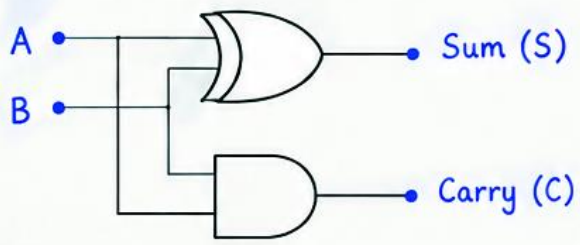
COMBINATIONAL LOGIC CIRCUITS

ADDER – HALF ADDER AND FULL ADDER

1. Half Adder :

A Half Adder is a combinational circuit that adds two single-bit binary numbers and produces a Sum and Carry output.

Logic Circuit :



Truth Table :

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

Boolean Expressions :

$$\text{Sum } (S) = A \oplus B$$
$$= A'B + AB'$$

$$\text{Carry } (C) = A \cdot B$$

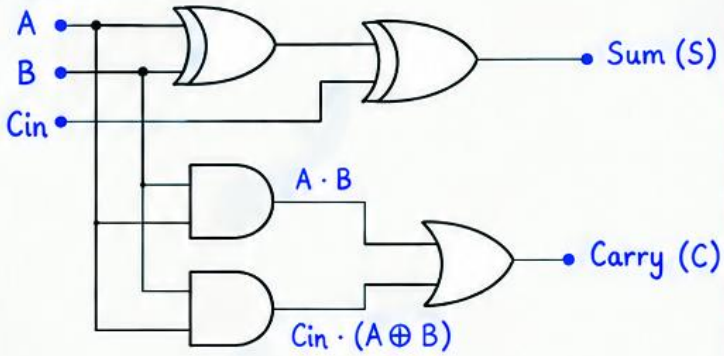
Note :

- The Sum output is obtained using XOR gate.
- The Carry output is obtained using AND gate.

2. Full Adder :

A Full Adder is a combinational circuit that adds three single-bit binary numbers and produces a Sum and Carry output.

Logic Circuit :



Truth Table :

A	B	Cin	Sum (S)	Carry (C)
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

Boolean Expressions :

$$\text{Sum } (S) = A \oplus B \oplus Cin$$

$$\text{Carry } (C) = A \cdot B + Cin \cdot (A \oplus B)$$

Note :

- The Sum output is obtained using two XOR gates.
- The Carry output is obtained using two AND gates and one OR gate.

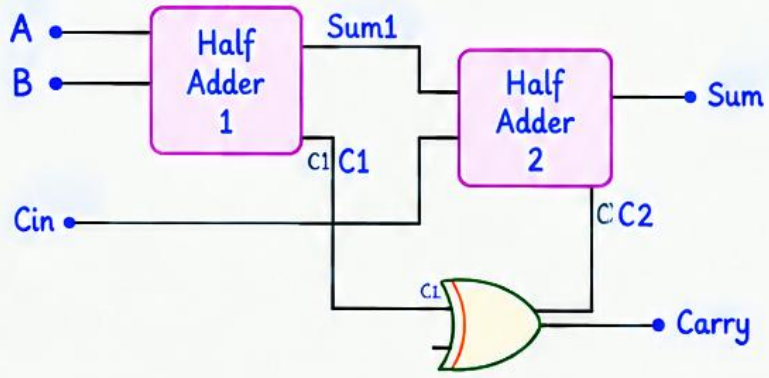
COMBINATIONAL LOGIC CIRCUITS

REALIZATION OF FULL ADDER USING 2 HALF ADDER

1. Concept :

A full adder can be implemented using two half adders and one OR gate.
The two half adders are used to add the three input bits (A, B and Cin) in two stages.
The carry outputs of both half adders are combined using an OR gate to get the final carry.

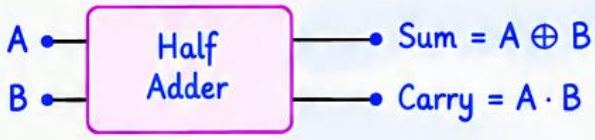
Logic Circuit :



Truth Table :

A	B	Cin	Sum	Carry
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

Half Adder (for reference) :



Output Expressions :

$$Sum = A \oplus B \oplus Cin$$

$$Carry = (A \cdot B) + (Cin \cdot (A \oplus B))$$

Step by Step Working :

(1) First Half Adder :
- A and B are added in the first half adder.
- Outputs: $Sum1 = A \oplus B$, $C1 = A \cdot B$

(2) Second Half Adder :
- Sum1 and Cin are added in the second half adder.
- Outputs: $Sum = Sum1 \oplus Cin = A \oplus B \oplus Cin$

$$C2 = Sum1 \cdot Cin = (A \oplus B) \cdot Cin$$

(3) Final Carry :
- The carry outputs C1 and C2 are combined using an OR gate.
- $Carry = C1 + C2 = (A \cdot B) + (Cin \cdot (A \oplus B))$

(4) Result :
- Using two half adders and one OR gate, we get a full adder.
- Sum and Carry are generated correctly for all input combinations.

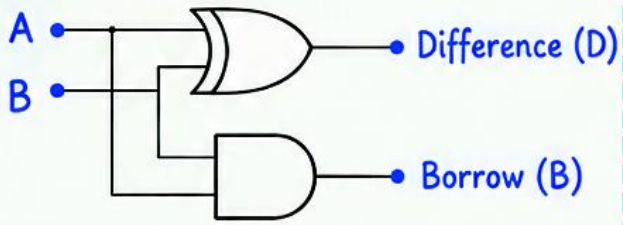
COMBINATIONAL LOGIC CIRCUITS

SUBTRACTOR – HALF SUBTRACTOR AND FULL SUBTRACTOR

1. Half Subtractor :

A Half Subtractor is a combinational circuit that subtracts two single-bit binary numbers and produces a Difference and a Borrow output.

Logic Circuit :



Truth Table :

A	B	D (A⊕B)	B ($\overline{A}$·B)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Boolean Expressions :

$$D = A \oplus B$$

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

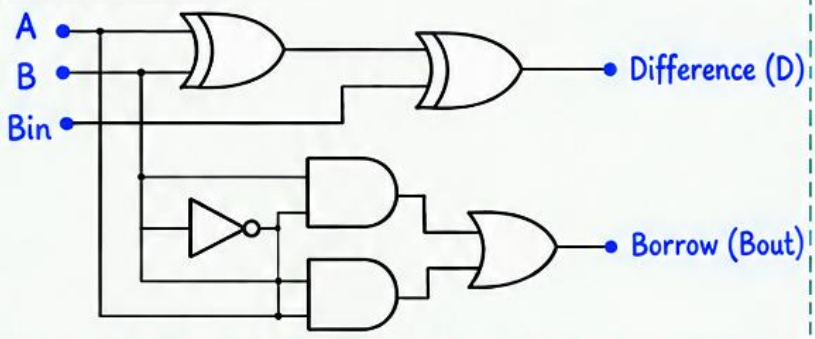
Note :

- Difference (D) is obtained using XOR gate.
- Borrow (B) is obtained using AND gate with $\overline{A}$ and B.

2. Full Subtractor :

A Full Subtractor is a combinational circuit that subtracts two single-bit binary numbers and takes a borrow input. It produces a Difference and a Borrow output.

Logic Circuit :



Truth Table :

A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

Boolean Expressions :

$$D = A \oplus B \oplus Bin$$

$$Bout = \overline{A} \cdot B + \overline{A} \cdot Bin + B \cdot Bin$$

Note :

- Difference (D) is obtained using 2 XOR gates.
- Borrow (Bout) is obtained using 3 AND gates, 1 OR gate and one NOT gate.

Key Points :

- Half Subtractor : 2 inputs (A, B)
  → Outputs : D, B
- Full Subtractor : 3 inputs (A, B, Bin)
  → Outputs : D, Bout

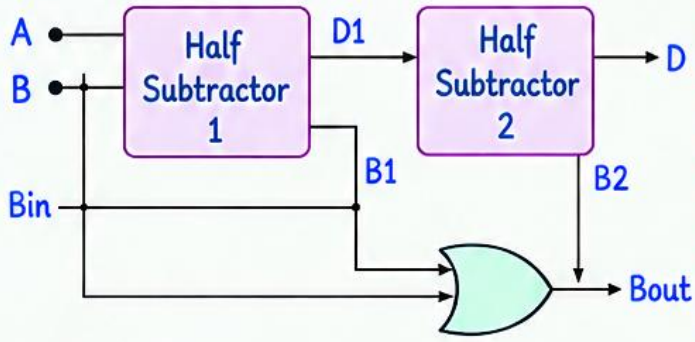
COMBINATIONAL LOGIC CIRCUITS

A FULL SUBTRACTOR USING 2 HALF SUBTRACTOR

1. Concept :

A full subtractor can be implemented using two half subtractors and one OR gate. The two half subtractors are cascaded and their borrow outputs are combined using an OR gate to get the final borrow output.

2. Logic Diagram :



3. Truth Table :

A	B	Bin	D (Difference)	Bout (Borrow)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

4. Boolean Expressions :

Difference (D) :

$$D = A \oplus B \oplus Bin$$

Borrow (Bout) :

$$Bout = \overline{A} \cdot B + \overline{A} \cdot Bin + B \cdot Bin$$

$$= (\overline{A} \cdot B) + (\overline{A} \cdot Bin) + (B \cdot Bin)$$

5. Step by Step Working :

① First Half Subtractor (A, B) :

- Difference : $D1 = A \oplus B$
- Borrow : $B1 = \overline{A} \cdot B$

② Second Half Subtractor (D1, Bin) :

- Difference : $D = D1 \oplus Bin = A \oplus B \oplus Bin$
- Borrow : $B2 = \overline{D1} \cdot Bin + D1 \cdot \overline{Bin}$

③ Final Borrow (OR gate) :

$Bout = B1 + B2 = (\overline{A} \cdot B) + (\overline{A} \cdot Bin) + (B \cdot Bin)$

6. Final Output :

Difference (D) = $A \oplus B \oplus Bin$

Borrow (Bout) = $(\overline{A} \cdot B) + (\overline{A} \cdot Bin) + (B \cdot Bin)$

7. Key Points :

- Uses 2 Half Subtractors + 1 OR gate.
- First HS gives D1 and B1.
- Second HS gives final D and B2.
- OR gate combines B1 and B2 to get Bout.

MULTIPLEXER (MUX) – TYPES, WORKING AND USES

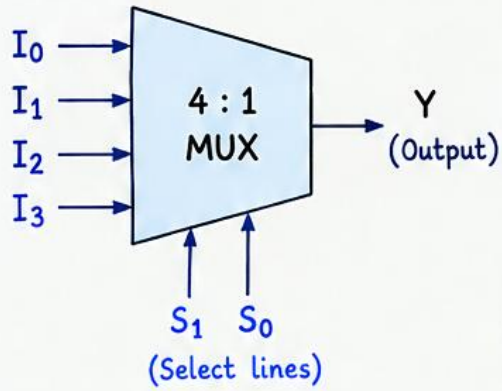
1. Definition :

A Multiplexer (MUX) is a combinational logic circuit that selects one of the many input signals and forwards it to a single output line, based on the select lines.
It is also called a data selector.

Key Points :

- It has 2^n input lines and 1 output line.
- It requires n select lines.
- Only one input is selected at a time.
- The output is the same as the selected input.

Example :

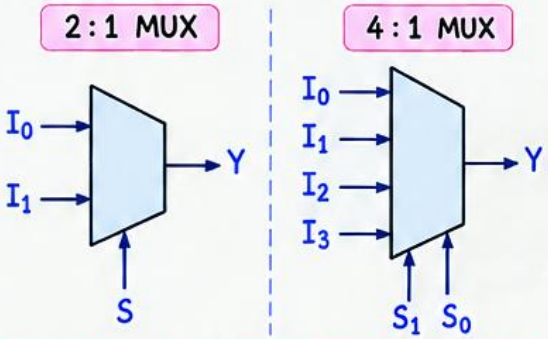


2. Types of Multiplexer :

Multiplexers are available in different sizes based on the number of input lines. Common types are:

- 2 : 1 Multiplexer (1 select line, 2 inputs)
- 4 : 1 Multiplexer (2 select lines, 4 inputs)
- 8 : 1 Multiplexer (3 select lines, 8 inputs)
- 16 : 1 Multiplexer (4 select lines, 16 inputs)

Types (with Diagram) :



3. Working :

1. The select lines (S) decide which input line will be connected to the output.
2. When the select lines have a particular value, the corresponding input is passed to the output.
3. All other inputs are ignored.

Truth Table (4 : 1 MUX) :

S_1	S_0	Y	Selected Input
0	0	I_0	I_0
0	1	I_1	I_1
1	0	I_2	I_2
1	1	I_3	I_3

4. Boolean Expression :

For 4 : 1 MUX, the output can be written as :

$$Y = I_0 \cdot \overline{S}_1 \cdot \overline{S}_0 + I_1 \cdot \overline{S}_1 \cdot S_0 + I_2 \cdot S_1 \cdot \overline{S}_0 + I_3 \cdot S_1 \cdot S_0$$

where, S_1 and S_0 are the select lines.

5. Uses :

- Data selection in digital systems.
- Memory address decoding.
- Implementing functions in computers and microprocessor systems.
- Used in ALU, registers, and buses.

2:1 MUX – Operation, Truth Table and Circuit diagram

Combinational Logic Circuits

★ Introduction

A Multiplexer (MUX) is a combinational circuit that selects one of the many input lines and forwards it to a single output line based on the select input.

A 2:1 MUX has two data inputs (I_0, I_1), one select input (S) and one output (Y).

★ Truth Table

Select (S)	I_0	I_1	Output (Y)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

When $S = 0$, $Y = I_0$
When $S = 1$, $Y = I_1$

★ Opexation

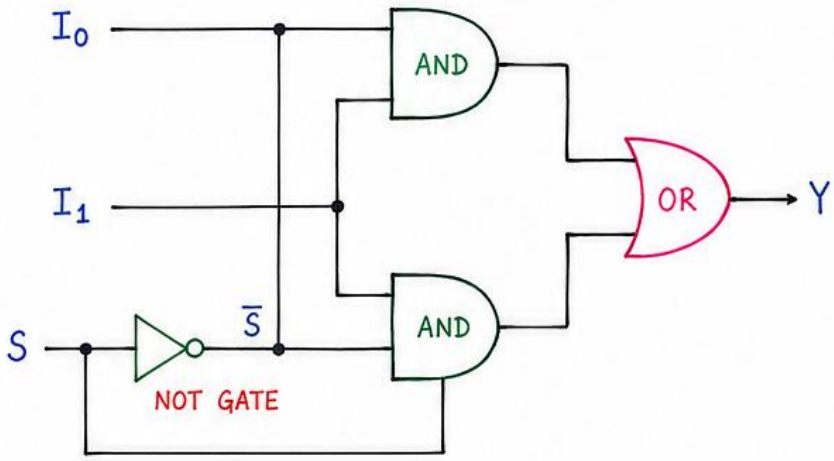
- If Select input (S) = 0, then I_0 is selected and appears at output Y.
- If Select input (S) = 1, then I_1 is selected and appears at output Y.

Boolean Expression:

$$Y = \overline{S}\, I_0 + S I_1$$

Where,
- $\overline{S}$ = Complement of S
- I_0 = Input 0
- I_1 = Input 1
- Y = Output

★ Circuit Diagram



★ Key Points

- A 2:1 MUX selects one of the two inputs and forwards it to the output.
- It is a basic building block for larger multiplexers (4:1, 8:1 etc.).
- It is widely used in data routing, data selection and digital system design.

4:1 MUX – Operation, Truth Table and Circuit diagram

Combinational Logic Circuits

★ Introduction

A Multiplexer (MUX) is a combinational circuit that selects one of the many input lines and forwards it to a single output line based on the select inputs.

A 4:1 MUX has four data inputs (I_0, I_1, I_2, I_3), two select inputs (S_1, S_0) and one output (Y).

★ Truth Table

Select Inputs		Data Inputs				Output (Y)
S_1	S_0	I_0	I_1	I_2	I_3	
0	0	I_0	I_1	I_2	I_3	I_0
0	1	I_0	I_1	I_2	I_3	I_1
1	0	I_0	I_1	I_2	I_3	I_2
1	1	I_0	I_1	I_2	I_3	I_3

When $S_1\ S_0 = 00 \rightarrow Y = I_0$
When $S_1\ S_0 = 01 \rightarrow Y = I_1$
When $S_1\ S_0 = 10 \rightarrow Y = I_2$
When $S_1\ S_0 = 11 \rightarrow Y = I_3$

★ Operation

- If $S_1\ S_0 = 00$, then I_0 is selected and appears at output Y.
- If $S_1\ S_0 = 01$, then I_1 is selected and appears at output Y.
- If $S_1\ S_0 = 10$, then I_2 is selected and appears at output Y.
- If $S_1\ S_0 = 11$, then I_2 is selected and appears at output Y.

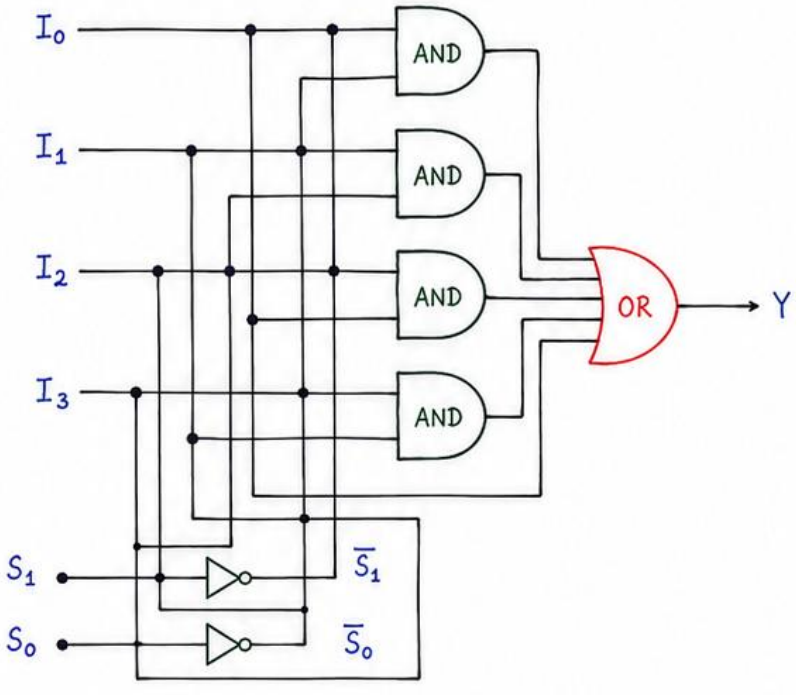
Boolean Expression:

$$Y = \overline{S}_1\,\overline{S}_0\,I_0 + \overline{S}_1\,S_0\,I_1 + S_1\,\overline{S}_0\,I_2 + S_1\,S_0\,I_3$$

Where,

- S_1, S_0 = Select inputs
- I_0, I_1, I_2, I_3 = Data inputs
- Y = Output

★ Circuit Diagram



★ Key Points

- A 4:1 MUX selects one of the four inputs (I_0, I_1, I_2, I_3) and forwards it to output.
- It uses two select lines (S_1, S_0) to determine which input is selected.
- It is a basic building block for larger multiplexers (8:1, 16:1 etc.).
- It is widely used in data routing, data selection and digital system design.

8:1 MUX – Operation, Truth Table and Circuit diagram

Combinational Logic Circuits

★ Introduction

A Multiplexer (MUX) is a combinational circuit that selects one of the many input lines and forwards it to a single output line based on the select inputs.

An 8:1 MUX has eight data inputs ($I_0 - I_7$), three select inputs (S_2, S_1, S_0) and one output (Y).

★ Truth Table

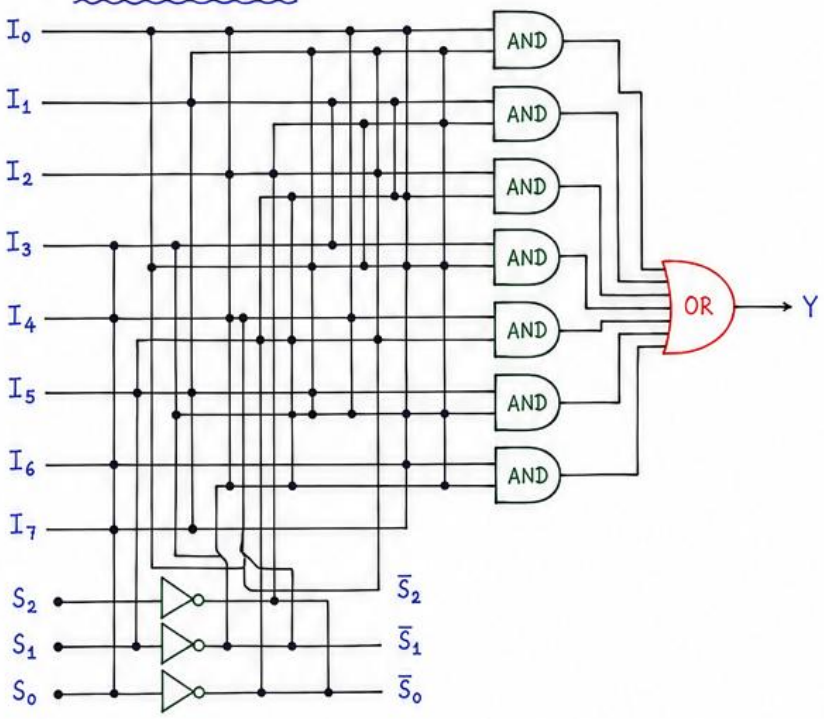
Select Inputs			Data Inputs								Output (Y)
S_2	S_1	S_0	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	
0	0	0	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_0
0	0	1	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_1
0	1	0	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_2
0	1	1	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_3
1	0	0	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_4
1	0	1	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_5
1	1	0	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_6
1	1	1	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_7

When $S_2\,S_1\,S_0 = 000 \rightarrow Y = I_0$
When $S_2\,S_1\,S_0 = 001 \rightarrow Y = I_1$
When $S_2\,S_1\,S_0 = 010 \rightarrow Y = I_2$
When $S_2\,S_1\,S_0 = 011 \rightarrow Y = I_3$
When $S_2\,S_1\,S_0 = 100 \rightarrow Y = I_4$
When $S_2\,S_1\,S_0 = 101 \rightarrow Y = I_5$
When $S_2\,S_1\,S_0 = 110 \rightarrow Y = I_6$
When $S_2\,S_1\,S_0 = 111 \rightarrow Y = I_7$

★ Operation

- If $S_2\,S_1\,S_0 = 000$, then I_0 is selected and appears at output Y.
- If $S_2\,S_1\,S_0 = 001$, then I_1 is selected and appears at output Y.
- If $S_2\,S_1\,S_0 = 010$, then I_2 is selected and appears at output Y.
- If $S_2\,S_1\,S_0 = 100$, then I_4 is selected and appears at output Y.
- If $S_2\,S_1\,S_0 = 101$, then I_5 is selected and appears at output Y.
- If $S_2\,S_1\,S_0 = 110$, then I_6 is selected and appears at output Y.
- If $S_2\,S_1\,S_0 = 111$, then I_7 is selected and appears at output Y.

★ Circuit Diagram



Boolean Expression:

$$Y = \overline{S}_2\,\overline{S}_1\,\overline{S}_0\,I_0 + \overline{S}_2\,\overline{S}_1\,S_0\,I_1 + \overline{S}_2\,S_1\,\overline{S}_0\,I_2 + \overline{S}_2\,S_1\,S_0\,I_3 + S_2\,\overline{S}_1\,\overline{S}_0\,I_4 + S_2\,\overline{S}_1\,S_0\,I_5 + S_2\,S_1\,\overline{S}_0\,I_6 + S_2\,S_1\,S_0\,I_7$$

Where,

S_2, S_1, S_0	=	Select inputs
$I_0 - I_7$	=	Data inputs
Y	=	Output
$\overline{S}$	=	Complement of S

★ Key Points

- An 8:1 MUX selects one of the eight inputs ($I_0 - I_7$) and forwards it to the output (Y).
- It uses three select lines (S_2, S_1, S_0) to determine which input is selected.
- It is a basic building block for larger multiplexers (16:1, 32:1, ... etc.).
- It is widely used in data routing, data selection and digital system design.

16:1 MUX – Just Truth Table and Circuit diagram

Combinational Logic Circuits

★ Truth Table

Select Inputs				Data Inputs	Output
S_3	S_2	S_1	S_0	$I_0 - I_{15}$	Y
0	0	0	0	I_0	I_0
0	0	0	1	I_1	I_1
0	0	1	0	I_2	I_2
0	0	1	1	I_3	I_3
0	1	0	0	I_4	I_4
0	1	0	1	I_5	I_5
0	1	1	0	I_6	I_6
0	1	1	1	I_7	I_7
1	0	0	0	I_8	I_8
1	0	0	1	I_9	I_9
1	0	1	0	I_{10}	I_{10}
1	0	1	1	I_{11}	I_{11}
1	1	0	0	I_{12}	I_{12}
1	1	0	1	I_{13}	I_{13}
1	1	1	0	I_{14}	I_{14}
1	1	1	1	I_{15}	I_{15}

★ When $S_3\ S_2\ S_1\ S_0 = n$ $(0 \le n \le 15)$, output $Y = I_n$

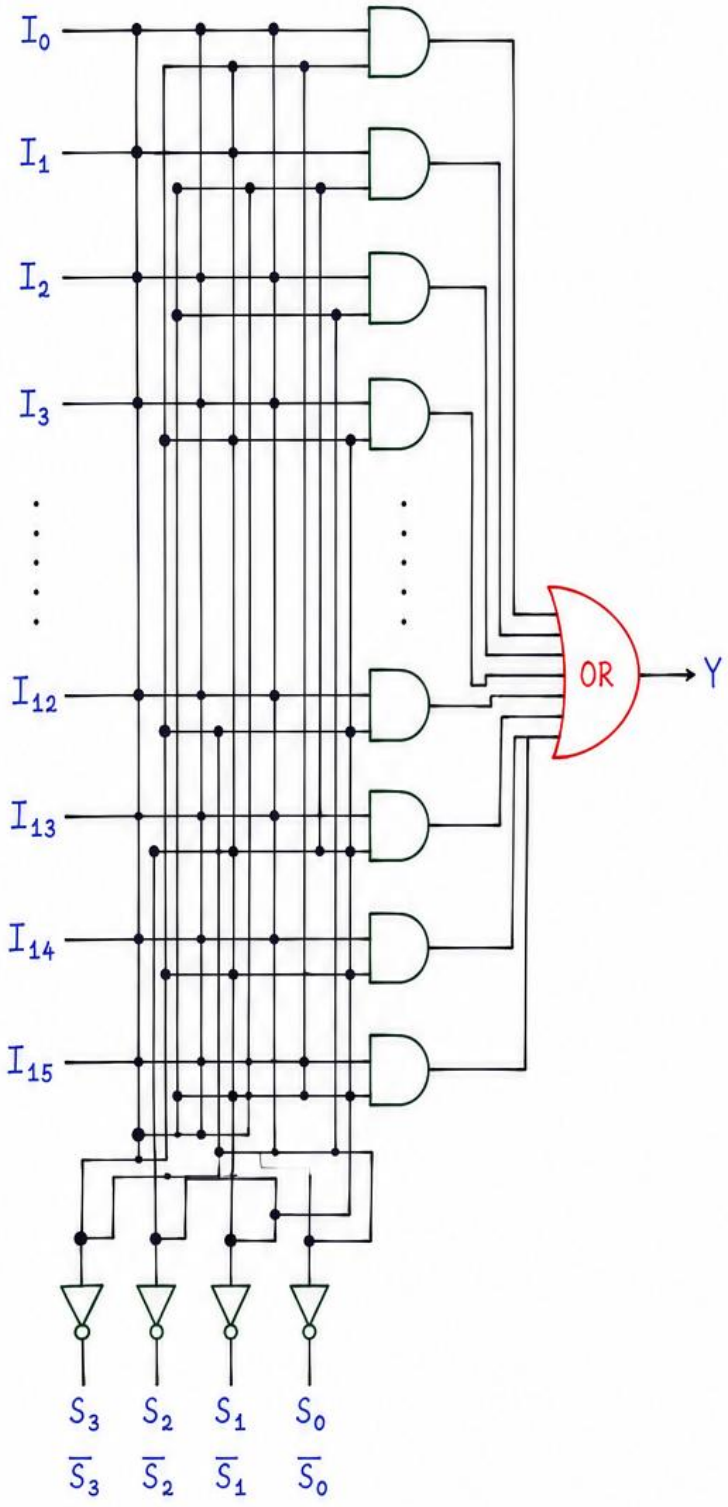
Where,

S_3, S_2, S_1, S_0 = Select inputs

$I_0 - I_{15}$ = Data inputs

Y = Output

★ Circuit Diagram



Design a 4:1 MUX using 2:1 MUX

Combinational Logic Circuits

★ Objective

To design a 4:1 Multiplexer using 2:1 Multiplexers.

★ 4:1 MUX

A 4:1 MUX has four data inputs (I_0, I_1, I_2, I_3), two select lines (S_1, S_0) and one output (Y). Based on S_1 and S_0, one of the four inputs is selected and connected to output Y.

★ Design Procedure

1. Use three 2:1 MUXs.
2. Use S_0 to select between each pair of inputs.
3. Use S_1 to select between the two intermediate outputs of step 2.
4. The final output is Y.

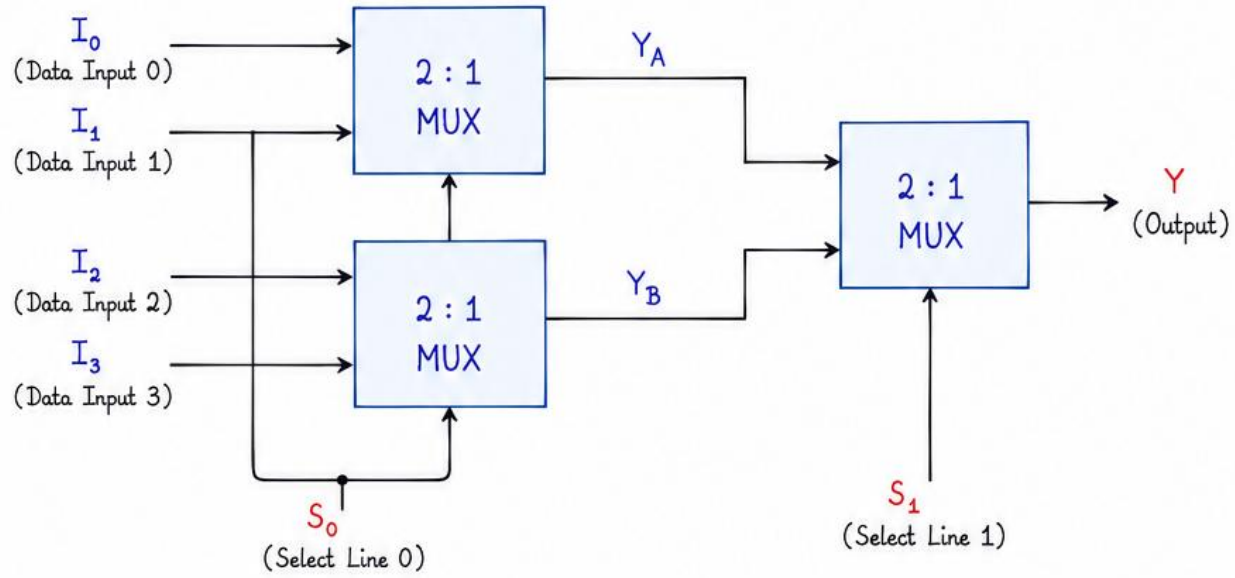
★ 4:1 MUX Truth Table

S_1	S_0	Selected Input	Output (Y)
0	0	I_0	$Y = I_0$
0	1	I_1	$Y = I_1$
1	0	I_2	$Y = I_2$
1	1	I_3	$Y = I_3$

Working

- When $S_1 = 0$, the upper 2:1 MUX selects between I_0 and I_1 using S_0.
- When $S_1 = 1$, the lower 2:1 MUX selects between I_2 and I_3 using S_0.
- The final 2:1 MUX selects between these two results using S_1.

★ Circuit Diagram



How it Works

1. S_0 selects the required input from each pair (I_0/I_1 and I_2/I_3).
2. S_1 selects the required output from the two intermediate outputs (Y_A and Y_B).
3. Thus, the output Y will be I_0, I_1, I_2 or I_3 depending on (S_1 S_0).

Implementation Summary

- 3 × 2:1 MUX are required.
- Select line S_0 is used by the first two MUXs.
- Select line S_1 is used by the last MUX.

Design an 8:1 MUX using 4:1 MUX

Combinational Logic Circuits

★ Objective

To design an 8:1 Multiplexer using four 4:1 Multiplexers.

★ 8:1 MUX

An 8:1 MUX has eight data inputs $I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7$, three select lines (S_2, S_1, S_0) and one output (Y). Based on (S_2, S_1, S_0), one of the eight inputs is selected and connected to output Y.

★ Design Procedure

1. Use four 4:1 MUXs.
2. Use S_1 and S_0 as the select lines for each 4:1 MUX.
3. The first two 4:1 MUXs select between I_0-I_3 and I_4-I_7 respectively.
4. Use the outputs of these MUXs as inputs to the third 4:1 MUX.
5. Use S_2 as the select line of the final 4:1 MUX to choose between the two groups.
6. The output of the final MUX is Y.

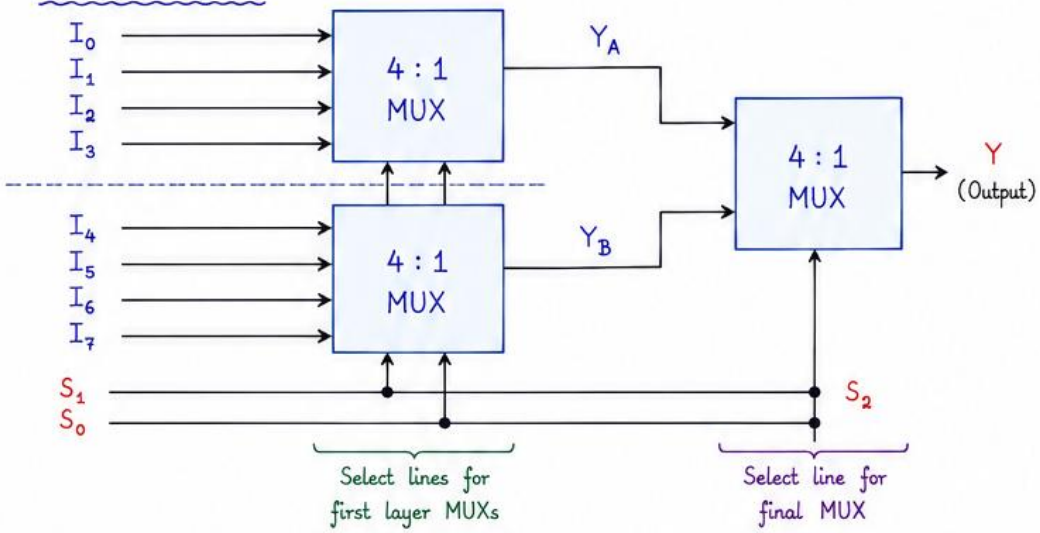
★ 8:1 MUX Truth Table

S_2	S_1	S_0	Selected Input	Output (Y)
0	0	0	I_0	$Y = I_0$
0	0	1	I_1	$Y = I_1$
0	1	0	I_2	$Y = I_2$
0	1	1	I_3	$Y = I_3$
1	0	0	I_4	$Y = I_4$
1	0	1	I_5	$Y = I_5$
1	1	0	I_6	$Y = I_6$
1	1	1	I_7	$Y = I_7$

How it Works

- S_1 S_0 select one input from each group of four using the first two layers of 4:1 MUXs.
- S_2 selects between the two groups using the final 4:1 MUX.

★ Circuit Diagram



Summary

- 4 (first layer) + 1 (second layer) = 5 numbers of 4:1 MUXs are used.
- S_1, S_0 go to the first two MUXs.
- S_2 goes to the final MUX.

Implementation Summary

- First layer: Two 4:1 MUXs select between I_0-I_3 and I_4-I_7 using S_1 and S_0.
- Second layer: One 4:1 MUX selects between Y_A and Y_B using S_2.
- Output Y will be I_0 to I_7 depending on S_2 S_1 S_0.

Note

This design can be extended to implement higher order MUXs (16:1, 32:1, ...) using the same hierarchical approach.

Design a NAND Gate using MUX

Combinational Logic Circuits

★ Aim :

To implement a 2-input NAND gate using a 4:1 Multiplexer.

★ NAND Gate

For inputs A and B, the output Y of a NAND gate is

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

★ Design Procedure

1. Choose a 4:1 MUX.
2. Use the inputs A and B as the select lines.
3. Connect the data inputs (I_0, I_1, I_2, I_3) of the MUX according to the NAND truth table.
4. The output of the MUX will be the NAND output Y.

★ MUX Connections

Let $S_1 = A$ and $\overline{S}_0 = \overline{B}$ (Select lines).

For each combination of A and B, connect the corresponding data input as per NAND output:

Select Lines		MUX Data Input	NAND Output Y
A (S_1)	B (S_0)		
0	0	I_0	1
0	1	I_1	1
1	0	I_2	1
1	1	I_3	0

Hence, connect: $I_0 = 1$, $I_1 = 1$, $I_2 = 1$, $I_3 = 0$

★ Verification (Operation)

The operation of the circuit is same as NAND truth table.

A (S_1)	B (S_0)	Selected Data Input	MUX Output Y
0	0	$I_0 = 1$	Y = 1
0	1	$I_1 = 1$	Y = 1
1	0	$I_2 = 1$	Y = 1
1	1	$I_3 = 0$	Y = 0

This matches exactly with the NAND gate truth table.

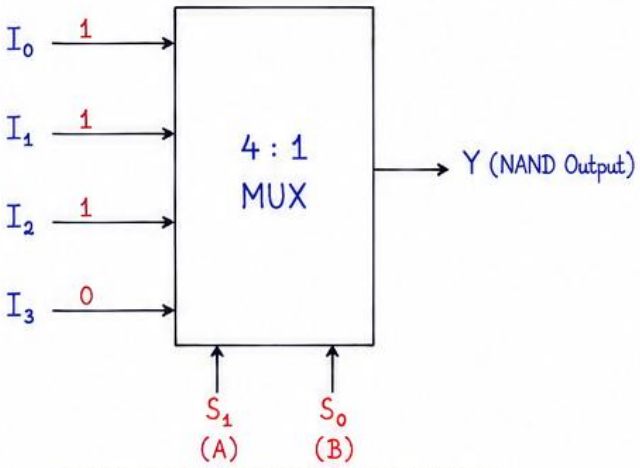
★ Truth Table of NAND Gate

A	B	A · B	$Y = \overline{A \cdot B}$ (NAND)
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Why 4:1 MUX ?

A 4:1 MUX has 2 select lines and can realize any logic function of two variables by properly connecting its data inputs.

★ Circuit Diagram



Select Lines: $S_1 = A$, $S_0 = B$

★ Result

A NAND gate is successfully designed using a 4:1 Multiplexer with the connections:

$I_0 = 1$ $I_1 = 1$ $I_2 = 1$ $I_3 = 0$

$S_1 = A$, $S_0 = B$

Y (MUX output) = A NAND B

★ Note

By fixing the data inputs of a MUX and using proper select lines, many logic gates can be implemented.

NOR Gate Designing using MUX (Circuit Diagram)

Combinational Logic Circuits

★ Objective

To implement a 2-input NOR gate using a 4:1 MUX.

★ NOR Gate

For inputs A and B, the output Y of a NOR gate is

$$Y = \overline{A + B}$$

★ NOR Truth Table

A	B	$Y = \overline{A + B}$ (NOR)
0	0	1
0	1	0
1	0	0
1	1	0

★ Implementation using 4:1 MUX

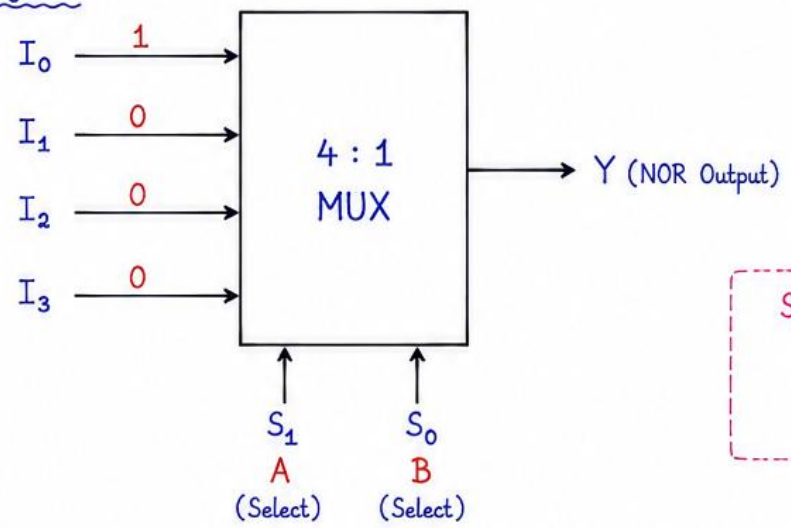
1. Use A and B as select lines.
2. Connect the data inputs (I_0 – I_3) according to the NOR truth table.
3. The output of the MUX will be the NOR output Y.

MUX Data Inputs

$I_0 = 1$ (when A = 0, B = 0)
$I_1 = 0$ (when A = 0, B = 1)
$I_2 = 0$ (when A = 1, B = 0)
$I_3 = 0$ (when A = 1, B = 1)

So, connect: $I_0 = 1$, $I_1 = 0$, $I_2 = 0$, $I_3 = 0$

★ Circuit Diagram



Select Lines

$S_1 = A$
$S_0 = B$

Explanation

- The select lines A and B determine which data input (I_0 to I_3) is selected by the MUX.
- The MUX output Y becomes $\overline{A + B}$, which is the output of a NOR gate.

Verification (Operation)

When A = 0, B = 0 → $S_1S_0 = 00$ → $Y = I_0 = 1$
When A = 0, B = 1 → $S_1S_0 = 01$ → $Y = I_1 = 0$
When A = 1, B = 0 → $S_1S_0 = 10$ → $Y = I_2 = 0$
When A = 1, B = 1 → $S_1S_0 = 11$ → $Y = I_3 = 0$

★ Note

By properly connecting the data inputs of a 4:1 MUX and using A and B as select lines, a NOR gate can be easily implemented.

Demultiplexer – Types, Working, and Uses

Combinational Logic Circuits

★ Introduction

A Demultiplexer (DEMUX) is a combinational circuit that takes one data input and forwards it to one of the many output lines based on the select inputs. In other words, it performs the reverse operation of a multiplexer.

A DEMUX has one data input (D), n select inputs ($S_{n-1} \dots S_0$) and 2^n output lines ($Y_0 \dots Y_{2^n-1}$). Only one output line is active at a time.

★ Working (General Concept)

- The data input D is applied to the demultiplexer.
- The select inputs ($S_{n-1} \dots S_0$) select one output line.
- The selected output line becomes equal to D.
- All other output lines remain 0.

General Expression:

$Y_m = D$, when $(S_{n-1} \dots S_0) = m$

$(0 \le m \le 2^n - 1)$

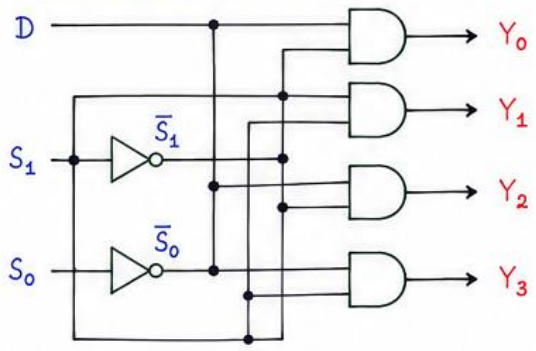
All other outputs = 0

Example (1:4 DEMUX)

If $D = 1$ and $S_1 S_0 = 10$,

then $Y_2 = 1$ and $Y_0 = Y_1 = Y_3 = 0$

★ Logic Circuit (1:4 DEMUX)

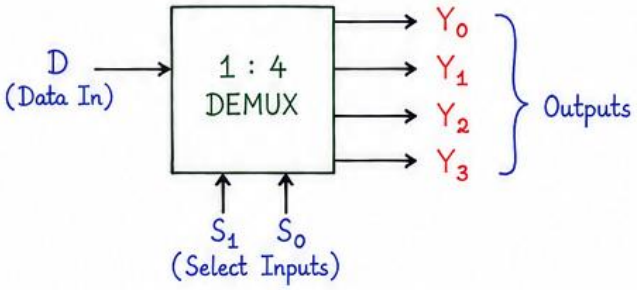


★ Types of Demultiplexer

Demultiplexers are classified by the number of output lines.

Type	Select Inputs (n)	Output Lines (2^n)	Notation
1 to 2 DEMUX	1	2	1:2 DEMUX
1 to 4 DEMUX	2	4	1:4 DEMUX
1 to 8 DEMUX	3	8	1:8 DEMUX
1 to 16 DEMUX	4	16	1:16 DEMUX
1 to 32 DEMUX	5	32	1:32 DEMUX
...	...	...	...

★ Block Diagram (1:4 DEMUX)



★ Truth Table (1:4 DEMUX)

Select Inputs		Data Input	Outputs			
S_1	S_0	D	Y_0	Y_1	Y_2	Y_3
0	0	D	D	0	0	0
0	1	D	0	D	0	0
1	0	D	0	0	D	0
1	1	D	0	0	0	D

Assume D = 1, then only one output = 1 based on select inputs.

★ Uses / Applications

- Data distribution: Routing one data input to multiple destinations.
- Serial to parallel conversion.
- Memory chip selection.
- Communication systems.
- Used in microprocessor and peripheral interfacing.

In simple words, DEMUX takes one input and sends it to the selected output line.

1:2 De-MUX – Operation, Truth Table and Circuit diagram

Combinational Logic Circuits

★ Introduction

A De-Multiplexer (De-MUX) is a combinational circuit that takes one data input and forwards it to one of the two output lines based on the select input. It performs the reverse operation of a 2:1 MUX.

A 1:2 De-MUX has one data input (D), one select input (S) and two output lines (Y_0, Y_1). Only one output is active at a time.

★ Operation

- If S = 0, the data input D is routed to Y_0 and Y_1 = 0.
- If S = 1, the data input D is routed to Y_1 and Y_0 = 0.
- Thus, only the selected output line gets the data input, while the other output remains 0.

General Expressions :

$Y_0 = \overline{S} \cdot D$ (When S = 0)

$Y_1 = S \cdot D$ (When S = 1)

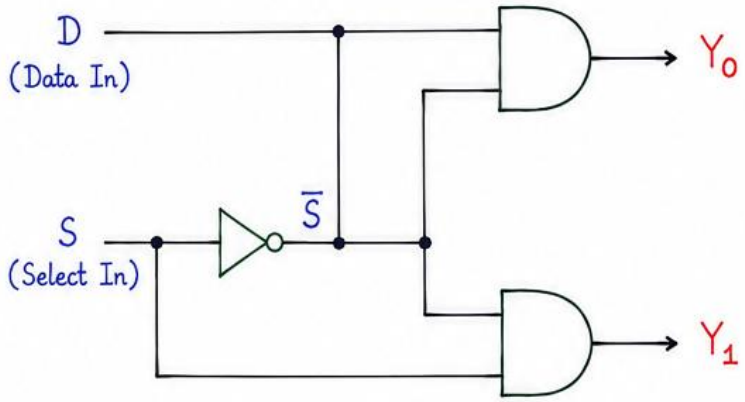
★ Truth Table (1:2 De-MUX)

Select Input S	Data Input D	Outputs	
		Y_0	Y_1
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

From the table :

- When S = 0 ⟶ $Y_0 = D$, $Y_1 = 0$
- When S = 1 ⟶ $Y_1 = D$, $Y_0 = 0$

★ Circuit Diagram (1:2 De-MUX)



Top AND gate gives $Y_0 = \overline{S} \cdot D$

Bottom AND gate gives $Y_1 = S \cdot D$

★ Key Points

- 1:2 De-MUX has 1 data input (D), 1 select input (S) and 2 outputs (Y_0, Y_1).
- It routes the data input to one of the two outputs based on the select input.
- It is the reverse function of a 2:1 MUX.
- Used in data distribution, communication and digital system design.

NOTES :

Only one output line is active (high) at a time. The other output remains low (0).

1:4 De-MUX – Operation, Truth Table and Circuit diagram

Combinational Logic Circuits

★ Introduction

A De-Multiplexer (De-MUX) is a combinational circuit that takes one data input and forwards it to one of the many output lines based on the select inputs. It performs the reverse operation of a Multiplexer.

A 1:4 De-MUX has one data input (D), two select inputs (S_1, S_0) and four output lines (Y_0, Y_1, Y_2, Y_3). Only one output is active (HIGH) at a time.

★ Operation

- The data input D is applied to the de-multiplexer.
- The select inputs (S_1, S_0) select one of the four outputs.
- The selected output gets the data input D.
- All other outputs remain 0.

General Expressions :

$Y_0 = \overline{S}_1\ \overline{S}_0\ D$ (When $S_1 = 0, S_0 = 0$)

$Y_1 = S_1\ \overline{S}_0\ D$ (When $S_1 = 0, S_0 = 1$)

$Y_2 = S_1\ \overline{S}_0\ D$ (When $S_1 = 1, S_0 = 0$)

$Y_3 = S_1\ S_0\ D$ (When $S_1 = 1, S_0 = 1$)

★ Truth Table (1:4 De-MUX)

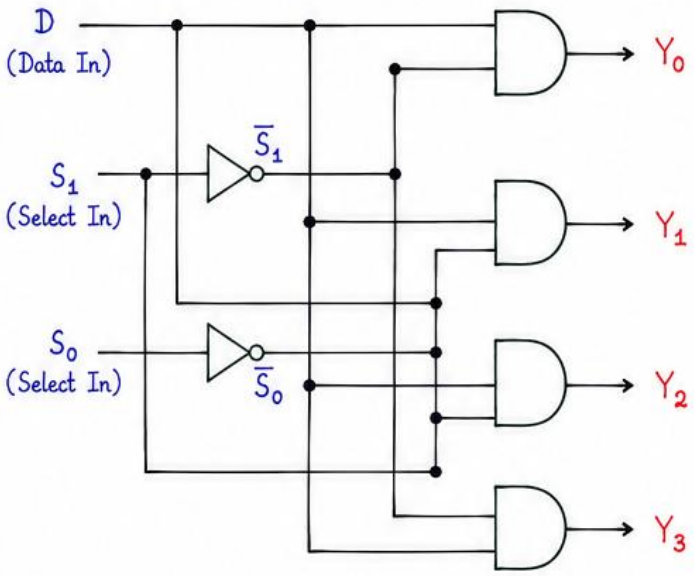
Select Inputs		Data Input	Outputs			
S_1	S_0	D	Y_0	Y_1	Y_2	Y_3
0	0	D	D	0	0	0
0	0	0	0	0	0	0
0	1	D	0	D	0	0
0	1	0	0	0	0	0
1	0	D	0	0	D	0
1	0	0	0	0	0	0
1	1	D	0	0	0	D
1	1	0	0	0	0	0

From the table :

- When $S_1\ S_0 = 00 \rightarrow Y_0 = D, Y_1 = Y_2 = Y_3 = 0$
- When $S_1\ S_0 = 01 \rightarrow Y_1 = D, Y_0 = Y_2 = Y_3 = 0$
- When $S_1\ S_0 = 10 \rightarrow Y_2 = D, Y_0 = Y_1 = Y_3 = 0$
- When $S_1\ S_0 = 11 \rightarrow Y_3 = D, Y_0 = Y_1 = Y_2 = 0$

Example :
If $D = 1, S_1 = 1, S_0 = 0$ (i.e. $S_1\ S_0 = 10$), then $Y_2 = 1$ and $Y_0 = Y_1 = Y_3 = 0$.

★ Circuit Diagram (1:4 De-MUX)



Top AND gate outputs $Y_0 = \overline{S}_1\ \overline{S}_0\ D$

2nd AND gate outputs $Y_1 = \overline{S}_1\ S_0\ D$

3rd AND gate outputs $Y_2 = S_1\ \overline{S}_0\ D$

Bottom AND gate outputs $Y_3 = S_1\ S_0\ D$

★ Key Points

- 1:4 De-MUX has 1 data input (D), 2 select inputs (S_1, S_0) and 4 outputs (Y_0, Y_1, Y_2, Y_3).
- It routes the data input to one of the four outputs based on the select inputs.
- Only one output line is active (HIGH) at a time; others remain LOW (0).
- It is the reverse operation of a 4:1 MUX.
- Used in data distribution, memory selection, communication systems, etc.

NOTE :
The selected output gets the data input (D). All other outputs are 0.

1:8 De-MUX – Operation, Truth Table and Circuit diagram

Combinational Logic Circuits

★ Introduction

A De-Multiplexer (De-MUX) is a combinational circuit that takes one data input and forwards it to one of the many output lines based on the select inputs. It performs the reverse operation of a Multiplexer.

A 1:8 De-MUX has one data input (D), three select inputs (S_2, S_1, S_0) and eight output lines (Y_0 – Y_7). Only one output is active (HIGH) at a time.

★ Operation

- The data input D is applied to the de-multiplexer.
- The select inputs (S_2, S_1, S_0) select one of the eight outputs.
- The selected output gets the data input D.
- All other outputs remain 0. (LOW).

General Expressions:

$Y_0 = \overline{S}_2\ \overline{S}_1\ \overline{S}_0\ D$ (When $S_2\ S_1\ S_0 = 000$)

$Y_1 = \overline{S}_2\ S_1\ S_0\ D$ (When $S_2\ S_1\ S_0 = 001$)

$Y_2 = \overline{S}_2\ S_1\ \overline{S}_0\ D$ (When $S_2\ S_1\ S_0 = 010$)

⋮

$Y_7 = S_2\ S_1\ S_0\ D$ (When $S_2\ S_1\ S_0 = 111$)

★ Truth Table (1:8 De-MUX)

Select Inputs			Data Input	Outputs							
S_2	S_1	S_0	D	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	D	D	0	0	0	0	0	0	0
0	0	1	D	0	D	0	0	0	0	0	0
0	1	0	D	0	0	D	0	0	0	0	0
0	1	1	D	0	0	0	D	0	0	0	0
1	0	0	D	0	0	0	0	D	0	0	0
1	0	1	D	0	0	0	0	0	D	0	0
1	1	0	D	0	0	0	0	0	0	D	0
1	1	1	D	0	0	0	0	0	0	0	D

From the table:

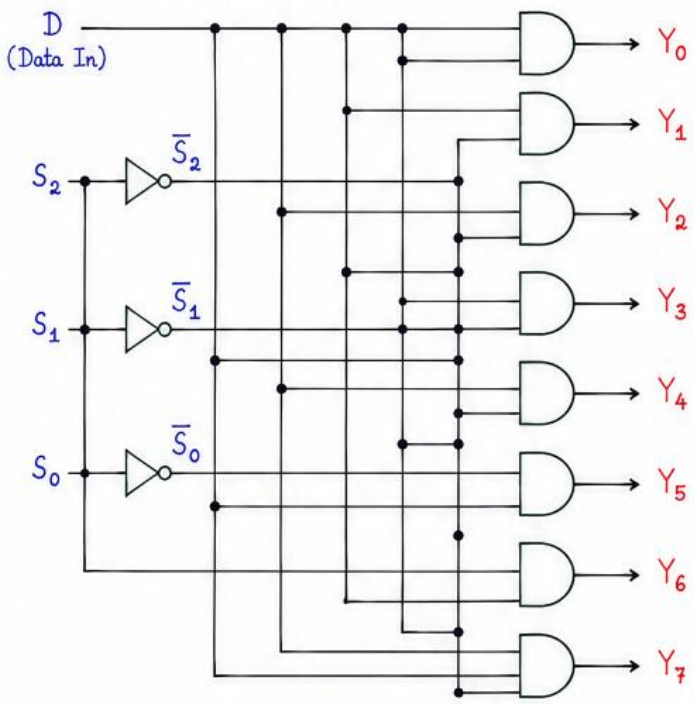
- When $S_2\ S_1\ S_0 = 000 \rightarrow Y_0 = D$, others = 0
- When $S_2\ S_1\ S_0 = 001 \rightarrow Y_1 = D$, others = 0
- When $S_2\ S_1\ S_0 = 010 \rightarrow Y_2 = D$, others = 0
- When $S_2\ S_1\ S_0 = 011 \rightarrow Y_3 = D$, others = 0

⋮

- When $S_2\ S_1\ S_0 = 111 \rightarrow Y_7 = D$, others = 0

Example:
If D = 1 and $S_2\ S_1\ S_0 = 101$ (i.e. $S_2 = 1$, $S_1 = 0$, $S_0 = 1$) then $Y_5 = 1$ and all other outputs are 0.

★ Circuit Diagram (1:8 De-MUX)



Each output is the result of a 4-input AND gate where inputs are D, S_2 (or $\overline{S}_2$), S_1 (or $\overline{S}_1$), S_0 (or $\overline{S}_0$).

★ Key Points

- 1:8 De-MUX has 1 data input (D), 3 select inputs (S_2, S_1, S_0) and 8 outputs (Y_0 – Y_7).
- It routes the data input to one of the eight outputs based on the select inputs.
- Only one output line is active (HIGH) at a time; all others are LOW (0).
- It is the reverse operation of an 8:1 MUX.
- Used in data distribution, memory chip enable, communication systems, etc.

NOTE:
The select inputs determine which output line will carry the data. All other output lines remain 0.

Design a 1:4 DeMUX using 1:2 DeMUX

Combinational Logic Circuits

★ Objective

To design a 1:4 DeMultiplexer using two 1:2 DeMultiplexers.

★ 1:4 DeMUX

A 1:4 DeMUX has one data input (D), two select lines (S_1, S_0) and four outputs (Y_0, Y_1, Y_2, Y_3). The data input D is routed to one of the four outputs based on the select lines.

★ Design Procedure

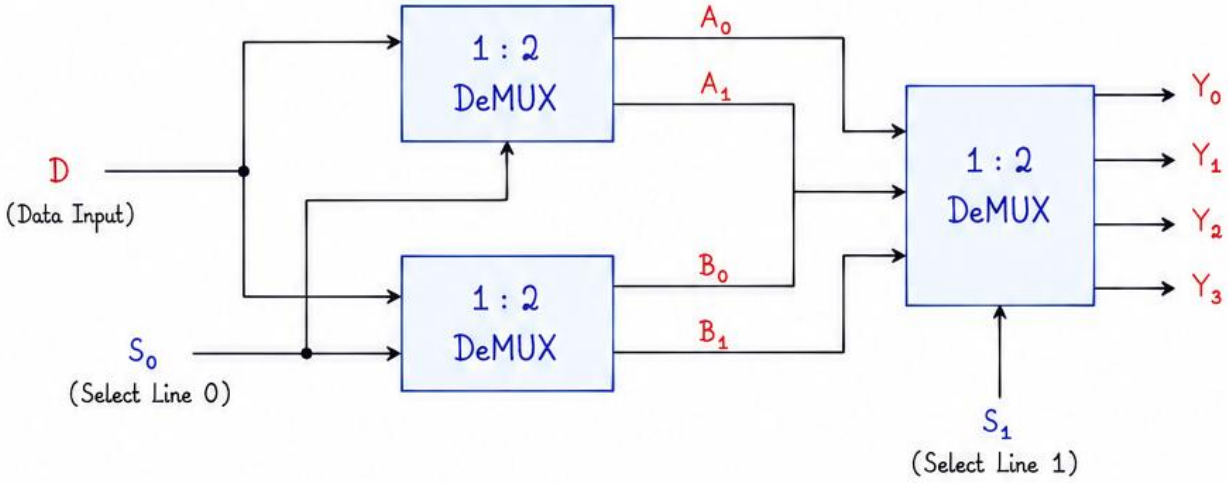
1. Use two 1:2 DeMUX.
2. Connect S_0 to the select input of both 1:2 DeMUX.
3. Connect D to the data input of both 1:2 DeMUX.
4. Use S_1 to select between the two 1:2 DeMUX outputs at the second stage.
5. The four outputs from both 1:2 DeMUXs become Y_0, Y_1, Y_2 and Y_3.

★ Truth Table of 1:4 DeMUX

S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

How it Works

- When $S_1 = 0$, the upper 1:2 DeMUX is enabled and D is sent to Y_0 or Y_1 based on S_0.
- When $S_1 = 1$, the lower 1:2 DeMUX is enabled and D is sent to Y_2 or Y_3 based on S_0.
- Thus, for each combination of (S_1, S_0), D appears at only one output.



Connection Summary

- D → Data input of both 1:2 DeMUXs
- S_0 → Select input of both 1:2 DeMUXs
- Outputs (A_0, A_1) and (B_0, B_1) → Inputs of second stage 1:2 DeMUX
- S_1 → Select input of second stage 1:2 DeMUX
- Outputs of second stage → Y_0, Y_1, Y_2, Y_3

Output Description

- If $S_1 = 0$:
  If $S_0 = 0$ → $Y_0 = D$, others = 0
  If $S_0 = 1$ → $Y_1 = D$, others = 0
- If $S_1 = 1$:
  If $S_0 = 0$ → $Y_2 = D$, others = 0
  If $S_0 = 1$ → $Y_3 = D$, others = 0

Design a 1:8 DeMUX using 1:4 DeMUX

Combinational Logic Circuits

★ Objective

To design a 1:8 DeMultiplexer using two 1:4 DeMultiplexers.

★ 1:8 DeMUX

A 1:8 DeMUX has one data input (D), three select lines (S_2, S_1, S_0) and eight outputs (Y_0 to Y_7). The data input D is routed to one of the eight outputs based on the select lines.

★ Design Procedure

1. Use two 1:4 DeMUX.
2. Connect D to the data input of both 1:4 DeMUX.
3. Connect S_1 and S_0 to the select inputs of both 1:4 DeMUX.
4. Connect the outputs of both 1:4 DeMUX to the outputs $Y_0 - Y_7$.
5. Use S_2 to enable one of the two 1:4 DeMUX.
6. When $S_2 = 0$, the upper 1:4 DeMUX is enabled → Y_0 to Y_3.
7. When $S_2 = 1$, the lower 1:4 DeMUX is enabled → Y_4 to Y_7.

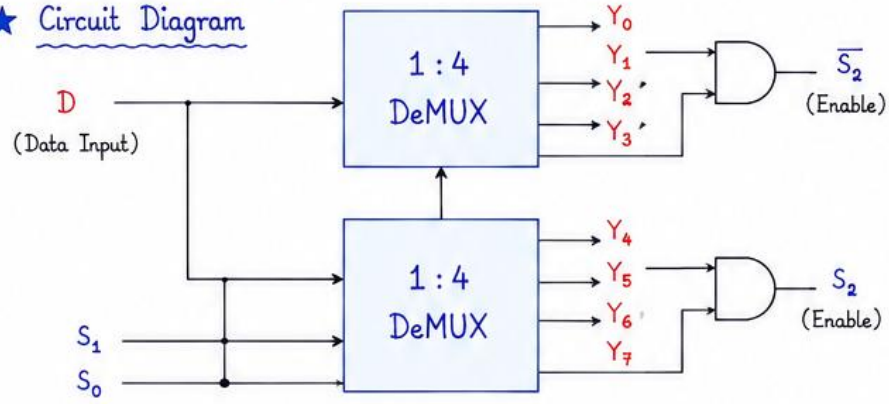
★ Truth Table of 1:8 DeMUX

S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	D	0	0	0	0	0	0	0
0	0	1	0	D	0	0	0	0	0	0
0	1	0	0	0	D	0	0	0	0	0
0	1	1	0	0	0	D	0	0	0	0
1	0	0	0	0	0	0	D	0	0	0
1	0	1	0	0	0	0	0	D	0	0
1	1	0	0	0	0	0	0	0	D	0
1	1	1	0	0	0	0	0	0	0	D

How it Works

- S_2 acts as the most significant select line.
- If $S_2 = 0$ → upper 1:4 DeMUX is enabled and D is distributed to $Y_0 - Y_3$ based on S_1, S_0.
- If $S_2 = 1$ → lower 1:4 DeMUX is enabled and D is distributed to $Y_4 - Y_7$ based on S_1, S_0.
- Only one output among $Y_0 - Y_7$ will be equal to D, all others will be 0.

★ Circuit Diagram



Select Lines

- S_1, S_0 → Connected to the select inputs of both 1:4 DeMUXs.
- S_2 → Used to enable one of the DeMUXs.

Note :

$S_2 = 0$ → Y_0 to Y_3 active

$S_2 = 1$ → Y_4 to Y_7 active

Connection Summary

- D → Data input of both 1:4 DeMUXs.
- S_1, S_0 → Select inputs of both 1:4 DeMUXs.
- Outputs of upper DeMUX → Y_0, Y_1, Y_2, Y_3.
- Outputs of lower DeMUX → Y_4, Y_5, Y_6, Y_7.
- S_2 → Enable control (0 for upper, 1 for lower).

Output Description

- For $S_2 = 0$:
  - $Y_0 = D$ (if $S_1S_0 = 00$)
  - $Y_1 = D$ (if $S_1S_0 = 01$)
  - $Y_2 = D$ (if $S_1S_0 = 10$)
  - $Y_3 = D$ (if $S_1S_0 = 11$)
- For $S_2 = 1$:
  - $Y_4 = D$ (if $S_1S_0 = 00$)
  - $Y_5 = D$ (if $S_1S_0 = 01$)
  - $Y_6 = D$ (if $S_1S_0 = 10$)
  - $Y_7 = D$ (if $S_1S_0 = 11$)