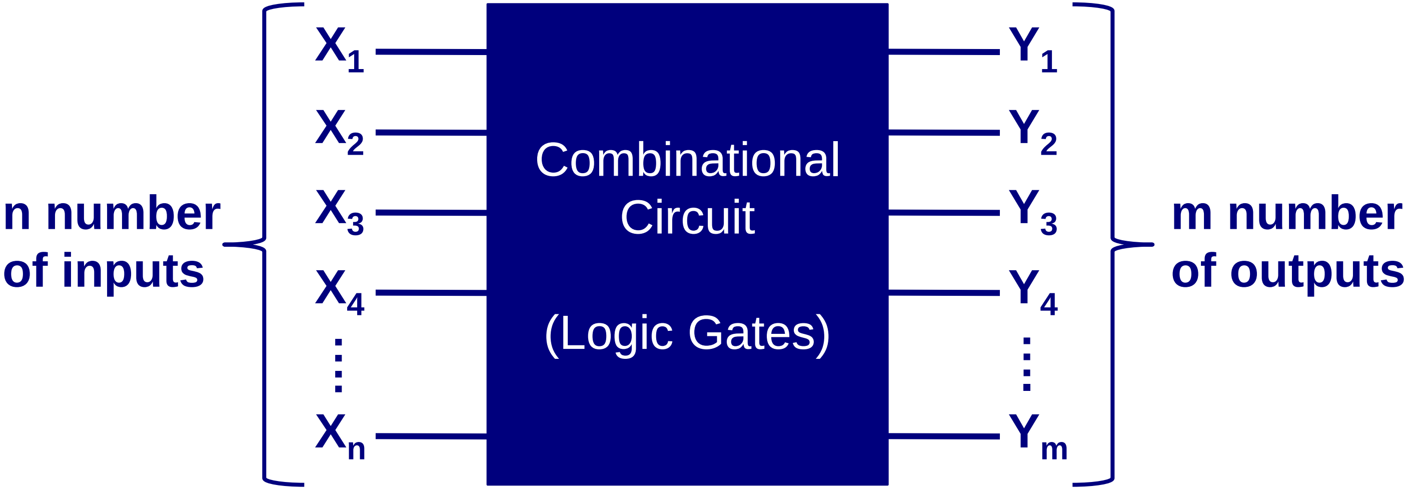
Combinational Circuits (Part-I)

Nilesh Patidar and Shiraz Husain

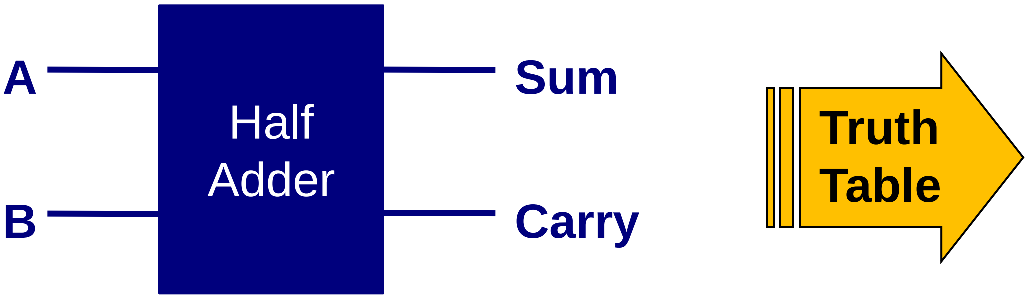
Combinational Circuits



Does not contain any memory element/feedback path

Half adder

- Half adder is used for addition of 2 bits.



A

B

Half Adder

Sum

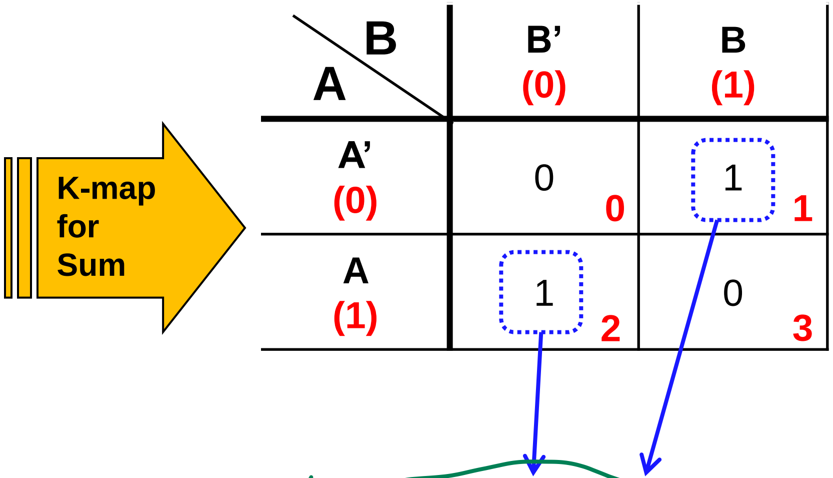
Carry

Truth Table

xor and

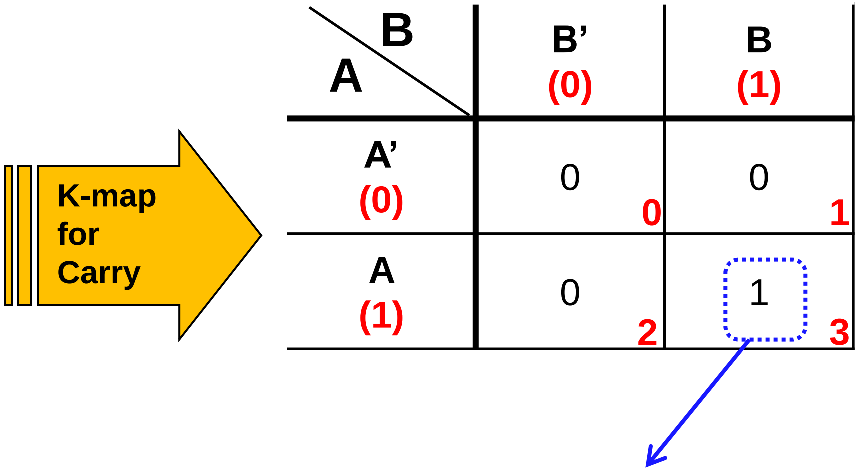
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half adder



K-map for Sum

B A	B’ (0)	B (1)
A’ (0)	0 0	1 1
A (1)	1 2	0 3



K-map for Carry

B A	B’ (0)	B (1)
A’ (0)	0 0	0 1
A (1)	0 2	1 3

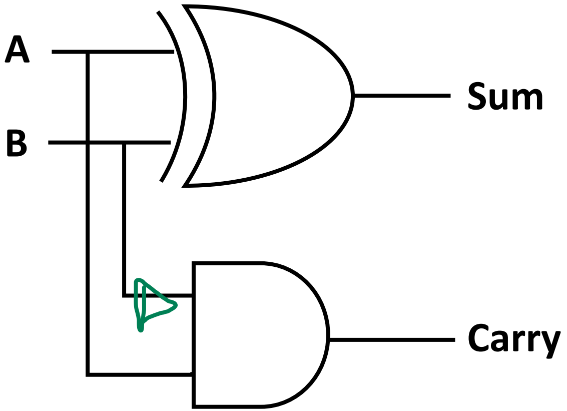
difference

Sum = AB’+A’B

= A⊕B

Carry = AB

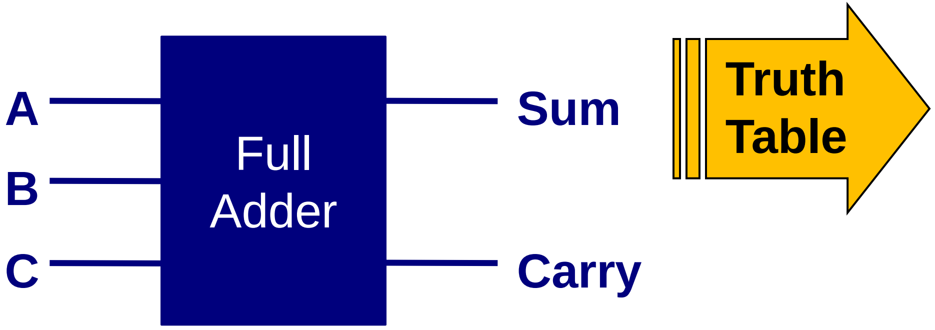
Half adder



Logic Circuit of Half adder

Full adder

- Full adder is used for addition of 3 bits.



			XOR	AND
A	**B**	**C**	**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

Full adder

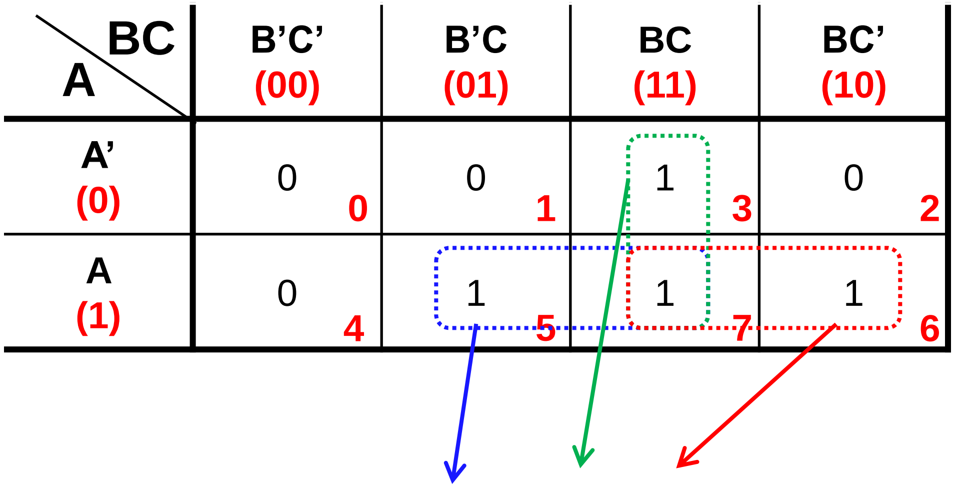
K-map for Sum

BC A	B’C’ (00)	B’C (01)	BC (11)	BC’ (10)
A’ (0)	0 0	1 1	0 3	1 2
A (1)	1 4	0 5	1 7	0 6

Sum = AB’C’+ABC+A’B’C+A’BC’
= A(B’C’+BC)+A’(B’C+BC’)
= A(B⊕C)’+ A’(B⊕C)
= A⊕B⊕C FULL SUBTRACT

Full adder

K-map for Carry



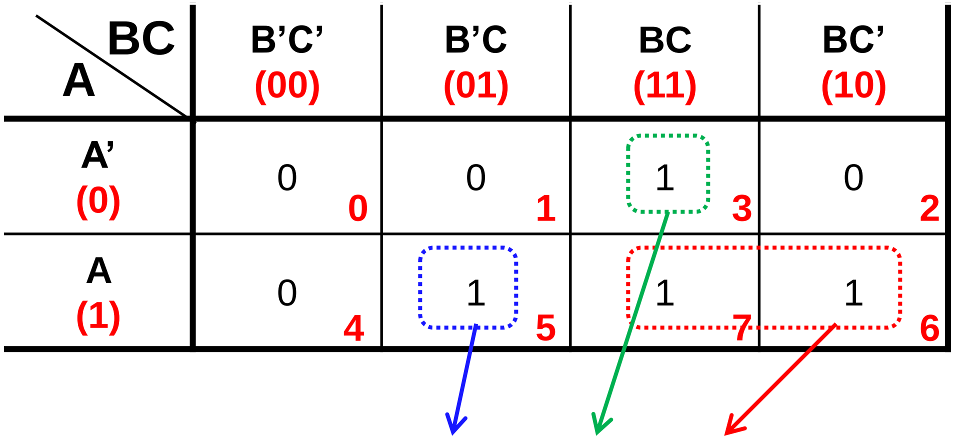
A \ BC	B’C’ (00)	B’C (01)	BC (11)	BC’ (10)
A’ (0)	0 0	0 1	1 3	0 2
A (1)	0 4	1 5	1 7	1 6

Carry = AC + BC + AB

A' =SUB.

Full adder

K-map for Carry



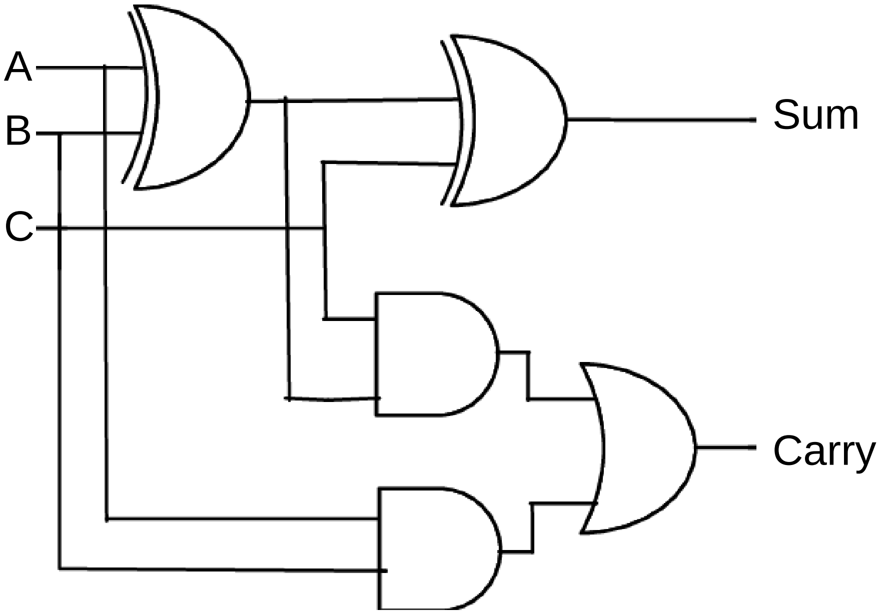
A \ BC	B'C' (00)	B'C (01)	BC (11)	BC' (10)
A' (0)	0 0	0 1	1 3	0 2
A (1)	0 4	1 5	1 7	1 6

Carry = AB'C+A'BC+AB

= (AB'+A'B)C+AB

= (A⊕B)C+AB

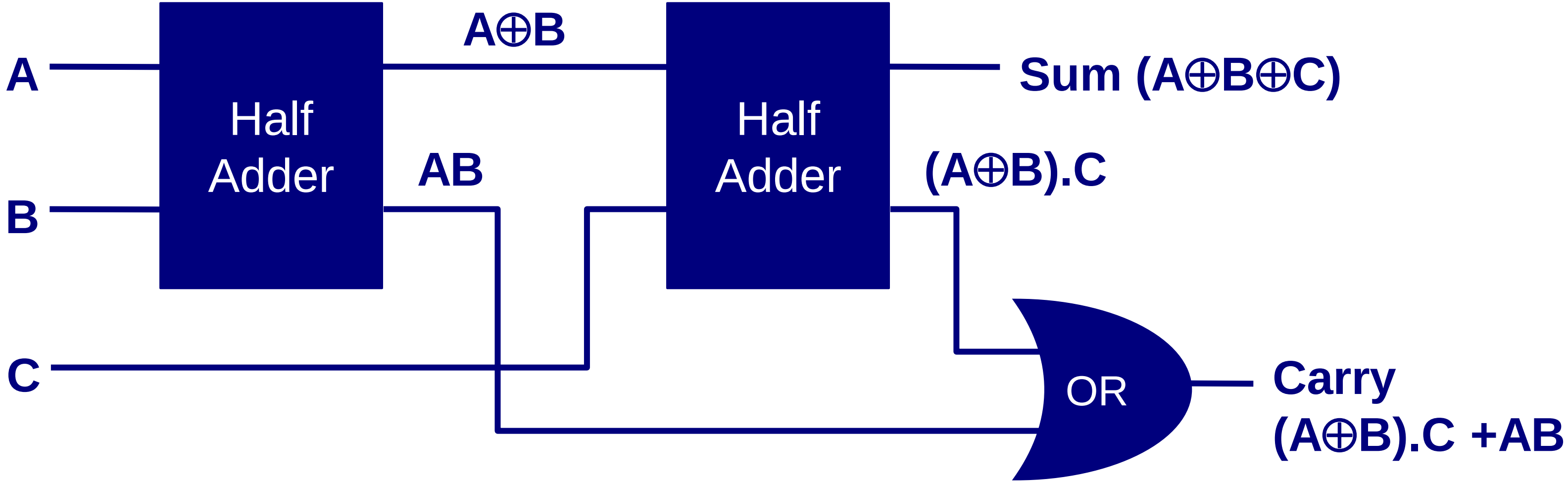
Full adder



logic circuit of full adder

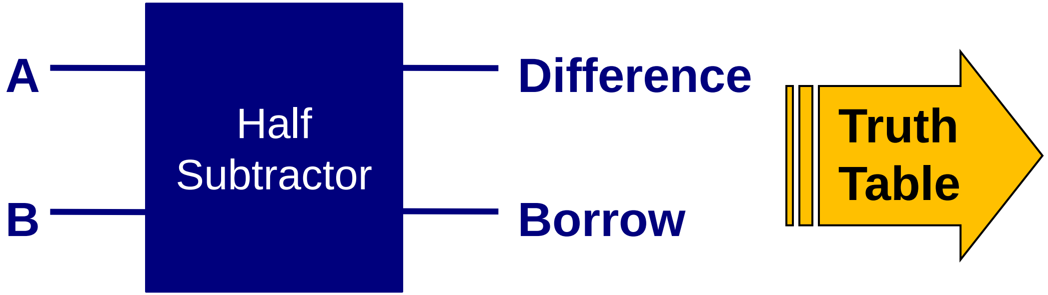
Design of Full adder using Half adders

- Full adder can be designed by using 2 half adders and 1 OR gate.



Half subtractor

- Half subtractor is used for subtraction of 2 bits.



A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Half subtractor

K-map for Difference

B A	B’ (0)	B (1)
A’ (0)	0 0	1 1
A (1)	1 2	0 3

Difference = AB’+A’B

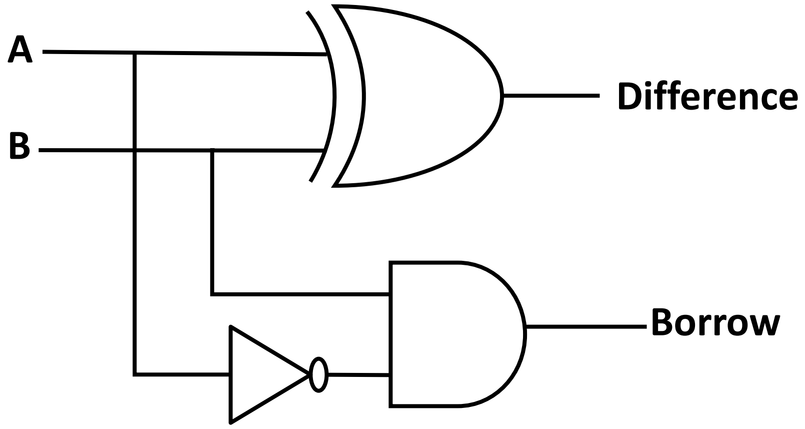
= A⊕B

K-map for Borrow

B A	B’ (0)	B (1)
A’ (0)	0 0	1 1
A (1)	0 2	0 3

Borrow = A’B

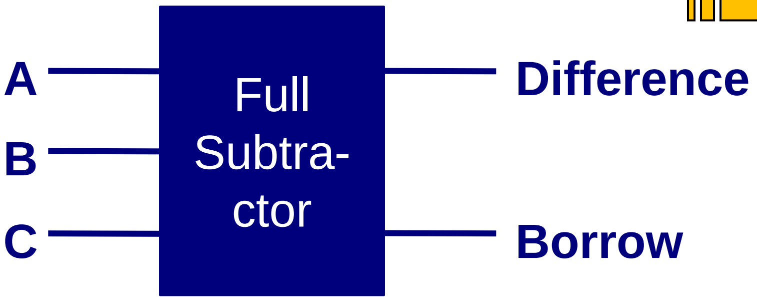
Half subtractor



Logic Circuit of Half subtractor

Full subtractor

- Full subtractor is used for subtraction of 3 bits.



A

B

C

Full Subtractor

Difference

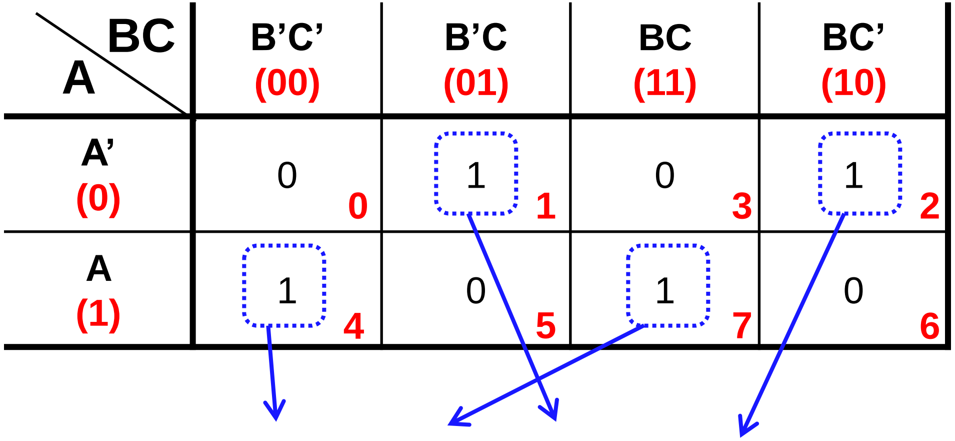
Borrow

Truth Table

A	B	C	Difference	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	0
1	1	1	1	1

Full subtractor

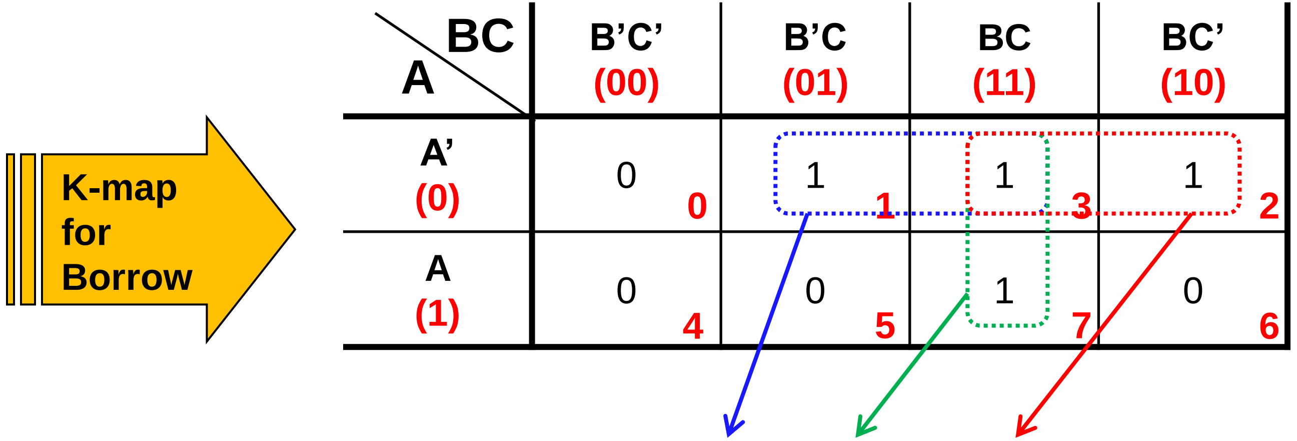
K-map for Diff.



A \ BC	B'C' (00)	B'C (01)	BC (11)	BC' (10)
A' (0)	0 0	1 1	0 3	1 2
A (1)	1 4	0 5	1 7	0 6

$$\begin{aligned}\text{Difference} &= AB'C'+ABC+A'B'C+A'BC' \\ &= A(B'C'+BC)+A'(B'C+BC') \\ &= A(B\oplus C)'+ A'(B\oplus C) \\ &= A\oplus B\oplus C\end{aligned}$$

Full subtractor



K-map for Borrow

BC A	B’C’ (00)	B’C (01)	BC (11)	BC’ (10)
A’ (0)	0 0	1 1	1 3	1 2
A (1)	0 4	0 5	1 7	0 6

Borrow = A’C + BC + A’B

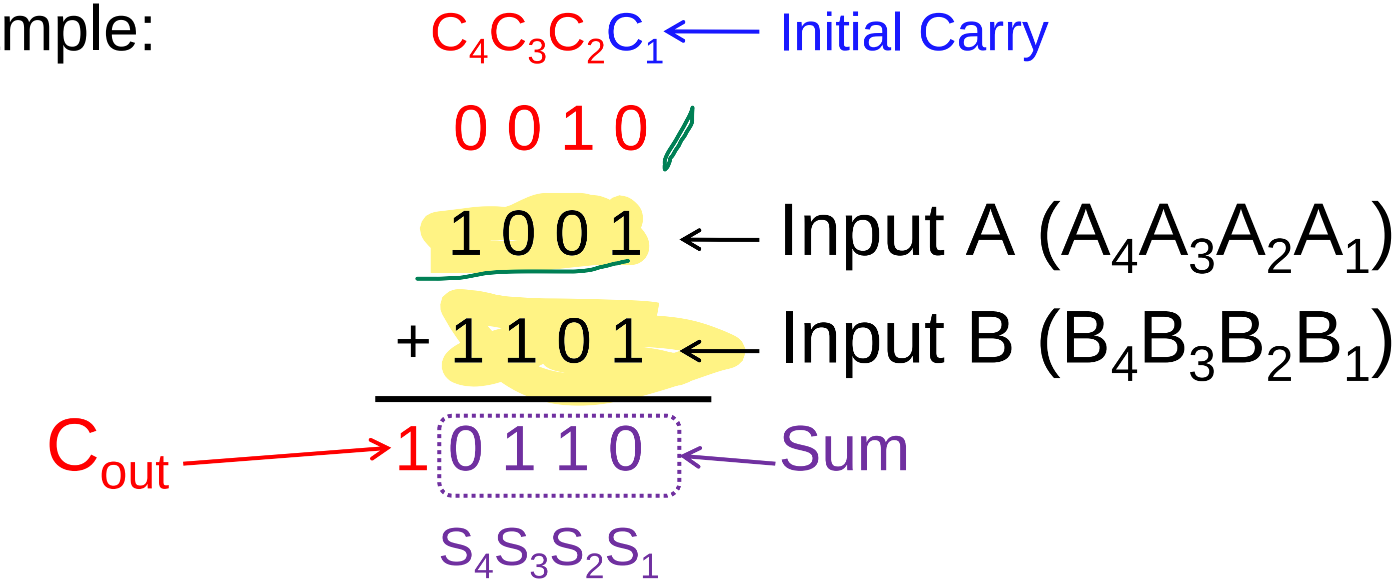
Full subtractor

?

Q.1 Design the logic circuit of full subtractor by using logic gates from obtained Boolean equations from K-map.

Parallel Adder (4-bit)

- Example:



$C_4C_3C_2C_1$ ← Initial Carry

0 0 1 0

1 0 0 1 ← Input A ($A_4A_3A_2A_1$)

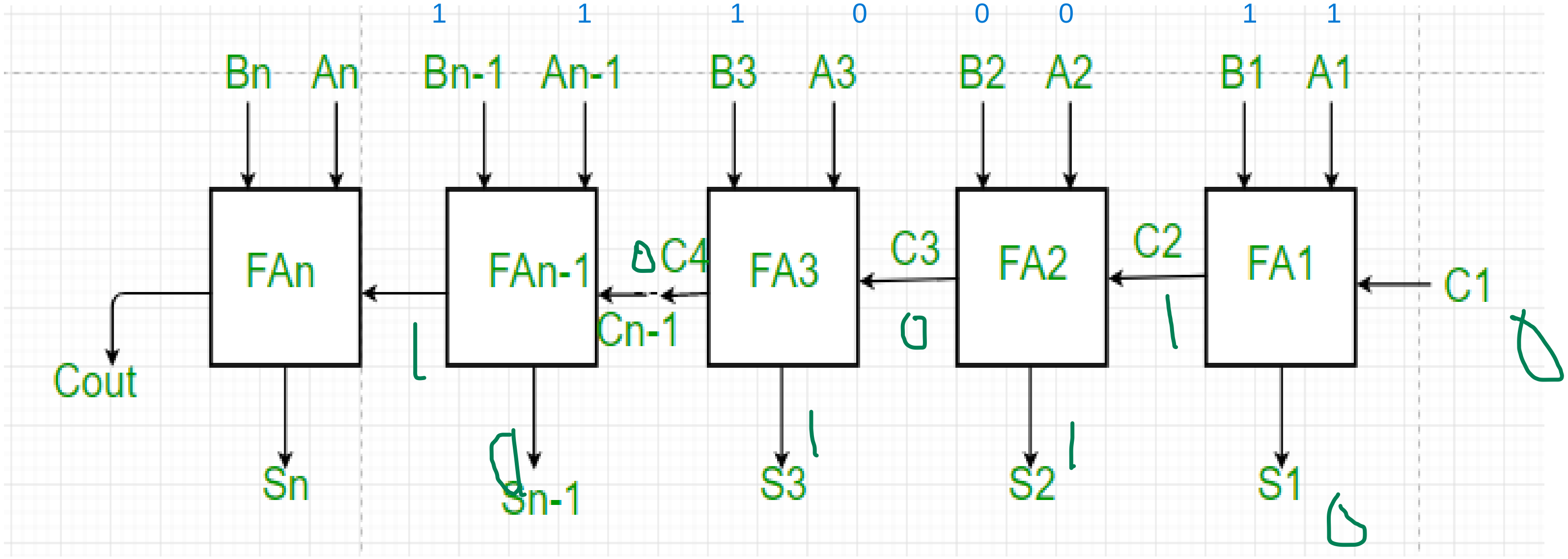
+ 1 1 0 1 ← Input B ($B_4B_3B_2B_1$)

C_{out} → 1 0 1 1 0 ← Sum

$S_4S_3S_2S_1$

Parallel Adder (n-bit)

COMBINATIONAL



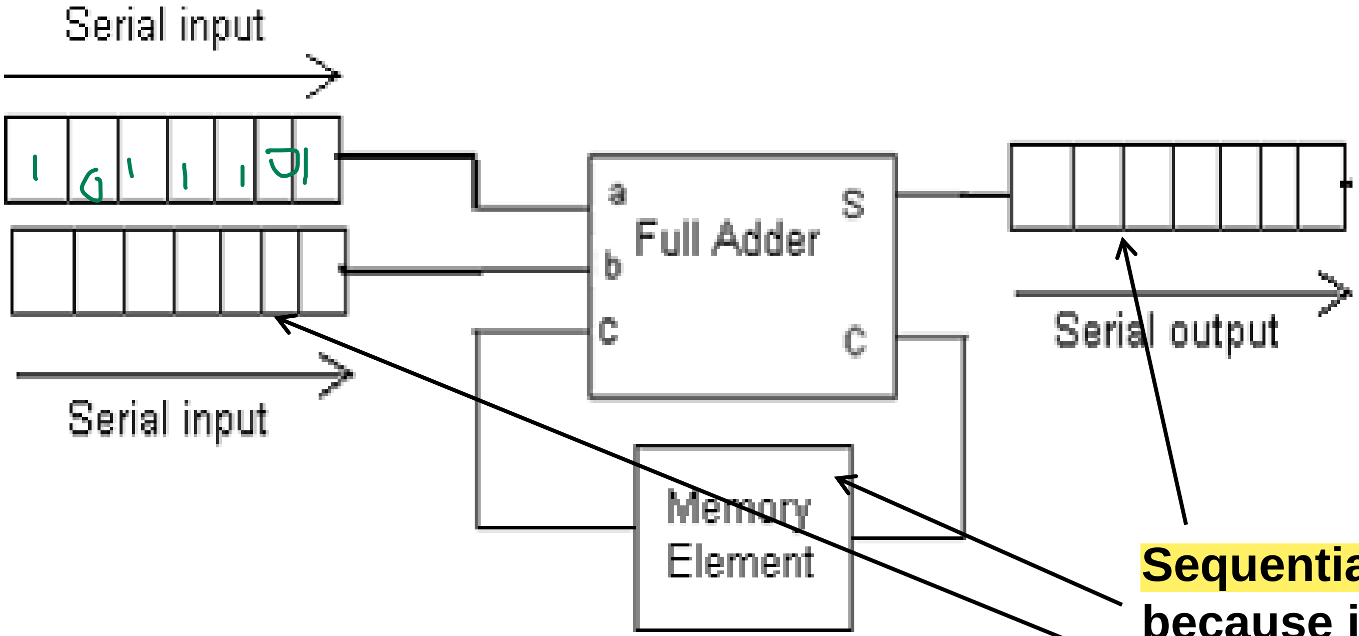
It is also known as ripple carry adder

Serial Adder

SQUE....

- **Serial binary adder** is a logic circuit that performs the addition of two binary numbers in serial form. Serial binary adder performs bit by bit addition. Two shift registers are used to store the binary numbers that are to be added.
- A single full adder is used to add one pair of bits at a time along with the carry. The carry output from the full adder is applied to a D flip-flop. After that output is used as carry for next significant bits. The sum bit from the output of the full adder can be transferred into a third shift register.

Serial Adder



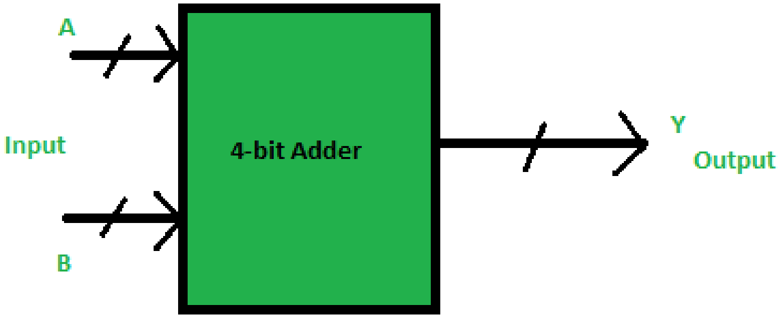
Sequential Circuit because it contain memory elements

Is it a combinational or sequential circuits?

BCD Adder

0110+

- BCD stand for binary coded decimal. (Range from 0 to 9)



Maximum value of any BCD numbers is 9. So, maximum sum of two BCD numbers are 18 (without any carry) and 19 (in case of previous carry)

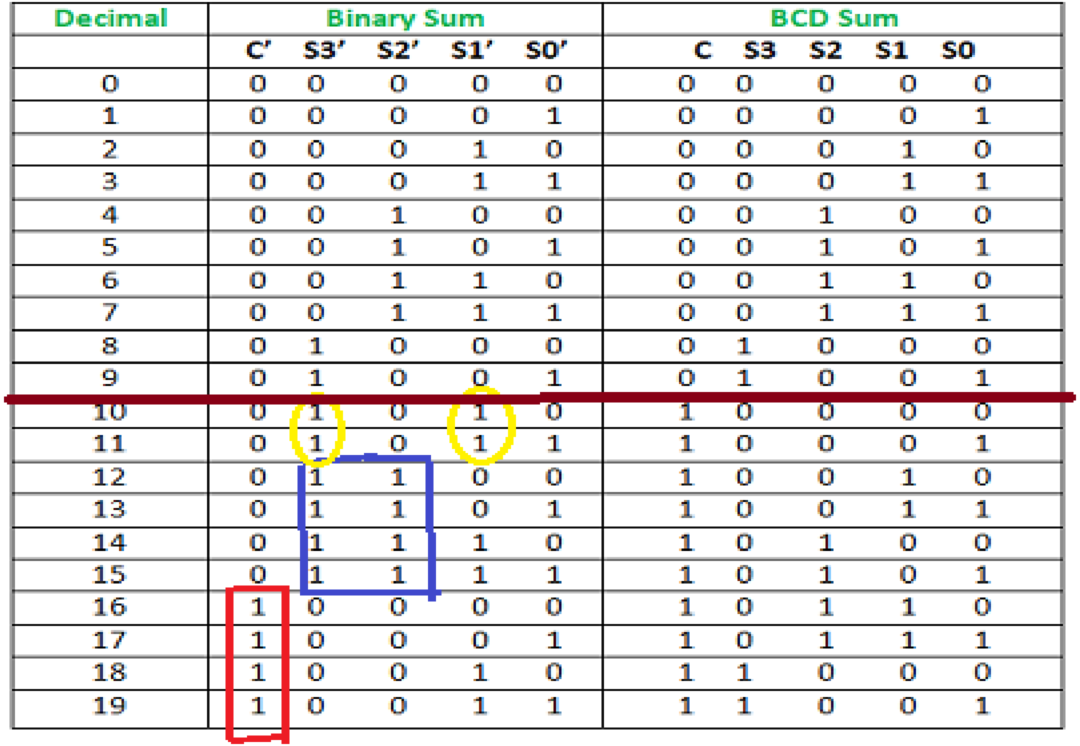
BCD Sum = 9 + 9 + 1(carry) = 19

BCD Adder

- Example: Add two BCD numbers.

```
                      1 0 0 1  ←  9 (in BCD)
                    + 0 1 0 1  ←  5 (in BCD)
                    ─────────
Binary Sum            1 1 1 0  ←  14 (in Decimal)←┐
                    + 0 1 1 0  ←  add 6 (in case of sum>9)
                    ─────────
BCD Sum     0 0 0 1   0 1 0 0  ←  1 4 (in BCD)
```

BCD Adder



Decimal	Binary Sum					BCD Sum				
	C'	S3'	S2'	S1'	S0'	C	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

BCD Adder

We are adding “0110” (=6) only to the second half of the table.

The conditions are:

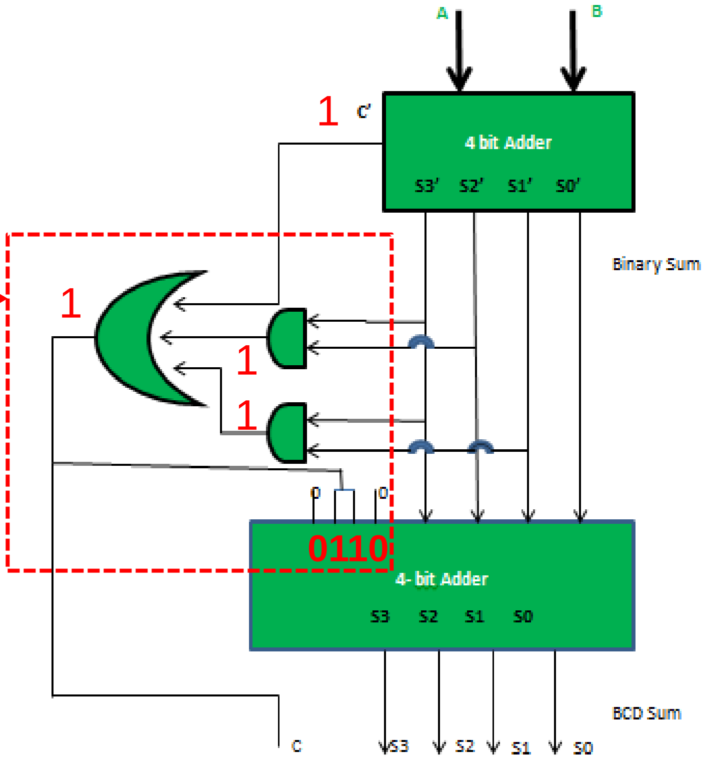
  - If C’ = 1 (Satisfies 16-19)
  - If S3′.S2′ = 1 (Satisfies 12-15)
  - If S3′.S1′ = 1 (Satisfies 10 and 11)

- So, our logic is

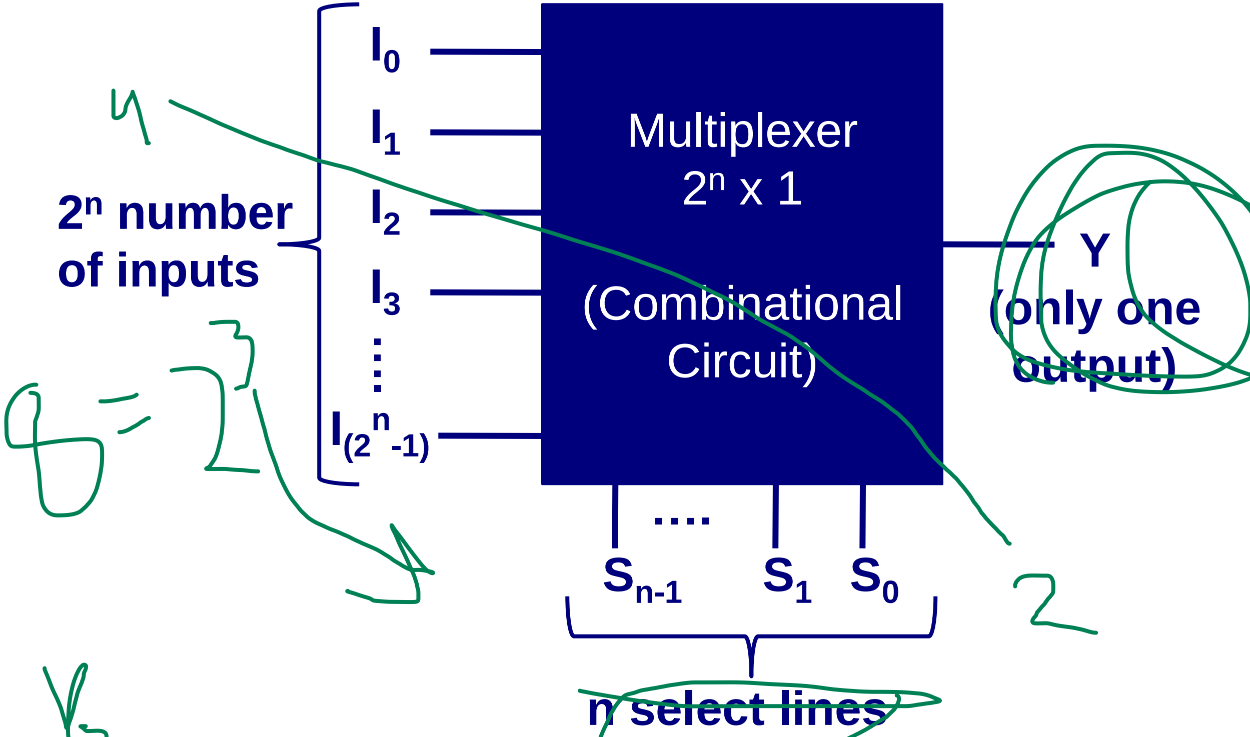
  C' + S3'.S2' + S3'.S1' = 1 (used to generate 6)

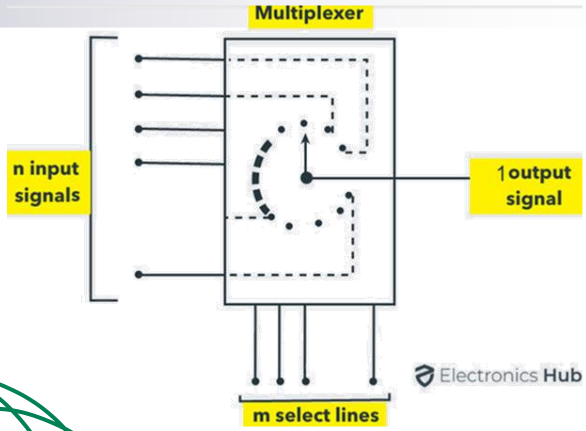
BCD Adder (Implementation)

C' + S3'.S2' + S3'.S1' = 1

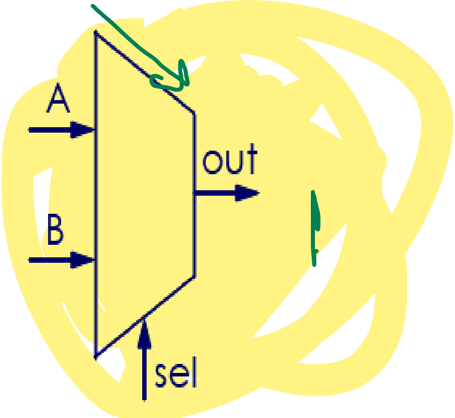


Multiplexer (MUX)





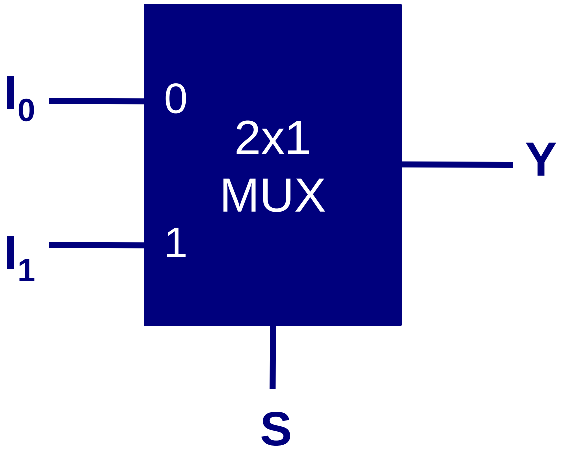
8 IS 1
4 IS1



Symbolic Representation

It is also called Data Selector

2x1 Multiplexer



Truth Table

S	I_0	I_1	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

Y=S'I0+SI1

Function Table

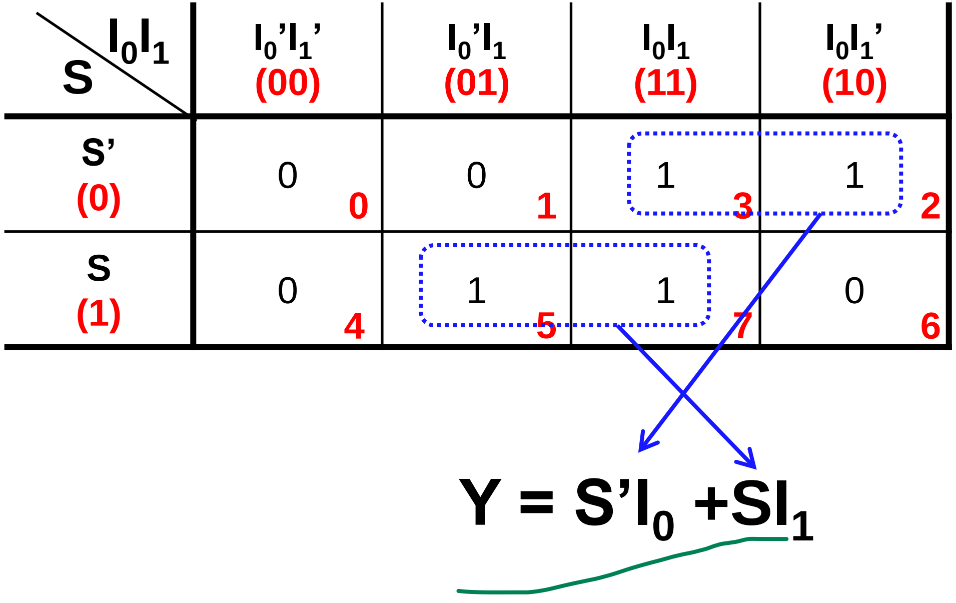
S	Y
0	I_0
1	I_1

S'I0+SI1=Y

$$Y = S'I_0 + SI_1$$

2x1 Multiplexer

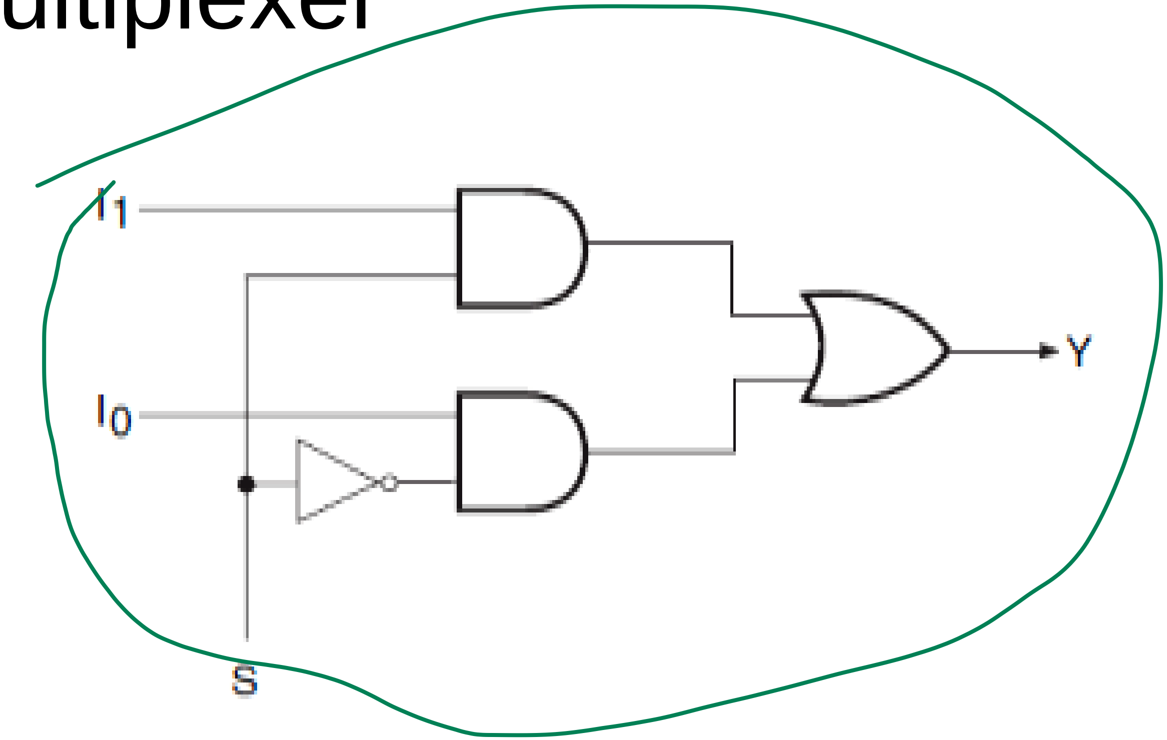
K-map
for
2x1 MUX



S \ I_0I_1	$I_0'I_1'$ (00)	$I_0'I_1$ (01)	I_0I_1 (11)	I_0I_1' (10)
S' (0)	0 0	0 1	1 3	1 2
S (1)	0 4	1 5	1 7	0 6

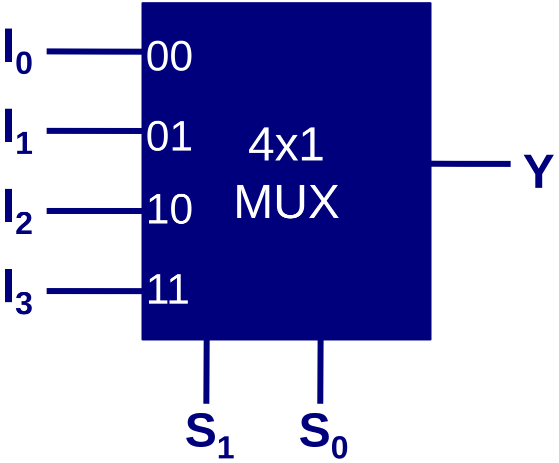
$$Y = S'I_0 + SI_1$$

2x1 Multiplexer



$Y = S'I_0 + SI_1$

4x1 Multiplexer

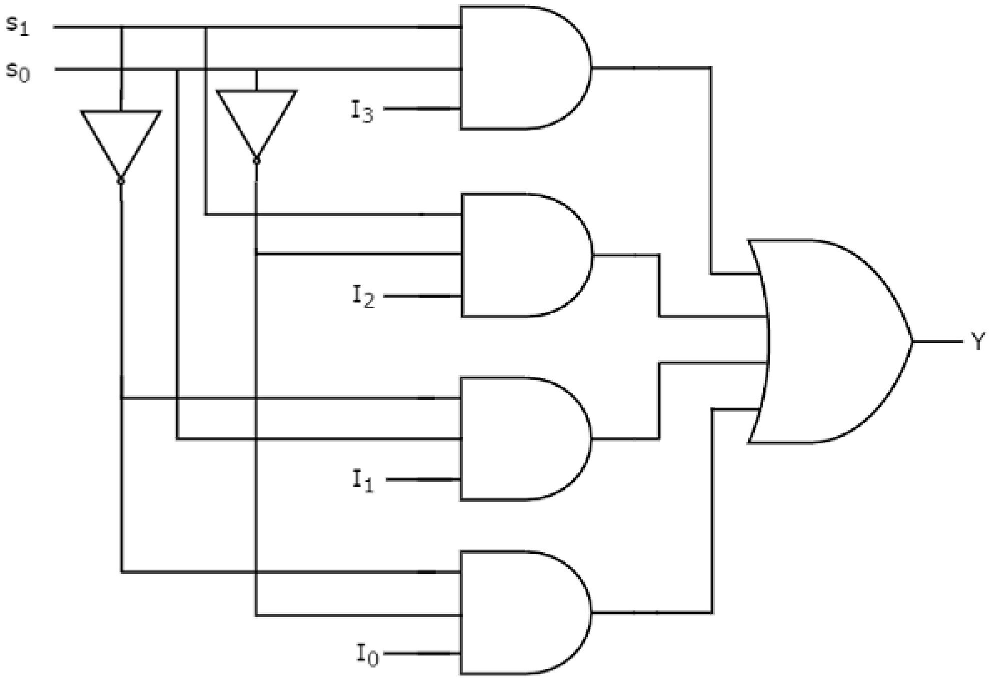


Function Table

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

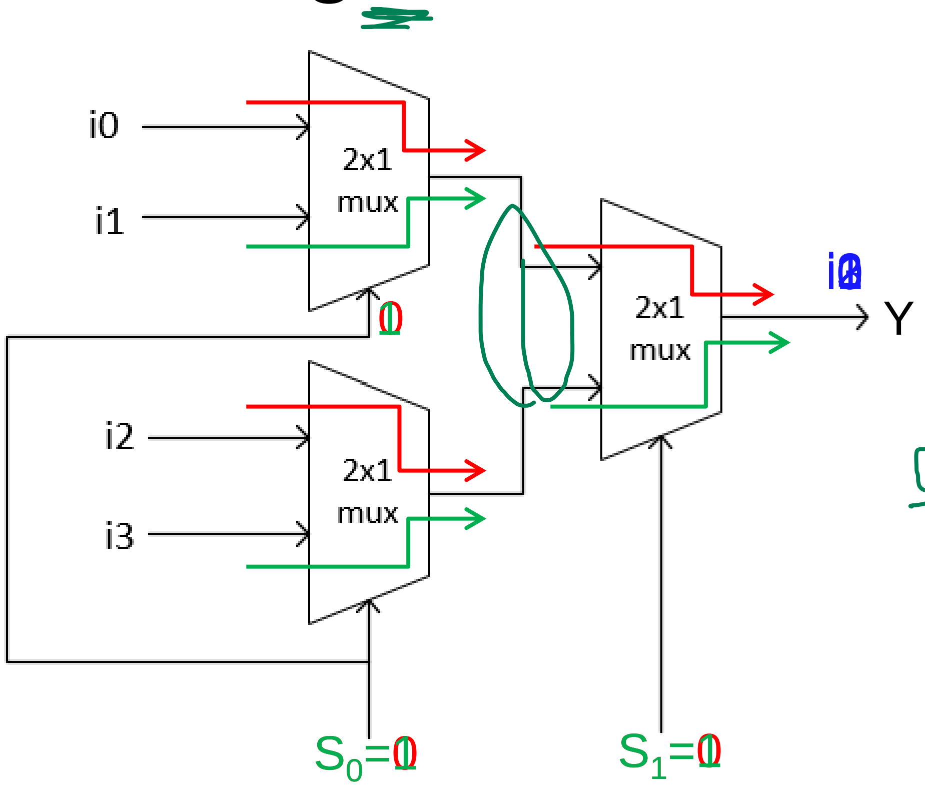
$$Y = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$$

4x1 Multiplexer



$$Y = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$$

4x1 MUX using 2x1 MUX



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

- To implement $2^n \times 1$ MUX using 2×1 MUX, the total number of 2×1 MUX required is $(2^n - 1)$.

∴ The number of 2 × 1 multiplexer required to implement 16 × 1 MUX will be:

$n = 16 - 1 = 15$

Or we can follow the below steps to calculate the same:

1^{st} stage $= \frac{16}{2} = 8$

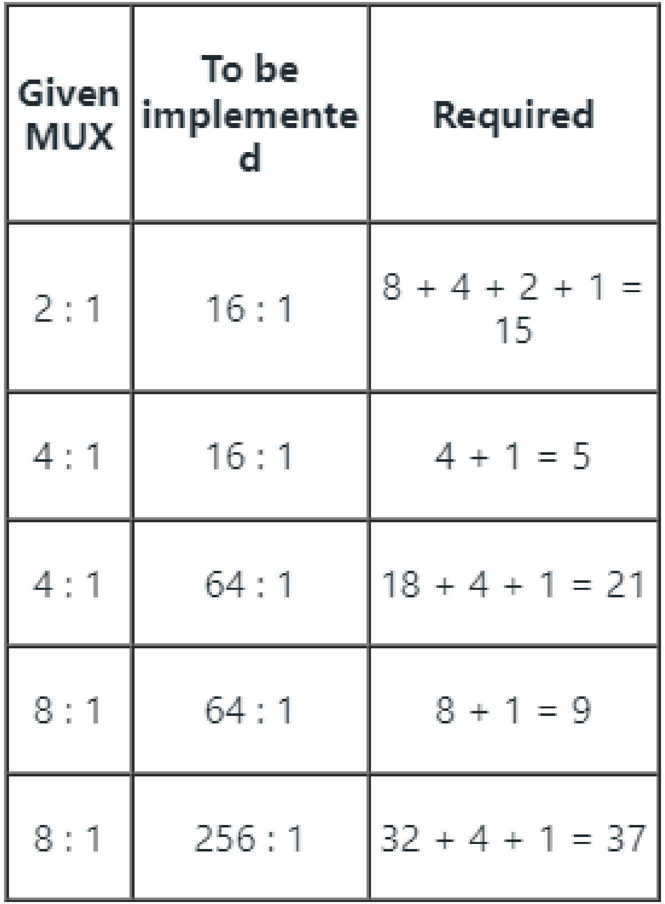
2^{nd} stage $= \frac{8}{2} = 4$

3^{rd} stage $= \frac{4}{2} = 2$

4^{th} stage $= \frac{2}{2} = 1$

The sum will give the total number of MUX required to implement 16 × 1 multiplexer using 2 × 1, i.e.

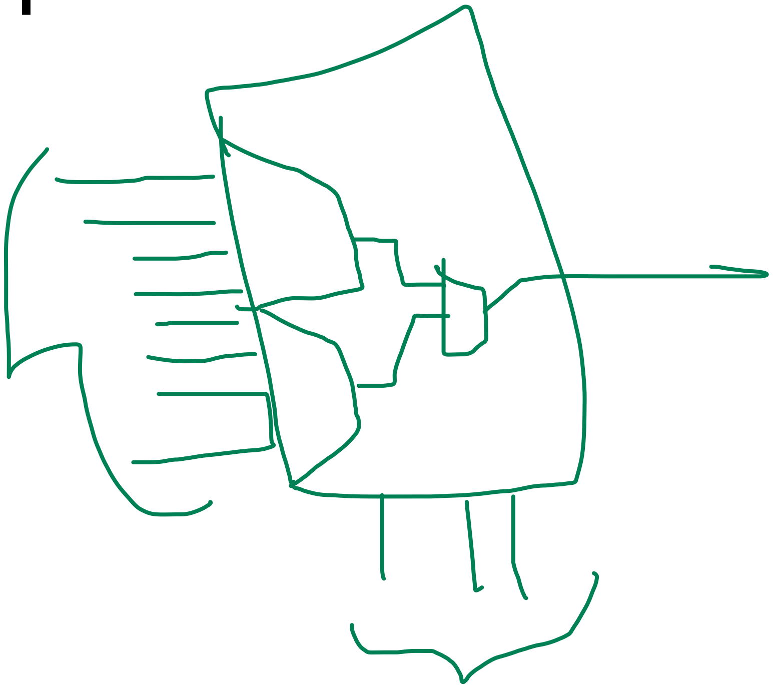
$n = 8 + 4 + 2 + 1 = 15$



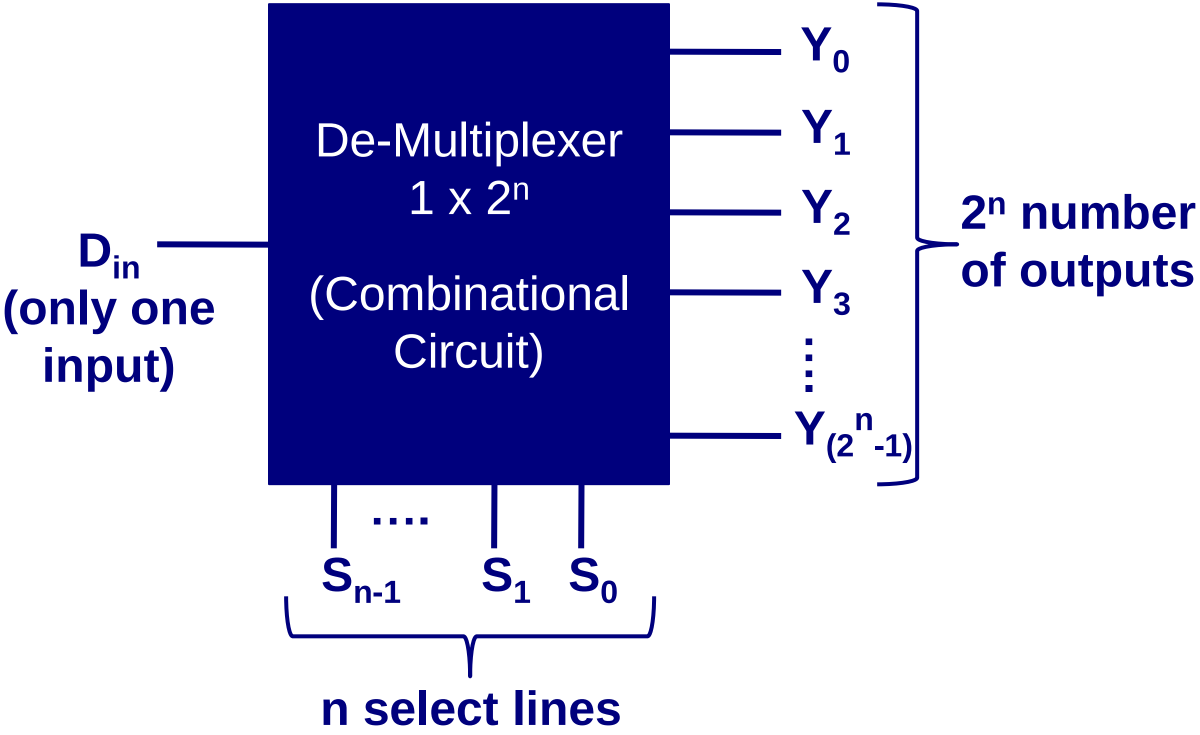
Given MUX	To be implemented	Required
2 : 1	16 : 1	8 + 4 + 2 + 1 = 15
4 : 1	16 : 1	4 + 1 = 5
4 : 1	64 : 1	18 + 4 + 1 = 21
8 : 1	64 : 1	8 + 1 = 9
8 : 1	256 : 1	32 + 4 + 1 = 37

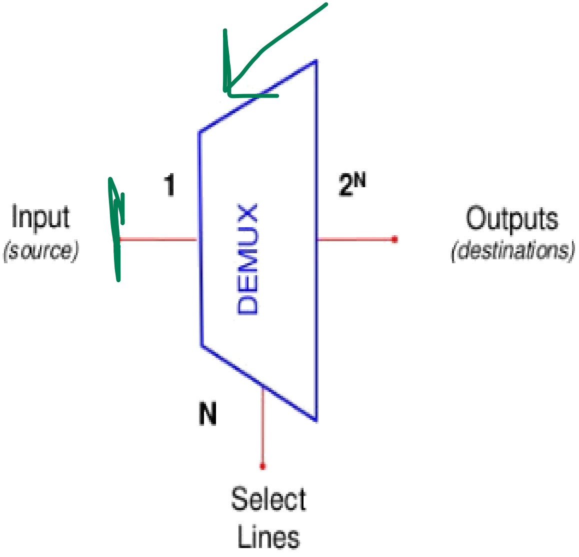
8x1 Multiplexer

8is 1 using 4:1



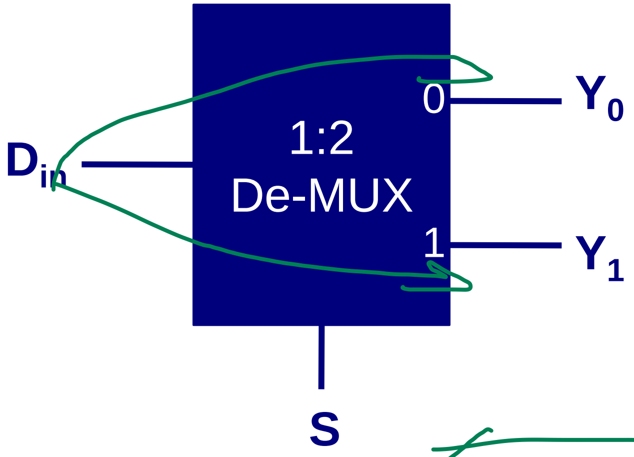
De-Multiplexer (De-MUX)





Symbolic Representation

1x2 De-Multiplexer



Truth Table

S	D_{in}	Y_0	Y_1
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

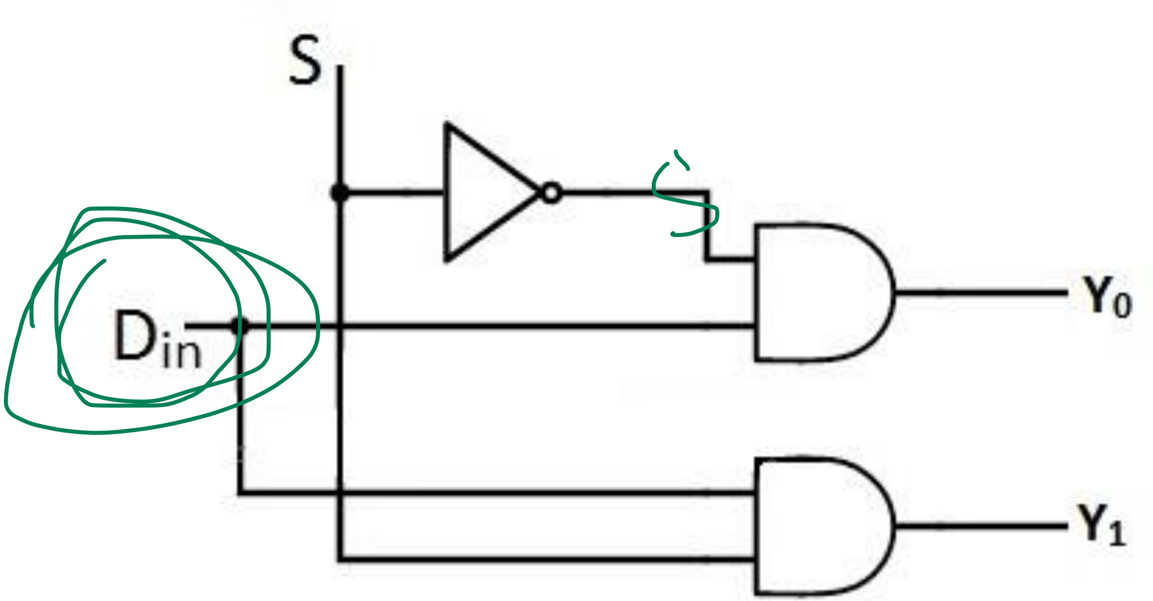
Function Table

S	Y_0	Y_1
0	D_{in}	0
1	0	D_{in}

$$Y_0 = S'D_{in}$$

$$Y_1 = SD_{in}$$

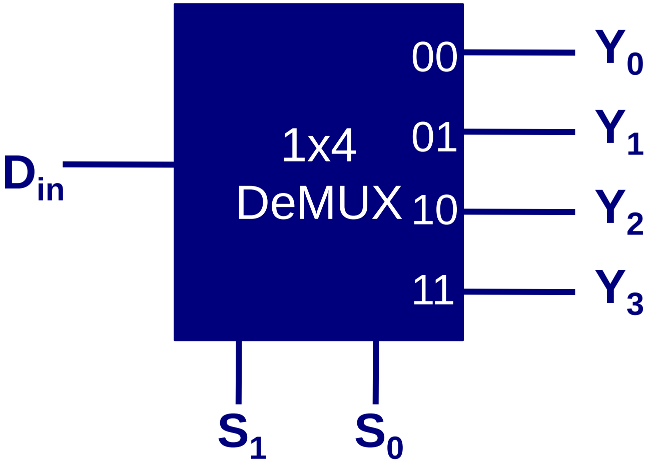
1x2 De-Multiplexer



$Y_0 = S'D_{in}$

$Y_1 = SD_{in}$

1x4 De-Multiplexer



Function Table

S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	D_{in}	0	0	0
0	1	0	D_{in}	0	0
1	0	0	0	D_{in}	0
1	1	0	0	0	D_{in}

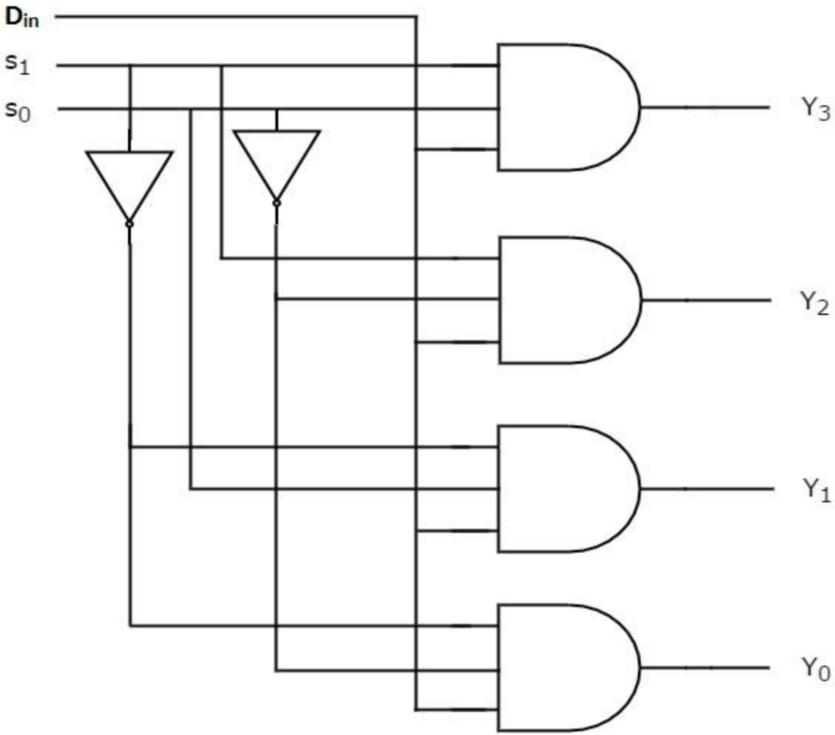
$Y_0 = S_1'S_0'D_{in}$

$Y_2 = S_1S_0'D_{in}$

$Y_1 = S_1'S_0D_{in}$

$Y_3 = S_1S_0D_{in}$

1x4 De-Multiplexer



$Y_3 = S_1S_0D_{in}$

$Y_2 = S_1S_0'D_{in}$

$Y_1 = S_1'S_0D_{in}$

$Y_0 = S_1'S_0'D_{in}$

Multiplexer (as universal logic gate)

- Multiplexer is also act as universal logic gate.
- It means that any of the logic circuit/gate can be implemented with the help of multiplexer.

m

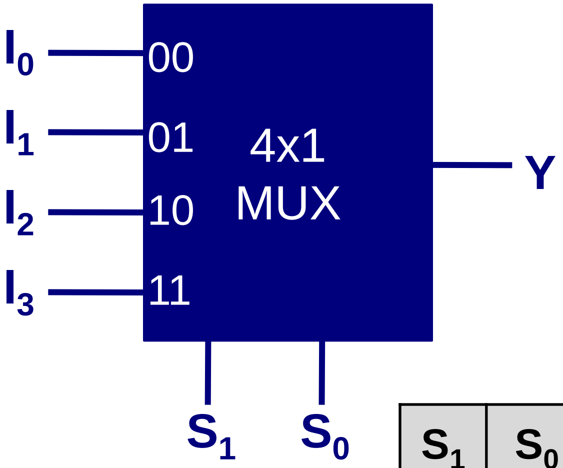
Implementation using MUX (Method 1)

- Example: Design 2-input OR gate with the help of MUX.
  - The Boolean equation for OR gate is-

    $$F(A,B) = A+B$$

  - First choose the size of MUX-
    - No. of select lines (n) = No. of variables in given Boolean function
    - No. of select lines (n) = 2 (A,B)
    - No. of inputs (2^n) = 4
    - It requires 4:1 MUX to implement OR gate

Implementation using MUX (Method 1)



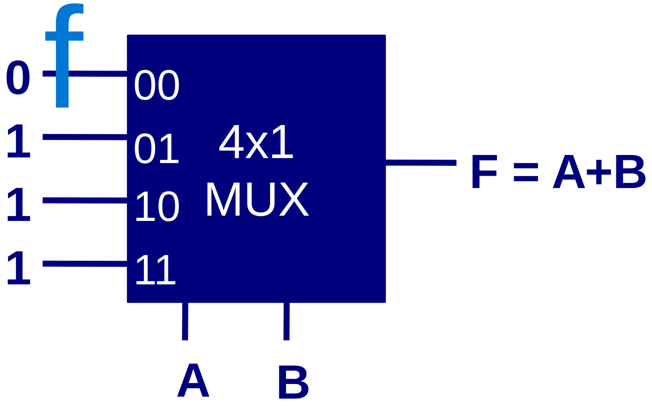
$F(A,B) = A+B$

Truth Table of OR

A	B	F	
0	0	0	I_0
0	1	1	I_1
1	0	1	I_2
1	1	1	I_3

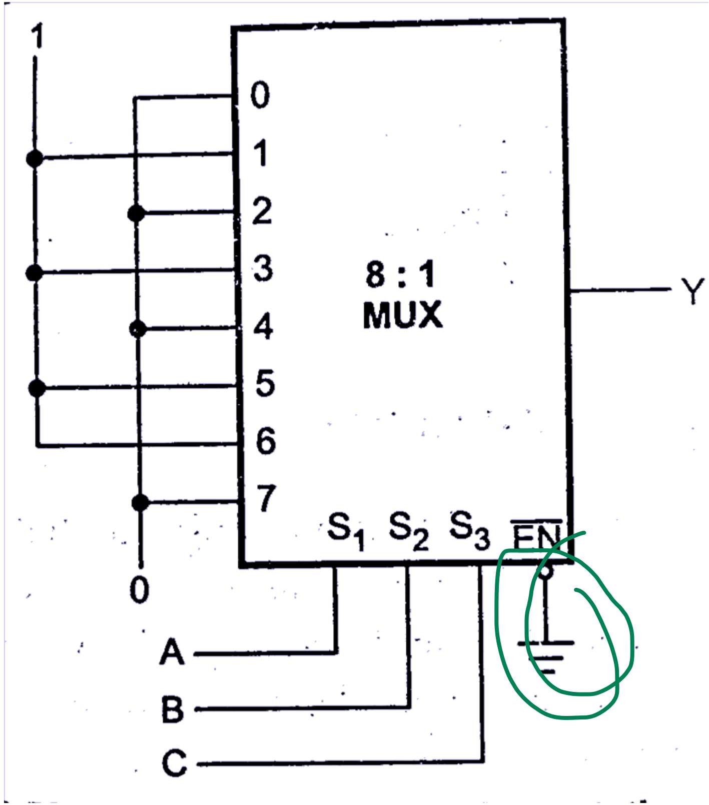
Function Table of MUX

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



Implementation using mux (Method 1)

- $F(A,B,C) = \sum m(1,3,5,6)$

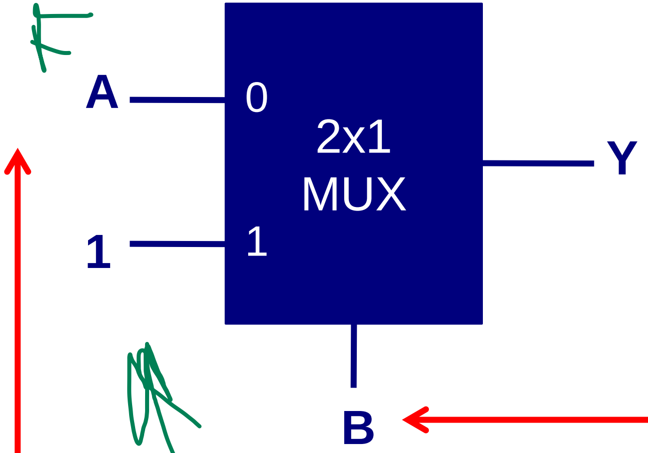


Implementation using MUX (Method 2)

- Example: Design 2-input OR gate with the help of 2:1 MUX.
  - The Boolean equation for OR gate is-

    F(A,B) = A+B
  - First choose the size of MUX-
    - No. of select lines (n) = No. of variables - 1
    - No. of select lines (n) = 2 – 1 = 1
    - No. of inputs (2^n) = 2
    - It requires 2:1 MUX to implement OR gate

Implementation using MUX (Method 2)



F(A,B) = A+B

Truth Table of OR

Decimal	A	B	F
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

Column	I0	I1
A'	0	1
A	2	3
	A	1

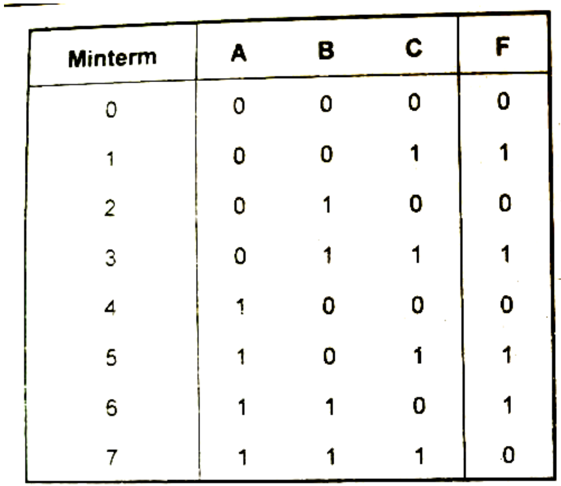
1

0 A'

S = B

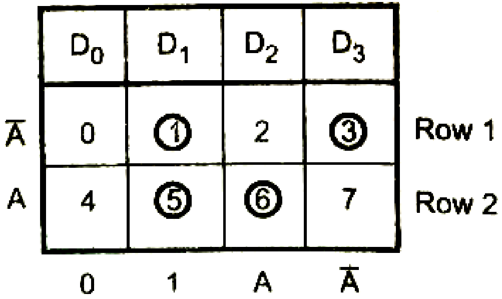
Implementation using mux (Method 2)

F(A,B,C) = ∑m(1,3,5,6)



Minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Fig. 3.57 (a) Truth table



	D_0	D_1	D_2	D_3	
$\bar{A}$	0	①	2	③	Row 1
A	4	⑤	⑥	7	Row 2
	0	1	A	$\bar{A}$	

Fig. 3.57 (c) Implementation table

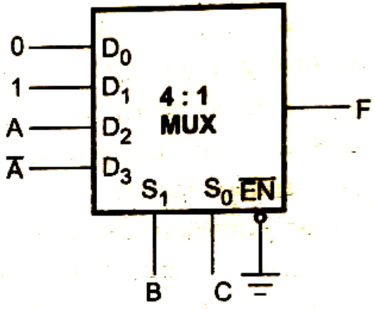


Fig. 3.57 (b) Multiplexer implementation

Implementation using MUX (Method 2)

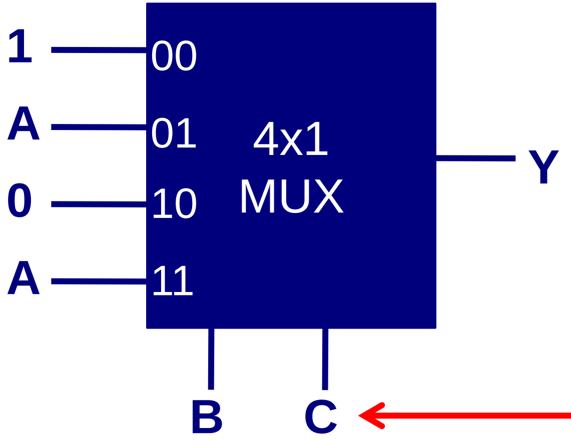
Design the following function by using MUX

$$F(A,B,C) = \Sigma m(0,4,5,7)$$

Solution: First choose the size of required MUX-

- No. of select lines (n) = No. of variables - 1
- No. of select lines (n) = 3 – 1 = 2
- No. of inputs (2^n) = 4
- It requires 4:1 MUX to implement the given function

Implementation using MUX (Method 2)



$F(A,B,C) = \Sigma m(0,4,5,7)$

Truth Table

Decimal	A	B	C	F
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

$S_1 = B$

$S_0 = C$

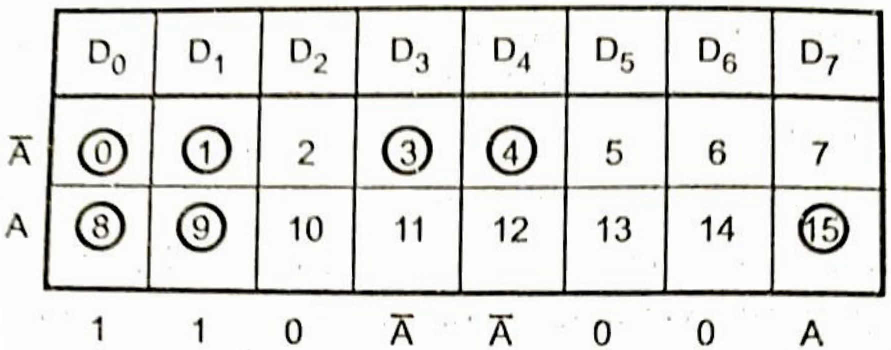
Column	I0	I1	I2	I3
A’	0	1	2	3
A	4	5	6	7
	1	**A**	**0**	**A**

Question

- Implement the following function using 8:1 Mux
  - $F(P,Q,R,S) = \sum m(0,1,3,4,8,9,15)$

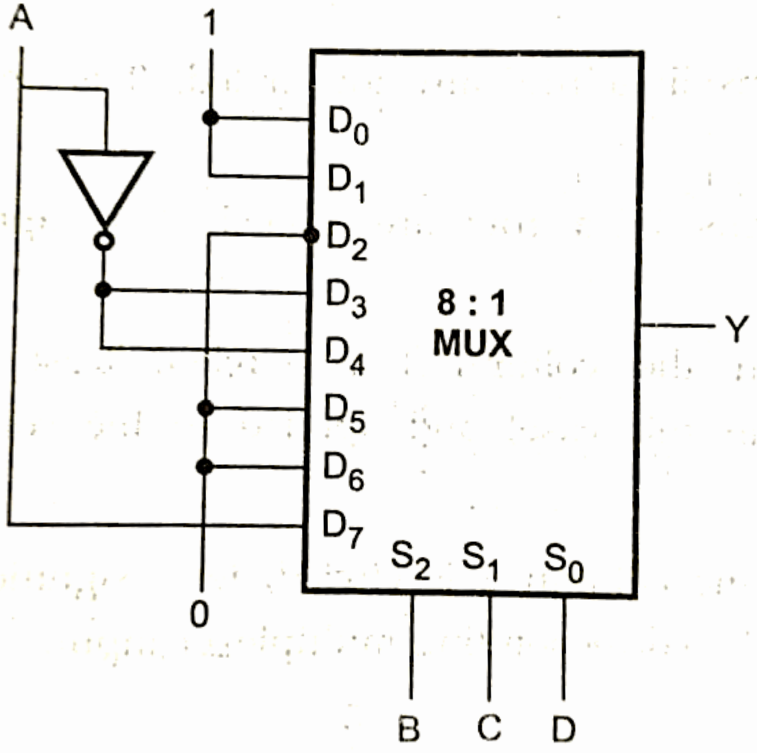
Question

- Implement the following function using 8:1 Mux
  - □ F(P,Q,R,S) = ∑m(0,1,3,4,8,9,15)



	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A

(a) Implementation table



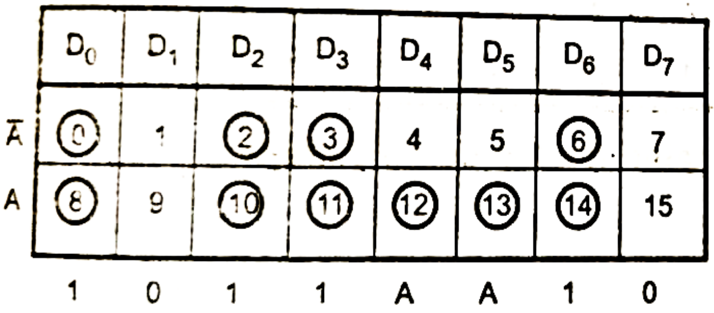
(b) Multiplexer implementation

Question

- Implement the following function using 8:1 Mux
  - F(A,B,C,D) = ∑m(0,2,6,10,11,12,13) +d(3,8,14)

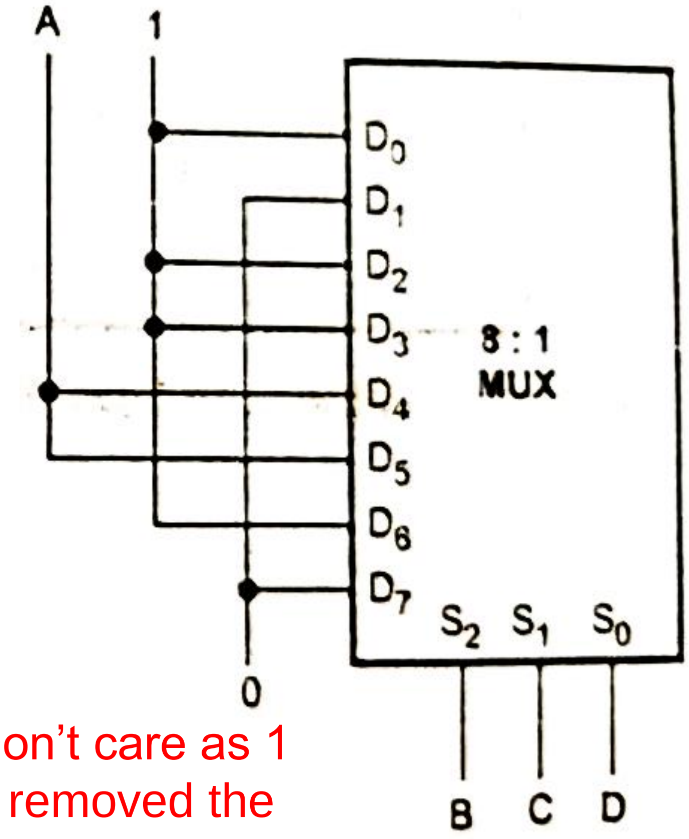
Question

- Implement the following function using 8:1 Mux
  - □ $F(A,B,C,D) = \sum m(0,2,6,10,11,12,13) + d(3,8,14)$



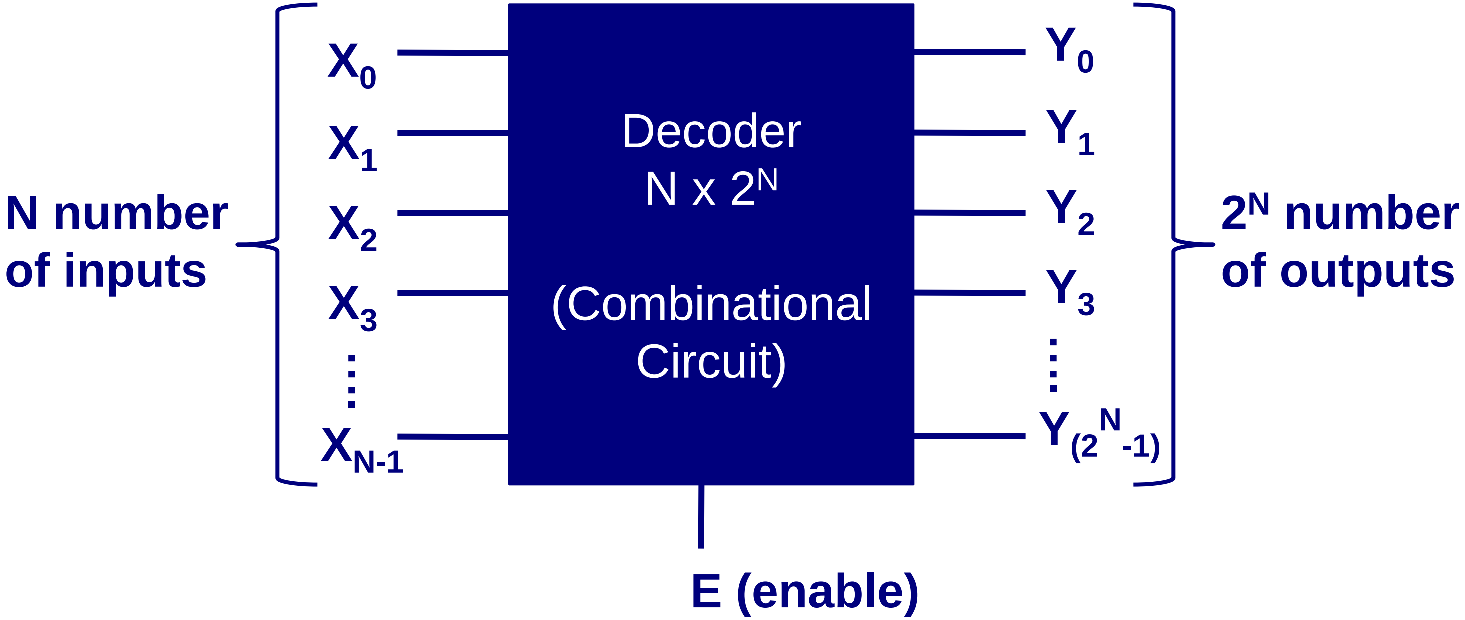
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	⓪	1	②	③	4	5	⑥	7
A	⑧	9	⑩	⑪	⑫	⑬	⑭	15
	1	0	1	1	A	A	1	0

Here, don't cares are treated as 1s

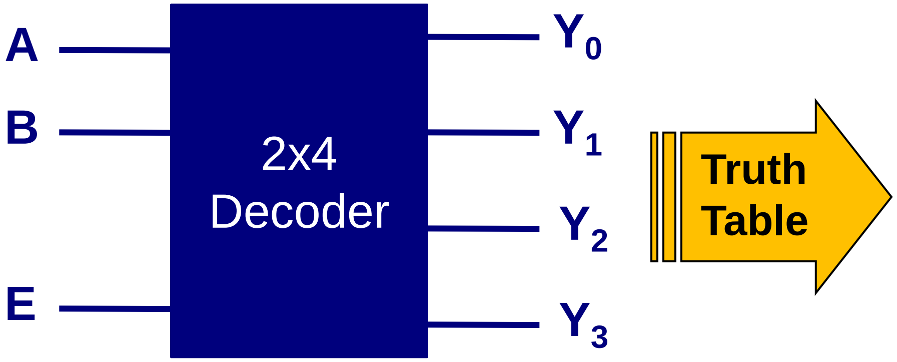


Taking Don't care as 1 we have removed the inverter for Abar

Decoder


N number of inputs
X_0
X_1
X_2
X_3
⋮
X_{N-1}
Decoder
N x 2^N
(Combinational Circuit)
Y_0
Y_1
Y_2
Y_3
⋮
$Y_{(2^N-1)}$
2^N number of outputs
E (enable)

2x4 Decoder

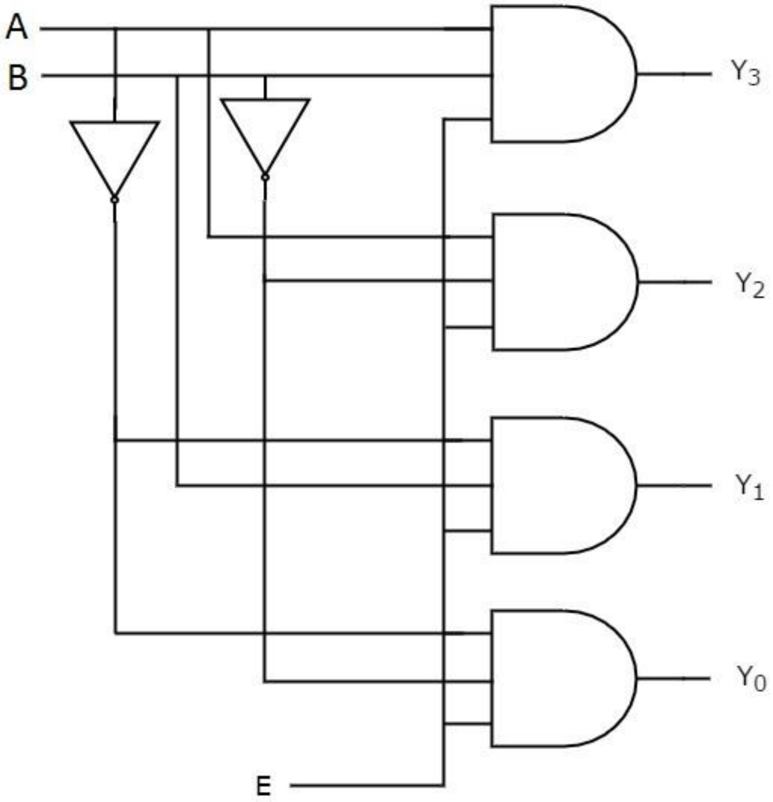


E	A	B	Y_0	Y_1	Y_2	Y_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

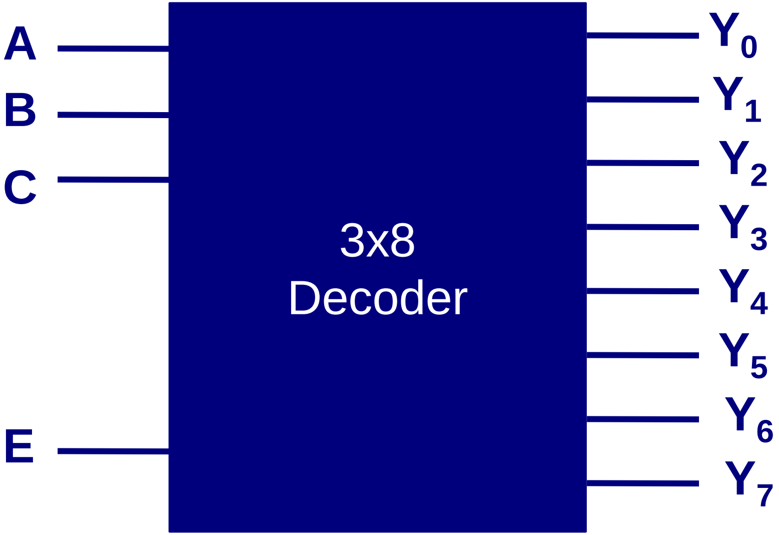
$Y_0 = EA'B'$ $Y_2 = EAB'$

$Y_1 = EA'B$ $Y_3 = EAB$

2x4 Decoder



3x8 Decoder

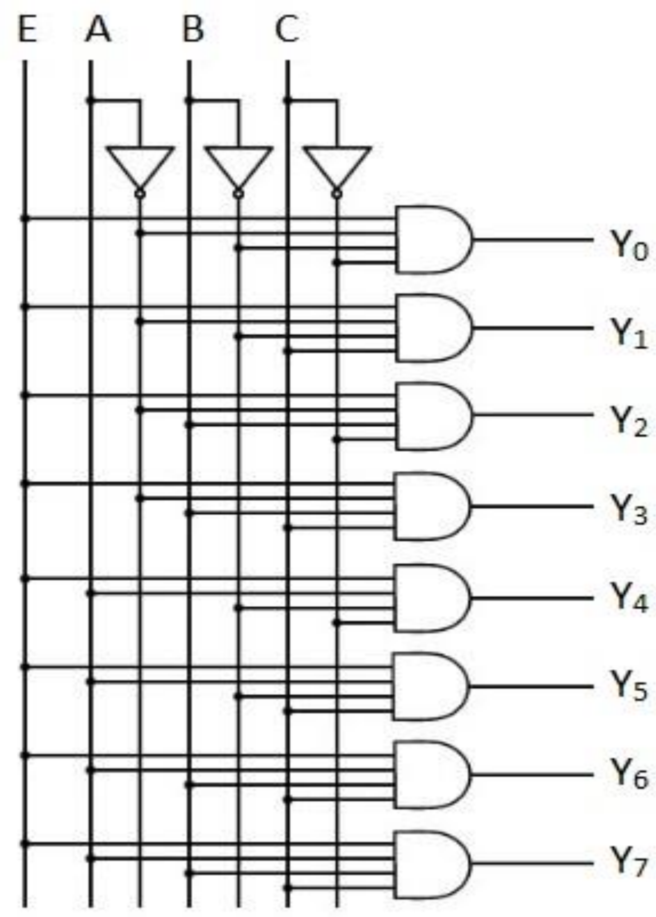


3x8 Decoder

Truth Table

E	A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

3x8 Decoder



Y_0 = EA’B’C’

Y_1 = EA’B’C

Y_2 = EA’BC’

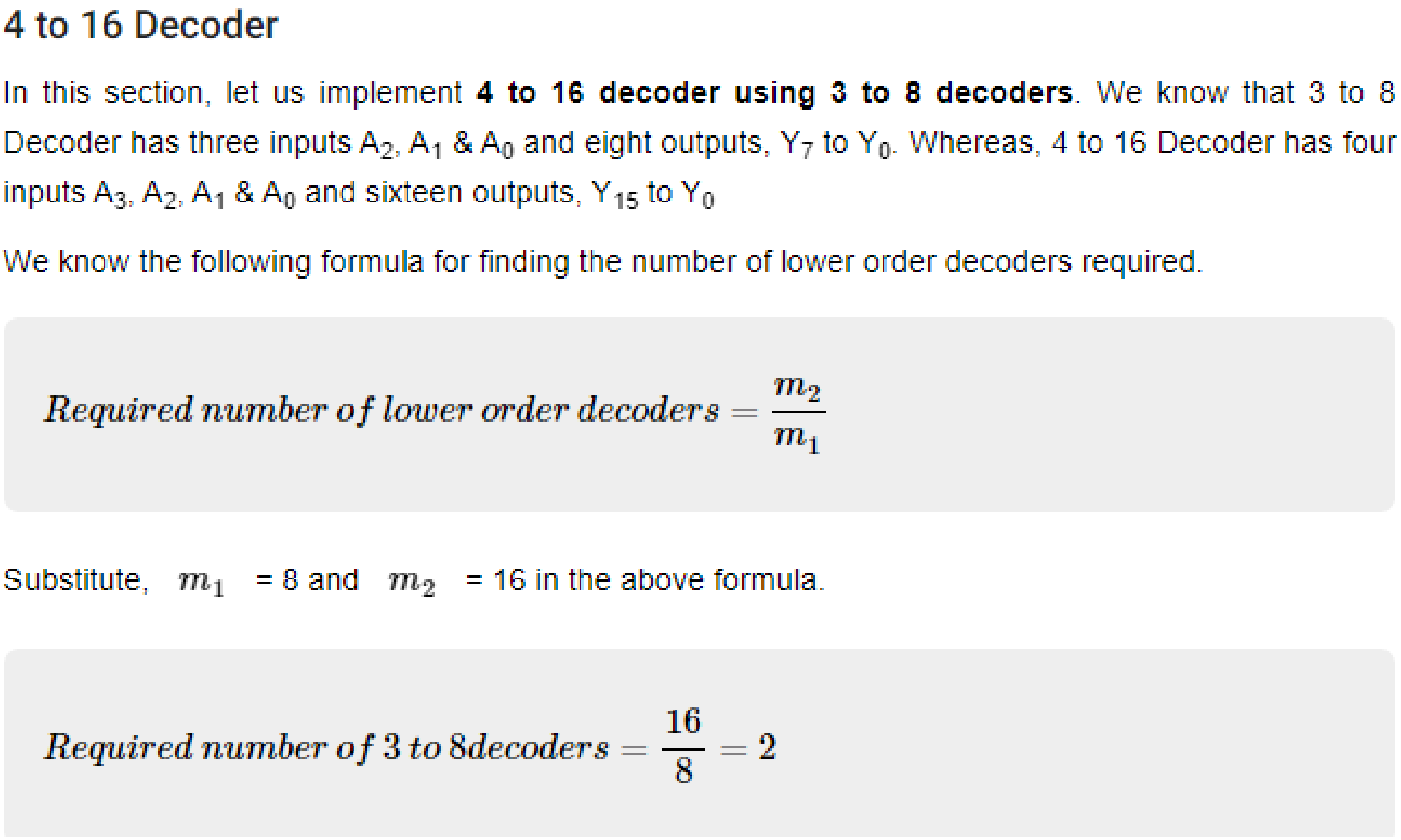
Y_3 = EA’BC

Y_4 = EAB’C’

Y_5 = EAB’C

Y_6 = EABC’

Y_7 = EABC



4 to 16 Decoder

In this section, let us implement **4 to 16 decoder using 3 to 8 decoders**. We know that 3 to 8 Decoder has three inputs A_2, A_1 & A_0 and eight outputs, Y_7 to Y_0. Whereas, 4 to 16 Decoder has four inputs A_3, A_2, A_1 & A_0 and sixteen outputs, Y_{15} to Y_0

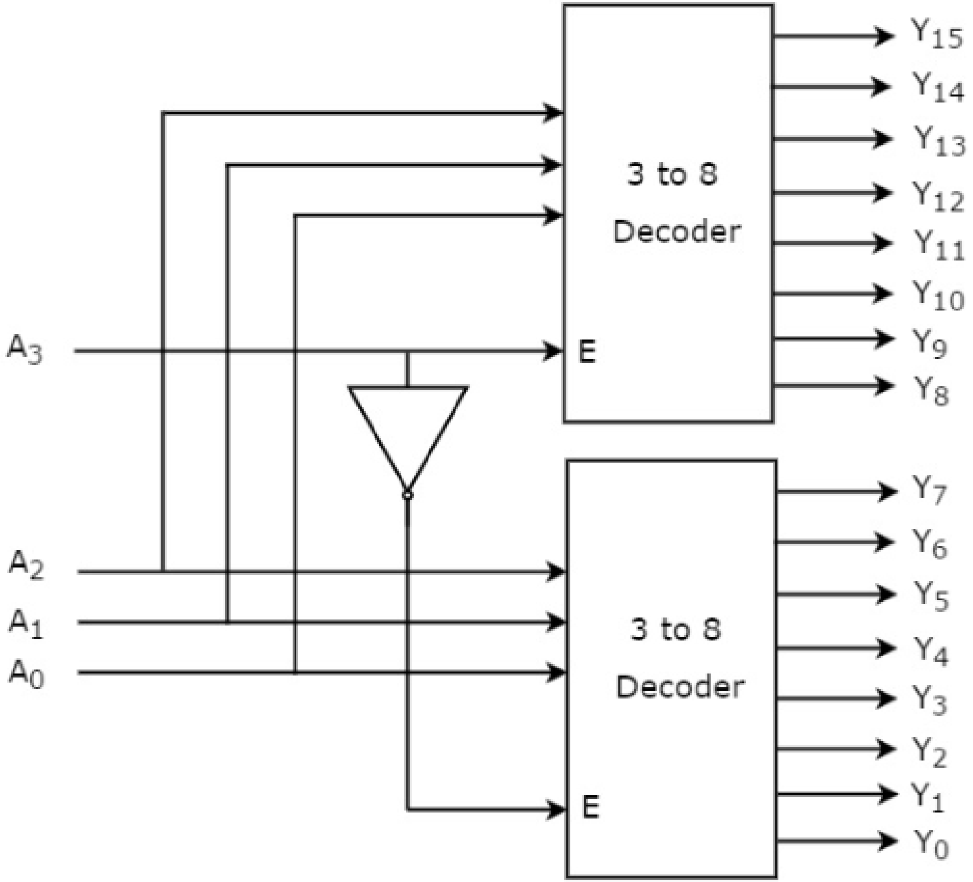
We know the following formula for finding the number of lower order decoders required.

$$Required\ number\ of\ lower\ order\ decoders = \frac{m_2}{m_1}$$

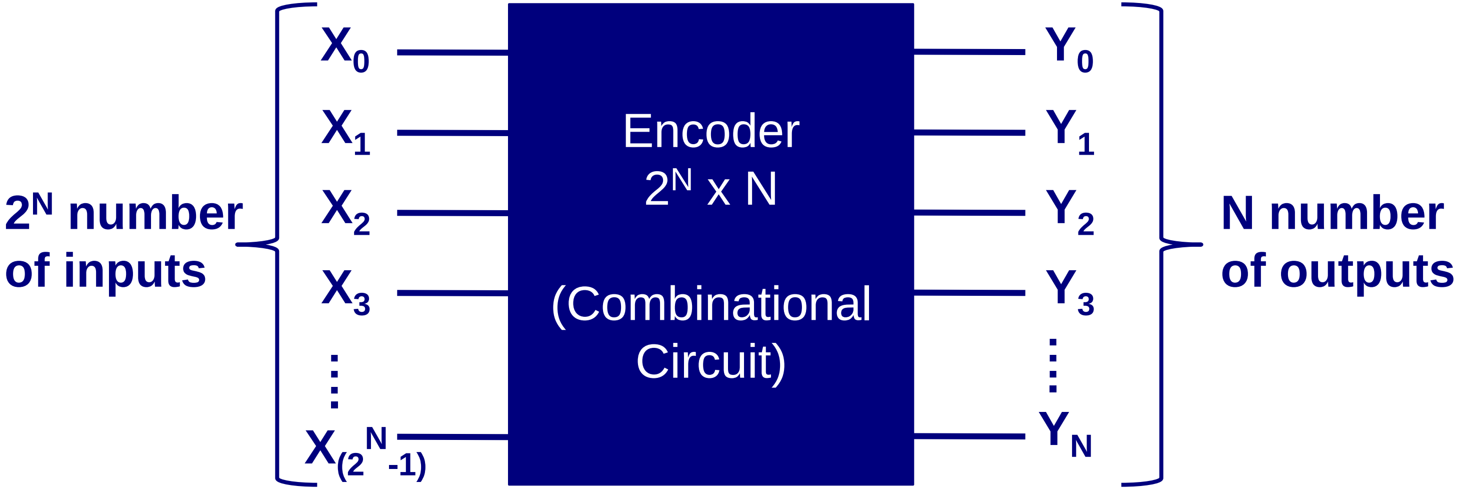
Substitute, m_1 = 8 and m_2 = 16 in the above formula.

$$Required\ number\ of\ 3\ to\ 8 decoders = \frac{16}{8} = 2$$

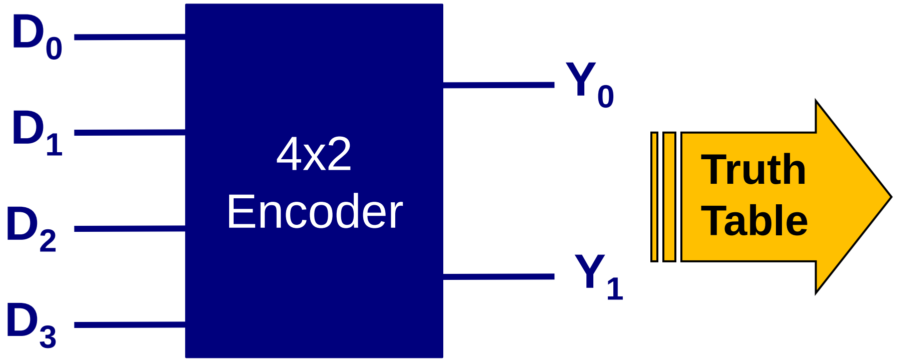
4:16 Decoder using 3:8 Decoder



Encoder


2^N number of inputs
X_0
X_1
X_2
X_3
⋮
$X_{(2^N-1)}$
Encoder
2^N x N
(Combinational Circuit)
Y_0
Y_1
Y_2
Y_3
⋮
Y_N
N number of outputs

4x2 Encoder



D_0	D_1	D_2	D_3	Y_1	Y_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1
X	X	X	X	X	X

$$Y_0 = D_1+D_3$$

$$Y_1 = D_2+D_3$$

4x2 Encoder

K-map for Y_0 of 4x2 Encoder

D_2D_3 D_0D_1				
	0	1	3	2
	4	5	7	6
	12	13	15	14
	8	9	10	11

$$Y_0 = D_1 + D_3$$

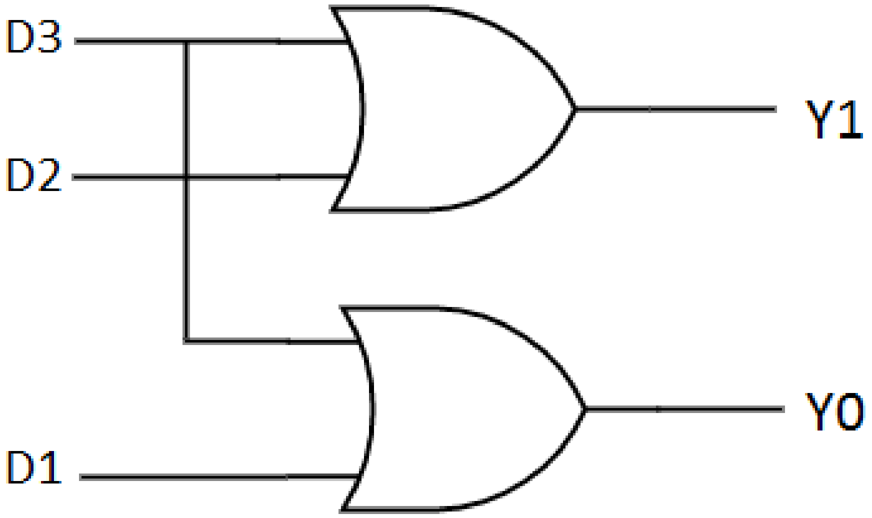
4x2 Encoder

K-map for Y_1 of 4x2 Encoder

D_2D_3 D_0D_1				
	0	1	3	2
	4	5	7	6
	12	13	15	14
	8	9	10	11

$$Y_1 = D_2+D_3$$

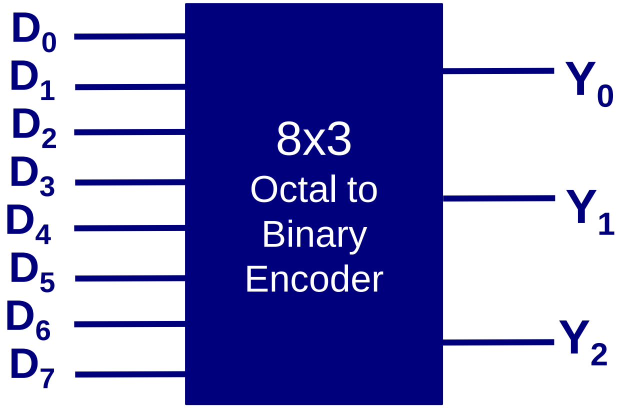
4x2 Encoder



$Y_1 = D_2+D_3$

$Y_0 = D_1+D_3$

8x3 Encoder (Octal to Binary)



$$Y_0 = D_1+D_3+D_5+D_7$$

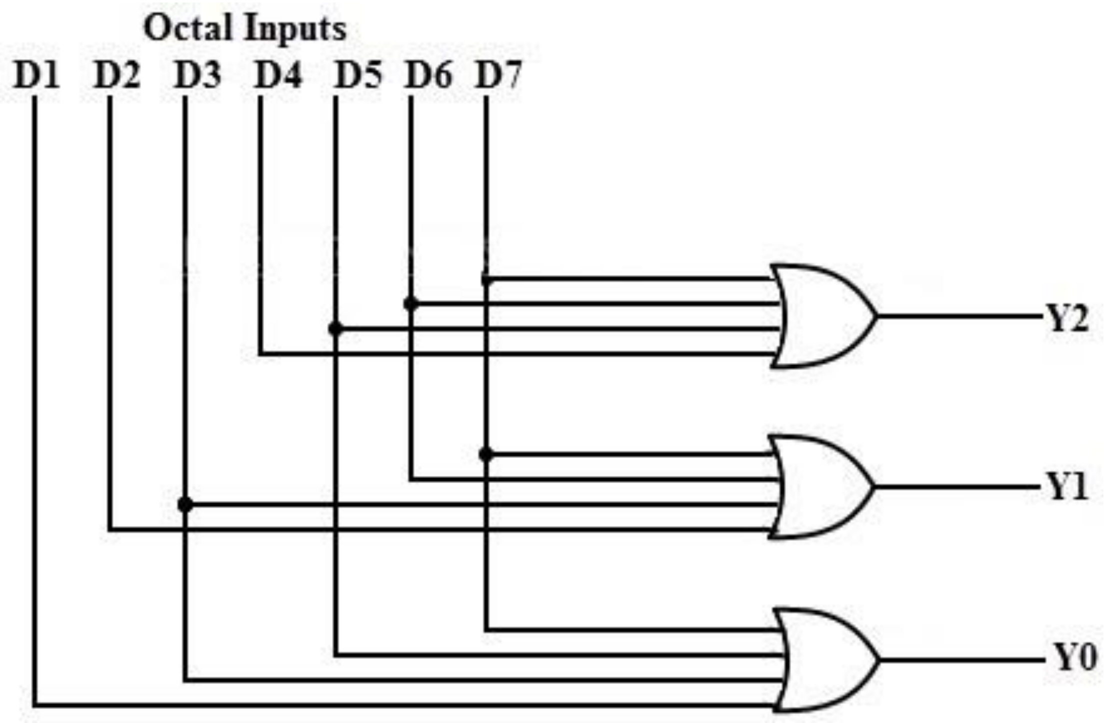
$$Y_1 = D_2+D_3+D_6+D_7$$

$$Y_2 = D_4+D_5+D_6+D_7$$

Truth Table

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1
X	X	X	X	X	X	X	X	X	X	X

8x3 Encoder (Octal to Binary)



$Y_2 = D_4+D_5+D_6+D_7$

$Y_1 = D_2+D_3+D_6+D_7$

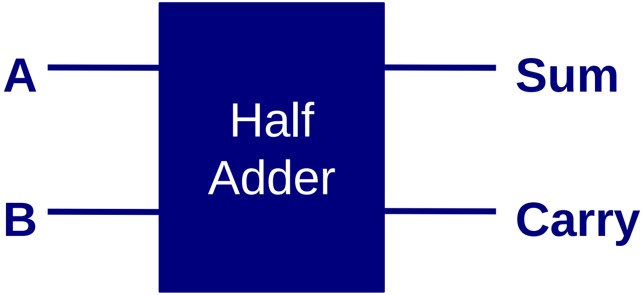
$Y_0 = D_1+D_3+D_5+D_7$

Decoder (as universal logic gate)

- Decoder is also act as universal logic gate.
- It means that any of the logic circuit/gate can be implemented with the help of Decoder.

Implementation of Half adder using decoder

- First, find out the no. of inputs required.



- So, half adder has 2 inputs.
- It means 2x4 decoder is required to design a half adder.

Implementation of Half adder using decoder

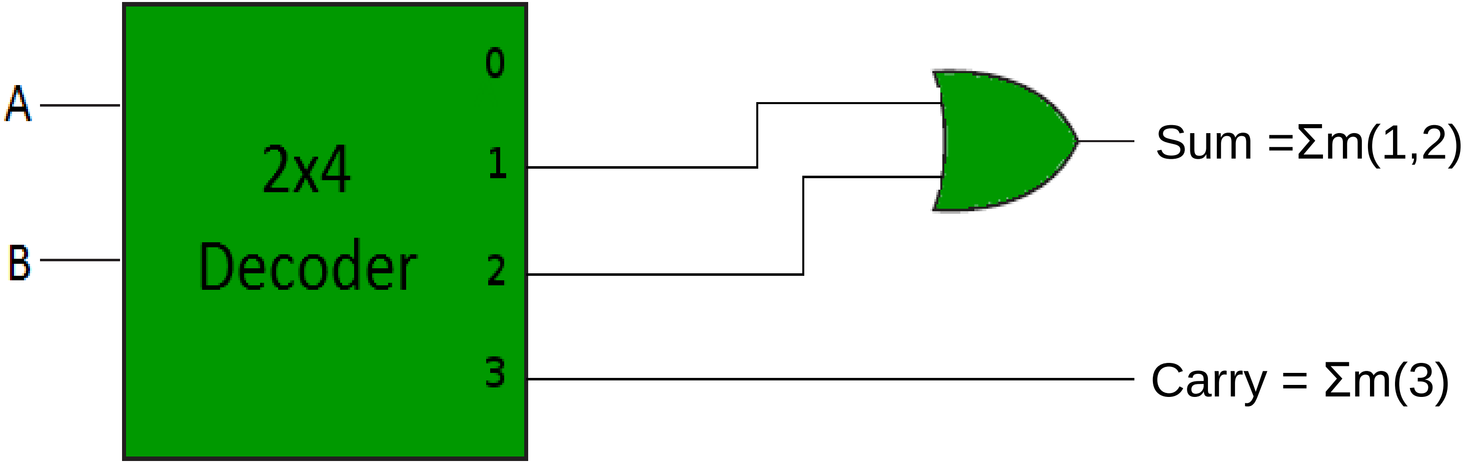
Truth Table of Half adder

Decimal	A	B	Sum	Carry
0	0	0	0	0
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1

Sum = Σm(1,2)

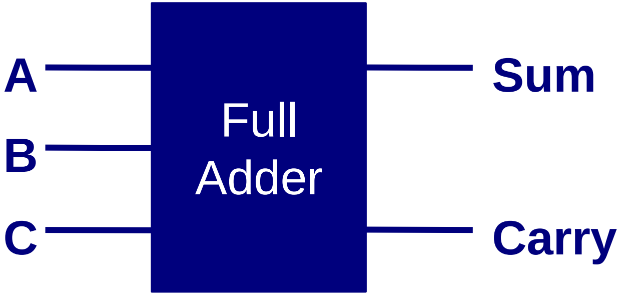
Carry = Σm(3)

Implementation of Half adder using decoder



Decoder (as universal logic gate)

- Implement a full adder with the help of decoder.
  - First, find out the no. of inputs required.



  - So, full adder has 3 inputs.
  - It means 3x8 decoder is required to design a full adder.

Implementation of full adder using decoder

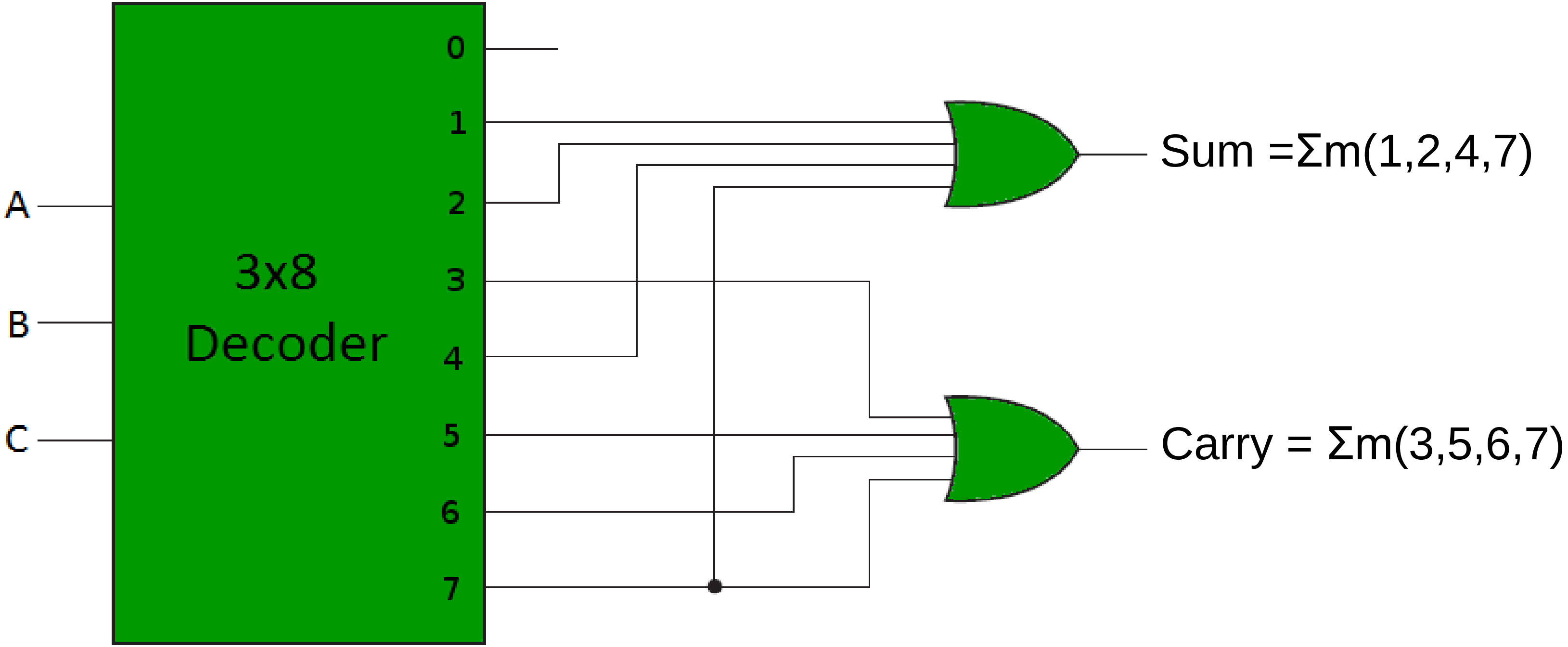
Truth Table of Full adder

Decimal	A	B	C	Sum	Carry
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

$$\textbf{Sum} = \Sigma m(\underline{1,2,4,7})$$

$$\textbf{Carry} = \Sigma m(\underline{3,5,6,7})$$

Implementation of full adder using decoder



Thank You