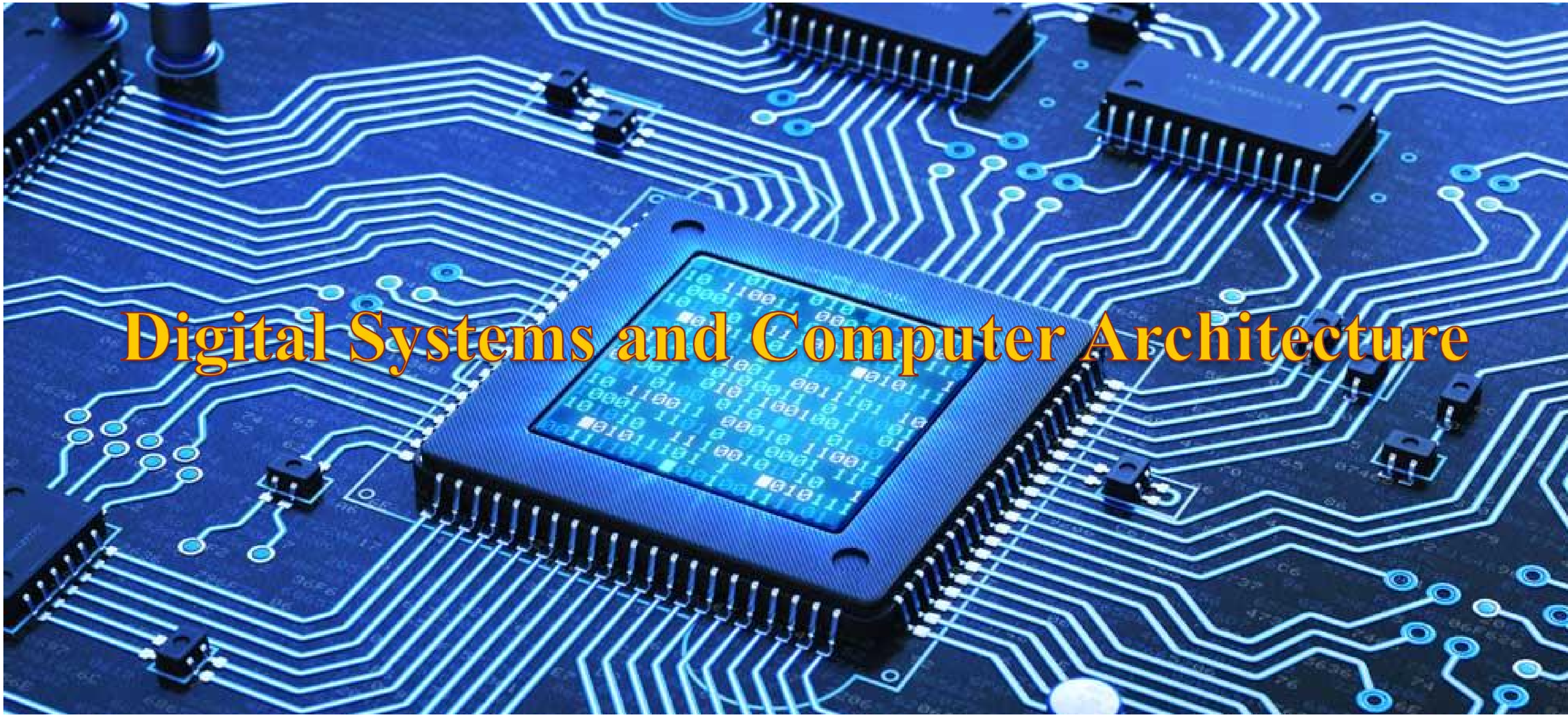


Digital Systems and Computer Architecture

Session 2.3

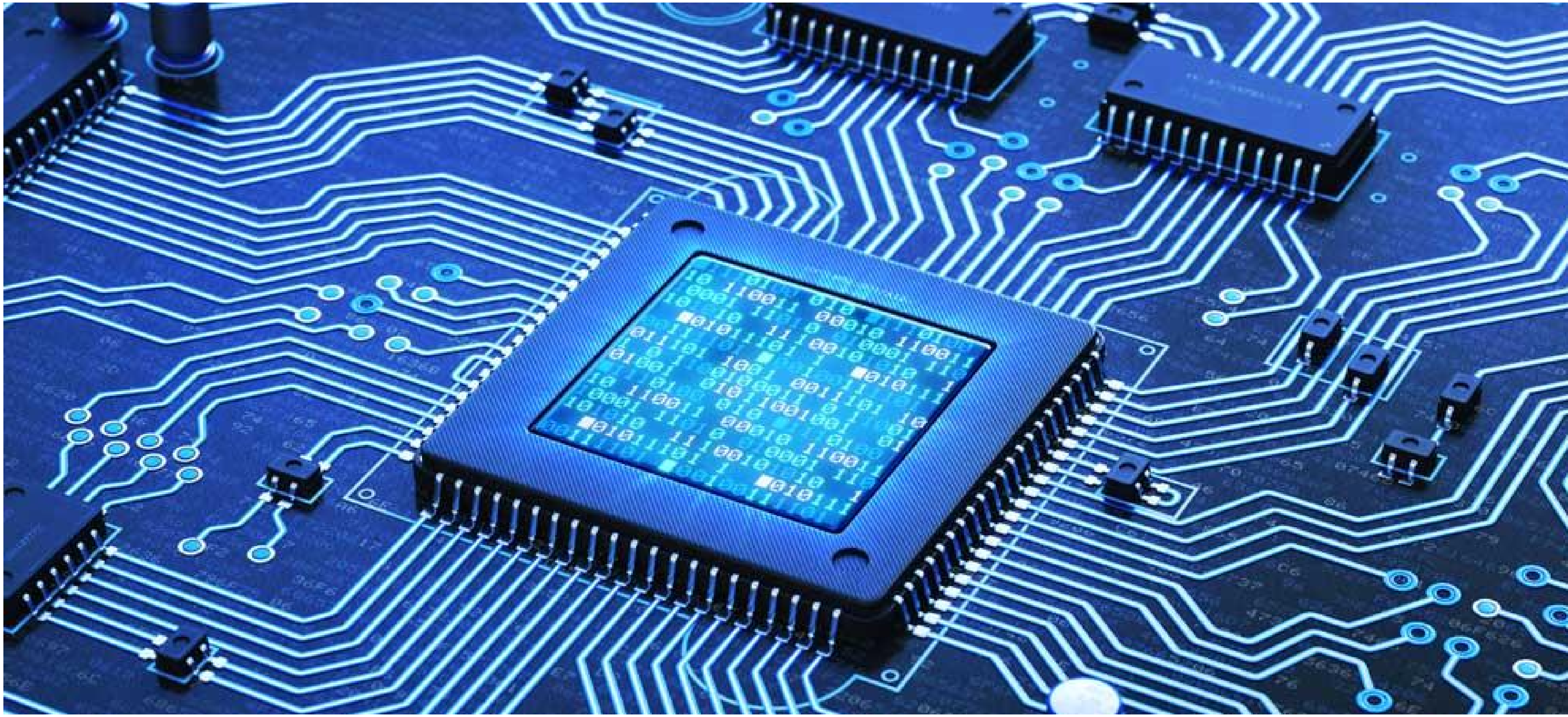
Module 2

Mouli Sankaran

Binary Multiplication and Division

Session 2.3: Focus

- Subtraction using 2's Complement
  - 2's Complement Notation
  - 2's complement subtraction
- Universal Gates
  - NAND and NOR
- Left Shift & Right Shift



Subtraction Using 2’ Complement Arithmetic

1’s and 2’s Complements

- **1’s Complement:**

1 0 1 1 0 0 1 0	Binary number
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	
0 1 0 0 1 1 0 1	1’s complement

- **2’s Complement:**

10110010	Binary number
01001101	1's complement
+ 1	Add 1
01001110	2's complement

2's Complement Explained

- 2's complement is a method of **performing subtraction** using **addition operation**
- Assume, we want to perform: **(9 – 5)**
- It is the same as: **(9)** + **(-5)** → **(9)** + **(0 -5)**
- Let us assume a **4 bit ALU**, for simplicity
- **Let us find out (0 – 5)**
- 0000 | 1 0000 → 1111 + 1 →
- - 0101 | - 0101
- --------- | ------------
- 1011

In binary, subtracting any number from all 1's is nothing but complementing each bit i.e., 1111 – 0101 → 1010 (complementing all bits)

1 } **Adding 1**

+ 1111
- 0101 } **Complementing All bits**

1011

2's complement of 5

It is same as (0 – 5)

So, 2's complement of a number (n) is nothing but (0 – n)

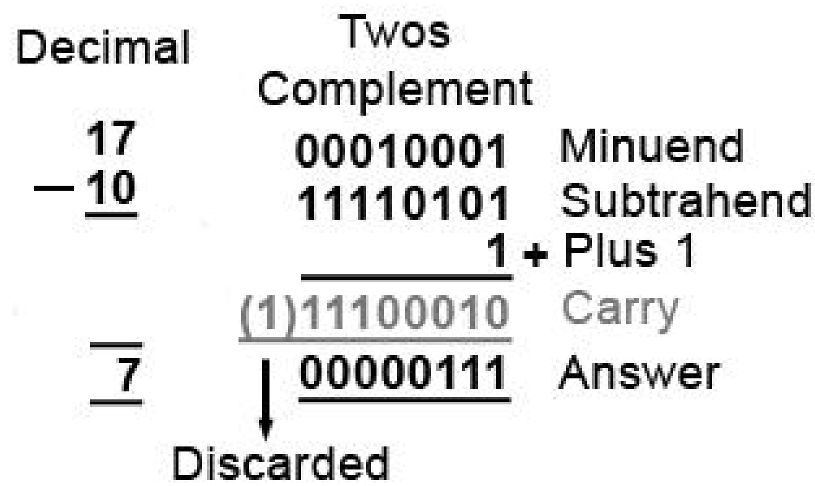
2's Complement Notation

- Negative numbers can be represented using 2's complement notation
- Negative of a number is achieved by complementing all the bits and adding a 1 to it
- For **example**, **-3** can be represented in 2's complement, by complementing 011-> 100 and then add 1 to it; **101**
  - Using three bits
- Only one representation for zero
- Simpler HW implementation for adder/subtractor

Bits	Unsigned Value	Two's Complement Value
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-4
101	5	-3
110	6	-2
111	7	-1

2's Complement Subtraction

- Subtraction is similar to addition.
- **Adding 2's complement** of the **subtrahend to** the **minuend** and **disregarding the carry**, if any, achieves **subtraction**
  - **Example: 17 – 10 = 7**
- In 2's complement arithmetic, the **answer** is also in **2's complement notation**



Quiz 1: Choose the Correct Option

- If a **32 bit** register has a value **0xFFFF FFFF** stored in it, as per 2's complement notation, the equivalent **decimal value** is:
  A. -1
  B. $(2^{32} - 1)$
  C. Both A and B
  D. None of the above

Answer: A

Note: Assume that the value stored is a **signed integer**.

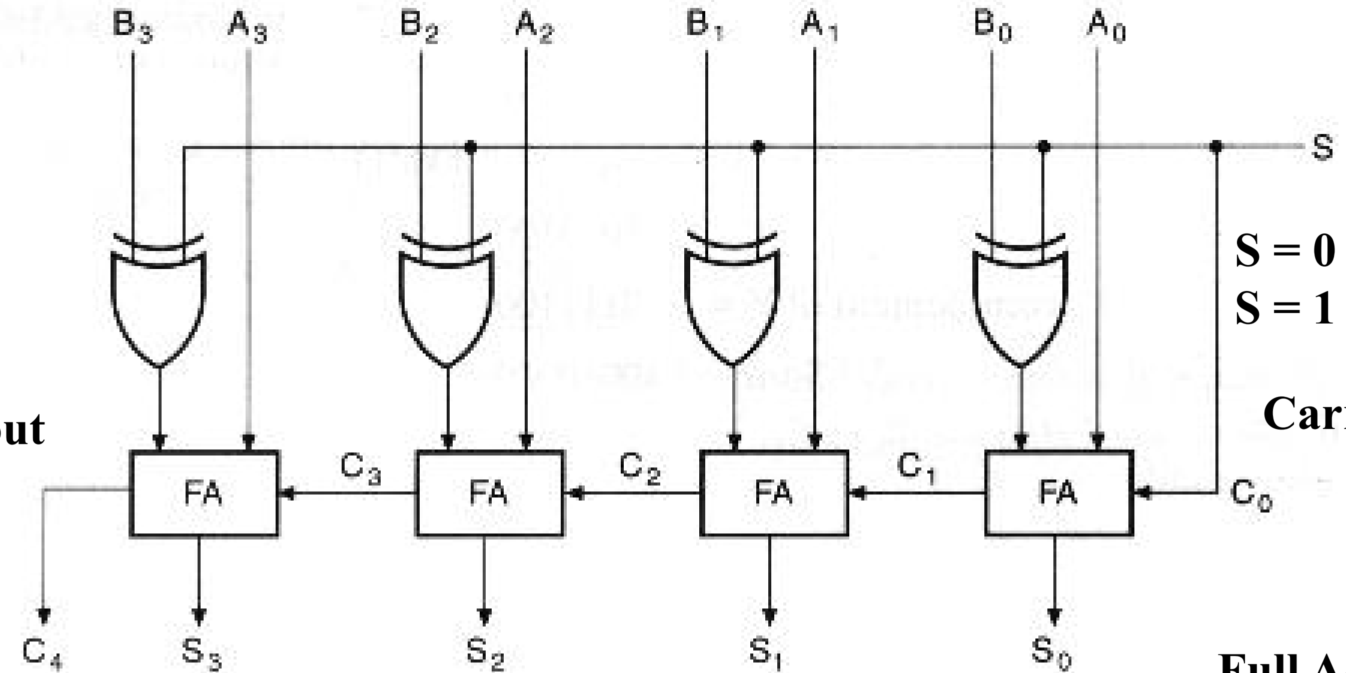
- If a **32 bit** register has a value **0xFFFF FFFF** stored in it, as per 2's complement notation, the equivalent **decimal value** is:
  A. -1
  B. $(2^{32} - 1)$
  C. Both A and B
  D. None of the above

Answer: B

Note: Assume that the value stored is a **unsigned integer**.

4-bit Adder/Subtractor Circuit

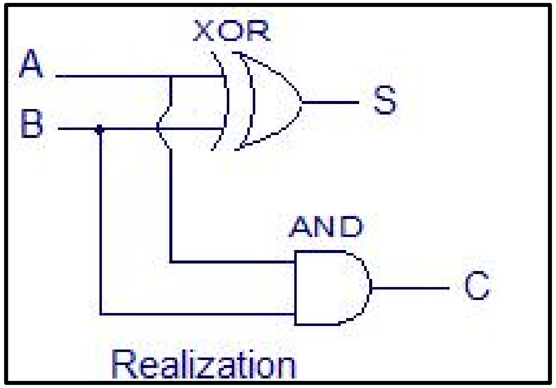
Carry out



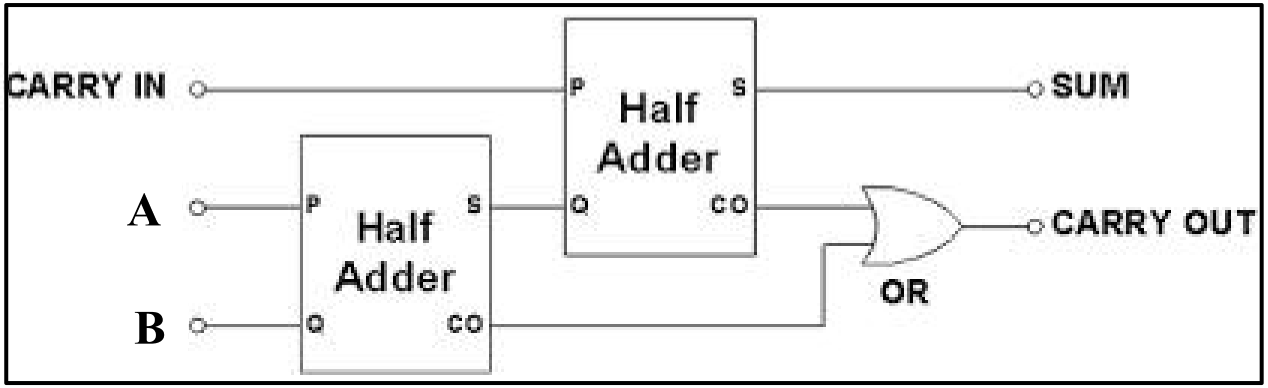
Carry in

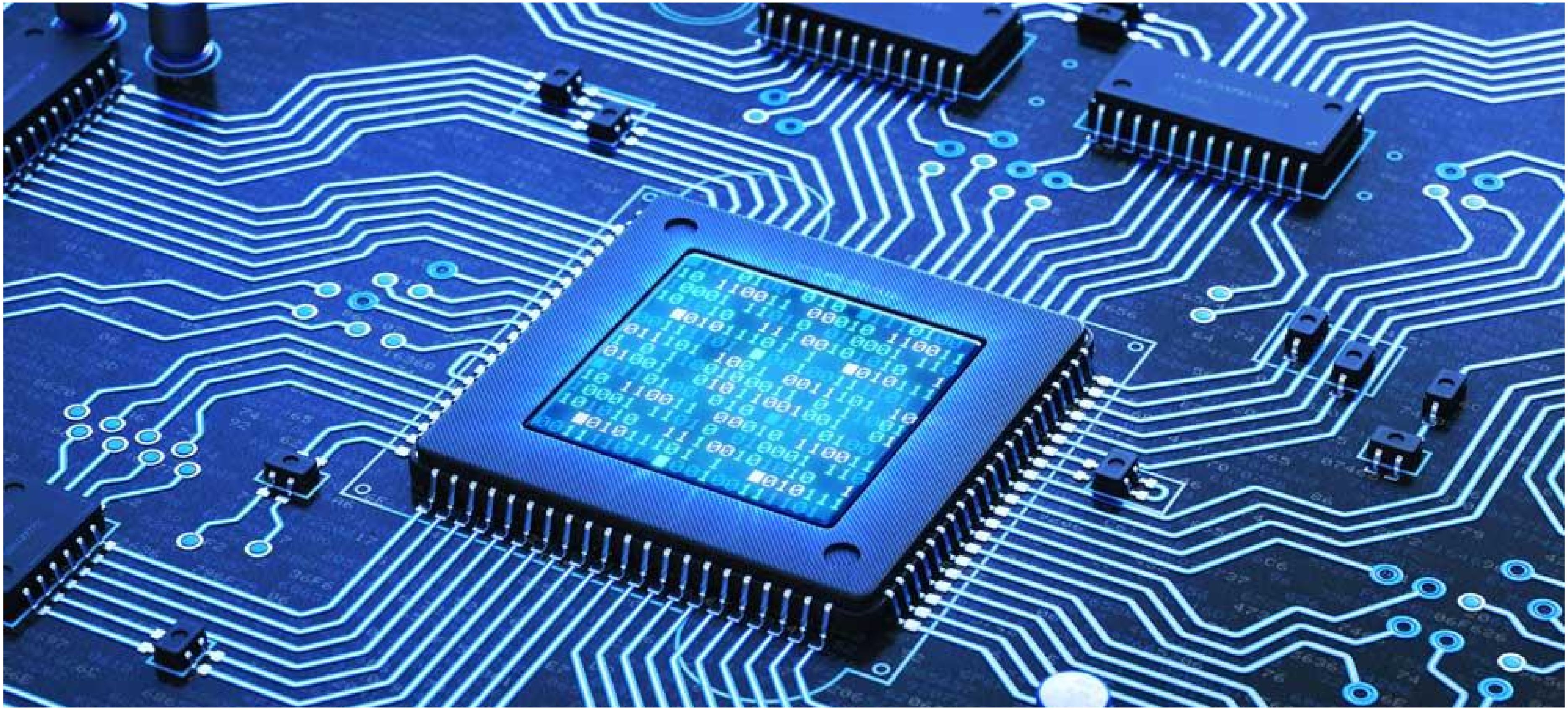
$S = 0$ (addition)
$S = 1$ (subtraction)

Half Adder



Full Adder (FA)



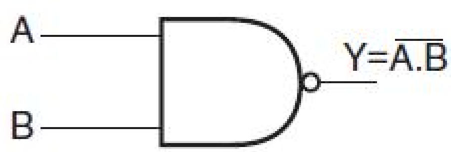


Universal Gates

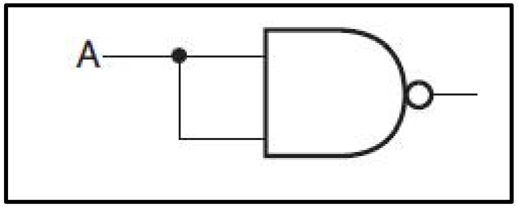
Universal Gates

- **Universal gate** is the one **with** which **all the basic logic gates** can be **built**
  - Basic logic gates are: **AND, OR, NOT**
- Can we build AND, OR, NOT gates **with** only **NAND** gates? **- Yes!!!**

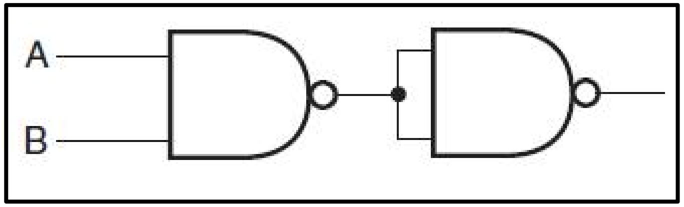
NAND



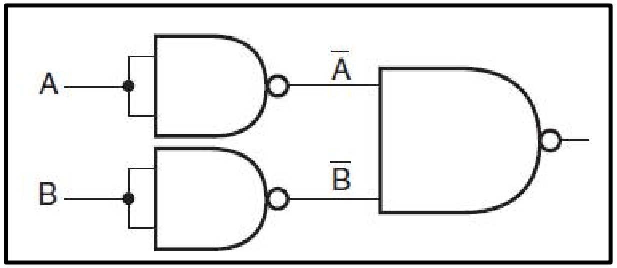
NOT: $Y = \overline{A}$



AND: A . B



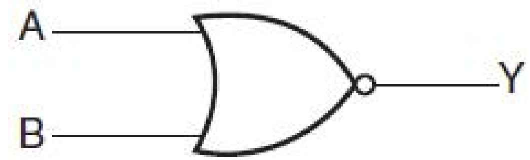
OR: A + B



Y = A + B

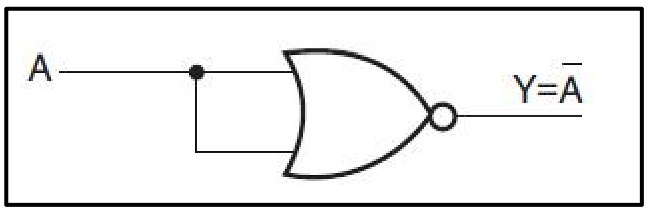
Quiz 1: Is NOR an Universal Gate?

NOR

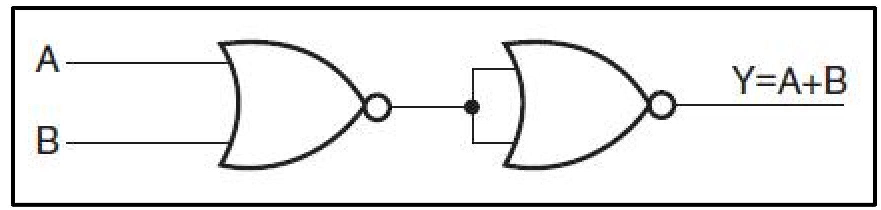


- Yes!!!

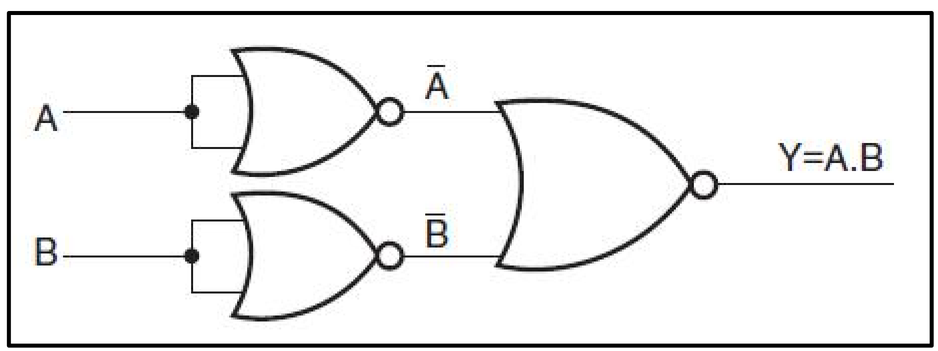
NOT: $Y = \overline{A}$



OR: A + B



AND: A . B



Binary Left Shift(<<)

- **Left Shift(<<):**

It is a binary operator that takes two numbers, left shifts the bits of the first operand, and the second operand decides the number of places to shift.

In other words, left-shifting an integer "a" with an integer "b" denoted as '(a<<b)'

is equivalent to multiplying a with 2^b (2 raised to power b).

Example: Let's take a=5; which is 101 in Binary Form.

Now, if "a is left-shifted by 2" i.e a=a<<2 then a will become a=a*(2^2). Thus, a=5*(2^2)=20 which can be written as 10100.

Binary Right Shift(<<)

Right Shift(>>):
It is a binary operator that takes two numbers, right shifts the bits of the first operand, and the second operand decides the number of places to shift.
In other words, right-shifting an integer "**a**" with an integer "**b**" denoted as '**(a>>b)**' is equivalent to dividing a with 2^b.

Example: let's take **a=5**; which is **101** in Binary Form. Now, if "*a is right-shifted by 2*" **i.e a=a>>2** then **a** will become **a=a/(2^2)**. Thus, **a=a/(2^2)=1** which can be written as **01**.

Session 2.3: Summary

- Subtraction using 2’s Complement
  - 2’s Complement Notation
  - 2’s complement subtraction
- Universal Gates
  - NAND and NOR