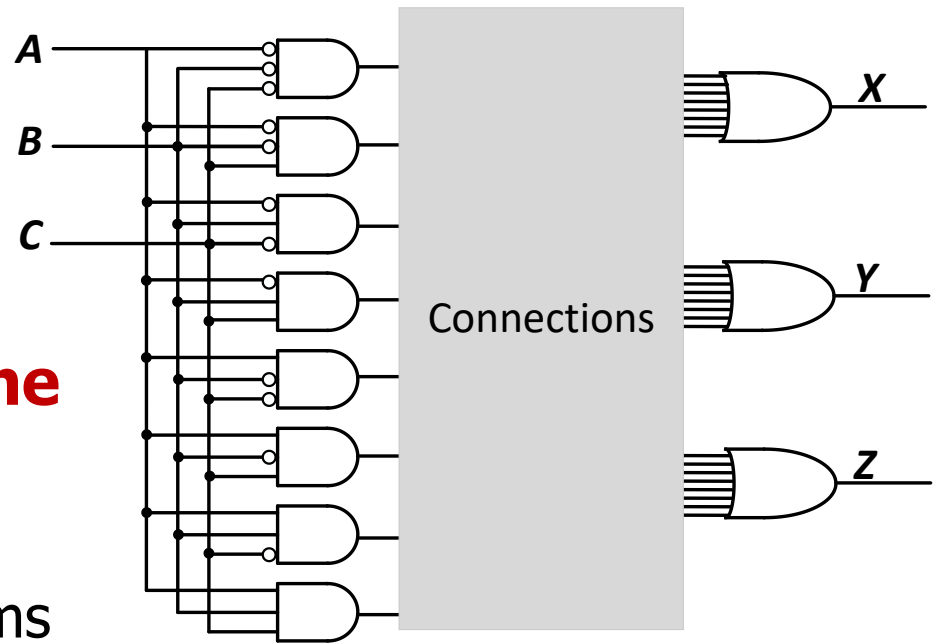
- ❖ The below logic structure is a very common building block for implementing any collection of logic functions one wishes to

- An **array** of AND gates followed by an **array** of OR gates
- **How do we determine the number of AND gates?**

- ❑ **Remember SOP:** the number of possible minterms

- ❑ For an n -input logic function, we need a PLA with 2^n n -input AND gates

- **How do we determine the number of OR gates?** The number of output columns in the truth table

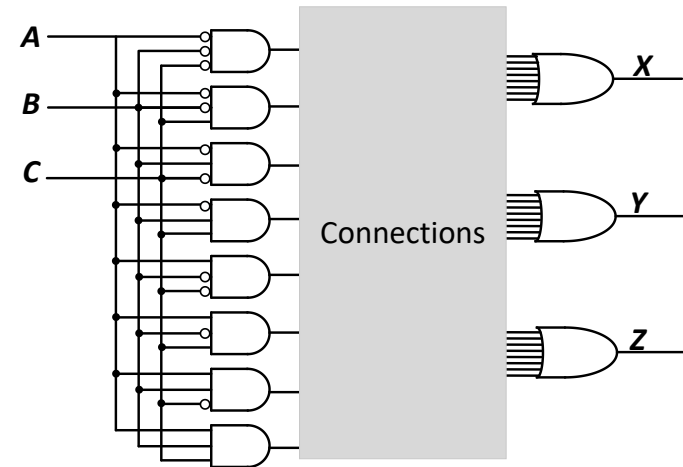




PROGRAMMABLE LOGIC ARRAY (PLA)

- ❖ How do we implement a logic function?
- ❖ Connect the output of an AND gate to the input of an OR gate if the corresponding minterm is included in the SOP
- ❖ This is a simple programmable logic construct

- **Programming a PLA:** we program the **connections** from AND gate outputs to OR gate inputs to implement a desired logic function



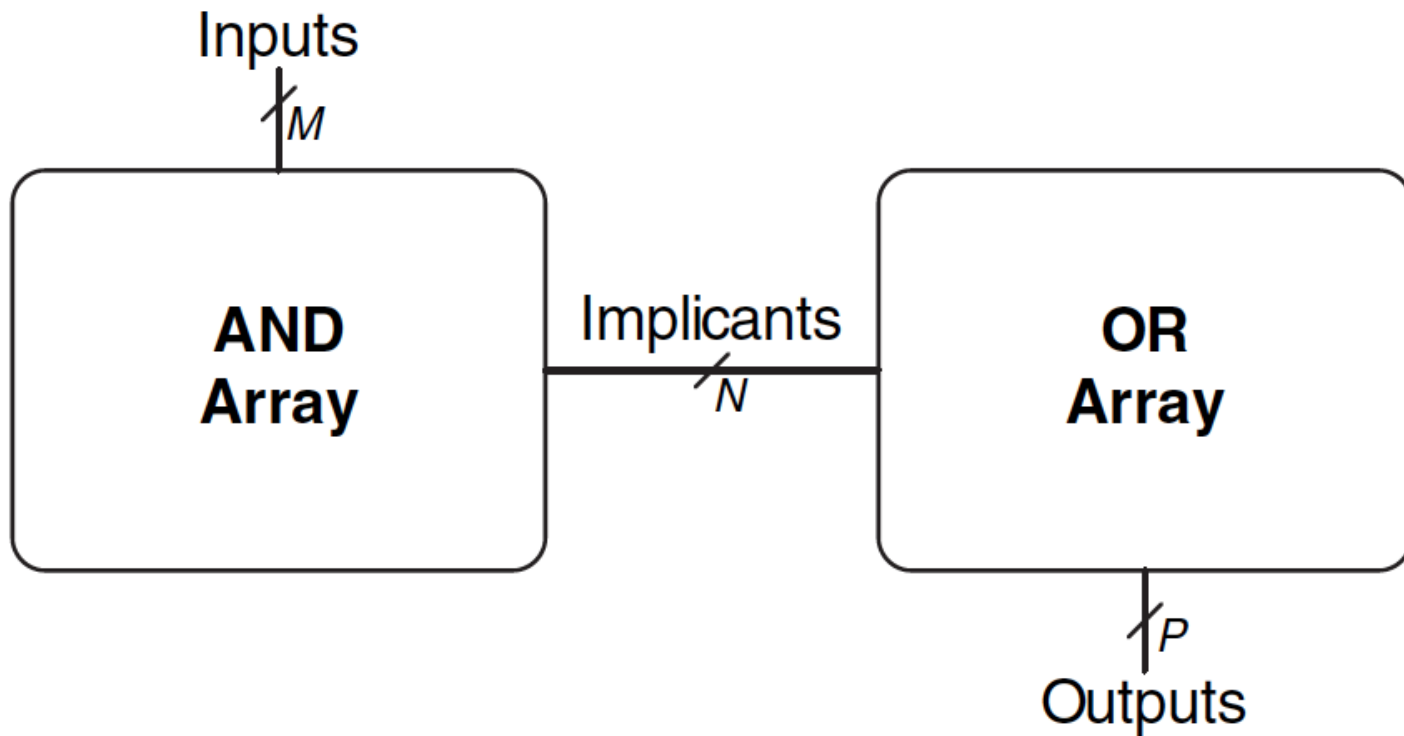
- Have you seen any other type of programmable logic?
 - ❑ Yes! An FPGA...
 - ❑ An FPGA uses more advanced structures, as we see in the labs





PROGRAMMABLE LOGIC ARRAY (PLA)

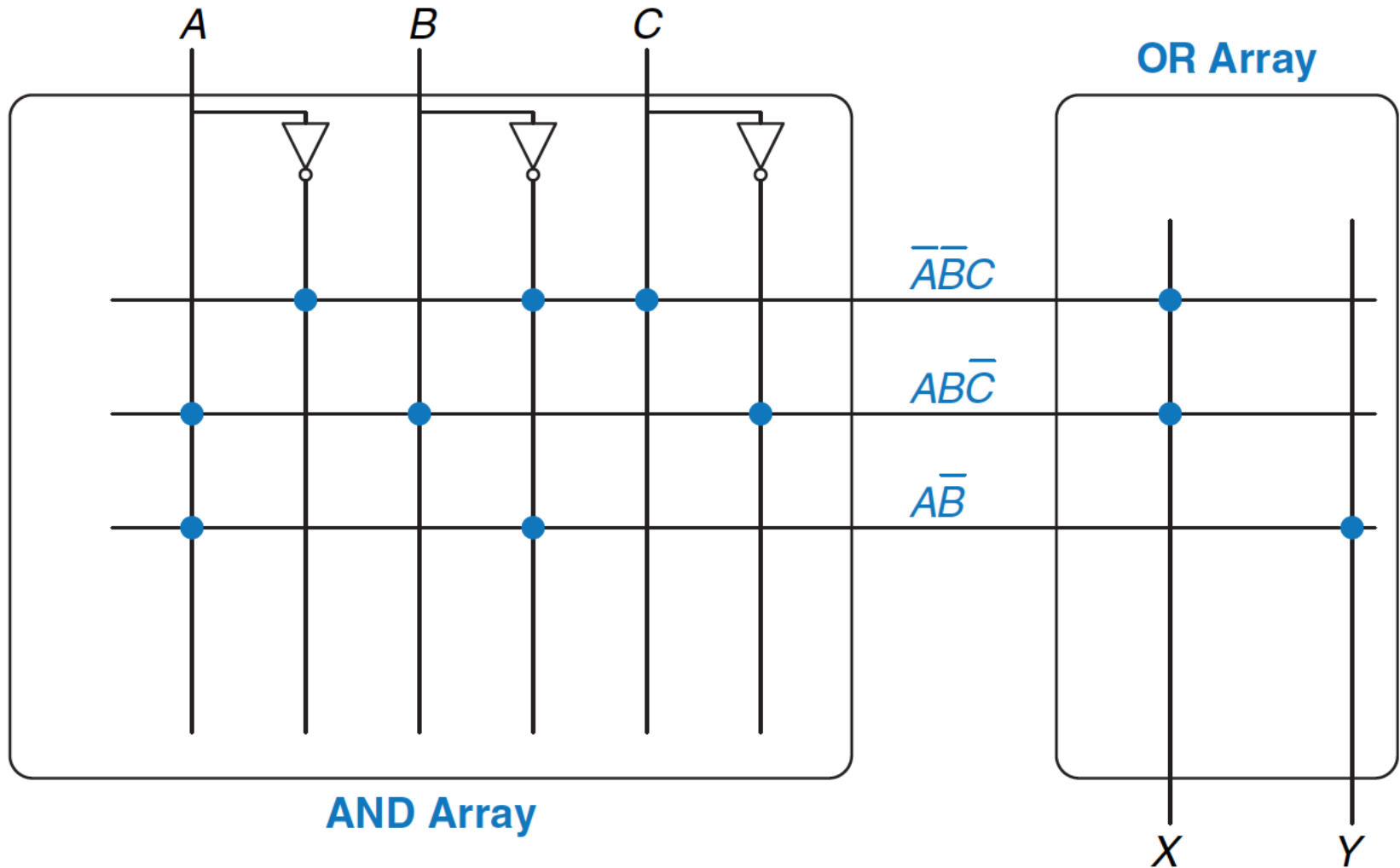
❖ PLA example





PROGRAMMABLE LOGIC ARRAY (PLA)

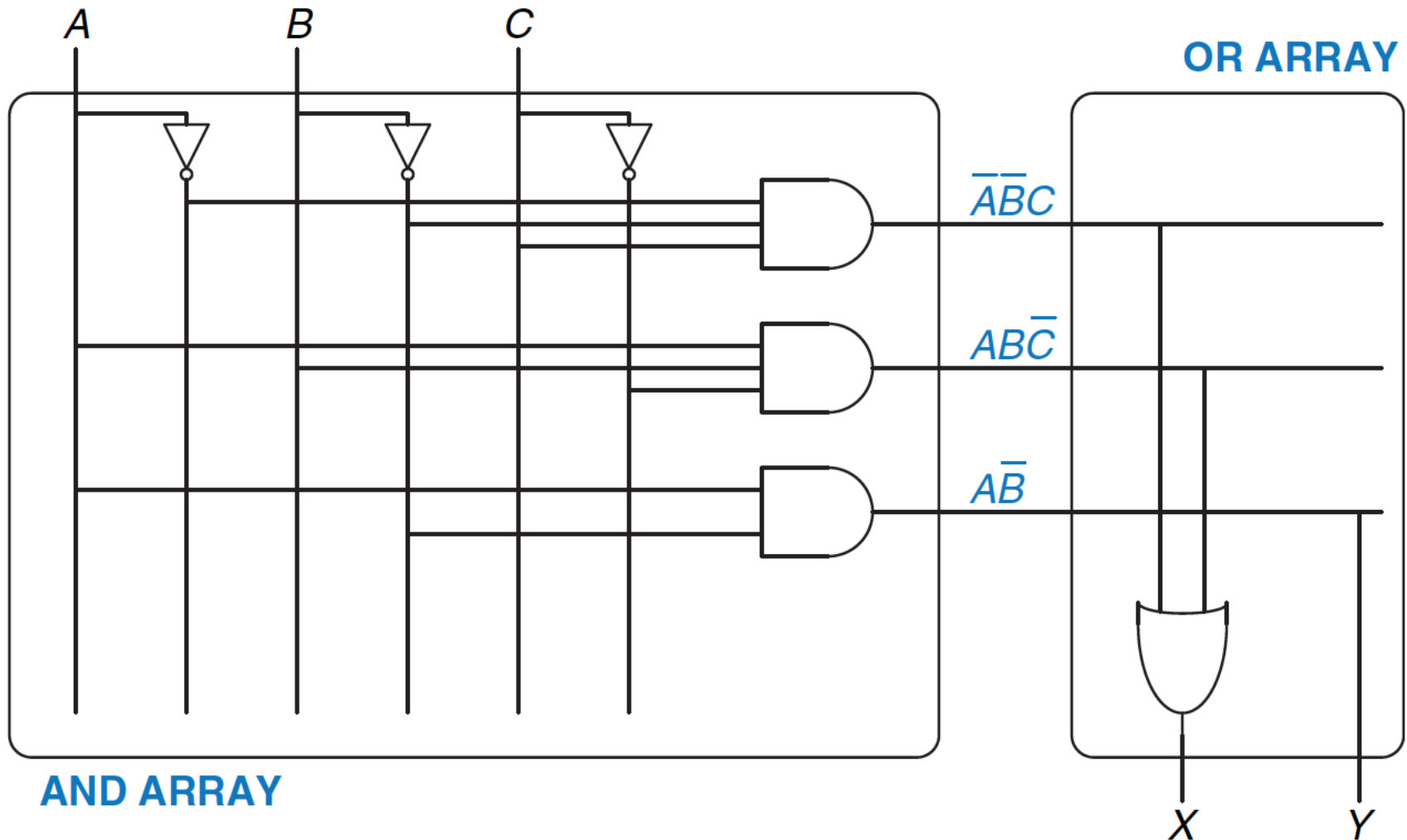
❖ PLA example





PROGRAMMABLE LOGIC ARRAY (PLA)

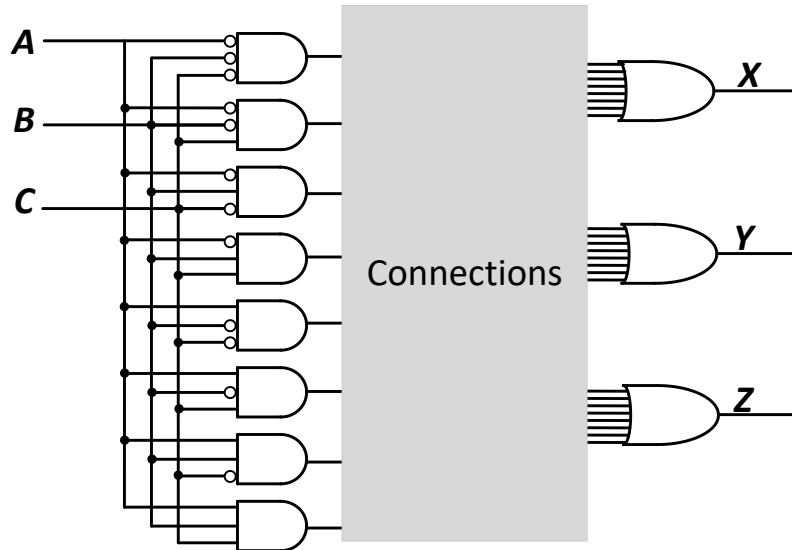
❖ PLA example





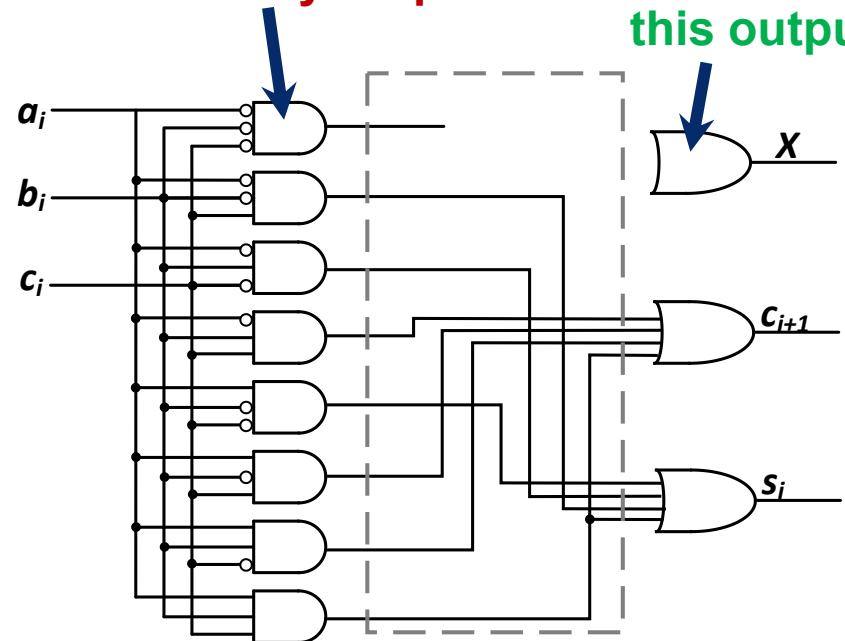
PROGRAMMABLE LOGIC ARRAY (PLA)

❖ Implementing a Full Adder Using a PLA



This input should not be connected to any outputs

We do not need this output



Truth table of a full adder

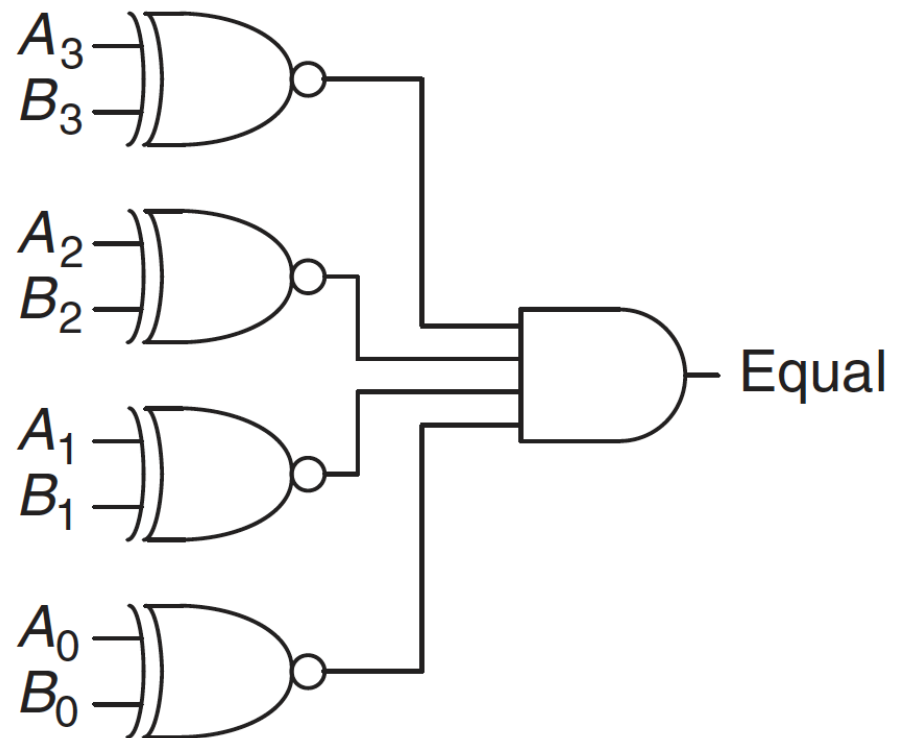
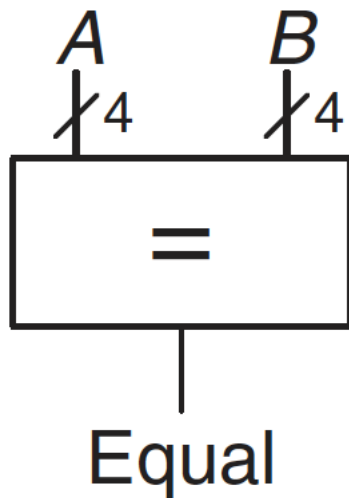
a_i	b_i	$carry_i$	$carry_{i+1}$	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1





COMPARATOR

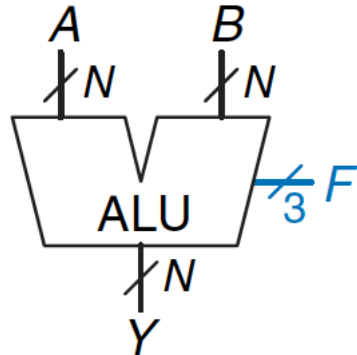
- ❖ Equality checker (compare if equal)
- ❖ Checks if two N-input values are exactly the same
- ❖ Example: 4-bit Comparator





ARITHMETIC LOGIC UNIT (ALU)

- ❖ Combines a variety of arithmetic and logical operations into a single unit (that performs only one function at a time)
- ❖ Usually denoted with this symbol:



ALU Symbol

ALU Operations

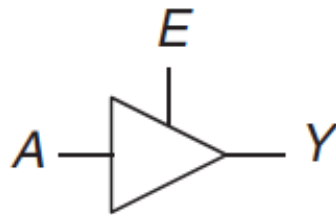
$F_{2:0}$	Function
000	A AND B
001	A OR B
010	A + B
011	not used
100	A AND \bar{B}
101	A OR \bar{B}
110	A - B
111	SLT





TRI-STATE BUFFER

- ❖ A tri-state buffer enables gating of different signals onto a wire
- ❖ A tri-state buffer acts like a switch



E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

- ❖ Floating signal (Z): Signal that is not driven by any circuit
 - Open circuit, floating wire





TRI-STATE BUFFER

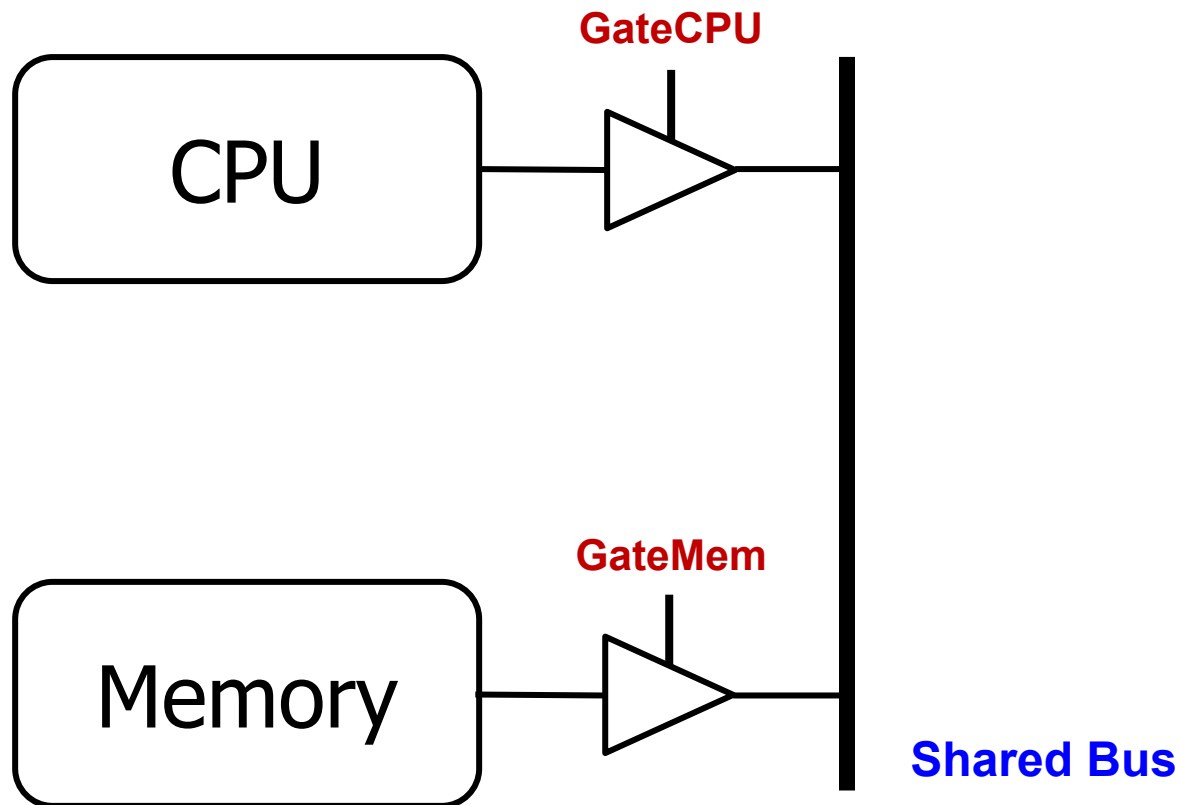
- ❖ Use of tri-state buffers
- ❖ Imagine a wire connecting the CPU and memory
 - At any time only the CPU or the memory can place a value on the wire, both not both
 - You can have two tri-state buffers: one driven by CPU, the other memory; and ensure at most one is enabled at any time





TRI-STATE BUFFER

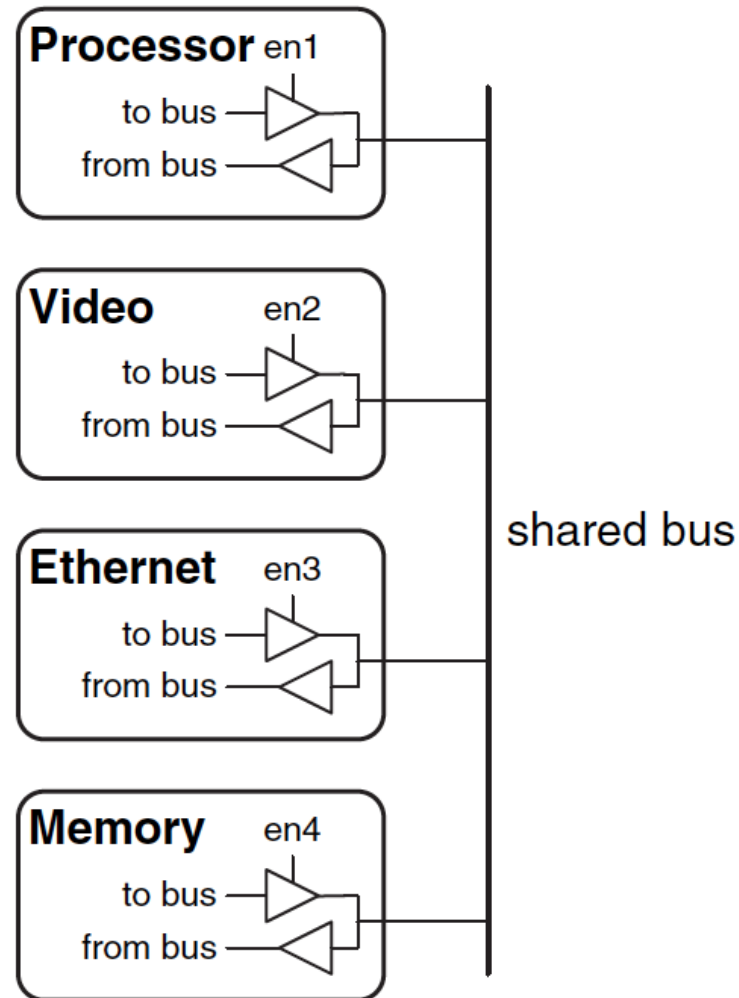
❖ Use of tri-state buffers





TRI-STATE BUFFER

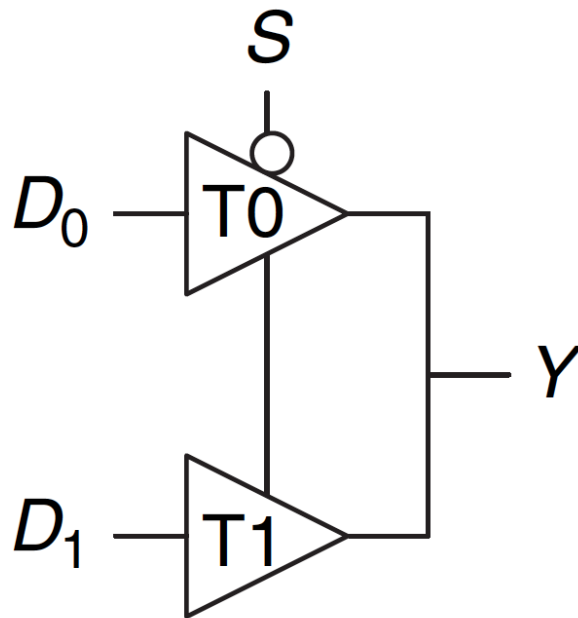
❖ Use of tri-state buffers



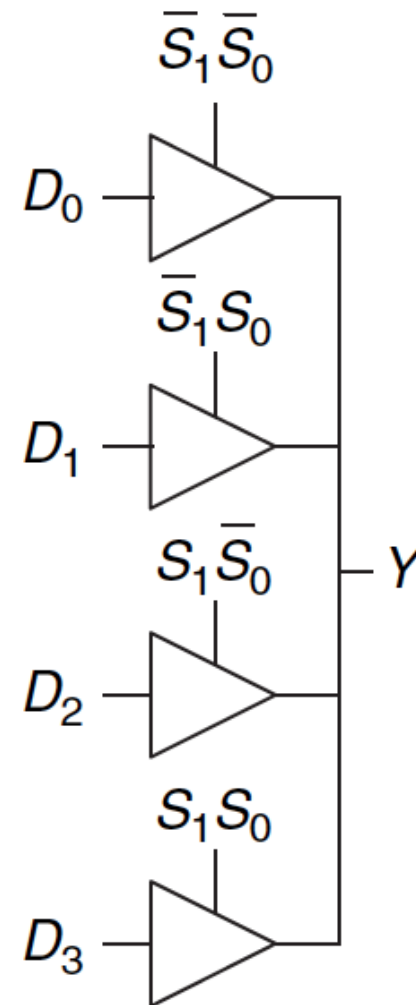


TRI-STATE BUFFER

- ❖ Use of tri-state buffers (MUX using tri-state buffers)



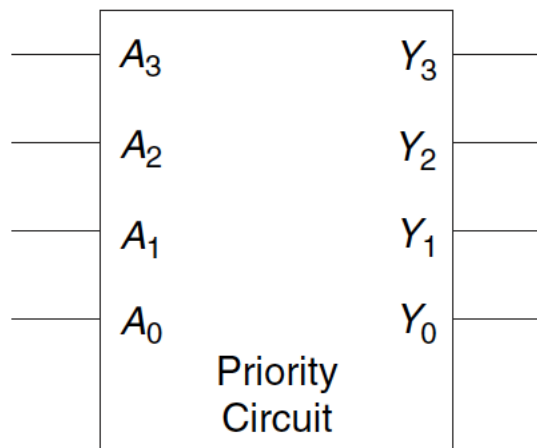
$$Y = D_0 \bar{S} + D_1 S$$





PRIORITY CIRCUIT

- Inputs: “Requestors” with priority levels
- Outputs: “Grant” signal for each requestor
- Example 4-bit priority circuit
- Real life example: Imagine a bus requested by 4 processors



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	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	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

