CS 271
Computer Architecture & Assembly Language

Lecture 18

Digital Logic Circuits

3/3/22, Thursday

Oregon State University

Odds and Ends

- Due 3/6 11:59 pm
  - Weekly Summary 9

Lecture Topics:

- Digital Logic
- Boolean Logic
- Digital Logic Circuits

Digital Logic
Boolean Logic

What is Digital Logic?

- Discrete circuitry, where the only valid values are a 1 or 0.
- Represented by building blocks of defined functionality.

- An application of boolean logic.

What is Boolean Logic?

- Rules for combining statements that are TRUE or FALSE
- Examples:
  - If statement A is TRUE, “not A” is FALSE
  - If statement A is TRUE, and statement B is TRUE, the combined statement “A and B” is also TRUE
  - If statement A is FALSE, and statement B is TRUE, the combined statement “A or B” is TRUE

Digital Logic, Boolean Logic, and Truth Tables

- AND, OR, and NOT are logical operators
- Define rules for combining statements that are TRUE and FALSE
- Operators can be defined by truth tables

A	NOT A
F	T
T	F

A	B	A AND B
F	F	F
F	T	F
T	F	F
T	T	T

A	B	A OR B
F	F	F
F	T	T
T	F	T
T	T	T

Boolean Notation

- Boolean variables (e.g., A, B, …)
  - Can have value of 0 or 1 (false or true)
- Boolean expressions
  - Join Boolean variables with AND, OR, etc.

Functional	Logical	Boolean
NOT(A)	~A	$\overline{A}$
AND(A, B)	A AND B	AB
OR(A, B)	A OR B	$A+B$
XOR(A, B)	A XOR B	$A \oplus B$
NAND(A, B)	A NAND B	$\overline{AB}$
NOR(A, B)	A NOR B	$\overline{A + B}$
XNOR(A, B)	A XNOR B	$\overline{A \oplus B}$

Using Truth Tables

- Define a Boolean function
  - Specify an output for each possible combination of inputs

- Prove that two functions are equivalent
  - Make a truth table for each function
  - Outputs are identical if and only if the functions are equivalent

Truth Tables: 0 = false, 1 = true

2 input, 1 output

$R = X\overline{Y}$

X	Y	R
0	0	0
0	1	0
1	0	1
1	1	0

3 input, 1 output

$R = X+(YZ)$

X	Y	Z	R
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

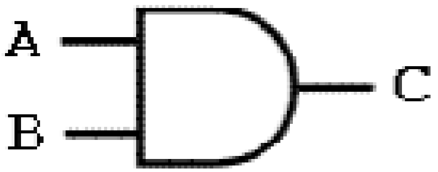
Internal (electric) Representation of Binary Codes

- Power source
- **<u>Gates</u>** (the "building blocks" with defined functionality)
  - Made of one or more transistors
  - Only 2 voltages are permitted
    - **Low (e.g., 0.5v) represents binary 0**
    - **High (e.g., 5.0v) represents binary 1**
  - Can "convert" low ←→ high using gates

- For any given set of inputs (0s and 1s), gates can be combined to produce specified output (0 or 1).
- These combinations of gates are called **digital circuits**.

- <u>Any</u> desired output can be produced by combinations (circuits) of these 3 gates:

AND

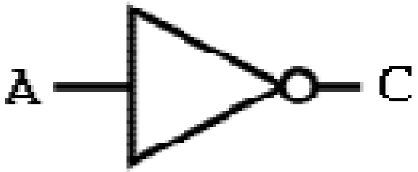


Inputs		Output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

OR

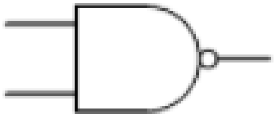
Inputs		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

NOT



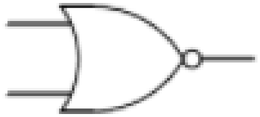
Input	Output
A	C
0	1
1	0

- These additional gates can simplify and/or reduce cost of a circuit



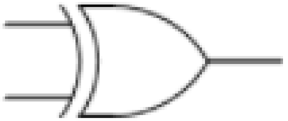
NAND

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



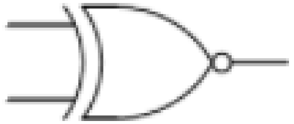
NOR

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



XOR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



XNOR

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Functions

- A function of ***n*** binary variables has 2^n possible combinations of values for the values.
  - E.g., f(A,B,C) has $2^3 = 8$ combinations of values for A, B, and C.

- So … a function can completely described by its **truth table**.

Example:

- Let *f*(A,B,C) be defined by the truth table at the right. To write this in Boolean notation
  - Select all rows with R=1
  - AND the values in each row, using X for 1, X for 0
  - OR the resulting terms
  - Set R = resulting expression

A	B	C	R
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

$$R = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$

- Connection! We have **gates** that can implement this function as a **circuit**.

- In its most primitive form:

$$R = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$

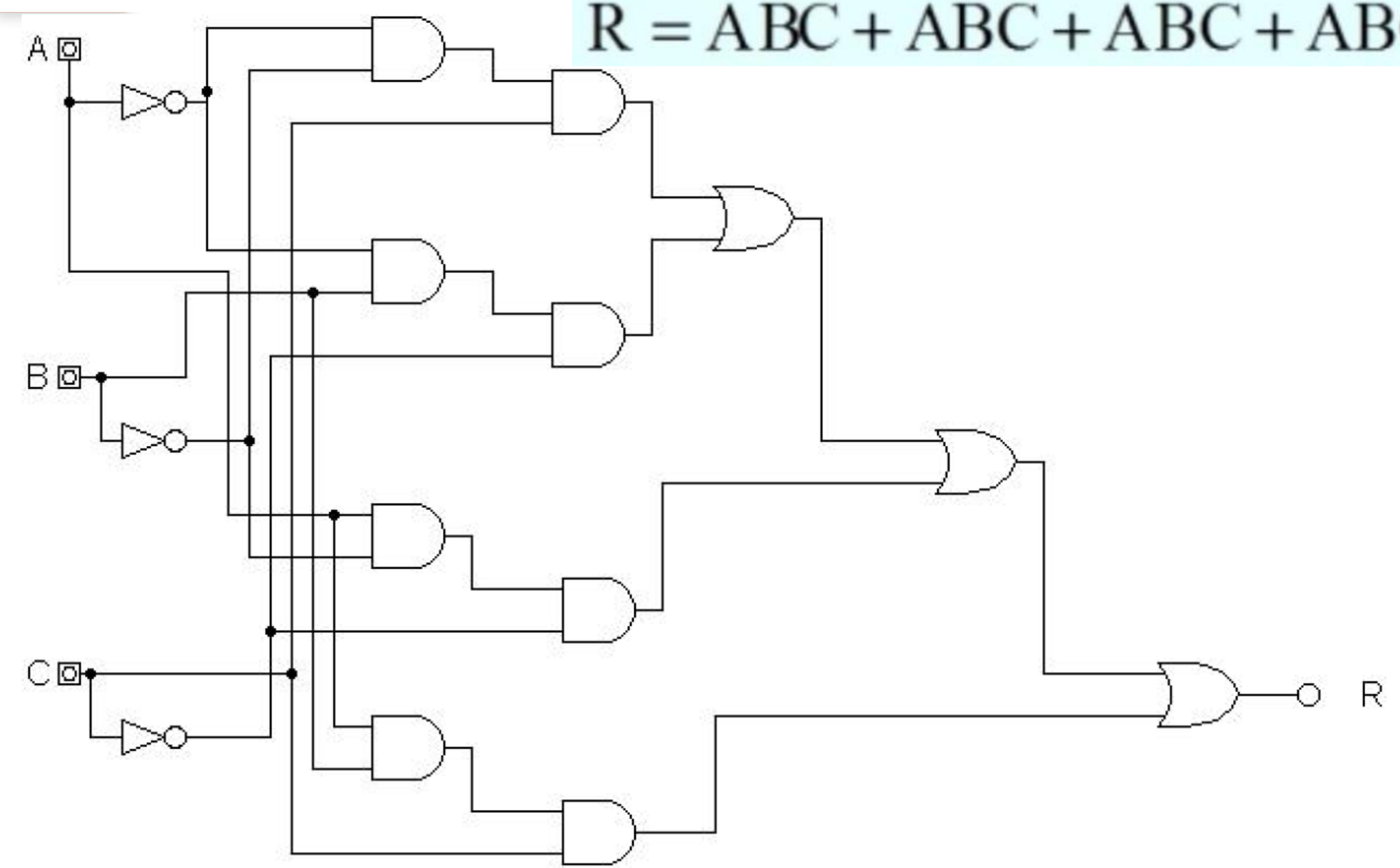
Verification

- Test all possible combinations of input to verify that the circuit implements the function

A	B	C	R
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

$$R = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$

$$R = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$



Simplifications:

- Multiple-input gates
- Equivalent gates
- Use Boolean Logic to simplify the function before implementing the circuit

Boolean Identities

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

Definition: $A \oplus B = A\overline{B} + \overline{A}B$

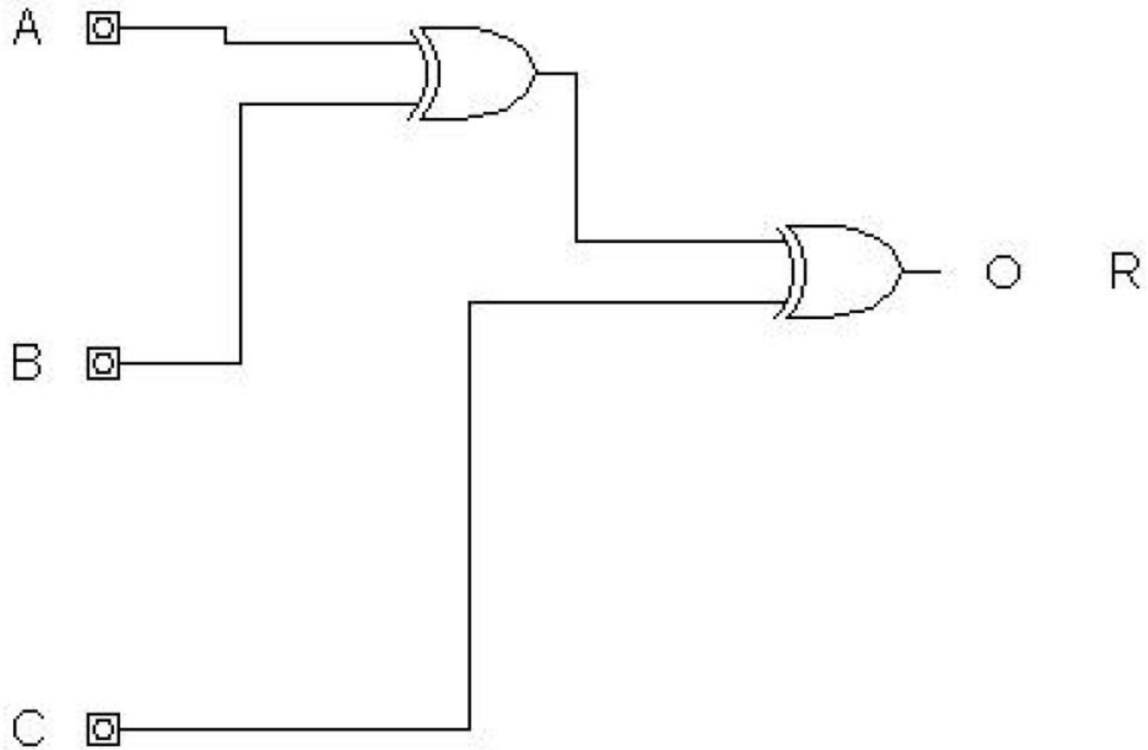
Simplify

$$R = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$

$$R = \overline{A}(\overline{B}C + B\overline{C}) + A(\overline{B}\,\overline{C} + BC)$$ Distributive

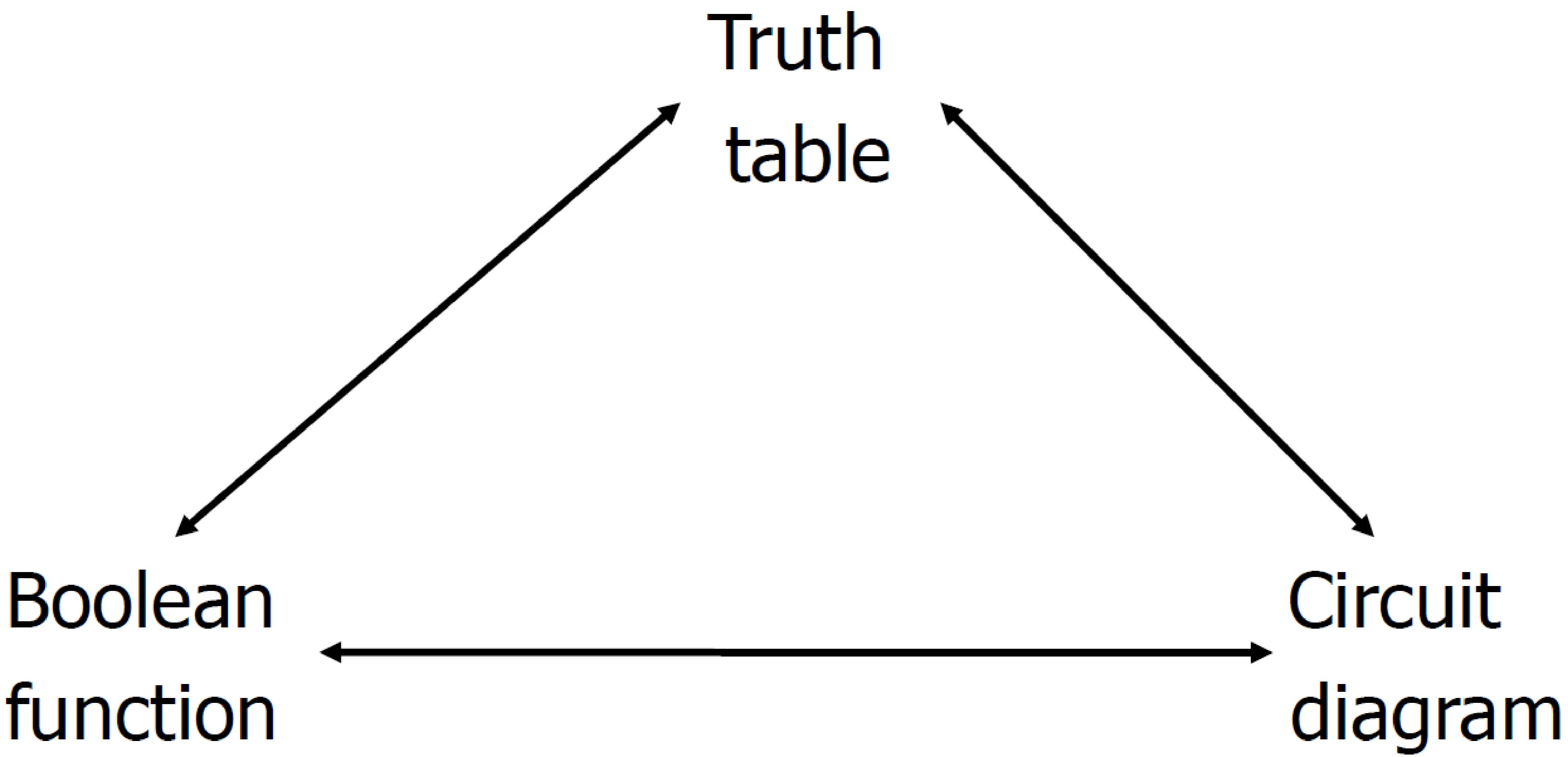
$$R = \overline{A}(B \oplus C) + A(\overline{B \oplus C})$$ Definition, distributive

$$R = A \oplus B \oplus C$$ Definition



$$R = A \oplus B \oplus C$$

Representation of Arithmetic / Logic Circuits

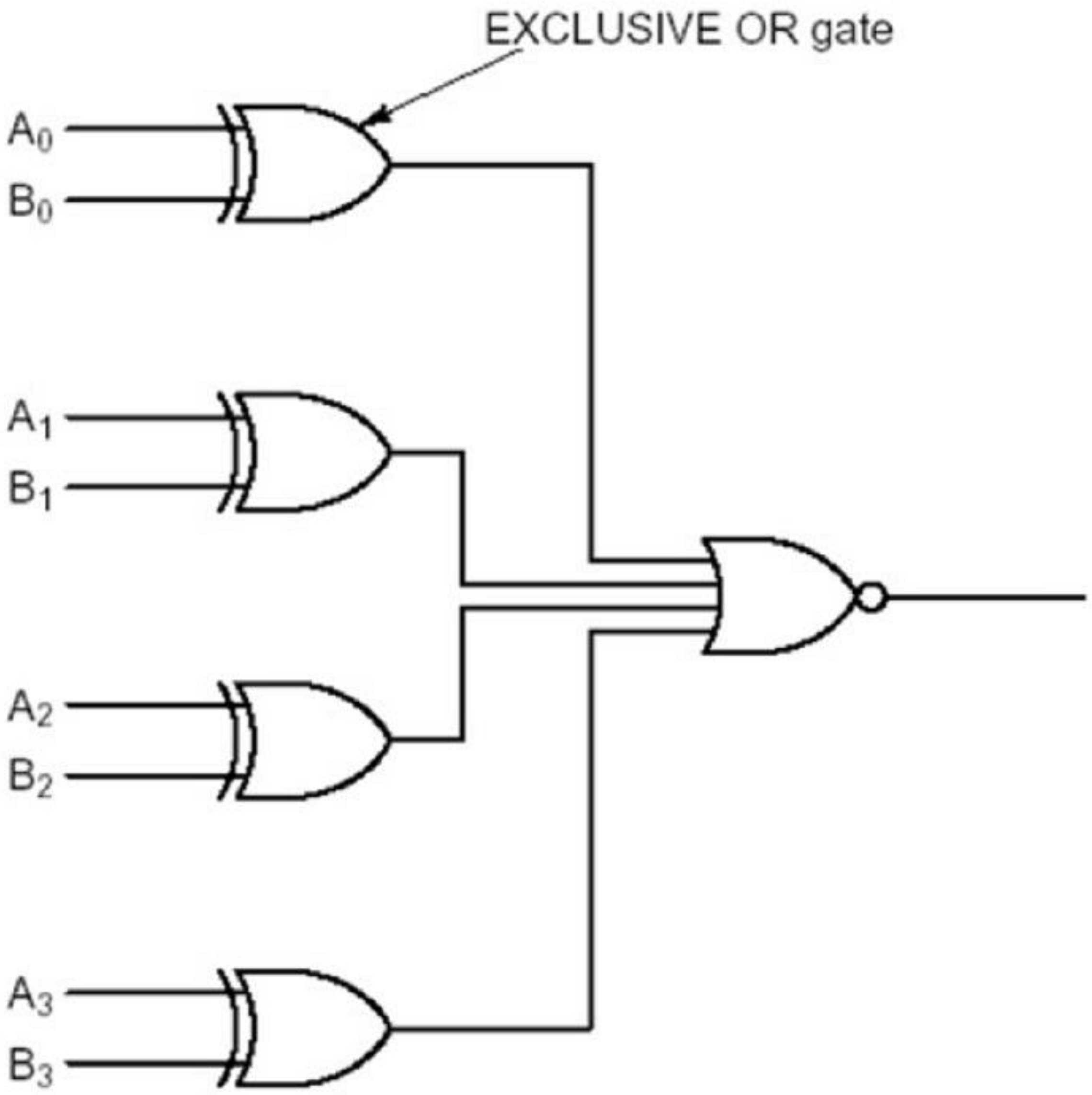


Hardware Circuits

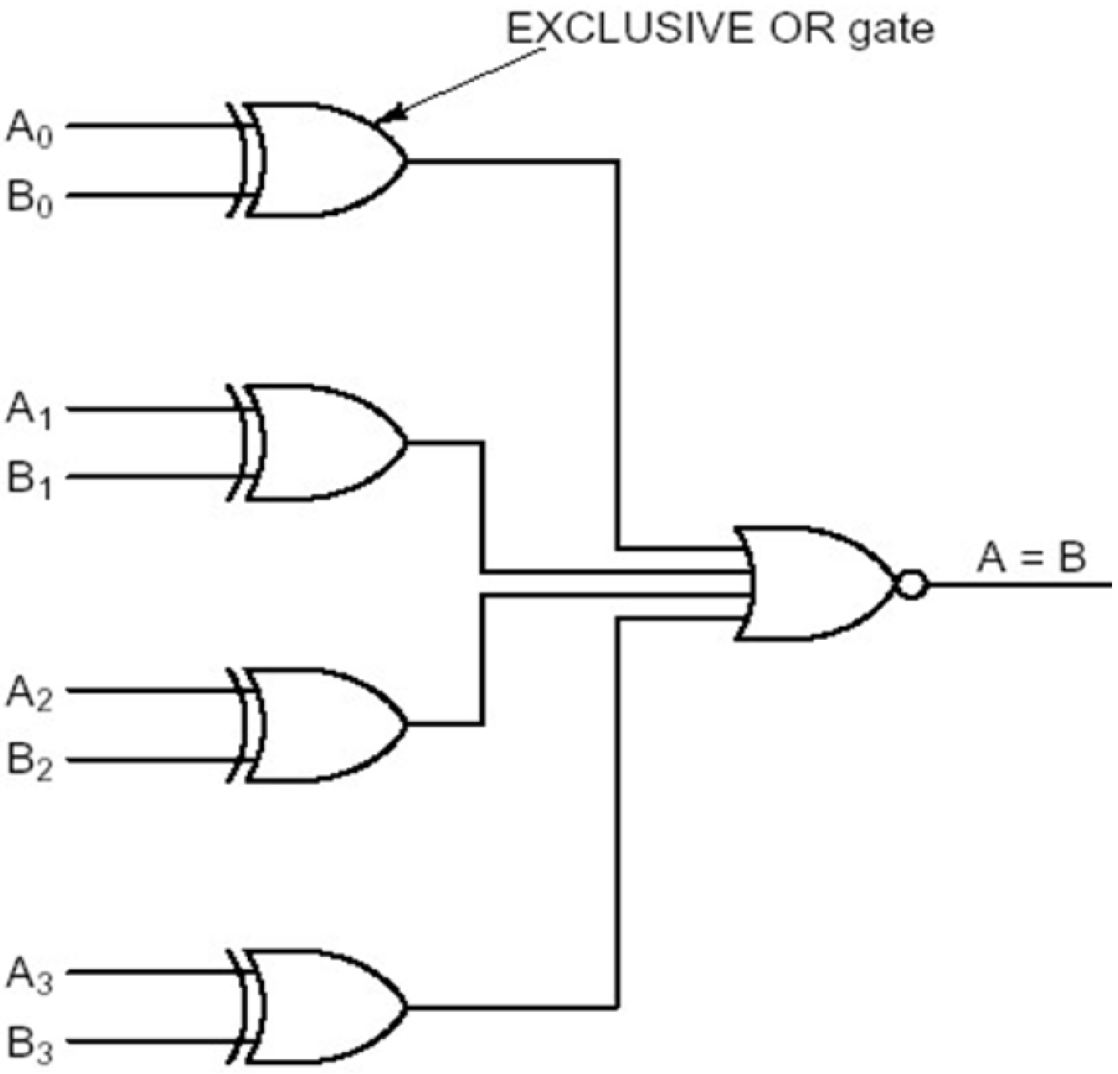
- We all well on our way to showing that an electrical operation can be performed on two electrical numeric representations to give an electrical result that is consistent with the rules of arithmetic.

Digital Logic Circuits

What does this circuit do?



4-bit Comparator



Adders

- Bitwise addition implements “sum bit” and “carry bit”

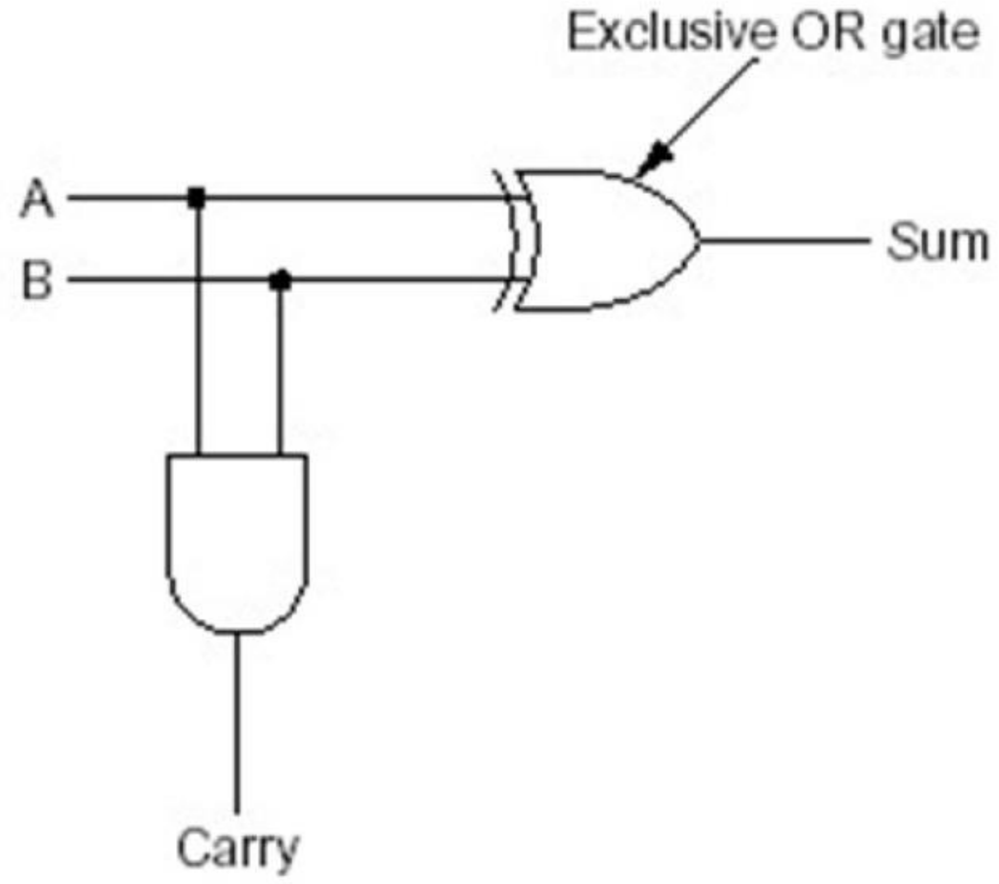
- Carry digits “ripple” to next adder

- Half adder: 2 inputs (corresponding bits of two numbers), 2 outputs (sum bit and carry-out bit)

- Full adder: 3 inputs (same as above, plus carry-in bit), 2 outputs (same as above)

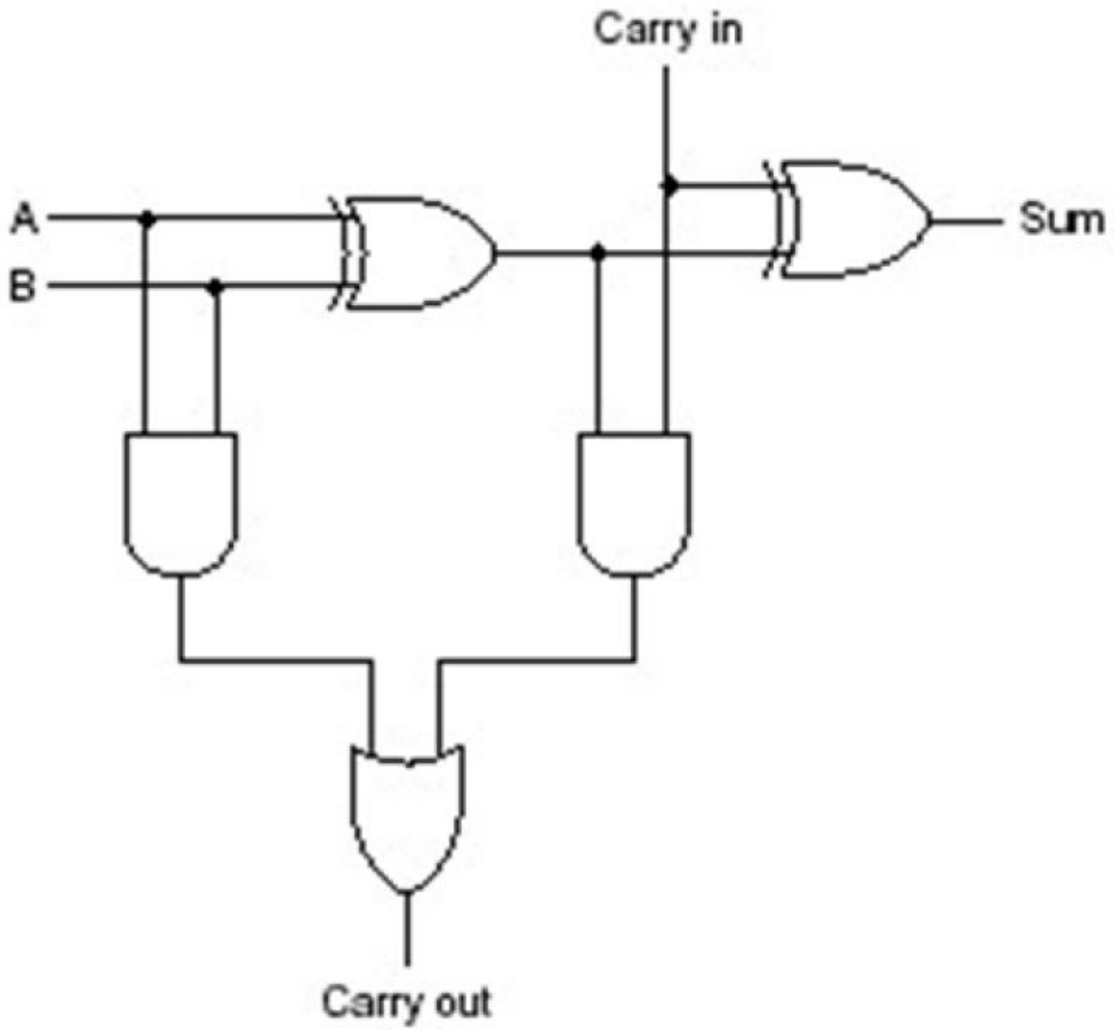
Half Adder

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Full Adder

A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



4-bit Ripple Carry Adder