Date __/__/____

Assignment-1

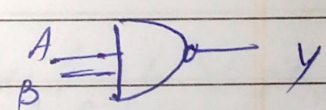
Logic gates :- Make decesions using 0 and 1.

MUX :- Chooses between options
Flip-flop :- Remembers one bit.
Register :- Remembers multiple bits.

Mission 1

Task :- Show how to create these gates using only NAND gate

↓
opposite of AND
$Y = \overline{A \cdot B}$



(a) The NOT Gate :-

The logic :- A NAND gate output is 0 only when both the inputs are 1. By connecting a single input signal to both input pins of the NAND gates, you force the gate to see either (0,0) or (1,1).

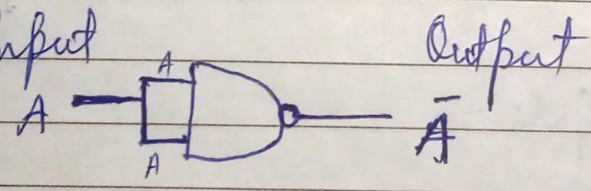
Date ___ / ___ / _____

- Input 0 : Both inputs become 0.
  NAND output is 1.

- Input 1 : Both inputs become 1.
  NAND output is 0.

The Circuit Construction :-

1. Take one NAN gate
2. Connect your input signal A to both input terminals of the gate
3. The output is $\bar{A}$.



Boolean $Y = NAND(A, A)$

$= \overline{A \cdot A} = \bar{A}$

Date ___/___/_____

(b) The AND Gate:-

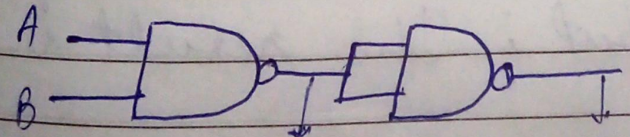
The Logic: A NAND gate is literally ($Y = \overline{A \cdot B}$). To get a regular AND, you simply need to invert the output of a NAND gate. Since we just learned how to make an inverter (NOT gate) in part (a). we can use that here.

The Circuit Construction:-

1. Stage 1:- Connect inputs A and B to a standard NAND gate.
   - Output: $\overline{A \cdot B}$

2. Stage 2:- Connect the output of Stage 1 to both inputs of a second NAND gate (configured) as a NOT Gate.

- Final Output: The double negative cancels out, leaving you with $A \cdot B$.



$\overline{A \cdot B}$ $\overline{\overline{A \cdot B}} = A \cdot B$

Date ___ / ___ / _____

(c) The OR Gate:

The Logic:- This requires a rule called De Morgan's Law, which states that if you invert the inputs before sending them into a NAND Gate, you get an OR function:

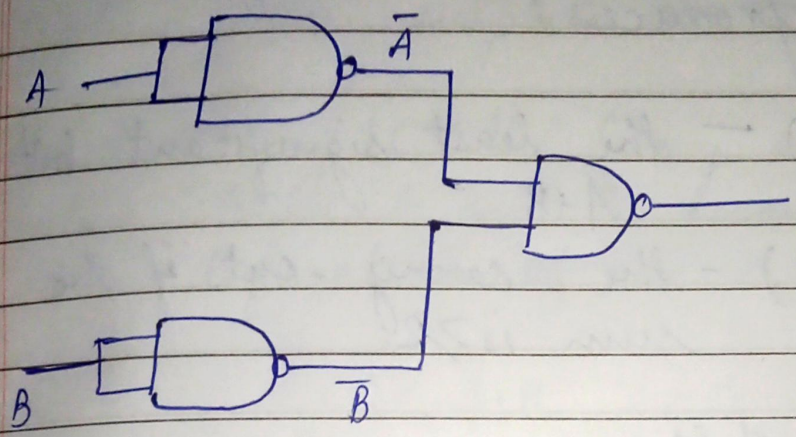
$$\overline{\bar{A} \cdot \bar{B}} = A + B$$

The Circuit Construction:

1. Stage 1 (Invert A): Connect A to both inputs of a NAND gate. Output $\bar{A}$.
2. Stage 2 (Invert B): Connect B to both inputs of a NAND gate. Output is $\bar{B}$.
3. Stage 3 (Combine): Connect the outputs from Stage 1 ($\bar{A}$) and Stage 2 ($\bar{B}$) into the inputs of a third NAND gate.

Final output: The result is A OR B.

Date ___/___/_____



$\overline{\bar{A} \cdot \bar{B}} = A + B$

Mission 2

A half adder takes two single bits, A and B, and produces:

- Sum (S) — the least significant bit of A+B
- Carry (C) - the carry-out if the sum is ≥ 2.

(a) Truth Table:-

Based on the rules of binary addition provided (0+0 = 0, 0+1 = 1, 1+0 = 1, 1+1 = 10 (that's two in binary)) here is truth table for input A and B.

A	B	Sum(S)	Carry(C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(b) Boolean Expression.

Date ___ / ___ / _____

To derive the expressions, we look at when the outputs are 1.

- For the Carry (C): The Carry output is 1 only when BOTH A and B are 1. Matches the defination of an AND gate:-

$$C = A \cdot B$$

- For the Sum (S): The Sum output is 1 only when the inputs are different (one is 0, the other is 1). Matches the defination of an XOR gate.

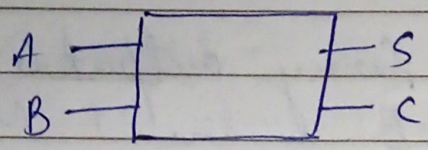
$$S = A \oplus B$$
$$= A\bar{B} + \bar{A}B$$

(c) The Circuit.
To build this circuit, you simply run your two input wires (A and B) into two parallel gates.

1. XOR Gate: Connect A and B to an XOR gate to produce the Sum.
2. AND Gate :- Connect A and B to an AND gate to produce the Carry.

Date ___ / ___ / _____

Block and Logic Diagram of half adder.

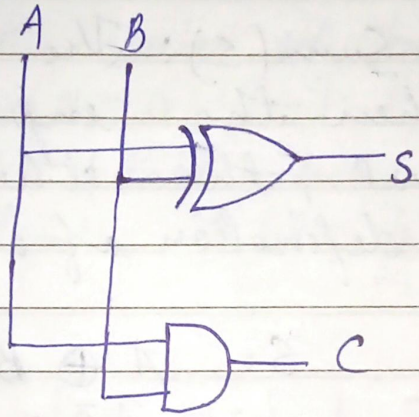


Block diagram

A, B - inputs

S - Sum

C - Carry



Logic Circuit Diagram.

Mission 3: Building a Smart Selector, you need to design a 4-to-1 Multiplexer (MUX) using three 2-to-1 MUX blocks, which is the component introduced in part 1.

A 4-to-1 MUX selects one of four data inputs (I_0, I_1, I_2, I_3) using a 2 bit control signal (S_1 and S_0). This is achieved by thinking hierarchically.

The Hierarchical Circuit Design

You will use three 2-to-1 MUXes, labeled MUX_A, MUX_B and MUX_C.

1. First Stage: Selecting Pairs (Using S_0).

The first control bit, S_0, is used to select between the two inputs within each pair.

- MUX_A (Input Pair 0 & 1):
- Inputs: I_0 and I_1

Date ___ / ___ / _____

- Select Line : S_0
- Output : Y_A (This output will be I_0 if $S_0 = 0$, or I_1 if $S_0 = 1$).

- MUX$_B$ (Input Pair 2 and 3):

Inputs : I_2 and I_3
: Select Line : S_0
.. Output : Y_B (This output will be I_2 if $S_0 = 0$ or I_3 if $S_0 = 1$)

2. Second Stage: Final Selection (Using S_1)

The second control bit, S_1, is used to select which of the first-stage outputs (Y_A or Y_B) will become the final output Y.

- MUX$_C$ (Final Selector):

Inputs: Y_A and Y_C.
Select Line : S_1

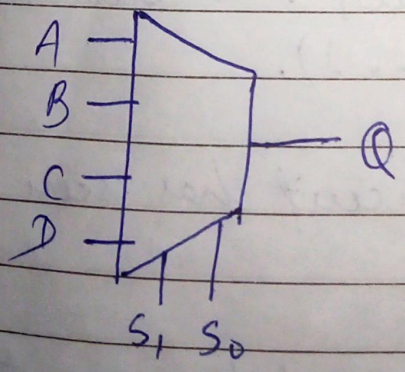
Date ___/___/_____

• Final Output: Y (This output will be Y_A if $S_1 = 0$, or Y_B if $S_1 = 1$).

Truth table of Control Logic.

S_1	S_0	MUX's selects	Final Output Y
0	0	Y_A (where $S_0 = 0$ choose I_0)	I_0
0	1	Y_A (where $S_0 = 1$ choose I_1)	I_1
1	0	Y_B (where $S_0 = 0$ choose I_2)	I_2
1	1	Y_B (where $S_0 = 0$ choose I_3)	I_3

MUZTIPLEXER 4×1



S_1	S_0	Q
0	0	A
0	1	B
1	0	C
1	1	D

Date ___/___/_____

Mission 4: Pattern Detector requires designing a sequential circuit that detects the pattern '111' in an input stream X and outputs P = 1. Sequential circuits, unlike combinational circuits, use memory elements (D Flip-Flops) to "remember" past inputs.

(a) Determining the Number of Flip-Flops

You need to "remember" enough previous states to distinguish between the pattern stages.

- State 0 (S0): Nothing detected yet, or the pattern was broken (e.g., ...0).
- State 1 (S1): The circuit has seen one '1' (...1).
- State 2 (S2): The circuit has seen two consecutive '1's (..11).
- State 3 (S3): The circuit has seen three consecutive '1s' (...111), triggering the

output P = 1.

You have four distinct states to track ($S0, S1, S2, S3$).

The number of flip flops (N) required to represent M states is given by

$$2^N \geq M.$$

$$2^N \geq 4$$

- $2^1 = 2$ (not enough)
- $2^2 = 4$ (sufficient)

Therefore, you need two D Flip-Flops (Q_1 and Q_0) to represent the four states (00, 01, 10, 11).

(b) The Complete Circuit Design :

The sequential circuit consists of two parts: the memory (Flip-Flops) and the combinational logic that generates the next state and the output.

Note: The pattern is overlaping - detecting a '111' means if the next input is '1', the circuit remains in the state of having seen two '1s' (S3 is the state after the pattern is found, and it must hold that state if X=1, but reset if X=0).

2. Boolean Expressions (Karnaugh Maps not shown, but result derived).

The expressions for the flip-flop inputs (D_1, D_0) and the output (P) are derived from the table above:

Output / Input	Expression
D_1	$D_1 = X \cdot Q_0 + X \cdot Q_1$
D_0	$D_0 = X \cdot \overline{Q_1} \cdot \overline{Q_0} + X \cdot Q_1 \cdot Q_0$
P	$P = X \cdot Q_1 \cdot Q_0$

The simplified expression for D_1 is ~~$D_1 = X$~~ $D_1 = X \cdot (Q_0 + Q_1)$. The simplified expression for D_0 is $D_0 = X \cdot (\overline{Q_1} \cdot \overline{Q_0} + Q_1 \cdot Q_0)$, which is $D_0 = X \cdot \overline{(Q_1 \oplus Q_0)}$ or $D_0 = X \cdot (Q_1 \oplus Q_0)$.

Date ___ / ___ / _____

3. Circuit Diagram.

The circuit uses the derived Boolean expressions implemented with combinational logic (AND, OR, NOT, XOR/XNOR) driving the inputs of the two D Flip-Flops.

- Memory: Two D Flip-Flops (FF_1 and FF_0).
  - The output of FF_1 is Q_1.
  - The output of FF_0 is Q_0.
  - Both are controlled by the same CLK signal.

- Combinational Logic:
  - D_1 Logic: Input X ANDed with (Q_1 OR Q_0).
  - D_0 Logic: Input X ANDed with the XNOR of Q_1 and Q_0.

- Output P Logic: Input X ANDed with Q_1 ANDed with Q_0. (P is only 1 when the current state is S3 ($Q_1 = 1$, $Q_0 = 1$) AND the next input X is 1).

Date ___ / ___ / _____

Mission 5 : The Counter Mystery, which asks you to design a 2-bit synchronous up-counter using two D Flip-Flops. The counter sequence is $00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00$ (and repeats).

(a) State Transition Table.

The table maps the current state (outputs Q_1, Q_0) to the next state (Q_{1next} Q_{0next}) after one clock pulse. Since we are using D Flip-Flops, the next state is equal to the inputs (i.e., $D_1 = Q_{1next}$ and $D_0 = Q_{0next}$).

Decimal Count	Current State ($Q_1 Q_0$)	Next State ($Q_{1next} Q_{2next}$)	Flip-Flop Inputs ($D_1 D_0$)
0	0 0	0 1	0 1
1	0 1	1 0	1 0
2	1 0	1 1	1 1
3	1 1	0 0	0 0

(b) Boolean Expressions for D_1 and D_0.

We derive the simplified Boolean expressions for the D inputs by looking at the D_1 and D_0 columns in the table above (where 1s are produced):

Expression for D_0:

D_0 is 1 when the current state is 00 ($\overline{Q_1}\,\overline{Q_0}$) OR 10 ($Q_1\,\overline{Q_0}$).

- $D_0 = (\overline{Q_1}\,\overline{Q_0}) + (Q_1\,\overline{Q_0})$
- Factor out $\overline{Q_0}$: $D_0 = \overline{Q_0} \cdot (\overline{Q_1} + Q_1)$
- Since $(\overline{Q_1} + Q_1)$ is always 1:

$$D_0 = \overline{Q_0}$$

(This means the least significant bit Q_0 must always toggle (NOT its current state) on every clock pulse.)

Date ___ / ___ / _____

Expression for D_1

D_1 is 1 when the current state 01 ($\bar{Q}_1, Q_0$) or 10 ($Q_1, \bar{Q}_0$).

- $D_1 = (\bar{Q}_1, Q_0) + (Q_1, \bar{Q}_0)$
- This expression exactly matches the defination of an XOR Gate.

$$D_1 = Q_1 \oplus Q_0$$

(This means the most significant bit Q_1 only changes its state when the least significant bit Q_0 is 1, causing a "carry" to the next bit.)

(c) The Complete Circuit.

The circuit uses the derived Boolean expressions to connect the outputs (Q_1, Q_0) back to the D inputs (D_1, D_0) via logic gates.

- D_0 connection: The output Q_0 is connected

through a NOT Gate (Inverter) to input D_0 of Flip-Flop 0.

- D_1 Connection: The outputs Q_1 and Q_0 are connected to an XOR Gate, and the output of the XOR Gate is connected to the input D_1 of Flip-Flop 1.

- Both Flip-Flops are connected to the single Clock (CLK) signal.