Lecture -3
Universal Gates & DeMorgan’s Duality Theorem

Prepared By: Dr. Shahriyar Masud Rizvi

Universal Gates

- NAND and NOR gates are considered universal gates. That means all the other gates can be built purely from NAND or purely from NOR gates.

- It might seem counter intuitive to consider NAND and NOR as universal gates, as NAND and NOR are defined as a NOT operation on an AND gate and a NOT operation on an OR gate, respectively.

- The primary reason behind using NAND and NOR as universal gate is that the dominant implementation technology (circuit level topology) for digital circuits including logic gates is CMOS (Complimentary Metal Oxide Semiconductor) transistor technology.

- In CMOS implementation, the simplest logic gate you can build are NOT, NAND and NOR. You need to add a NOT gate to the output of a NAND gate to realize an AND gate. Similarly, you must add a NOT gate to the output of a NOR gate to realize an OR gate.

- So, a technology library for an industrial IC design tool is sure to have a NOT, NAND and NOR gate in their cell library. Other gates will be built from these 3 gates.

Universal Gates

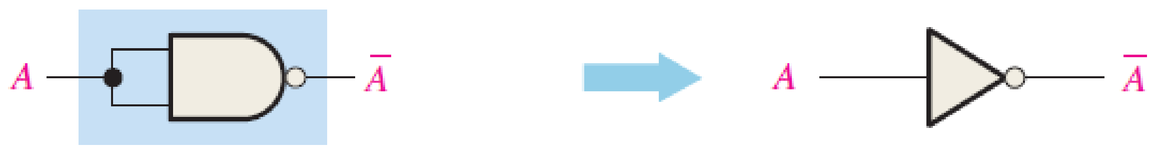
- One of the most important Boolean Algebraic relationship in digital design is DeMorgan’s Duality Theorem.
- DeMorgan’s Duality Theorem relates NAND with OR ($\overline{X.Y} = \bar{X} + \bar{Y}$) and NOR with AND ($\overline{X+Y} = \bar{X} \,.\, \bar{Y}$).
- However, the relationship requires complemented inputs.

- To get a proper OR and AND operation with uncomplimented inputs, one can replace X and Y above with $\bar{X}$ and $\bar{Y}$.
- So, $\overline{\bar{\bar{X}}.\bar{\bar{Y}}} = X + Y$,
- $\overline{\bar{\bar{X}} + \bar{\bar{Y}}} = X.Y$
- One can easily use NAND-NOT topology for an AND gate and NOR-NOT topology for an OR gate.
- NOT operation can be generated from both NAND and NOR by observing the truthtable.

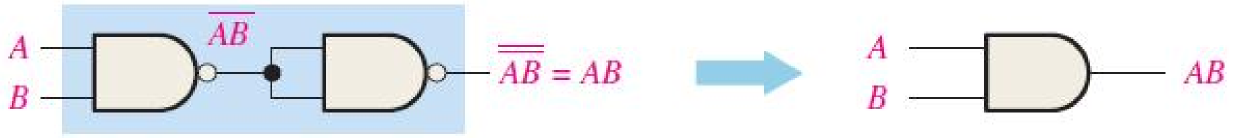
- Note that DeMorgan’s Duality Theorem also relates Pull-Down Network (PDN) and Pull-Up Network (PUN) of CMOS logic. This will be discussed later.

NAND Gate Implementation of Basic Gates

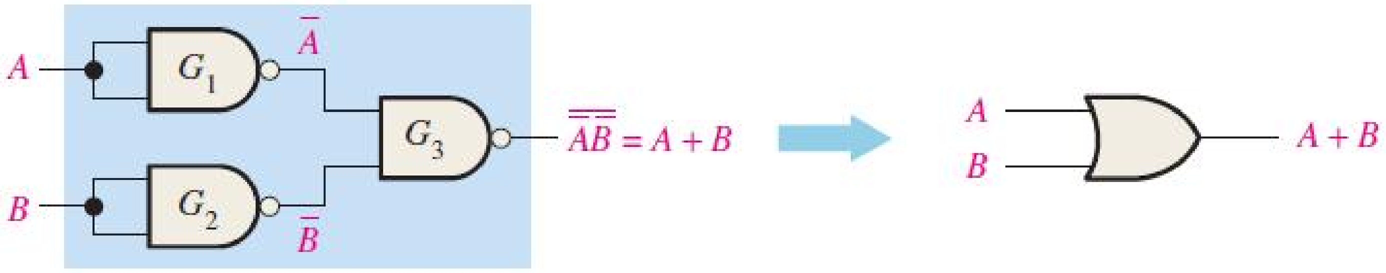
- NOT Gate



- AND Gate

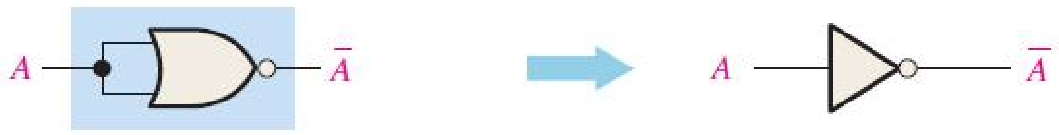


- OR Gate

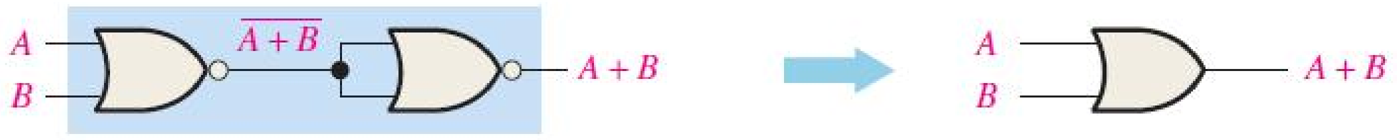


NOR Gate Implementation of Basic Gates

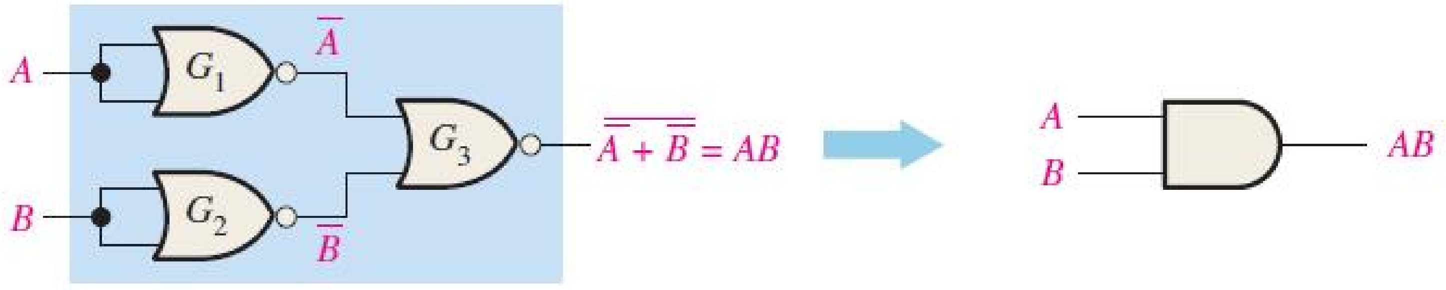
- NOT Gate



- OR Gate



- AND Gate



Comparing NAND, NOR and NOT

- The following table lists the area consumed by basic logic gates in CMOS technology. Here, n is the minimum manufacturable length for a MOS gate terminal. The area calculation will be covered in VLSI Circuit Design course. This table provided here only to make certain points.
- **Observation1:** Implementing a NOT gate with a universal gate such as a 2-input NAND gate is expensive as NOT requires just 30% area of a 2-input NAND gate. There is no speed or area benefit to implementing NOT with universal gates. We will only use universal gate for NOT when NOT gates are not available but universal gates are.

Logic Gate	Area in CMOS Technology	Normalized Area in CMOS Technology (base cell 2-input NAND)	Normalized Area in CMOS Technology (base cell 3-input NAND)
NOT	$6n^2$	0.3	0.17
2-input NAND	$20n^2$	**1**	0.57
2-input AND	$56n^2$	2.8	1.6
2-input NOR	$25n^2$	**1.25**	0.71
2-input OR	$64n^2$	3.2	1.82
3-input NAND	$35n^2$	1.75	**1**
3-input AND	$80n^2$	4	2.28
3-input NOR	$49n^2$	2.45	**1.4**
3-input OR	$100n^2$	5	2.85

Comparing NAND, NOR and NOT

- **Observation2:** A 2-input NOR gate consumes 25% more area than a 2-input NAND gate. When inputs are more, NOR gates consume even more area than NAND gates. A 3-input NOR gate consumes 40% more area than a 3-input NAND gate.

Logic Gate	Area in CMOS Technology	Normalized Area in CMOS Technology (base cell 2-input NAND)	Normalized Area in CMOS Technology (base cell 3-input NAND)
NOT	$6n^2$	0.3	0.17
2-input NAND	$20n^2$	**1**	0.57
2-input AND	$56n^2$	2.8	1.6
2-input NOR	$25n^2$	**1.25**	0.71
2-input OR	$64n^2$	3.2	1.82
3-input NAND	$35n^2$	1.75	**1**
3-input AND	$80n^2$	4	2.28
3-input NOR	$49n^2$	2.45	**1.4**
3-input OR	$100n^2$	5	2.85

Bubble Pushing

- Bubble Pushing utilizes DeMorgan’s Duality Theorem to realize AND-OR networks to primarily NAND networks and OR-AND networks to primarily NOR networks.

- One can also realize NAND-NOR networks from the original logic circuit.

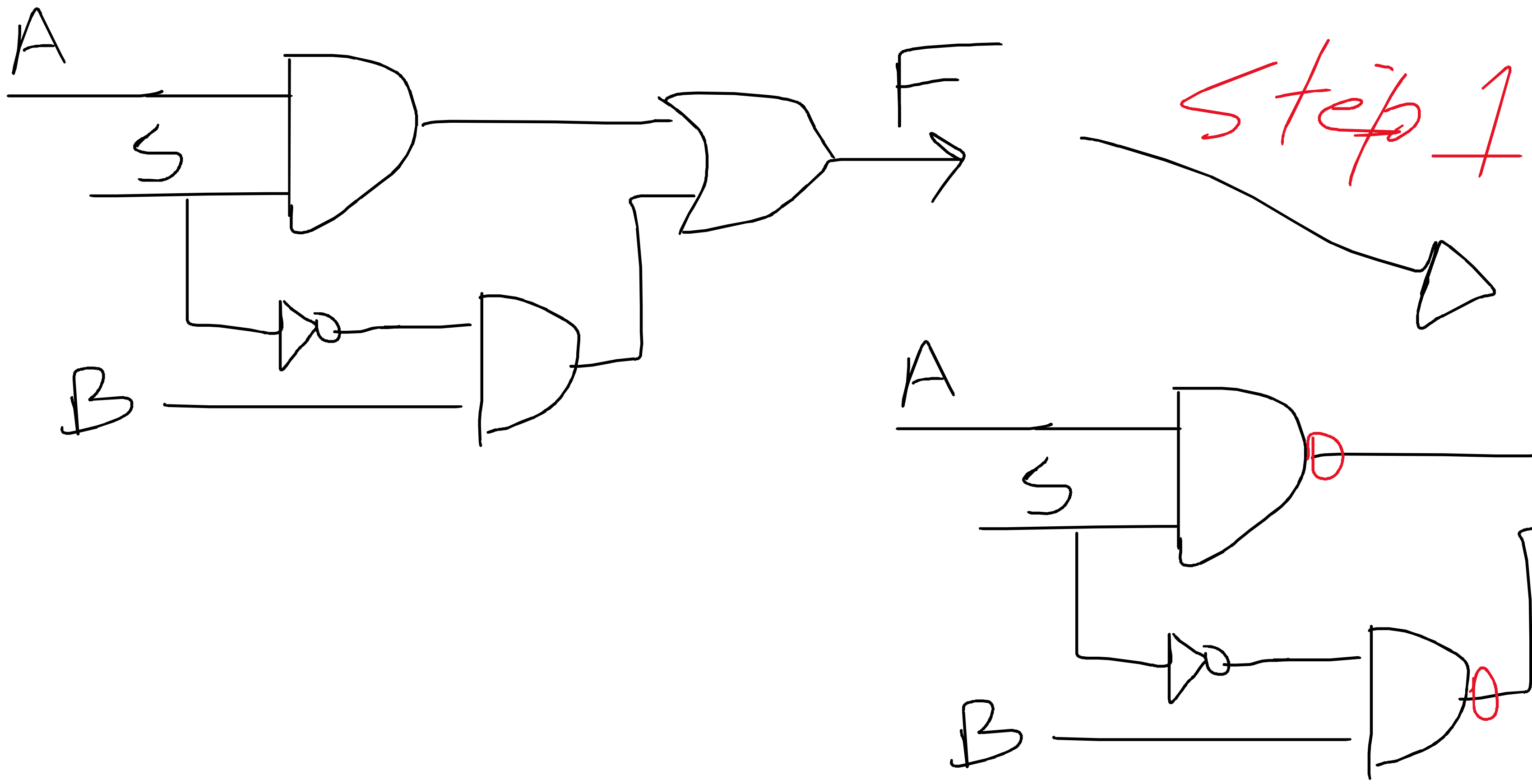
- Here, bubbles (NOT gates) are pushed after AND or OR gates in the first stage and then they are cancelled out by adding additional bubbles before AND or OR gates of the second stage.

- The first stage bubbles convert AND to NAND and OR to NOR.

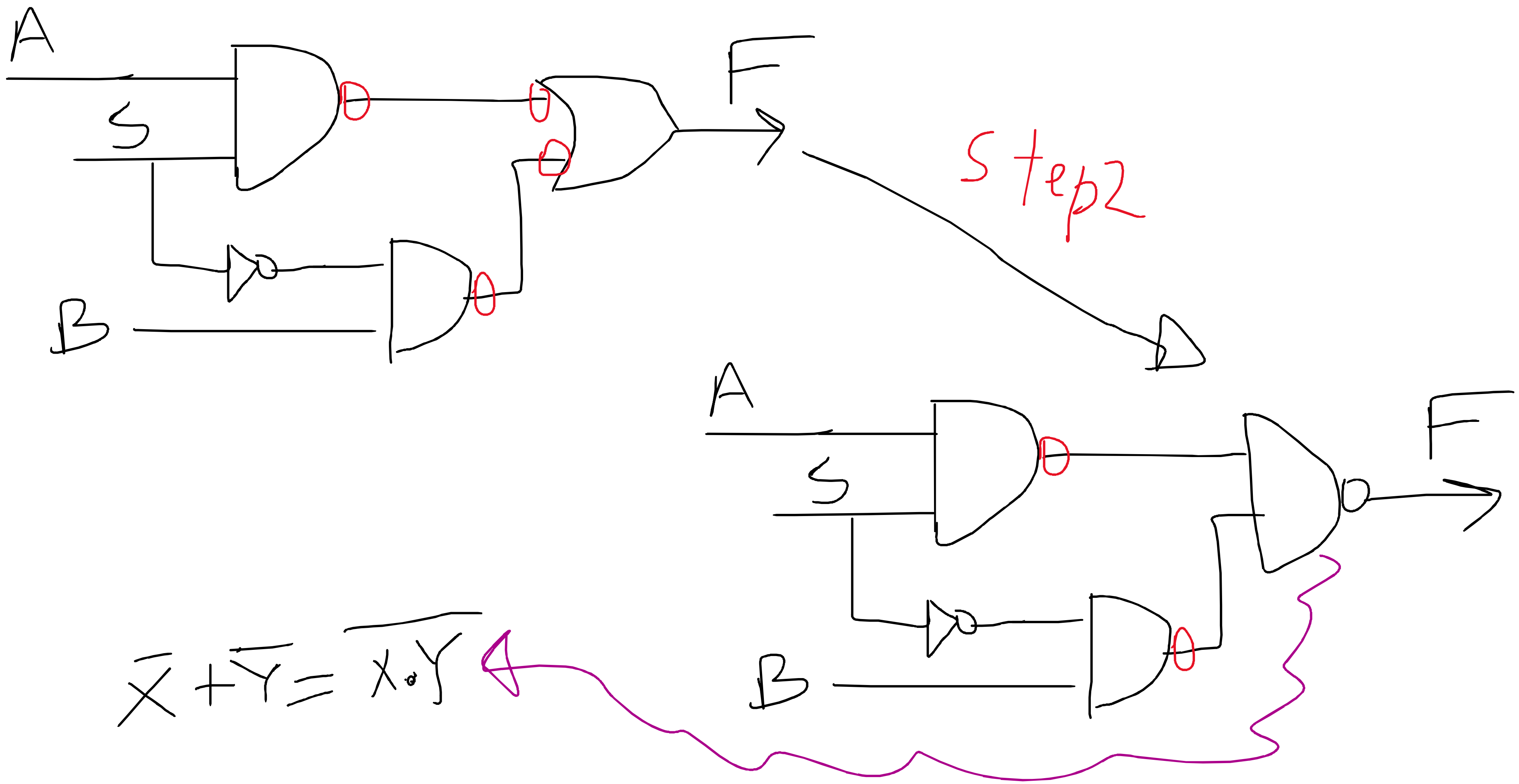
- The second stage bubbles utilize DeMorgan’s Duality Theorem to convert OR to NAND ($\bar{X} + \bar{Y} = \overline{X.Y}$) or AND to NOR ($\bar{X} . \bar{Y} = \overline{X + Y}$).

Bubble Pushing (Step 1)

- Realize the following expression with only NAND and NOT gates using bubble pushing.
- $F = S.A + \bar{S}B$



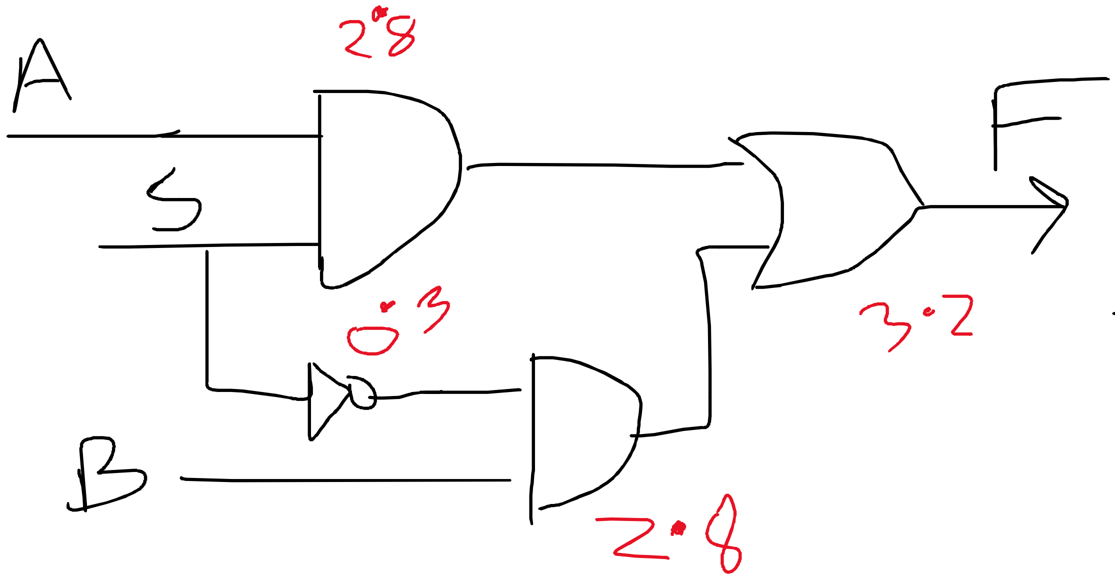
Bubble Pushing (Step 2)

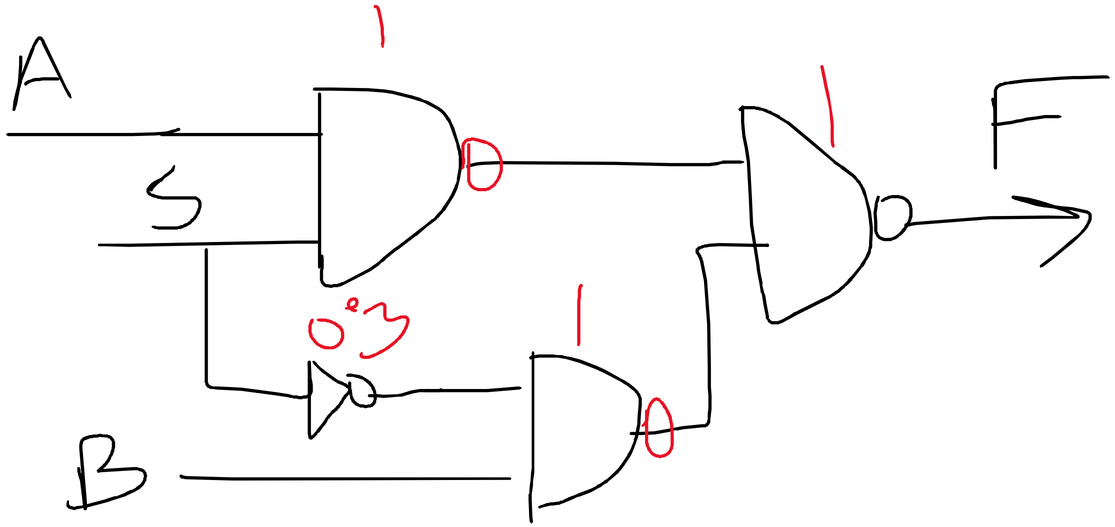


$\overline{X} + \overline{Y} = \overline{X \cdot Y}$

Bubble Pushed Circuit vs. Original Circuit

- Let us compare normalized area of the original circuit and the "bubble-pushed" circuit (using 2-input NAND gate as the basic cell).
- Area of the original circuit = (2.8+2.8+3.2+0.3) = 9.1
- Area of the "bubble-pushed" circuit = (1+1+1+0.3) = 3.3
- So, a bubble-pushed circuit consumes only 36% of the area as compared to the original circuit. In other words, there is a 2.75x improvement (reduction) in area.





Verilog HDL code for the “original” and “bubble-pushed” circuits

<u>Original Circuit</u>

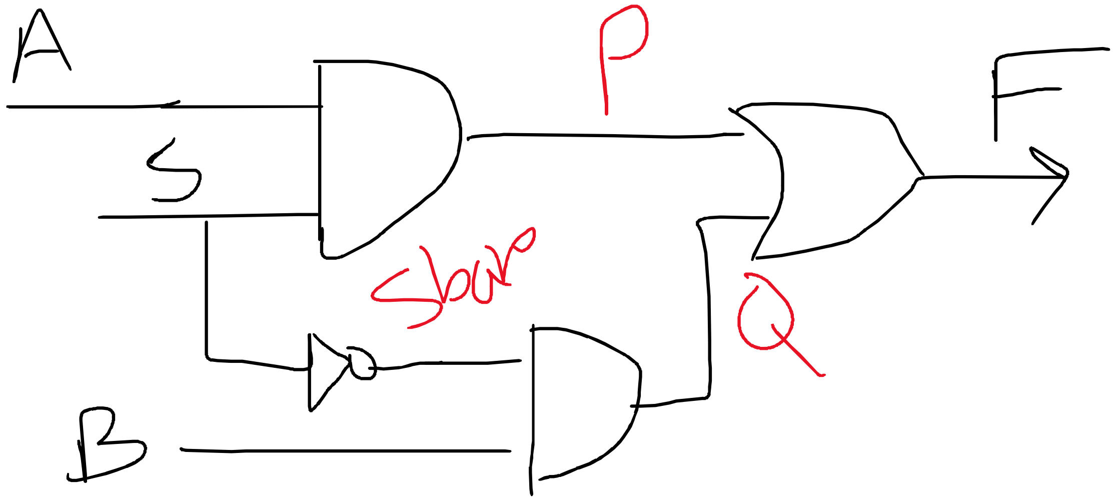
```
//Device name and I/O ports
module original (input wire A, B, S,
                 output wire F);

//Define internal signals
wire Sbar, P, Q;

//Define behavior/structure of the circuit

assign S_bar = ~S;
assign P= A & S;
assign Q= B & Sbar;
assign F= P | Q;

endmodule
```



Verilog HDL code for the “original” and “bubble-pushed” circuits

“Bubble-pushed” Circuit

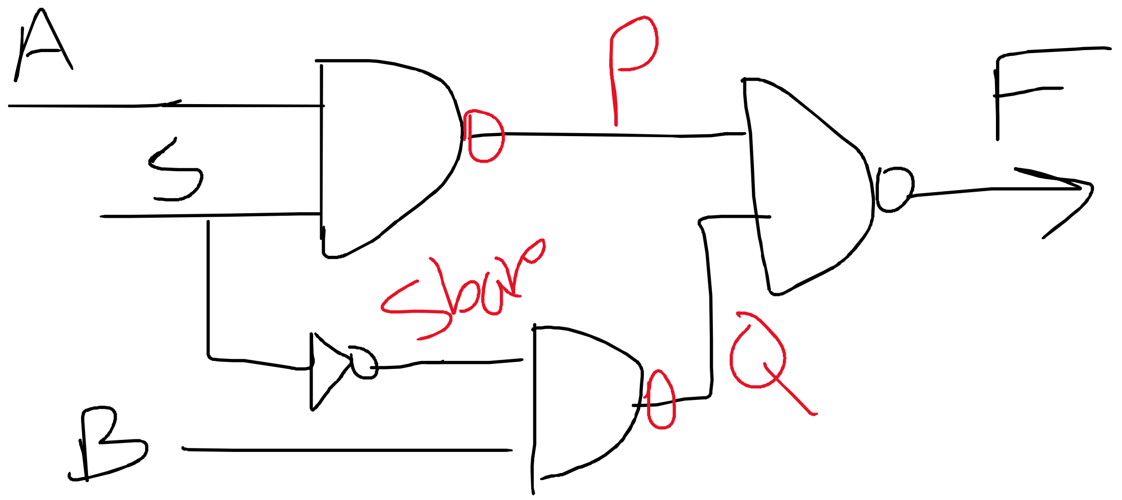
```
//Device name and I/O ports
module bubble_pushed (input wire A, B, S,
                      output wire F);

//Define internal signals
wire Sbar, P, Q;

//Define behavior/structure of the circuit

assign S_bar = ~S;
assign P= A  ~& S;
assign Q= B  ~& Sbar;
assign F= P  ~& Q;

endmodule
```



NAND/NOR Gate Implementation of Boolean Logic

- There are three basic tricks/ways to convert any Boolean Logic Circuit to its NAND/NOR equivalent:
  1. First implement the Boolean Logic with basic AND, OR and NOT gates and then replace the basic gates with its NAND/NOR equivalent.
  2. Convert the Boolean Logic expression to its NAND/NOR form using De Morgan's theorem and then implement it with NAND/NOR gates directly.
  3. Use Bubble Pushing Technique.
- Implement the following Boolean expression with NOR logic:
  a) $\mathrm{F = AB + CD}$
  b) $\mathrm{F = \overline{AB} + CD}$
  c) $\mathrm{F = A(B + CD)}$
  d) $\mathrm{F = C(\overline{A + BD})}$
  e) $\mathrm{F = A\overline{B} + \overline{A}B}$

- Implementation will be shown in class

References

1. Thomas L. Floyd, “Digital Fundamentals” 11th edition, Prentice Hall – Pearson Education.

Thank You