Disclaimer

The slides presented here are a combination of the CS251 course notes from previous terms, the work of Xiao-Bo Li, and material from the required textbook "Computer Organization and Design, ARM Edition," by David A. Patterson and John L. Hennessy. It is being used here with explicit permission from the authors.

CS251 course policy requires students to delete all course files after the term. Therefore, please do not post these slides to any website or share them.

CS251 - Computer Organization and Design

Digital Combinational Logic Design

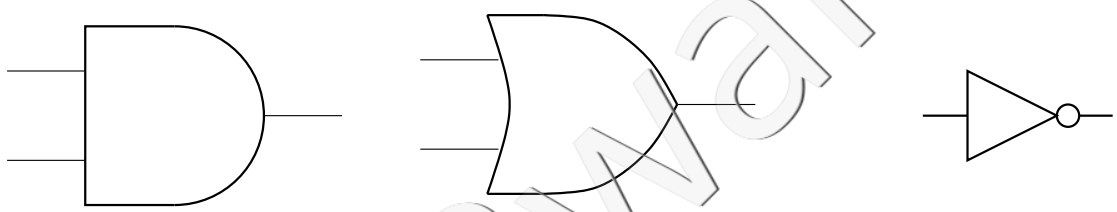
Instructor: Zille Huma Kamal

University of Waterloo

Spring 2023

Using Gates in Logic Design

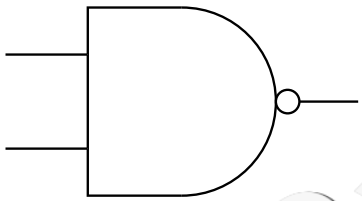
- Here are symbols for AND, OR, NOT gates

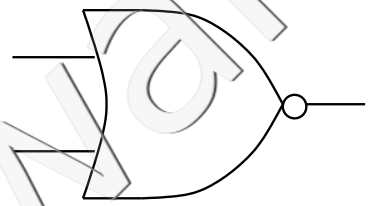


- NOT often drawn as "bubble" on input or output
- AND, OR can be generalized to many inputs (useful)
- We can design using AND, OR, NOT, and optimize afterwards

NAND and NOR

- In practice, logic minimization software works with NAND or NOR gates, or at transistor level
- Here are symbols for NAND, NOR:





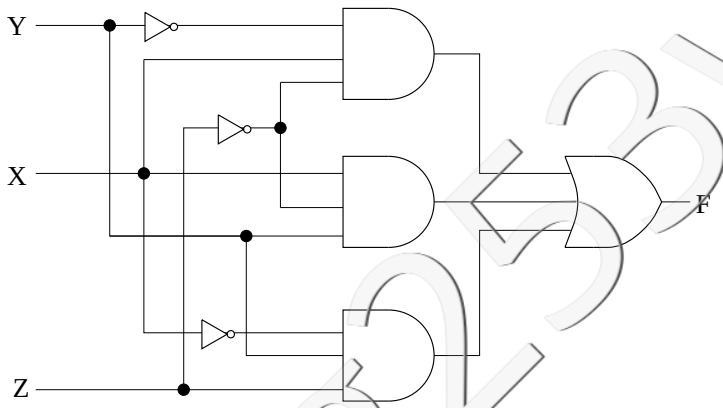
NAND		
A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

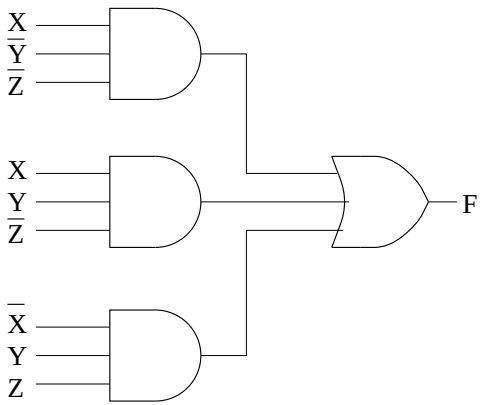
- Two level NAND circuit

Good Style in Circuit Drawing

- Assume all literals (variables and their negations) are available
- Rectilinear wires, dots when wires split
- Do not draw spaghetti wires for inputs; instead, write each literal as needed



harder to read



preferred

Example

Try this

Draw the combinational logic circuit that implements F using only NAND gates:

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

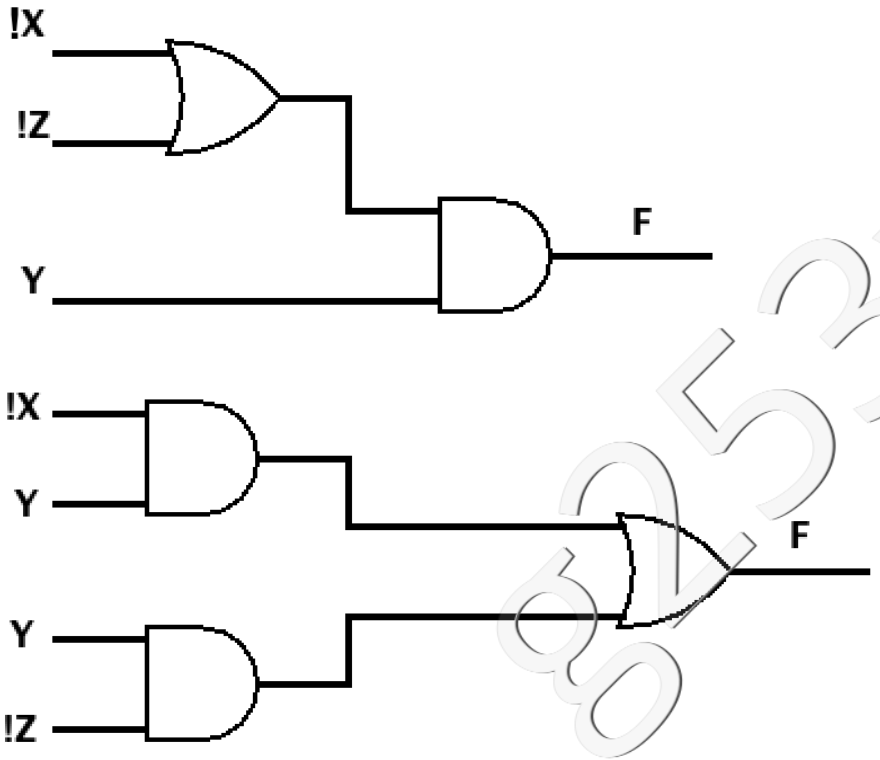
Solution: Derive Boolean expression for F

- Derive the Boolean expression for F
  - Sum of Products Form:
    $F = \bar{X}Y\bar{Z} + \bar{X}YZ + XY\bar{Z}$
  - Simplify the expression using Laws of Boolean Algebra:
    $F = \bar{X}Y\bar{Z} + \bar{X}YZ + XY\bar{Z} + \bar{X}Y\bar{Z}$
    $F = \bar{X}Y(\bar{Z} + Z) + Y\bar{Z}(X + \bar{X})$
    $F = \bar{X}Y + Y\bar{Z}$
    $F = Y(\bar{X} + \bar{Z})$
  - Product of Sums Form:
    $F = (X + Y + Z)(X + Y + \bar{Z})(\bar{X} + Y + Z)(\bar{X} + Y + \bar{Z})$
- Draw the circuit using ANDs and ORs
- Convert ANDs and ORs to NAND equivalent

Solution: Circuit design

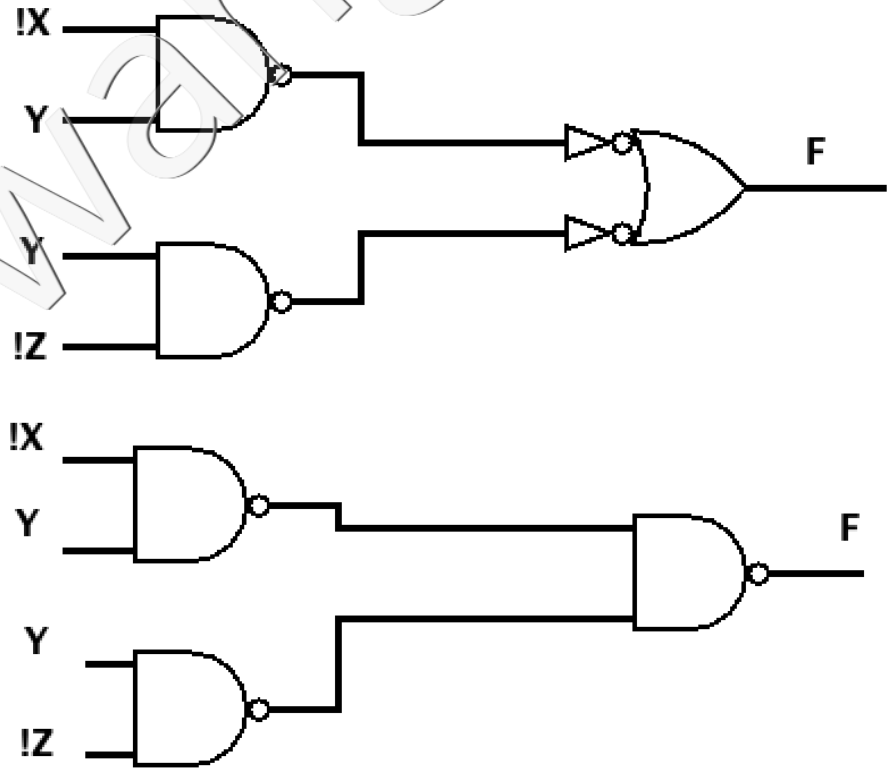
Design using ANDs and ORs for SOP or POS form of F

Note: $\bar{X}$ is represented as $!X$ in the circuit



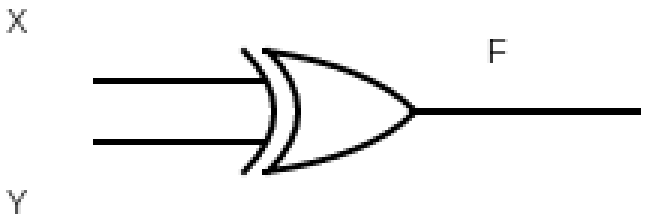
Design using NAND equivalent or NAND for F

Note: $\bar{X}$ is represented as $!X$ in the circuit



XOR Gate

The gate symbol and truth table for XOR



X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

The XOR gate has the following mathematical symbol:

$$F = X \oplus Y$$

XOR and Parity

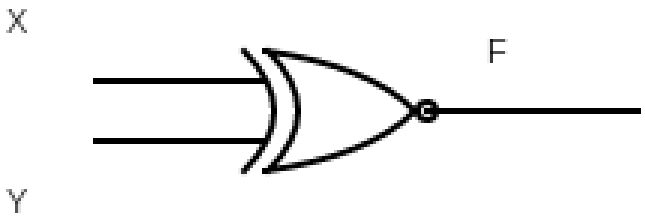
In general, the output of XOR is 1 if the input has an odd number of 1's.

$$1 \oplus 0 \oplus 1 = (1 \oplus 0) \oplus 1 = 1 \oplus 1 = 0$$
$$1 \oplus 1 \oplus 1 = (1 \oplus 1) \oplus 1 = 0 \oplus 1 = 1$$

Practical uses of XOR for error detection using odd parity (see the textbook's error correction section page A-64 for details)

XNOR Gate

The complement of the XOR gate is the XNOR Gate:



X	Y	F
0	0	1
0	1	0
1	0	0
1	1	1

The XNOR gate has the following mathematical symbol:

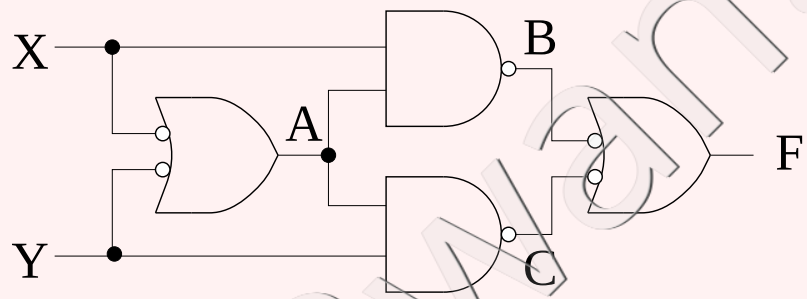
$$F = \overline{X \oplus Y}$$

- the output of XNOR is 1 if the input has an even number of 1's
  - 0 is an even number
  - when all inputs are 0, there are 0 number of 1's that is an even number of 1's, which implies, XNOR is asserted (1)
- Practical uses of XNOR for implementing a full-adder in ALU and the equality comparator.

Example: Deriving Truth Table from Circuit

Try this

Given the following circuit:



Complete the following truth table with intermediate outputs.

X	Y	A	B	C	F
0	0				
0	1				
1	0				
1	1				

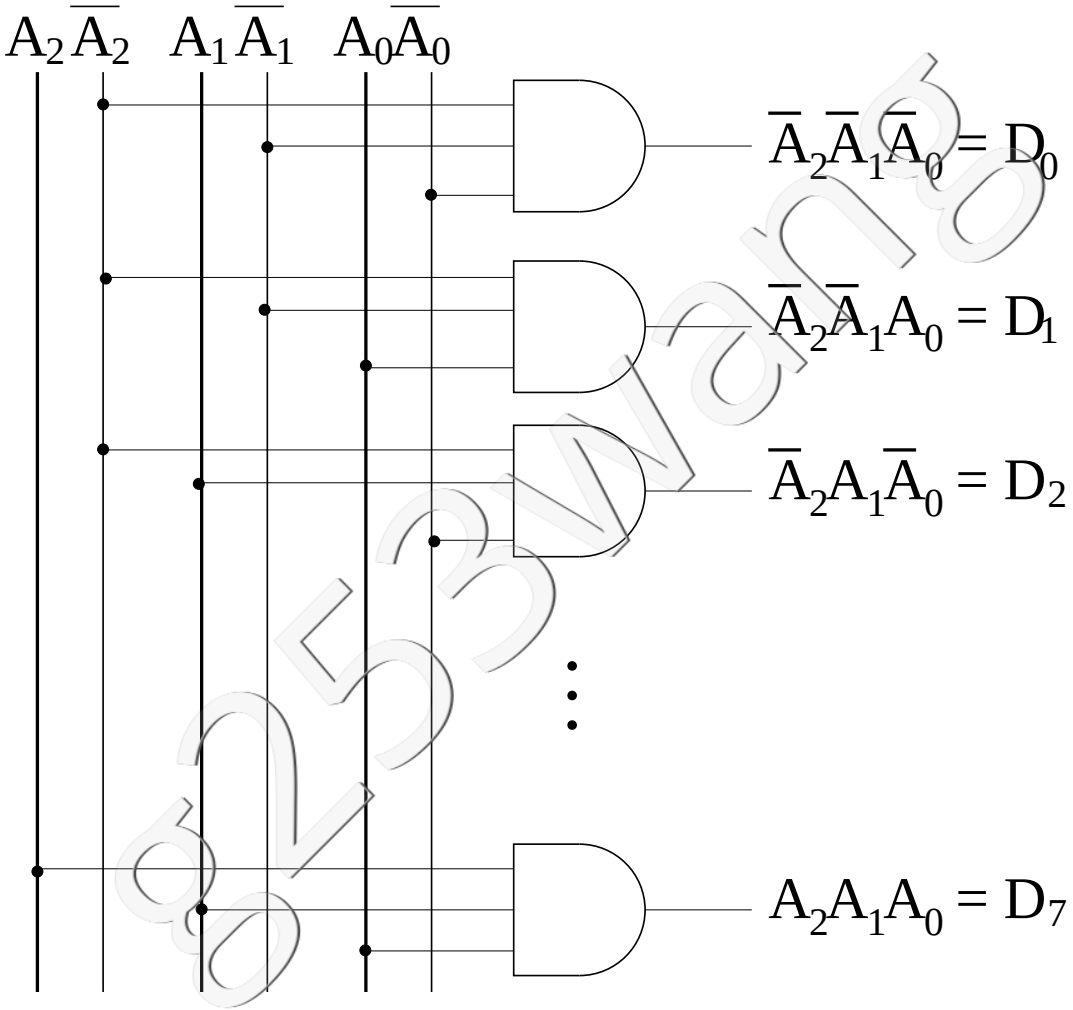
Useful Components: Decoders

- n inputs, 2^n outputs (converts binary to "unary")
- Example: 3-to-8 (or 3-bit) decoder

A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

- SOP form for each output function, is one minterm

Circuit Design based on SOP form for each output D_0-D_7



3-to-8 (or 3-bit) Decoder

Practical use of decoders

Think About It

- Decoding "addresses", selecting a unit from a number of units of the same kind
- For example:
  - A register from 32 registers
  - A memory chip from a set of memory chips

Use of Decoder

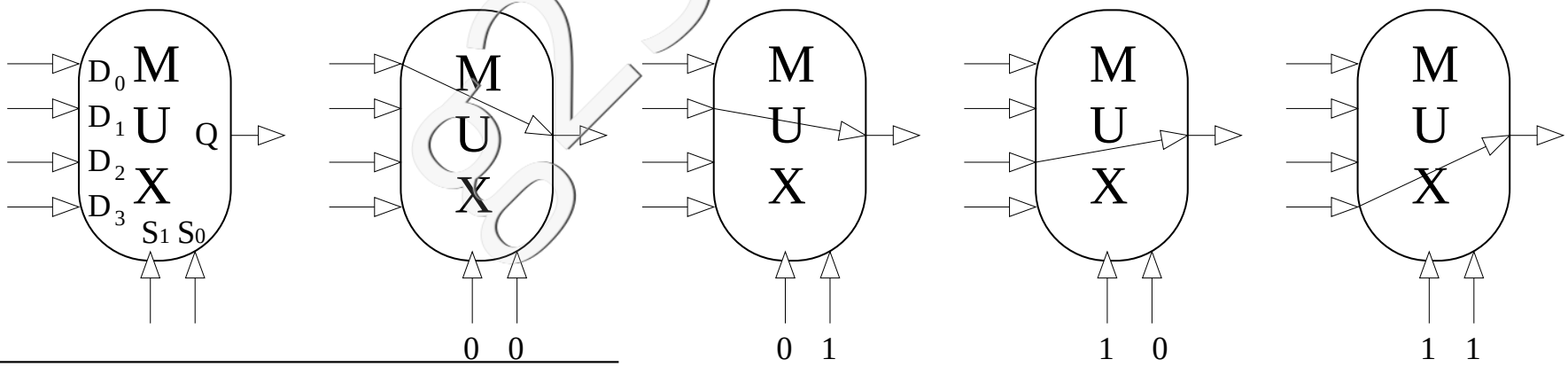
Think About It

- Construct a 1KB Memory from 256×8 memory chips
- Make sure it is a unified address space
- How many address lines to each chip?
- How to select a chip?

Multiplexors[1]

- Inputs: 2^n lines $(D_0, \ldots, D_{2^n-1})$
  n select lines $(S_{n-1}, \ldots, S_0)$
- Output: The value of the D_S line
- Example: 4-1 Multiplexer

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3



[1] also Multiplexers

Partial Truth Table for Multiplexer

Example: 4-to-1 MUX

S_1	S_0	D_3	D_2	D_1	D_0	Y
0	0	0	0	0	0	0
0	0	0	0	0	1	1
0	0	0	0	1	0	0
0	0	0	0	1	1	1
0	0	0	1	0	0	0
0	0	0	1	0	1	1
0	0	0	1	1	0	0
0	0	0	1	1	1	1
0	0	1	0	0	0	0
0	0	1	0	0	1	1
0	0	1	0	1	0	0
0	0	1	0	1	1	1
0	0	1	1	0	0	0
0	0	1	1	0	1	1
0	0	1	1	1	0	0
0	0	1	1	1	1	1

S_1	S_0	D_3	D_2	D_1	D_0	Y
0	1	0	0	0	0	0
0	1	0	0	0	1	0
0	1	0	0	1	0	1
0	1	0	0	1	1	1
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	1	0	1
0	1	0	1	1	1	1
0	1	1	0	0	0	0
0	1	1	0	0	1	0
0	1	1	0	1	0	1
0	1	1	0	1	1	1
0	1	1	1	0	0	0
0	1	1	1	0	1	0
0	1	1	1	1	0	1
0	1	1	1	1	1	1

Partial Compressed Truth Table for Multiplexer

Example: 4-to-1 MUX, when $S_1 = 0$ and $S_0 = 0$

S_1	S_0	D_3	D_2	D_1	D_0	Y
0	0	0	0	0	0	0
0	0	0	0	0	1	1
0	0	0	0	1	0	0
0	0	0	0	1	1	1
0	0	0	1	0	0	0
0	0	0	1	0	1	1
0	0	0	1	1	0	0
0	0	0	1	1	1	1
0	0	1	0	0	0	0
0	0	1	0	0	1	1
0	0	1	0	1	0	0
0	0	1	0	1	1	1
0	0	1	1	0	0	0
0	0	1	1	0	1	1
0	0	1	1	1	0	0
0	0	1	1	1	1	1

S_1	S_0	D_3	D_2	D_1	D_0	Y
0	0	X	X	X	0	0
0	0	X	X	X	1	1

Rearrange truth table

S_1	S_0	Y
0	0	D_0

Partial Truth Table for Multiplexer

Example: 4-to-1 MUX when $S_1 = 0$ and $S_0 = 1$

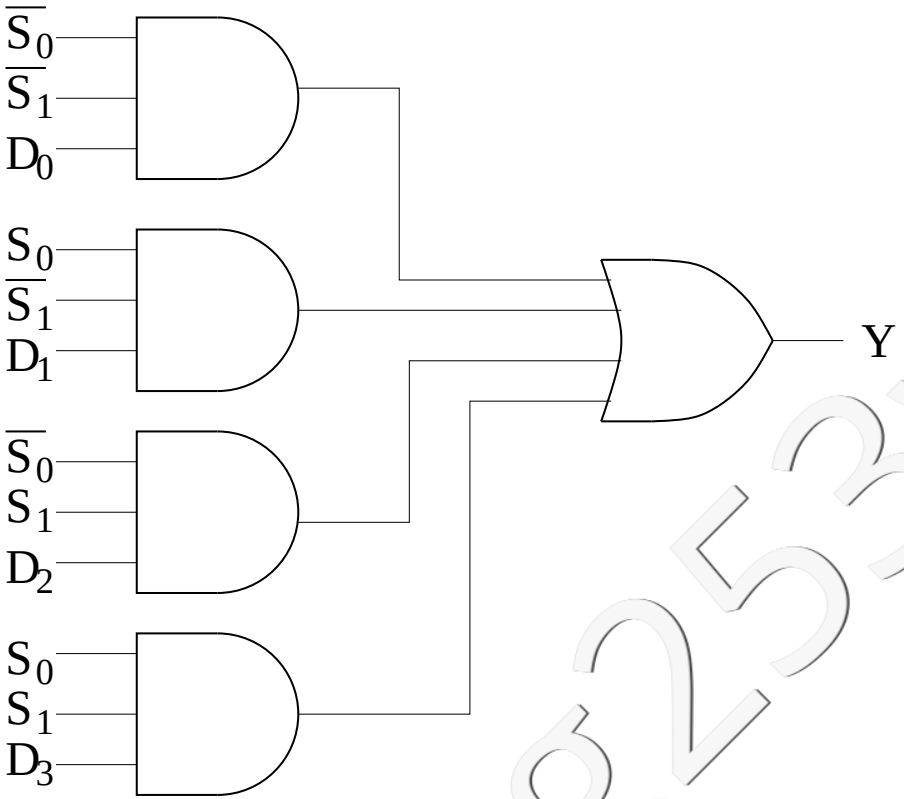
S_1	S_0	D_3	D_2	D_1	D_0	Y
0	1	0	0	0	0	0
0	1	0	0	0	1	0
0	1	0	0	1	0	1
0	1	0	0	1	1	1
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	1	0	1
0	1	0	1	1	1	1
0	1	1	0	0	0	0
0	1	1	0	0	1	0
0	1	1	0	1	0	1
0	1	1	0	1	1	1
0	1	1	1	0	0	0
0	1	1	1	0	1	0
0	1	1	1	1	0	1
0	1	1	1	1	1	1

S_1	S_0	D_3	D_2	D_1	D_0	Y
0	1	X	X	0	X	0
0	1	X	X	1	X	1

Rearrange truth table

S_1	S_0	Y
0	1	D_1

Circuit design of 4:1 MUX using SOP form for Y



Truth Table for 4-1 Multiplexer

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

This truth table is more concise and operation of MUX is clear.

Practical use of MUX

Think About It

- Connect multiple inputs to a single output
- At any time, one of the inputs is selected to be passed to the output
  - If a functional unit is to be used with multiple inputs, a MUX will allow us to select the input to the functional unit
- Perform logic functions
- MUX - a universal combinational logic circuit
  - NAND - a universal logic gate

Example 1: Use of MUX to select input to a functional unit

Think About It

- If a functional unit is to be used with multiple inputs, a MUX will allow us to select the input to the functional unit
- Consider the functional unit ALU to perform ADD. The input can be:
  - contents of two registers
  - content of a register and an `Immediate`

Example 2: Use of MUX to perform logic functions

Think About It

Given the truth table below, use a 2:1 MUX to implement the function F.

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

- inspect the truth table
- combine pairs of rows to suppress right most literal
- use right most literal to express output of MUX

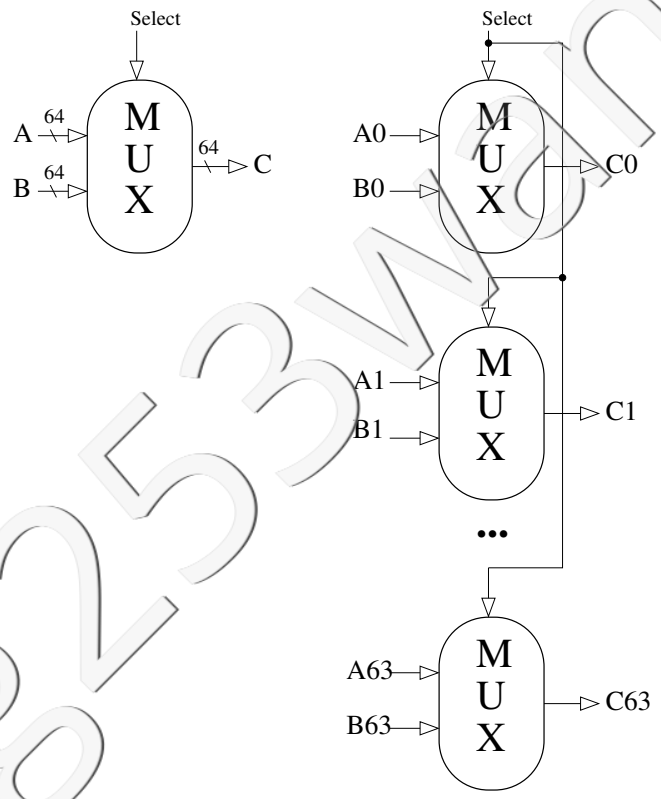
Example 3: MUX - universal logic circuit

Think About It

Use 4 4:1 MUX to design a circuit to achieve circular left shift, with a 4-bit input A and a 4-bit output F.

Arrays of Logic Elements

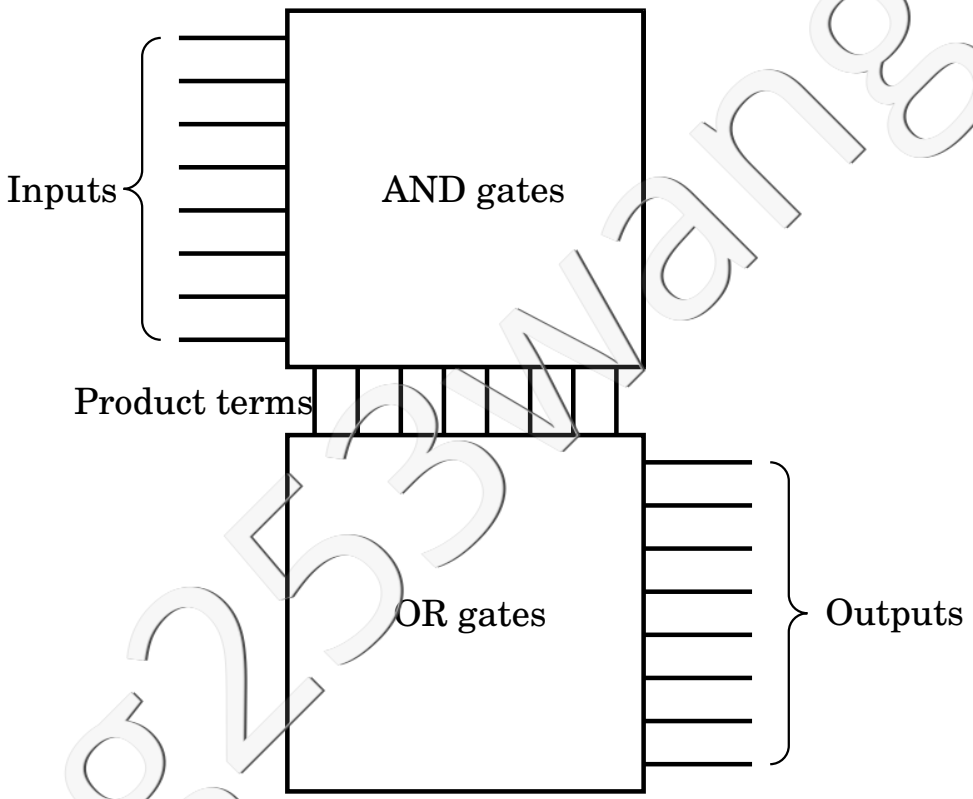
- "Slash" notation is used to indicate lines carrying multiple bits, and to imply parallel constructions



64-bit wide, 2:1 multiplexor expands to 64, 1 bit, 2:1 multiplexors

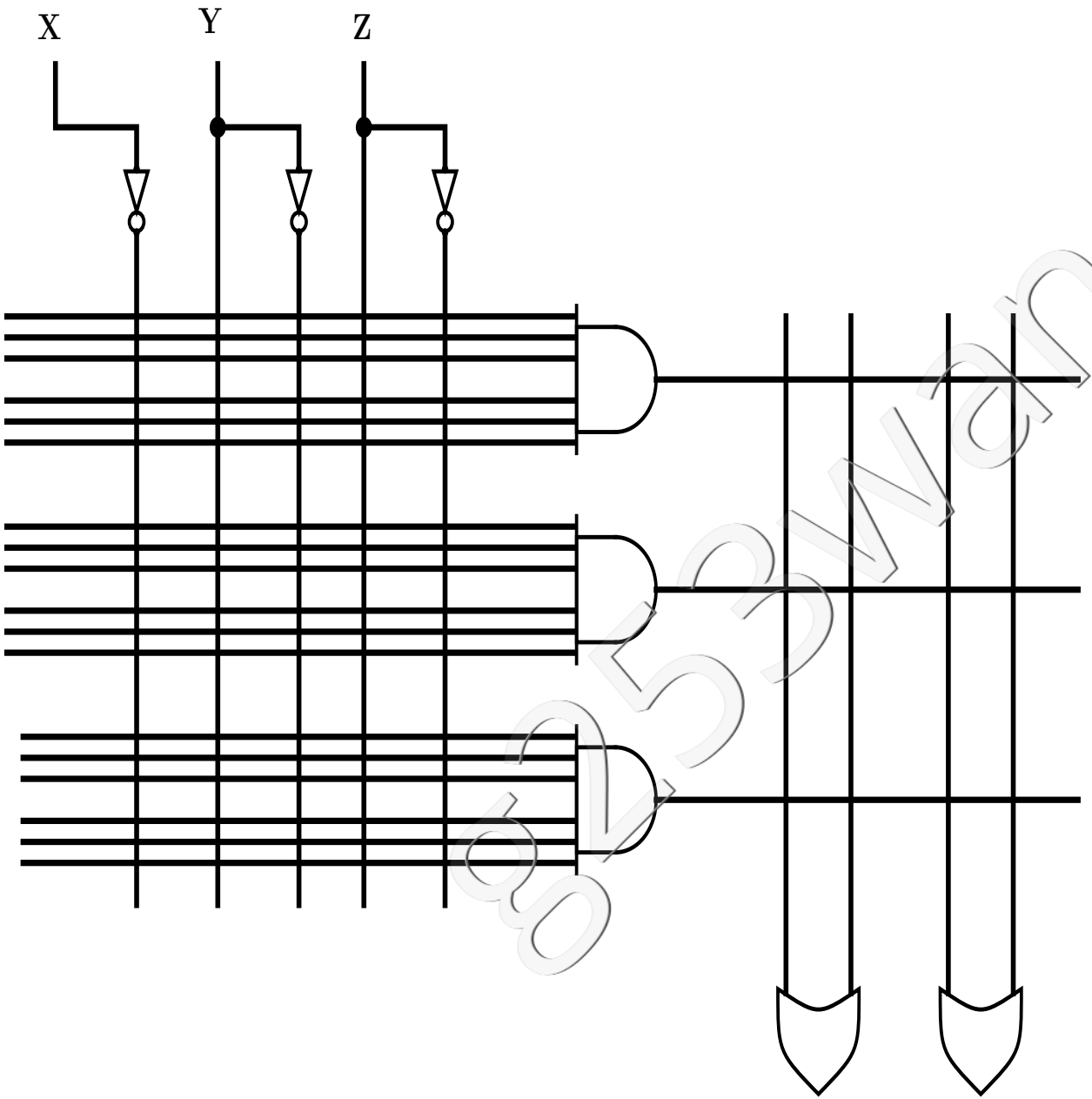
Implementing Boolean Functions: PLAs

- A PLA (Programmable Logic Array) implements a two-level function

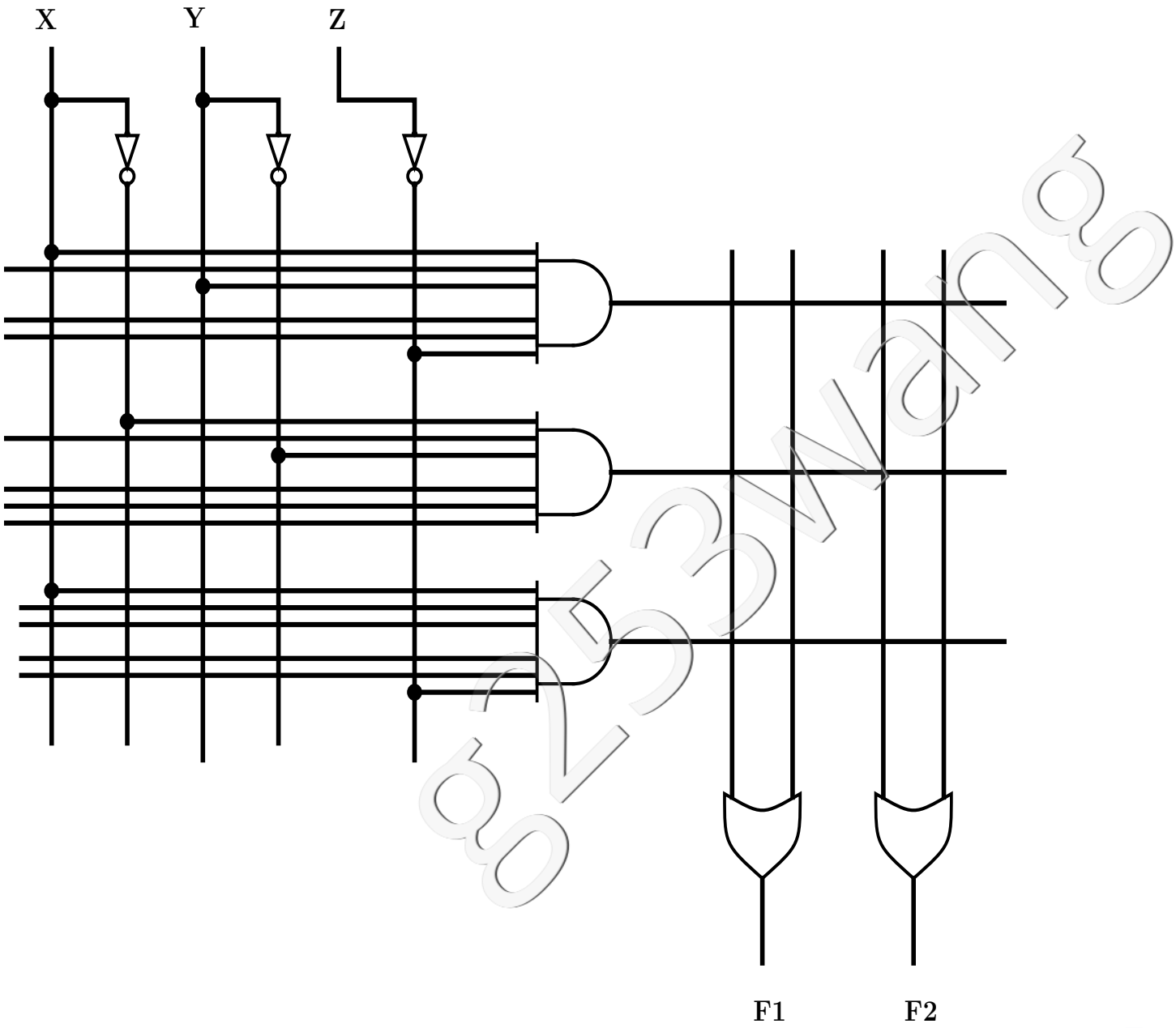


- Typically a PLA has fixed number of product terms and outputs available

Example: 3 input 2 output PLA



Example: Programmed 3 input 2 output PLA



Textbook Readings

- Appendix A-3