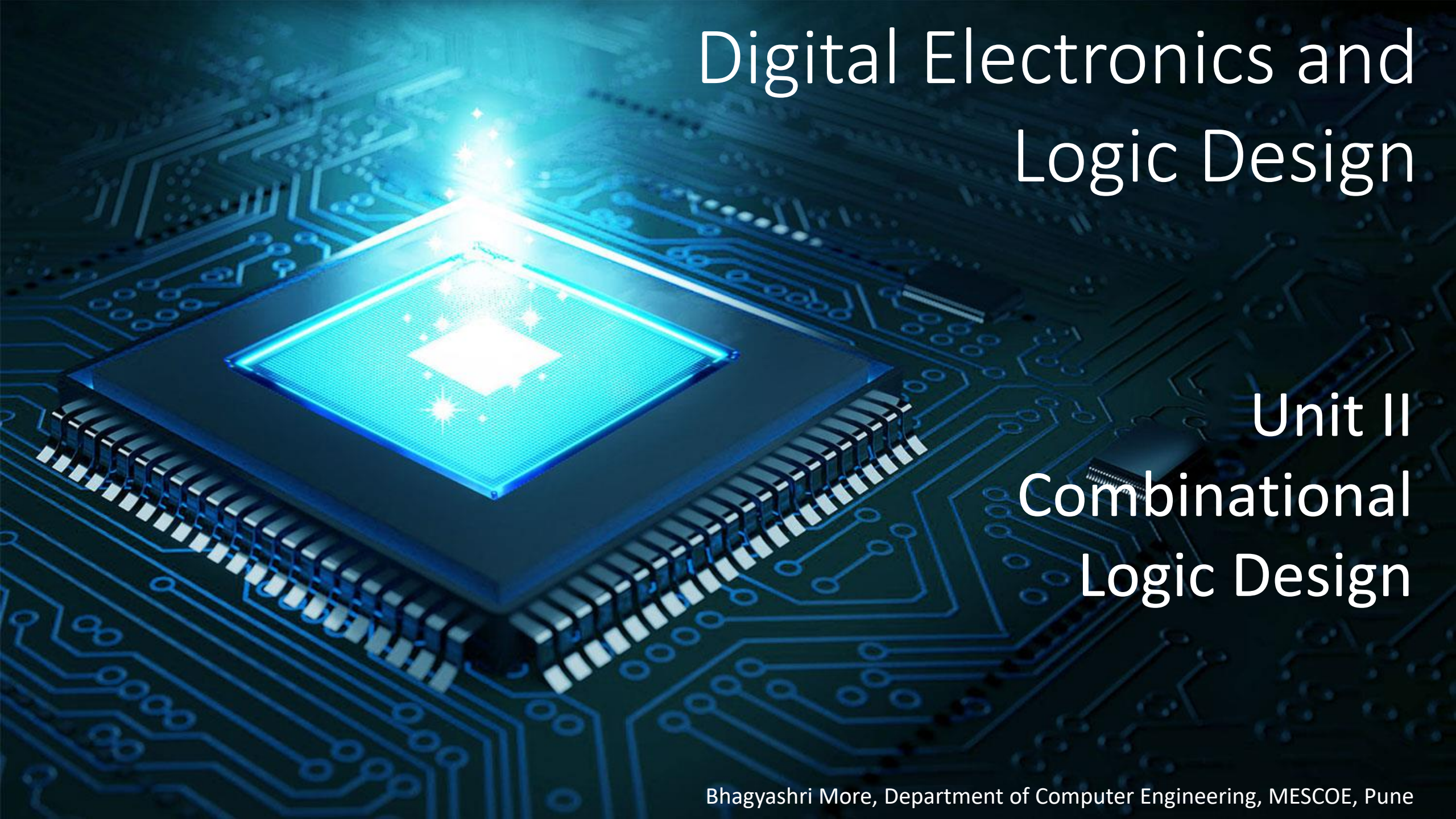


Digital Electronics and Logic Design

Unit II
Combinational Logic Design

Bhagyashri More, Department of Computer Engineering, MESCOE, Pune

Agenda

01 Code Converter: BCD, Excess-3, Gray, Binary Code

02 Half Adder, Full Adder, Half Subtractor, Full Subtractor

03 Binary Adder (IC 7483), BCD Adder, Look ahead carry generator

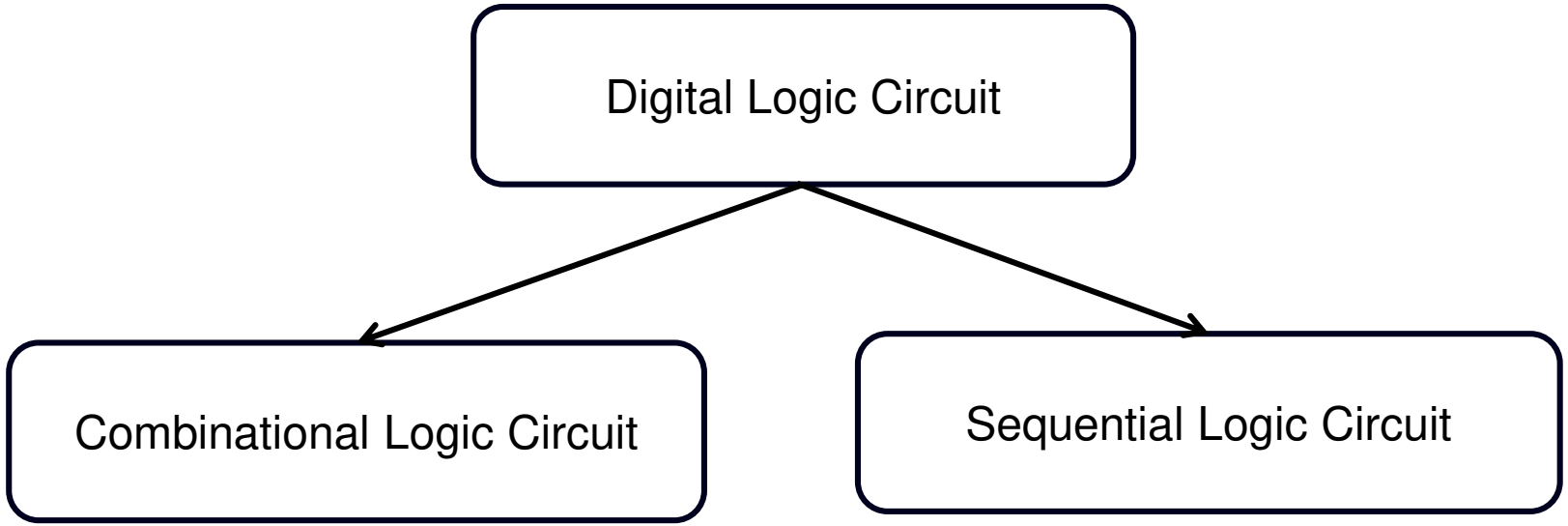
04 Multiplexer (MUX): MUX IC (74153, 74151), Cascading multiplexer

05 Demultiplexer (DEMUX)- Decoder (IC 74138, 74154), Implementation of SOP and POS using MUX , DEMUX

06 Comparators (2 bit), Parity Generators and Checkers.

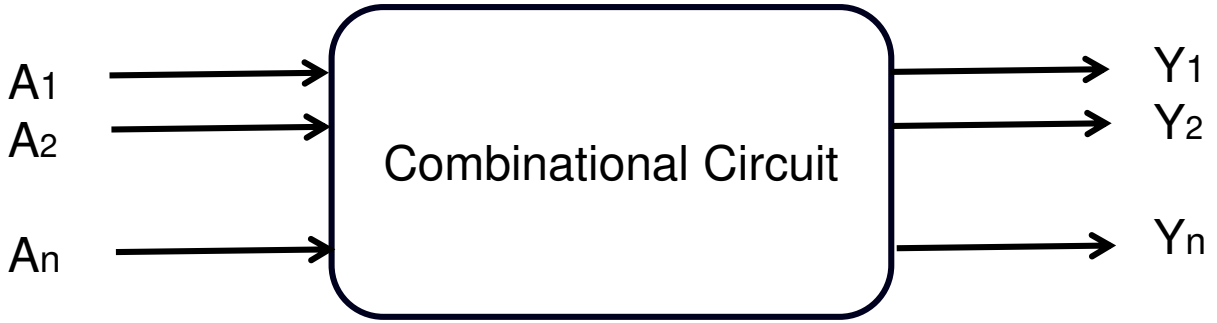
Digital Logic Circuit

Digital Logic Circuit: A digital logic circuit is a circuit formed when one or more gates are combined together to form a complex switching circuit.



Combinational Circuit

Combinational Circuit: Combinational Logic Circuit are defined as the circuits whose output depends on its present input at that instant of time.



- **Examples**: Code converter, Half Adder, Full Adder, Half Subtractor, Full Subtractor, Binary Adder, BCD Adder, Multiplexer, De-multiplexer.
- This circuits do not need any memory as the outputs are dependent on the present inputs at any instant of time.

Combinational Circuit

Procedure for Designing Combinational Circuit:

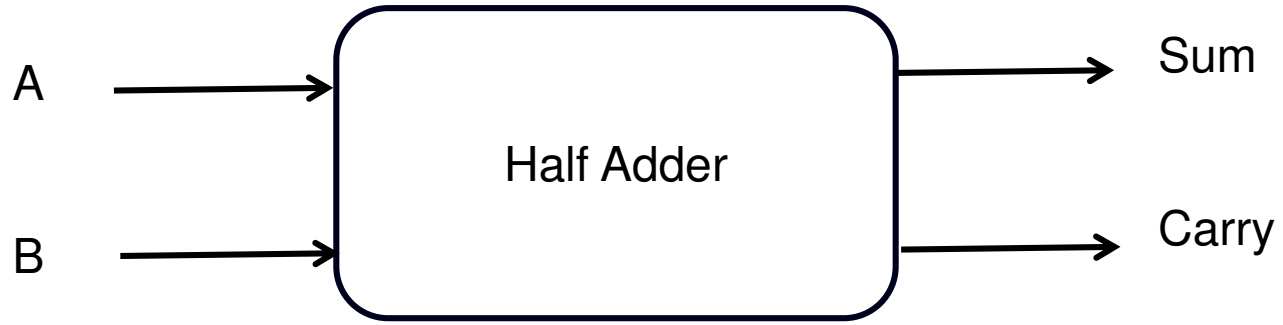
1. Define and understand problem
2. Identify and determine the number of input variables available and output variables required; assign labels to inputs and outputs.
3. Derive truth table for describing the relationship between the input and output variables.
4. Obtain the Boolean expression for every output variable in terms of the input variables. Simplify the Boolean expression for output variables.
5. Draw the logic diagram

Half Adder

An adder is a digital logic circuit in electronics that is extensively used for the addition of numbers.

- Types of Binary Adder:
    1. Half Adder
    2. Full Adder

- Half Adder: The half adder circuit has two inputs: A and B, which add two input digits and generates a carry and a sum.

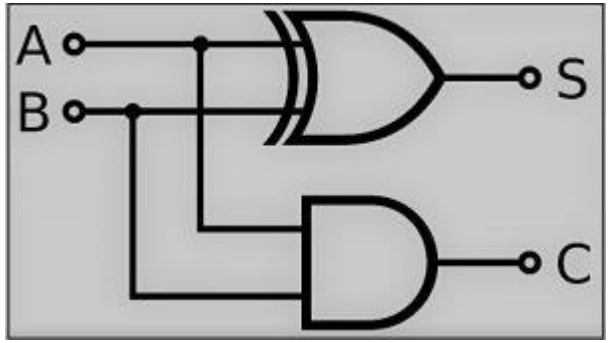


Half Adder

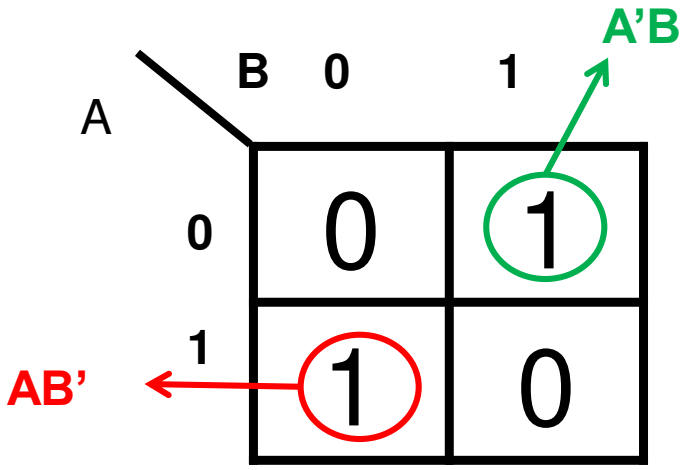
Truth Table:

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

Logic Diagram:

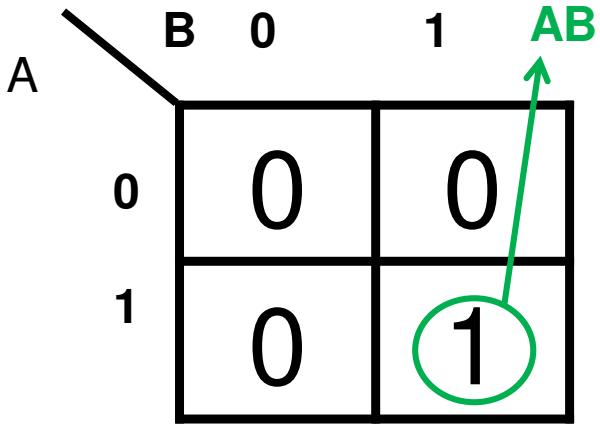


K-map: For Sum:



Sum = A'B + AB'
= A ⊕ B

K-map: For Carry:



Carry = AB

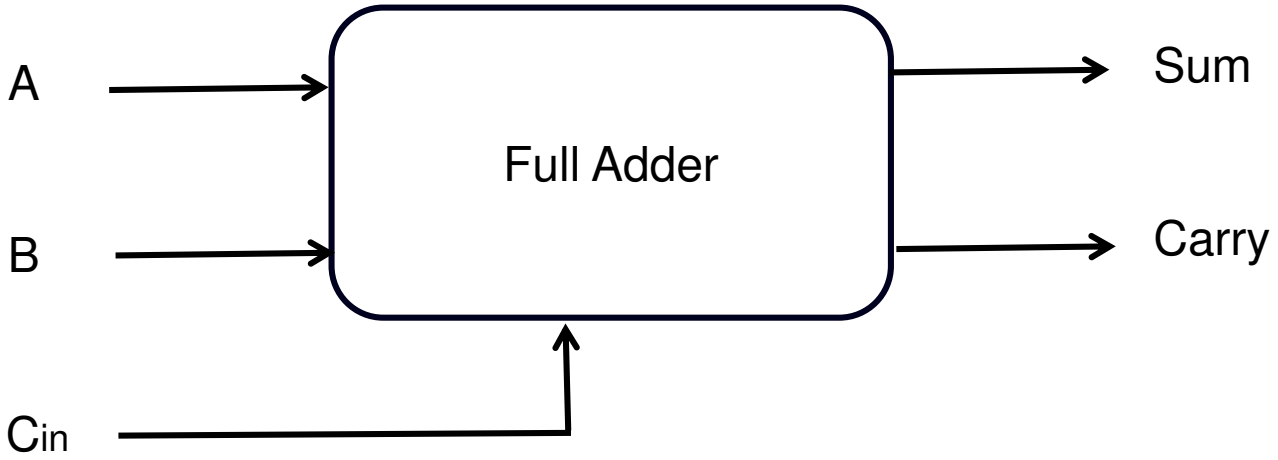
Half Adder

Half Adder Limitations:

The main reason to call these binary adders like Half Adders is, that there is no range to include the carry bit using an earlier bit. So, this is a main limitation of HAs once used like binary adder particularly in real-time situations which involve adding several bits. So this limitation can be overcome by using the full adders.

Full Adder

- **Full Adder:** The difference between a half-adder and a full-adder is that the full-adder has three inputs and two outputs, whereas half adder has only two inputs and two outputs.
- The first two inputs are A and B and the third input is an input carry as Cin.

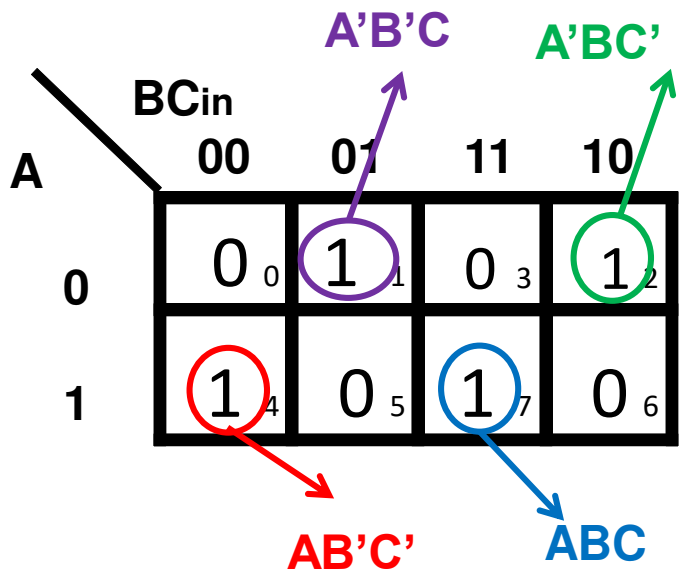


Full Adder

Truth Table:

A	B	Cin	S(Sum)	Cout(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

K-map:
For Sum:



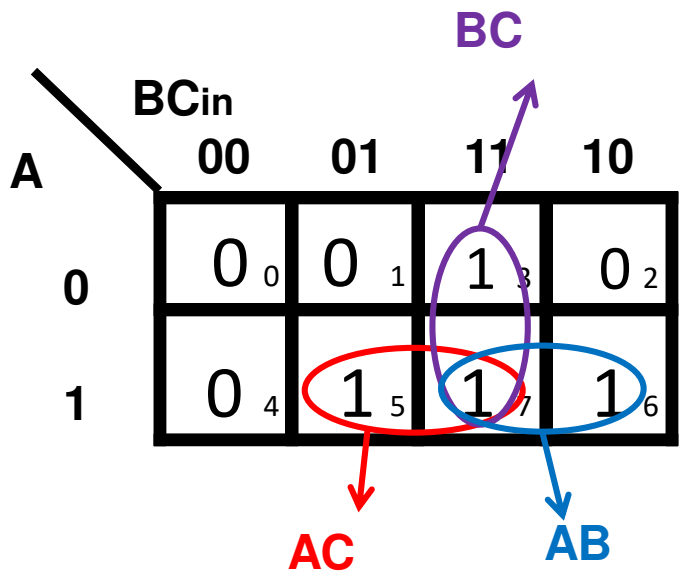
Sum = AB'Cin' + ABCin + A'B'Cin + A'BCin'
= A (B' Cin' + BCin) + A' (B'Cin + BCin')
= **A** (B EXNOR Cin) + A' (B ⊕ Cin)
= A (B ⊕ Cin)' + A' (B ⊕ Cin)
= AX' + A'X .....(B⊕Cin)= x
= A ⊕ X
= A ⊕ (B ⊕ Cin)

Full Adder

Truth Table:

A	B	Cin	S(Sum)	Cout(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**

K-map:
For Sum:



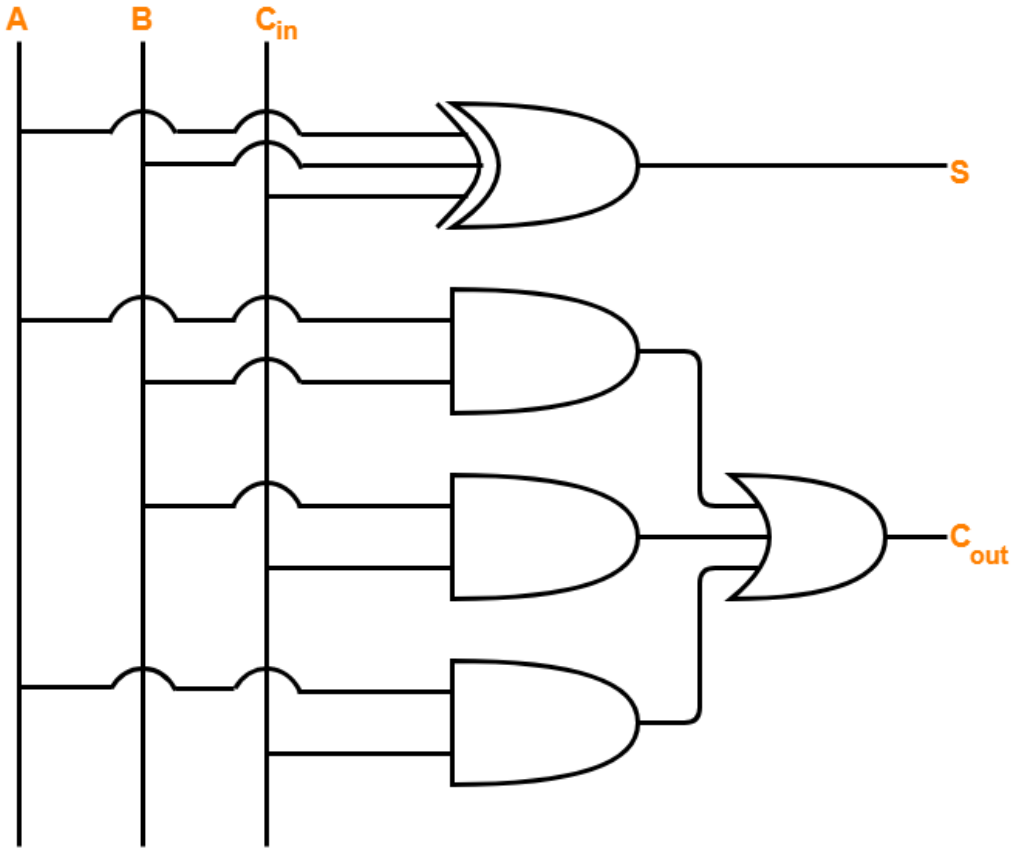
$C_{out} = AB + BC_{in} + AC_{in}$

Full Adder

$Sum = A \oplus B \oplus C_{in}$

$C_{out} = AB + BC_{in} + AC_{in}$

Logic Diagram:



Full Adder Logic Diagram

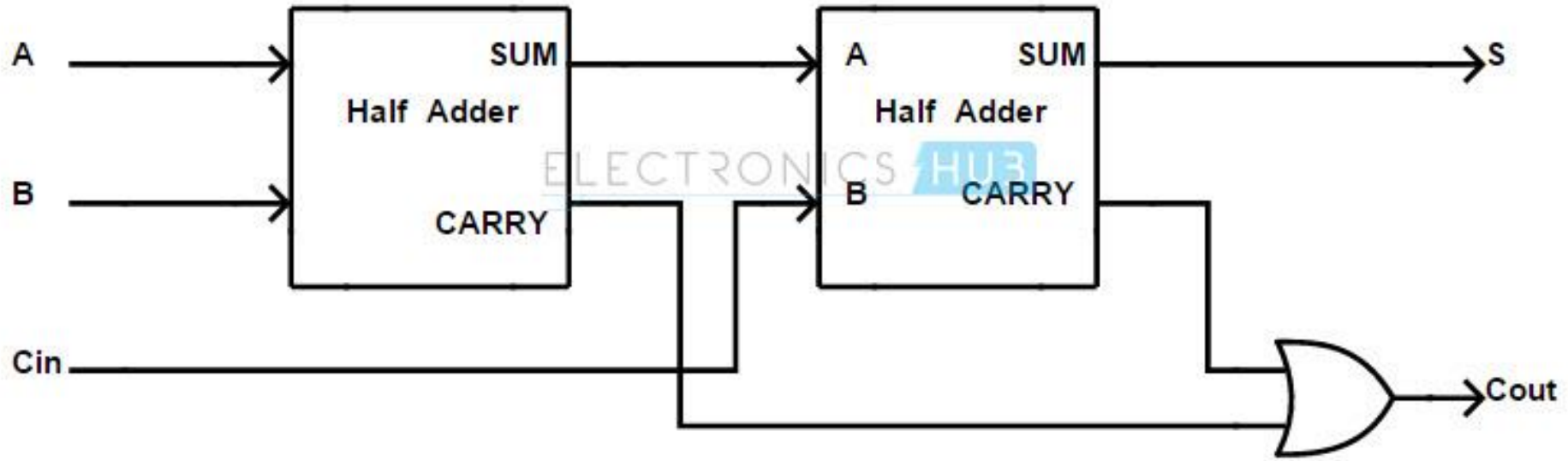
Full Adder

Advantages:
The addition of multiple bits can be obtained by increasing the number of the Full Adder in a circuit.

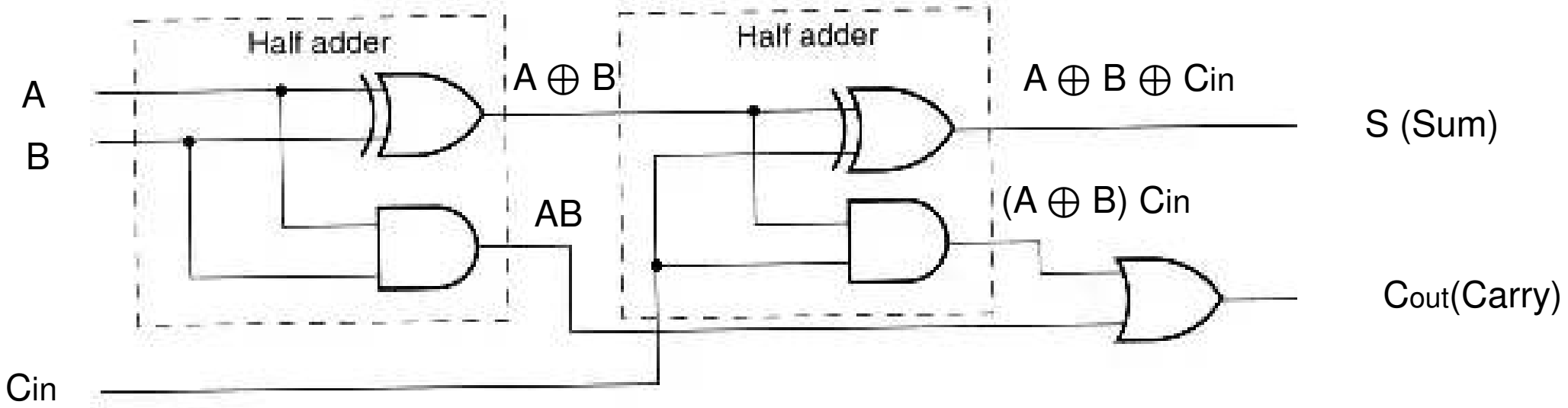
Applications:
1. Arithmetic Logic Unit use to compute binary addition.
2. Ripple ahead carry adder
3. Digital calculator
4. Microcontroller to calculate PC (program counter)
5. In Digital Signal Processor and Networking system.

Full Adder

Full Adder using Half Adder:



Full Adder



From above diagram,
$S = A \oplus B \oplus Cin$
$Cout = (A \oplus B)\ Cin + AB$

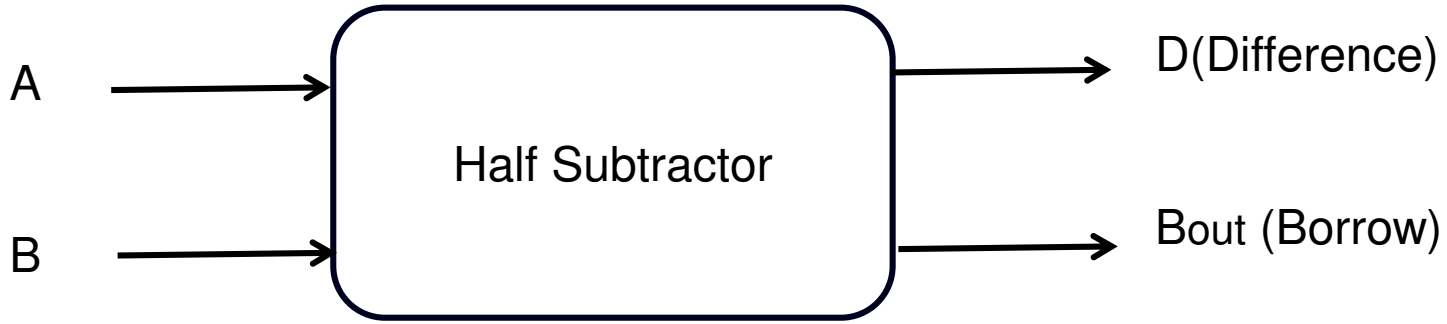
Full Adder Expression
$S = A \oplus B \oplus Cin$
$Cout = AB + BCin + ACin$

$Cout = (A \oplus B)\ Cin + AB$
$= (A'B + AB')\ Cin + AB$
$= A'B\ Cin + AB'\ Cin + AB\ (Cin + 1)$ ……A+1=1
$= \mathbf{A'B\ Cin} + AB'\ Cin + \mathbf{AB\ Cin} + AB$
$= B\ Cin\ (A' + A) + AB'\ Cin + AB$
$= B\ Cin + AB'\ Cin + AB(Cin + 1)$
$= B\ Cin + \mathbf{AB'\ Cin} + \mathbf{AB\ Cin} + AB$
$= B\ Cin + A\ Cin\ (B' + B) + AB$
$= B\ Cin + A\ Cin + AB$

Subtractor

Subtractor: It is combinational circuit used to perform the subtraction of the two binary bits**.**
Types of Subtractor:
1. Half Subtractor 2. Full Subtractor

Half Subtractor: Half substractor is a combinational circuit which performs substraction of single bit binary numbers. It has two input A and B with two output as differences (D) and borrow (Bout).

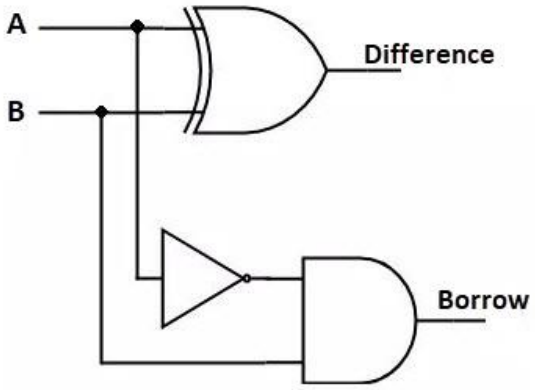


Half Subtractor

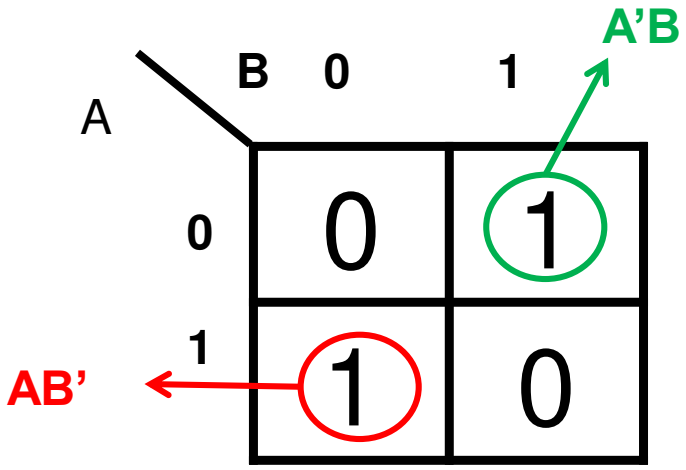
Truth Table:

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

Logic Diagram:

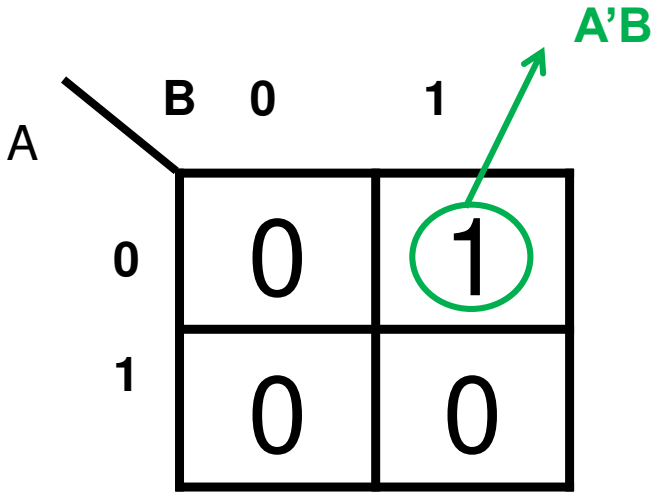


K-map: For D:



Difference = A'B + AB'
= A ⊕ B

K-map: For Bout:



Bout = A'B

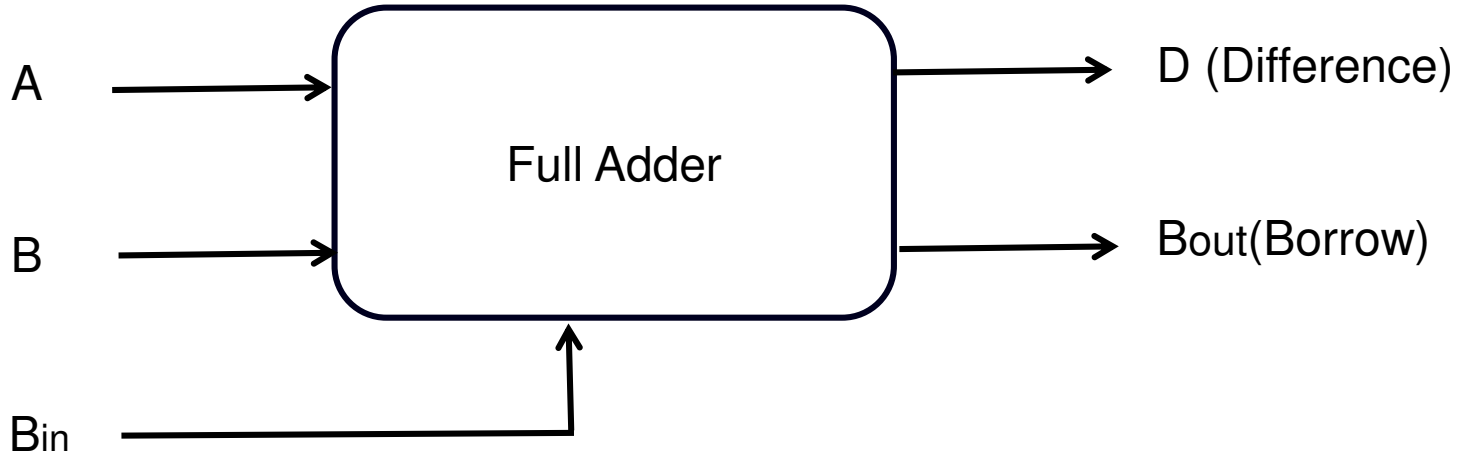
Half Subtractor

Half Subtractor Limitations:

One major disadvantage of the Half Subtractor circuit when used as a binary subtractor, is that here is no provision for a “Borrow-in” from the previous circuit when subtracting multiple data bits from each other. Then we need to produce what is called a “full binary subtractor” circuit to take into account this borrow-in input from a previous circuit.

Full Subtractor

- **Full Subtractor:** The main difference between the **Full Subtractor** and the previous **Half Subtractor** circuit is that a full subtractor has three inputs and two outputs.
- The first two inputs are A and B and the third input is an input borrow as Bin.

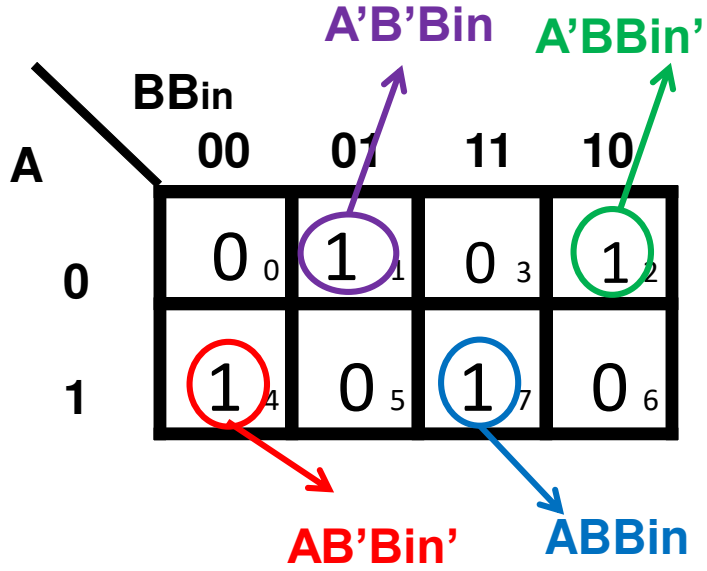


Full Subtractor

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	0
1	1	1	1	1

K-map: For Difference:



Difference = AB'Bin' + ABBin + A'B'Bin + A'BBin'

= A (B' Bin' + BBin) + A' (B' Bin + BBin')

= A (B' Bin' + BBin) +A'(B ⊕ Bin)

= A (B ⊙ Bin) +A'(B ⊕ Bin)

= A (B ⊕ Bin)' +A'(B ⊕ Bin)

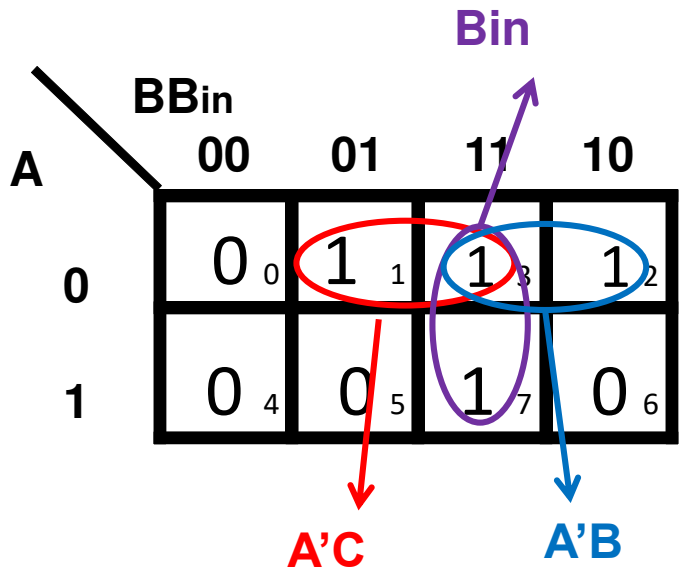
= A ⊕ B ⊕ Bin

Full Subtractor

Truth Table:

A	B	Bin	B	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	**0**
1	1	1	1	**1**

K-map: For Borrow:

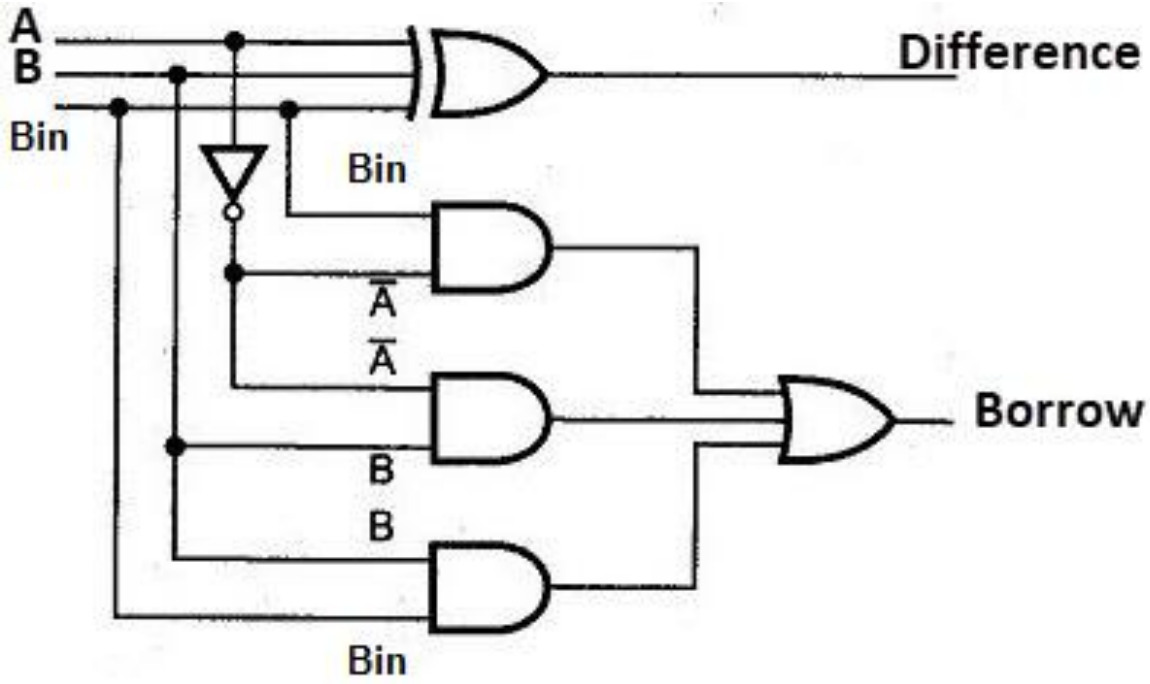


$B_{out} = A'B + BBin + A'Bin$

Full Subtractor

Difference = A ⊕ B ⊕ Bin
Borrow = A'B + BBin + A'Bin

Logic Diagram:



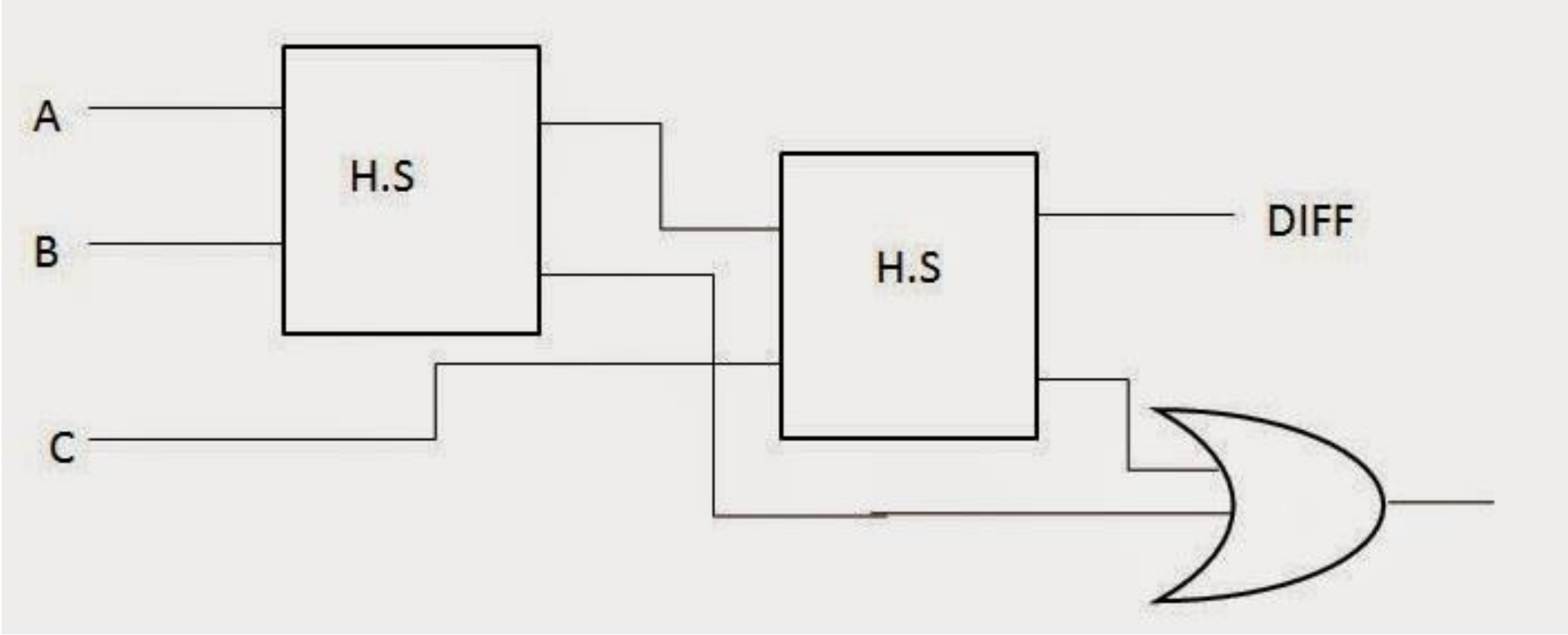
Full Subtractor

Advantages:
The subtraction of multiple bits can be obtained by increasing the number of the Full Subtractor in a circuit.

Applications:
1. Arithmetic Logic Unit use to compute binary subtraction.
2. Digital calculator
3. In microcontroller
4. In Digital Signal Processor and Networking system.
5. Subtractors are used in processors to compute tables, address, etc.

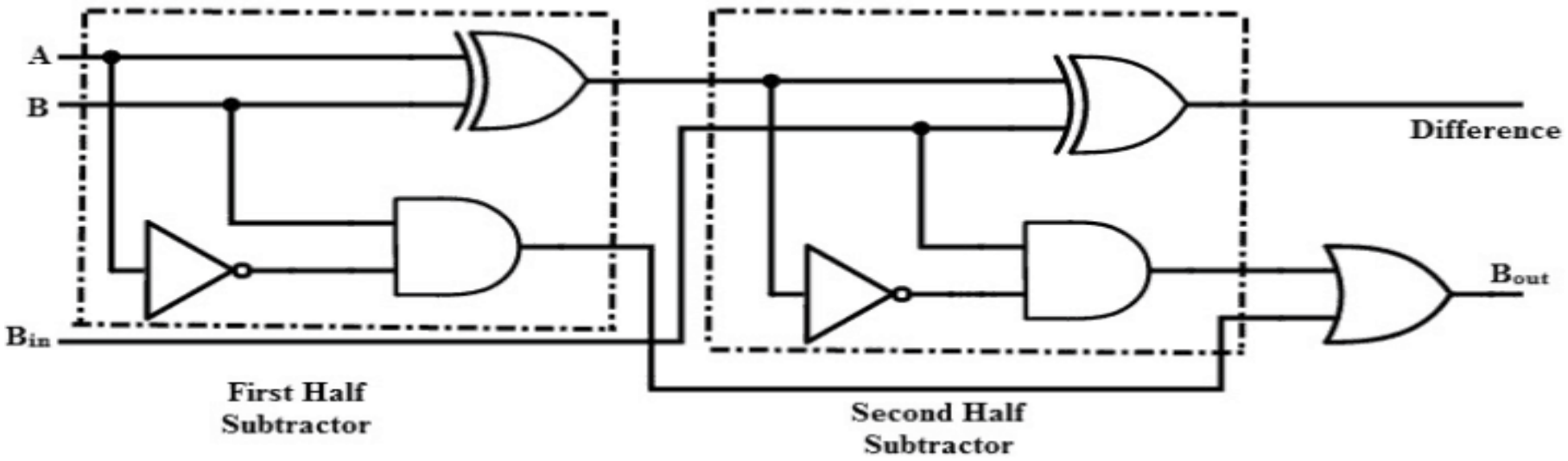
Full Subtractor

Full Subtractor using Half Subtractor:



Full Subtractor

Full Adder using Half Adder:



From above diagram,

$D = A \oplus B \oplus Bin$

$Bout = (A \oplus B)' Bin + A'B$

Full Sub Expression

$D = A \oplus B \oplus Bin$

$Bout = A'B + BBin + A'Bin$

$Bout = (A \oplus B)' Bin + A'B$

$= (AB + A'B') Bin + A'B$

$= ABBin + A'B'Bin + A'B(Bin + 1)$

$= \mathbf{ABBin} + A'B'Bin + \mathbf{A'BBin} + A'B$

$= BBin (A+A') + A'B'Bin + A'B$

$= BBin + A'B'Bin + A'B (Bin+1)$

$= BBin + \mathbf{A'B'Bin + A'BBin} + A'B$

$= Bbin + A'Bin (B'+B) + A'B$

$= BBin + A'Bin + A'B$

Agenda

01 Code Converter: BCD, Excess-3, Gray, Binary Code

02 Half Adder, Full Adder, Half Subtractor, Full Subtractor

03 Binary Adder (IC 7483), BCD Adder, Look ahead carry generator

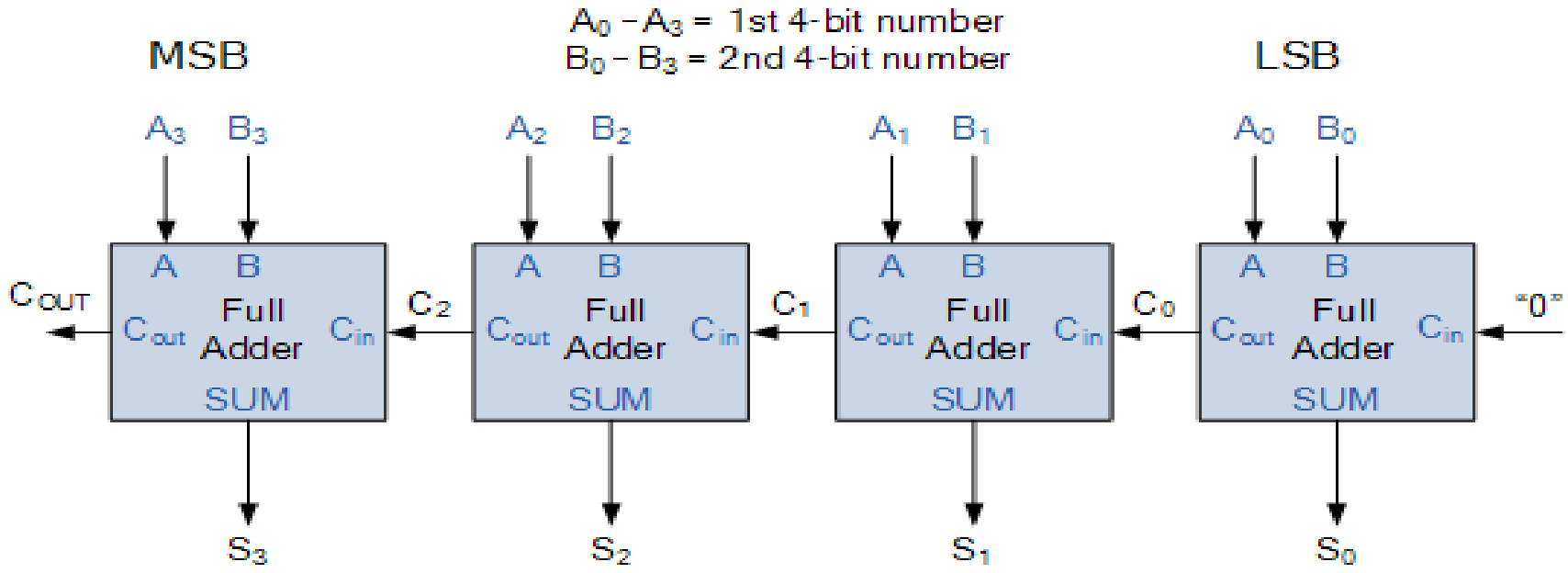
04 Multiplexer (MUX): MUX IC (74153, 74151), Cascading multiplexer

05 Demultiplexer (DEMUX)- Decoder (IC 74138, 74154), Implementation of SOP and POS using MUX , DEMUX

06 Comparators (2 bit), Parity Generators and Checkers.

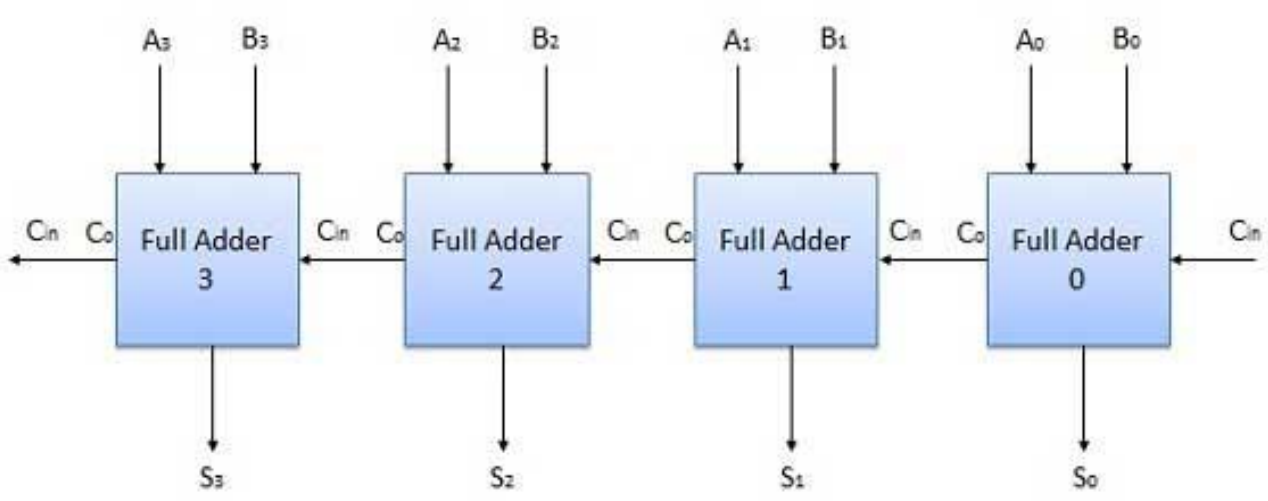
Parallel Adder

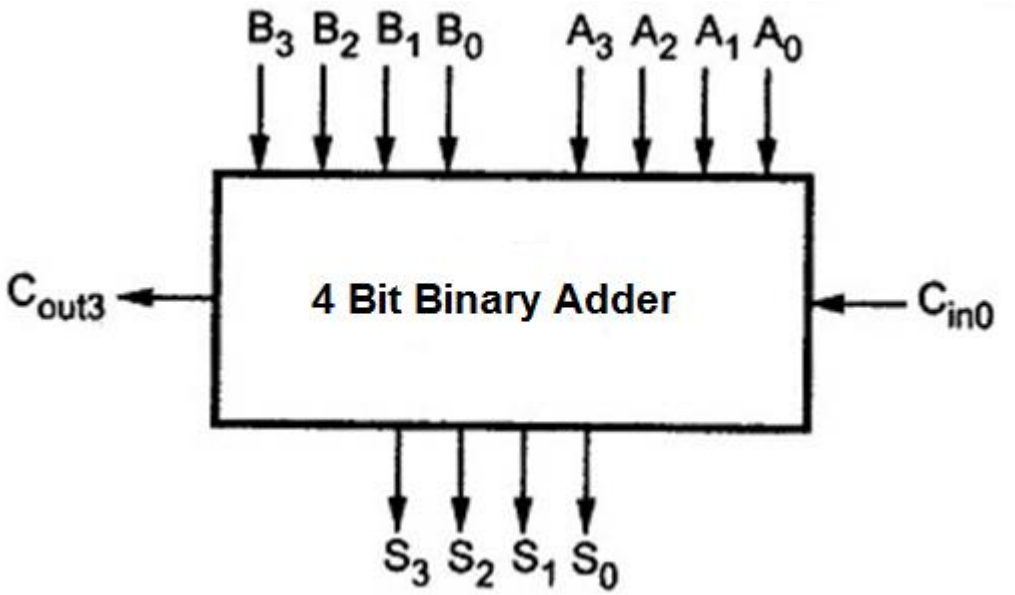
- A single full adder performs the addition of two one bit numbers and an input carry. But a **Parallel Adder** is a digital circuit capable of finding the arithmetic **sum** of two binary numbers that is **greater than one bit** in length by operating on corresponding pairs of bits in parallel.
- **A n bit parallel adder requires n full adders to perform the operation.**



Parallel Adder

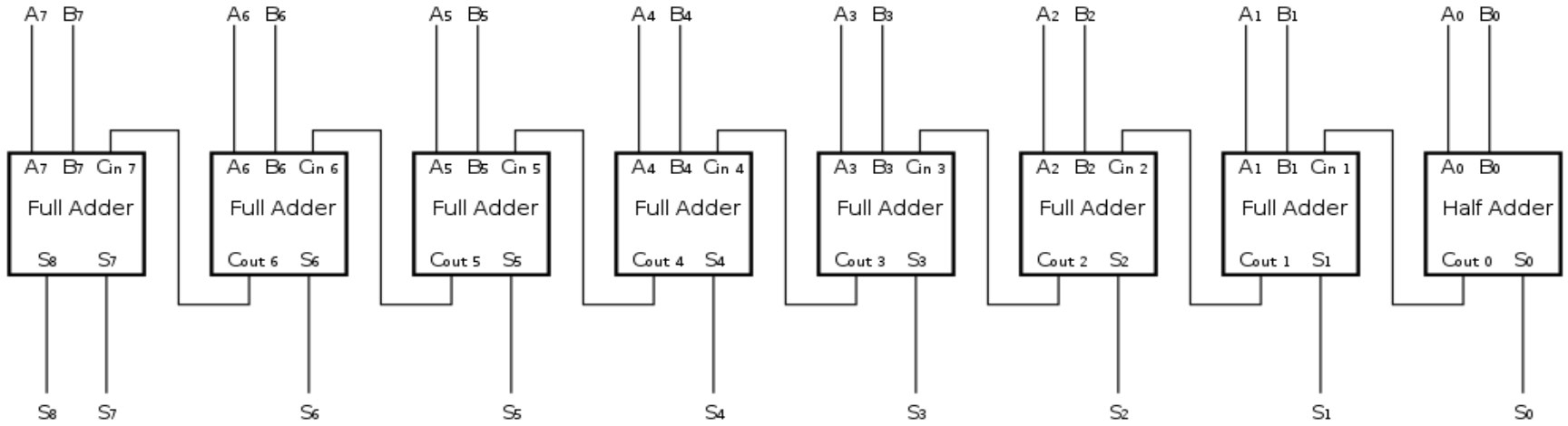
- **4 bit binary adder**
- **A 4 bit parallel adder requires 4 full adders to perform the operation.**



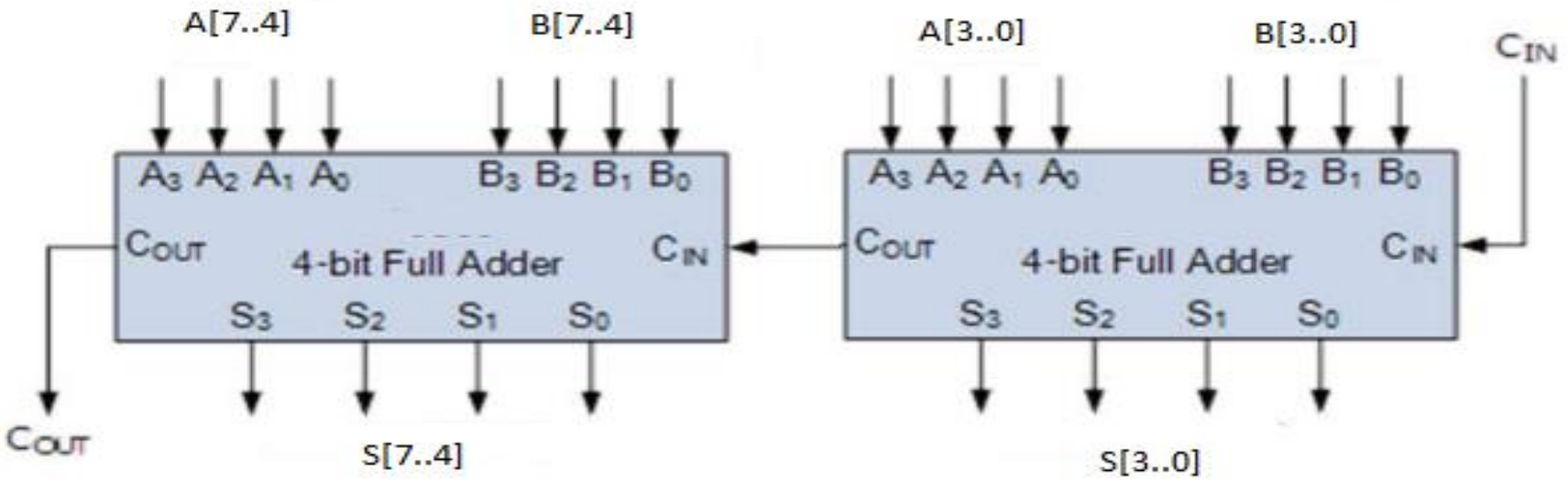


Parallel Adder

- **A 8 bit parallel adder requires 8 full adders to perform the operation.**

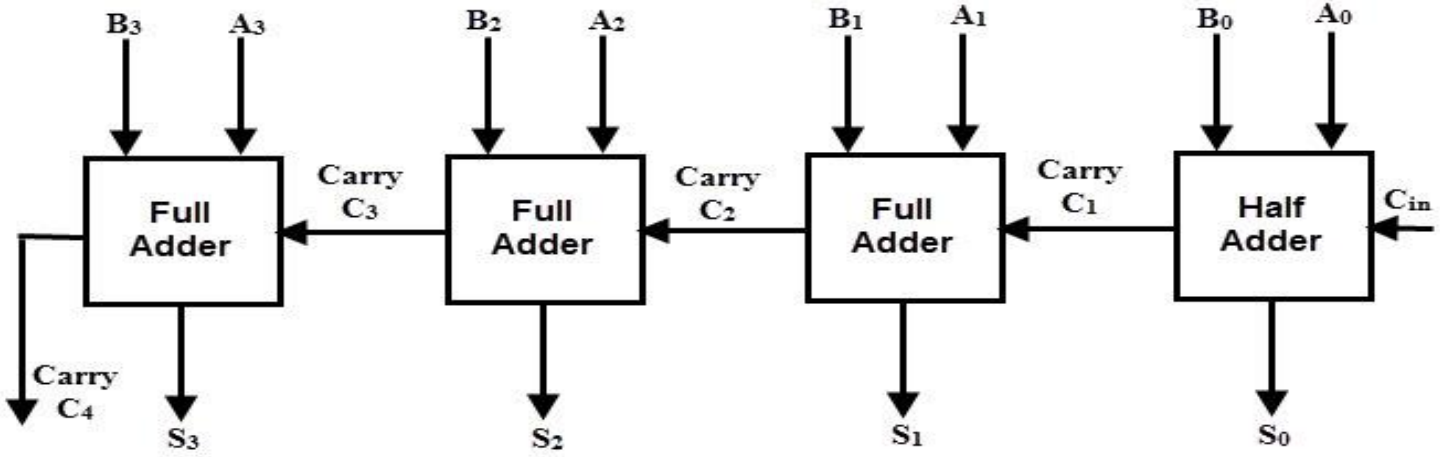


- **Block Diagram**



Propagation Delay in Parallel Adder

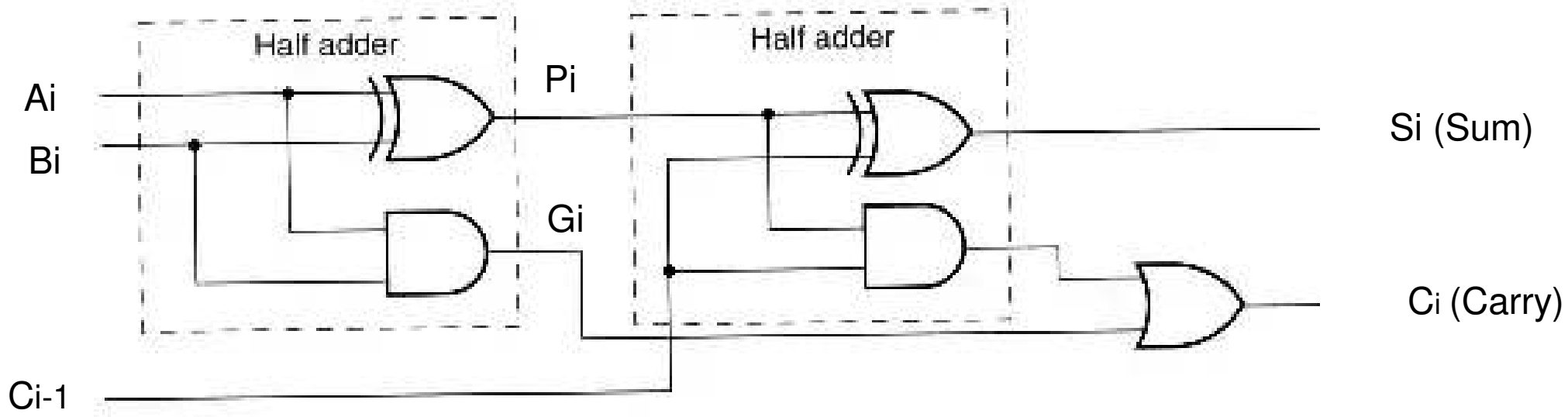
- One main disadvantage of “cascading” together 1-bit **binary adders** to add large binary numbers is that if inputs A and B change, the sum at its output will not be valid until any carry-input has “rippled” through every full adder in the chain because the MSB (most significant bit) of the sum has to wait for any changes from the carry input of the LSB (less significant bit).
- Consequently, there will be a finite delay before the output of the adder responds to any change in its inputs resulting in a accumulated delay.



A	A3	A2	A1	A0
B	B3	B2	B1	B0
+	C2	C1	C0	C-1
C3	S3	S2	S1	S0

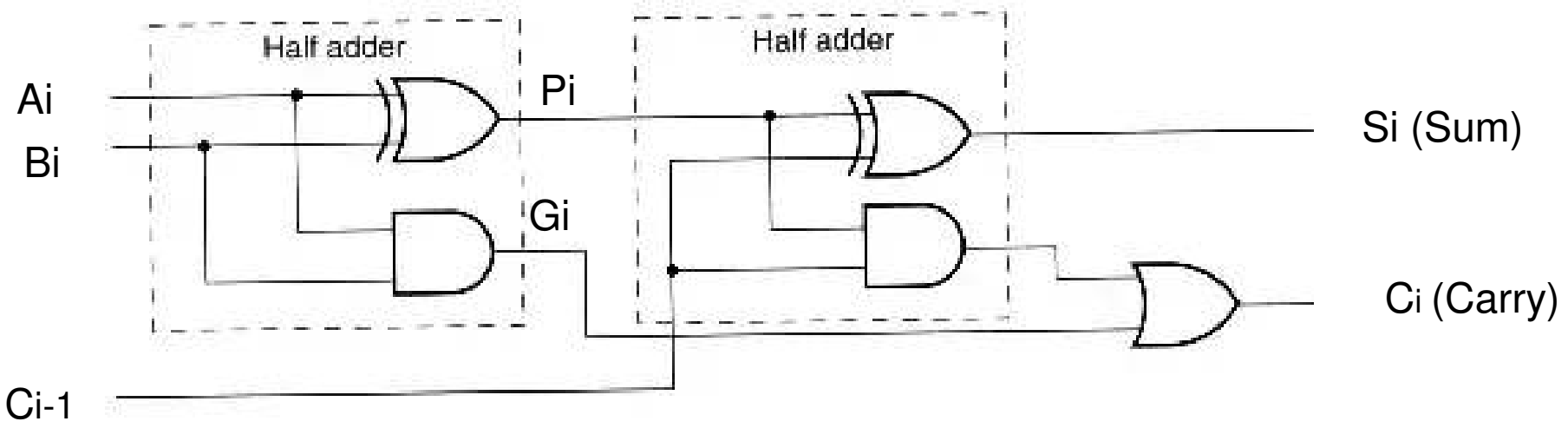
Look Ahead Carry Adder

- One solution is to generate the carry-input signals directly from the A and B inputs rather than using the ripple arrangement above. This then produces another type of binary adder circuit called a **Look Ahead Carry Binary Adder** where the speed of the parallel adder can be greatly improved using carry-look ahead logic.



- Gi is a **carry generate** which produces the carry when both Ai, Bi are one regardless of the input carry Ci-1.
- Pi is a **carry propagate** and it is associate with the propagation of carry from Ci-1 to Ci.

Look Ahead Carry Adder



From above diagram,

Pi = Ai⊕ Bi
Gi = Ai Bi
Si = Pi ⊕ Ci-1 = Ai ⊕ Bi ⊕Ci-1
Ci = Gi + Pi Ci-1

Need to calculate carry before to do addition by adder.

Look Ahead Carry Adder

The carry output Boolean function of each stage in a 4 stage carry-Look ahead adder can be expressed as,

Pi = Ai⊕ Bi

Gi = Ai Bi

Ci = Gi + Pi Ci-1

C0 = G0 + P0 C-1

C1 = G1 + P1 C0

= G1 + P1 (G0 + P0 C-1)

C1 = G1 + P1G0 + P1 P0 C-1

C2 = G2 + P2 C1

= G2 + P2 (G1 + P1G0 + P1 P0 C-1)

C2 = G2 + P2 G1 + P2 P1G0 + P2 P1 P0 C-1

C3 = G3 + P3 C2

= G3 + P3 (G2 + P2 G1 + P2 P1G0 + P2 P1 P0 C-1)

C3 = G3 + P3 G2 + P3 P2 G1 + P3 P2 P1G0 + P3 P2 P1 P0 C-1

Available inputs

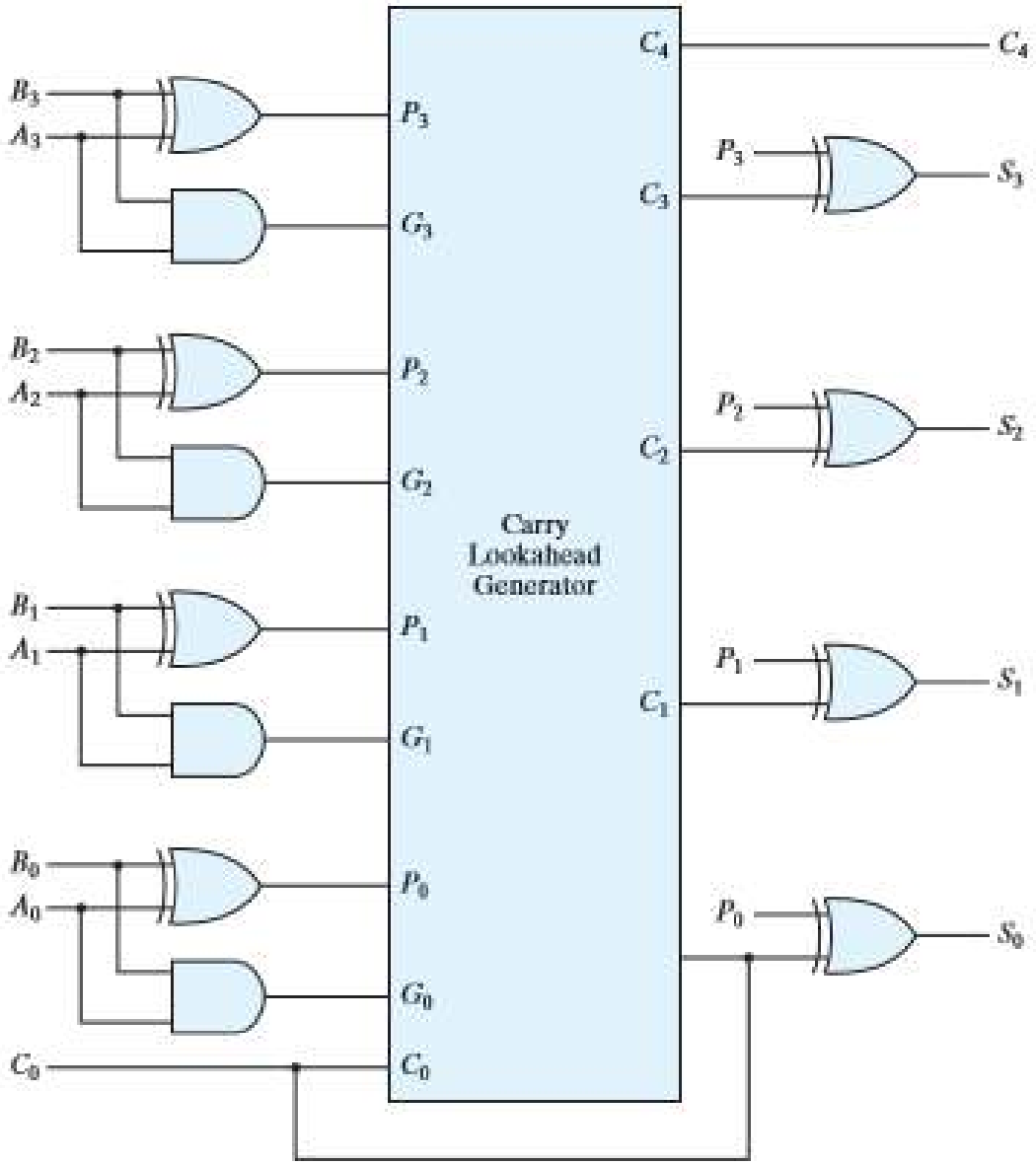
A A3 A2 A1 A0

B B3 B2 B1 B0

Cin C-1

- All above carry expressions generated by using G0,G1,G2,G3, P0,P1,P2,P3 & C-1.
- P is obtained by performing EX-OR operation of A , B.
- G is obtained by performing AND operation of A, B.
- If G, P, C-1 are available at a time then it is possible to produce the carry outputs, C0,C1,C2,C3 by using 2 level realization.

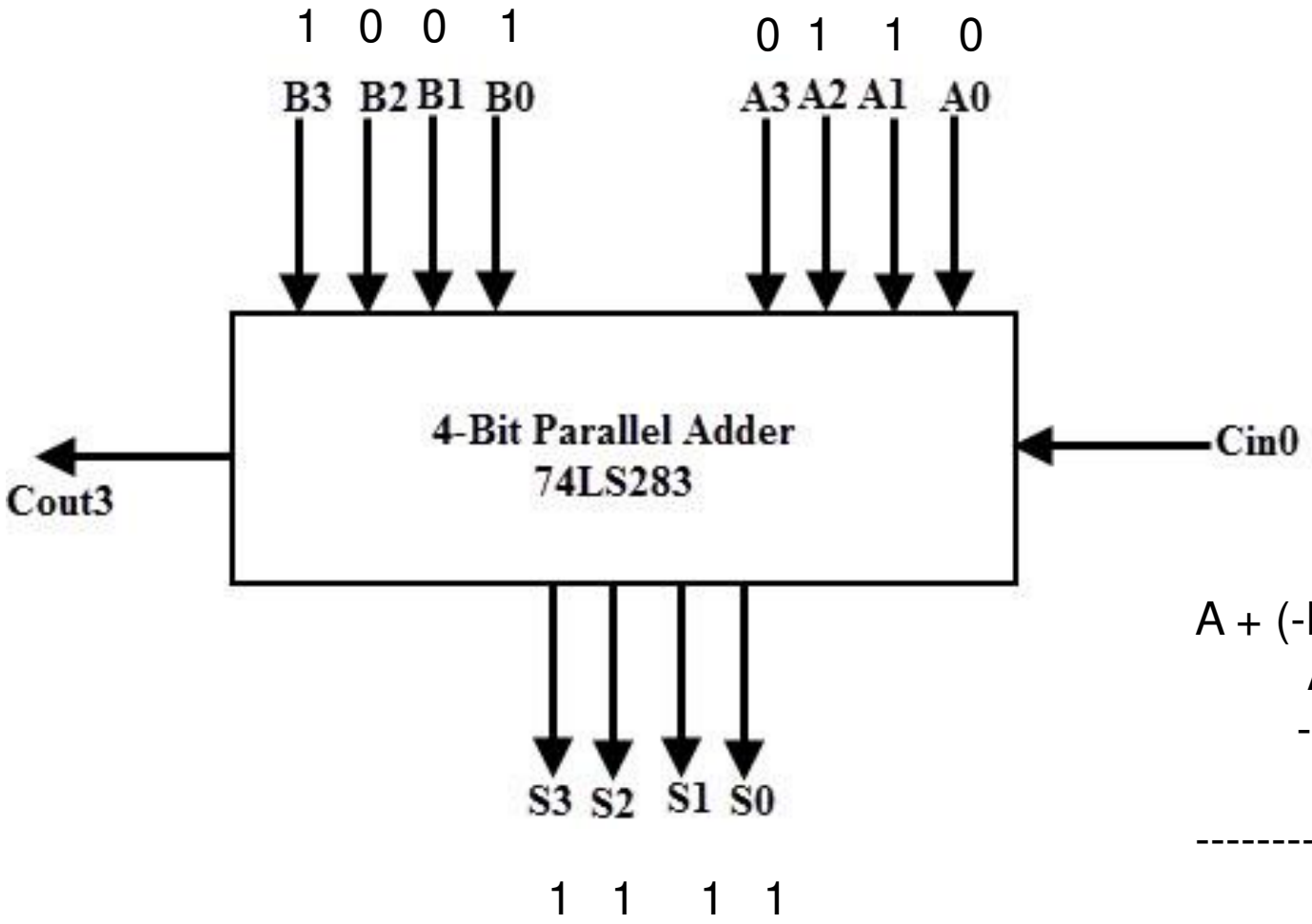
4 bit Parallel Adder using look ahead carry generator



4 Bit Parallel Adder using Look Ahead Carry Generator

4 Bit Parallel Adder using look ahead carry generator

Block diagram:



A+B A+(-B)

A 0110
B 1001 2'S COM 0111
2'S B 1 0 0 1
1. 1'S COMP 0 1 1 0
2. + 1
3. A + (-B)

A-B

```
  0 1 1 0
  1 0 0 1
  0   0 1
-----------------
1 1 1 0 1
```

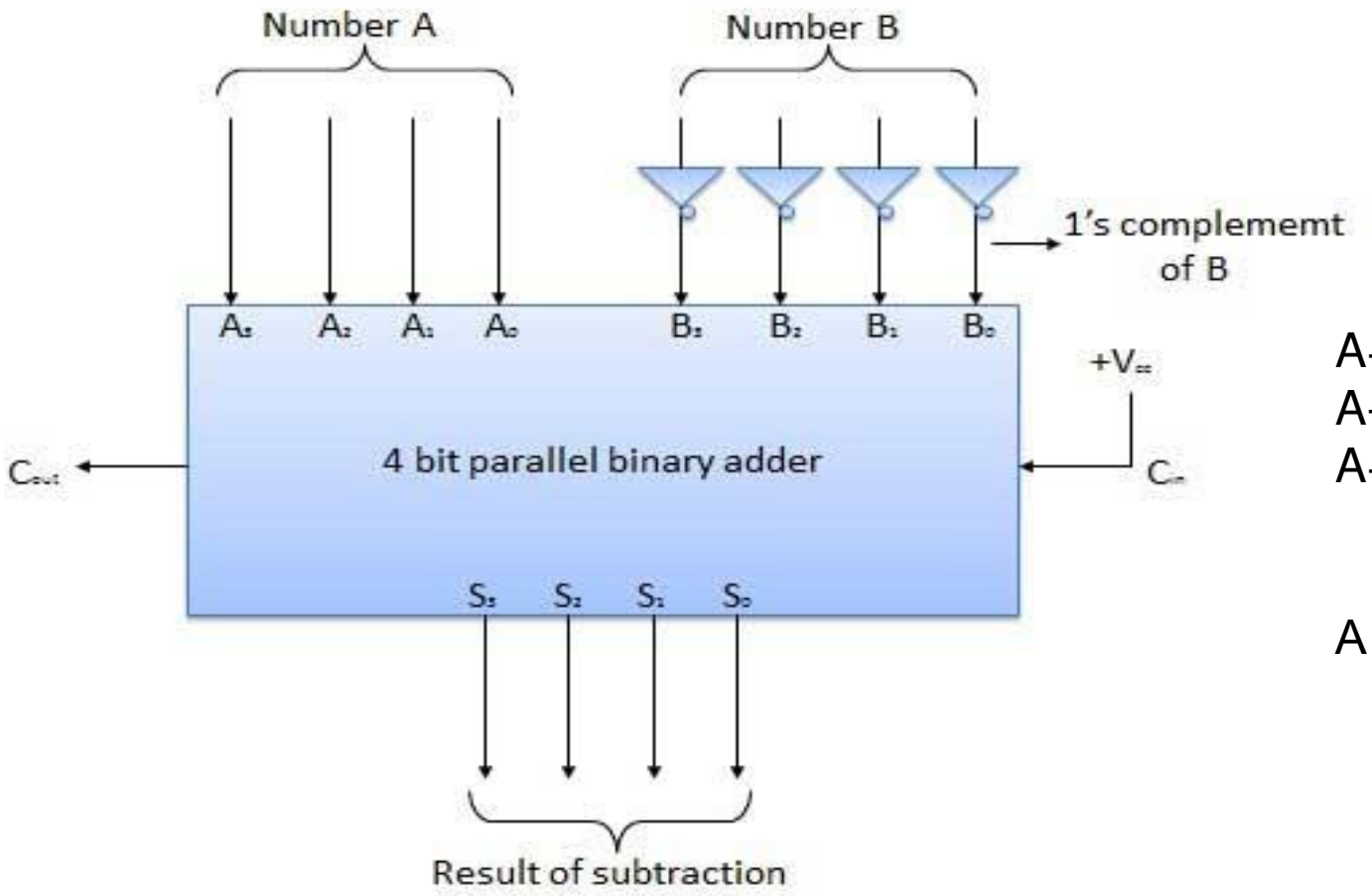
A + (-B)

```
    A   0 1 1 0
   -B   0 1 1 1
        1   1 0
-----------------------------
      1   1   0   1
```

4 Bit Parallel Subtractor

4 Bit Parallel Subtractor using 2's Complement Method

Block diagram:



A+B 7483
A-B 7483
A+(-B) SIGN
1'S OR 2'S

A + 2'S B 0101
1'S 1010
+1 1
---
10 11

BCD Adder

1. **Use the binary rules of addition to add the BCD numbers**
2. If the result of the addition is less than or equal to 9, then it is a valid BCD number.
3. if the result is greater than 9, then you need to convert it into a valid BCD code by adding 6(0110) in result.

Case 1:
Sum ≤ 9 , Carry = 0
Example: 3+ 6

3	**0**	**0**	**1**	**1**
6	**0**	**1**	**1**	**0**
	1	**1**	**0**	
9	**1**	**0**	**0**	**1**
	0	**0**	**0**	**0**
	0	**0**	**0**	
	1	**0**	**0**	**1**

Case 2:
Sum > 9 , Carry = 0
Example: 6+8

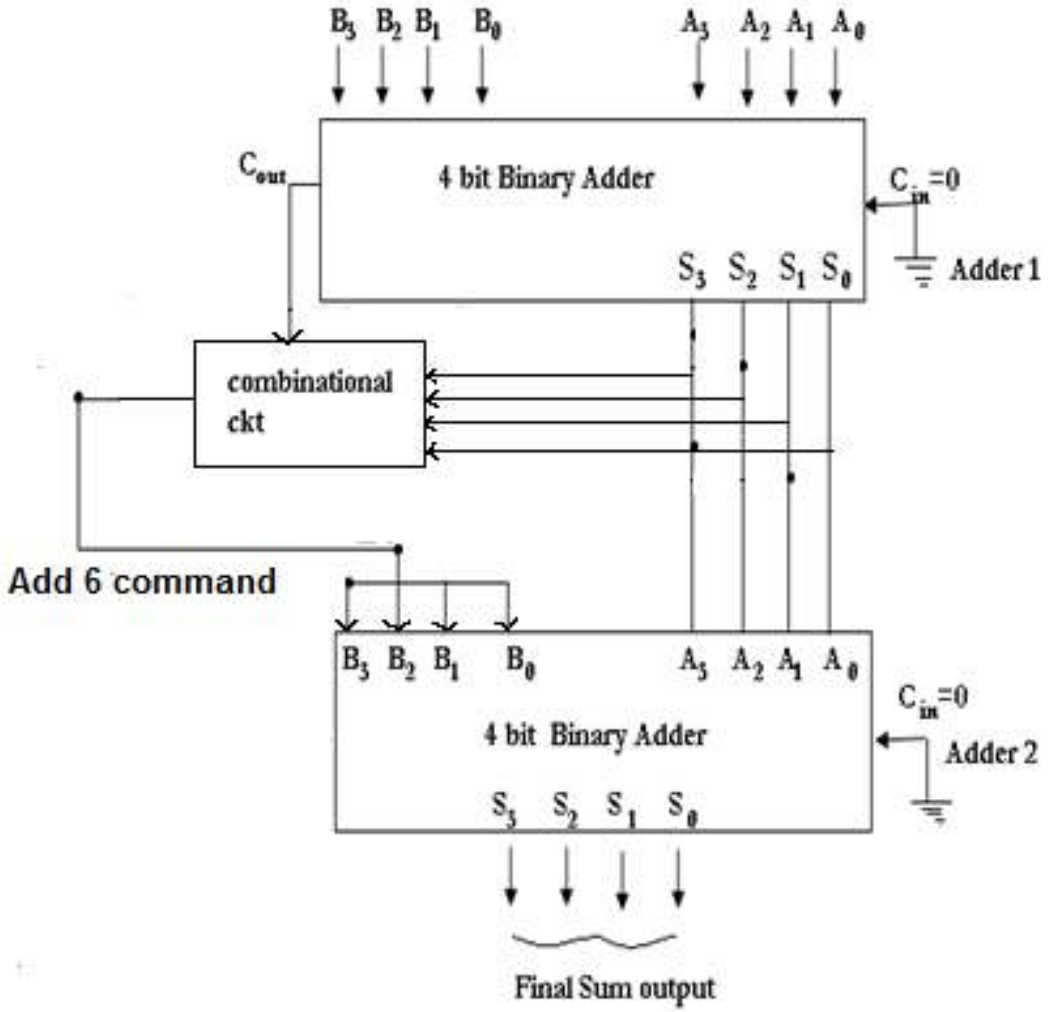
6	**0**	**1**	**1**	**0**
8	**1**	**0**	**0**	**0**
	0	**0**	**0**	
14	**1**	**1**	**1**	**0**
	0	1	1	0
	1	1	0	
1	**0**	**1**	**0**	**0**

Case 3:
Sum ≤ 9 , Carry = 1
Example: 8+9

8	**1**	**0**	**0**	**0**
9	**1**	**0**	**0**	**1**
	0	**0**	**0**	
17	**1**	**0**	**0**	**0**	**1**
		0	1	1	0
		0	0	0	
	1	0	1	1	1

BCD Adder

1. The 4 bit binary adder IC 7483 can be used to perform addition of BCD numbers.
2. In this, if the four-bit sum output is not a valid digit, or if a carry C3 is generated then decimal 6 (0110 binary) is to be added to the sum to get the correct result.
3. The output of combinational circuit should be 1 if the sum produced by adder 1 is greater than 9 i.e. 1001.



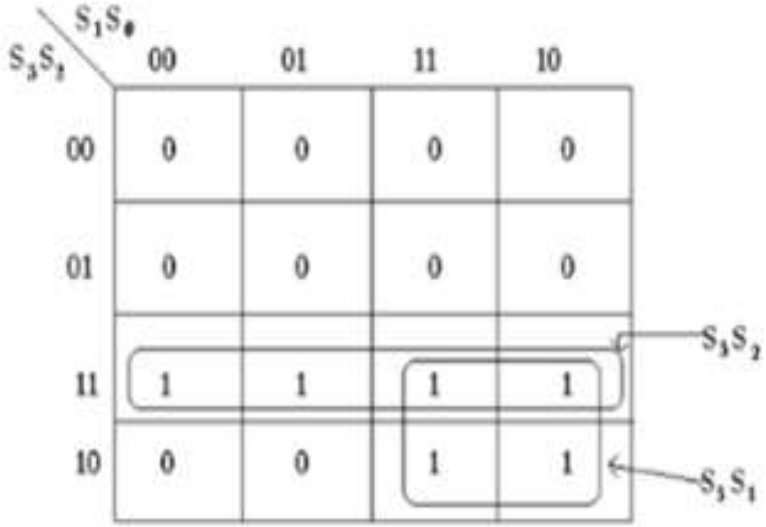
BCD Adder

Design Combinational Circuit

Inputs				Output
S_3	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Sum is invalid BCD number. Hence Y= 1

K-map:

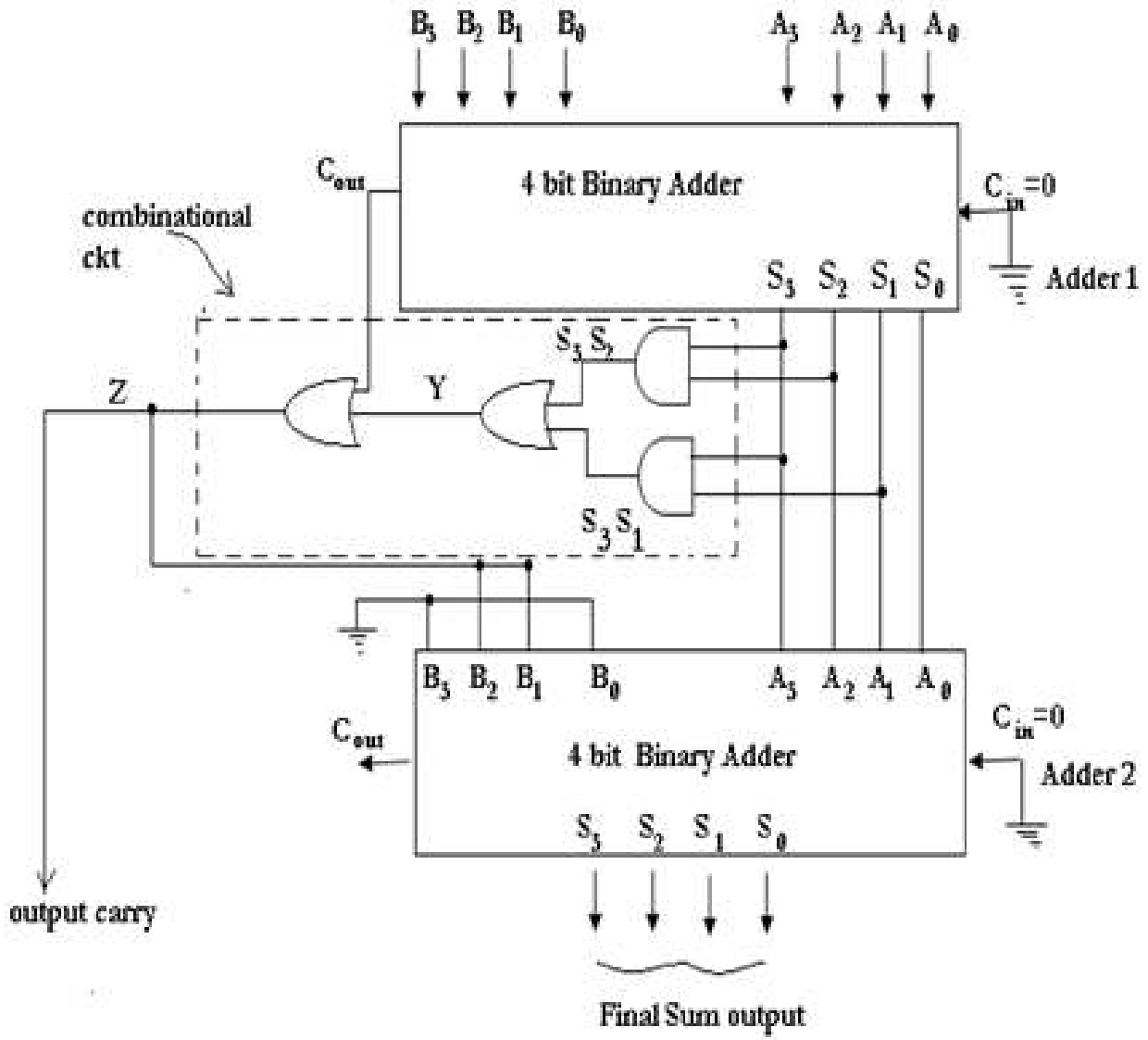


The Boolean expression is

$Y = S_3S_2 + S_3S_1$

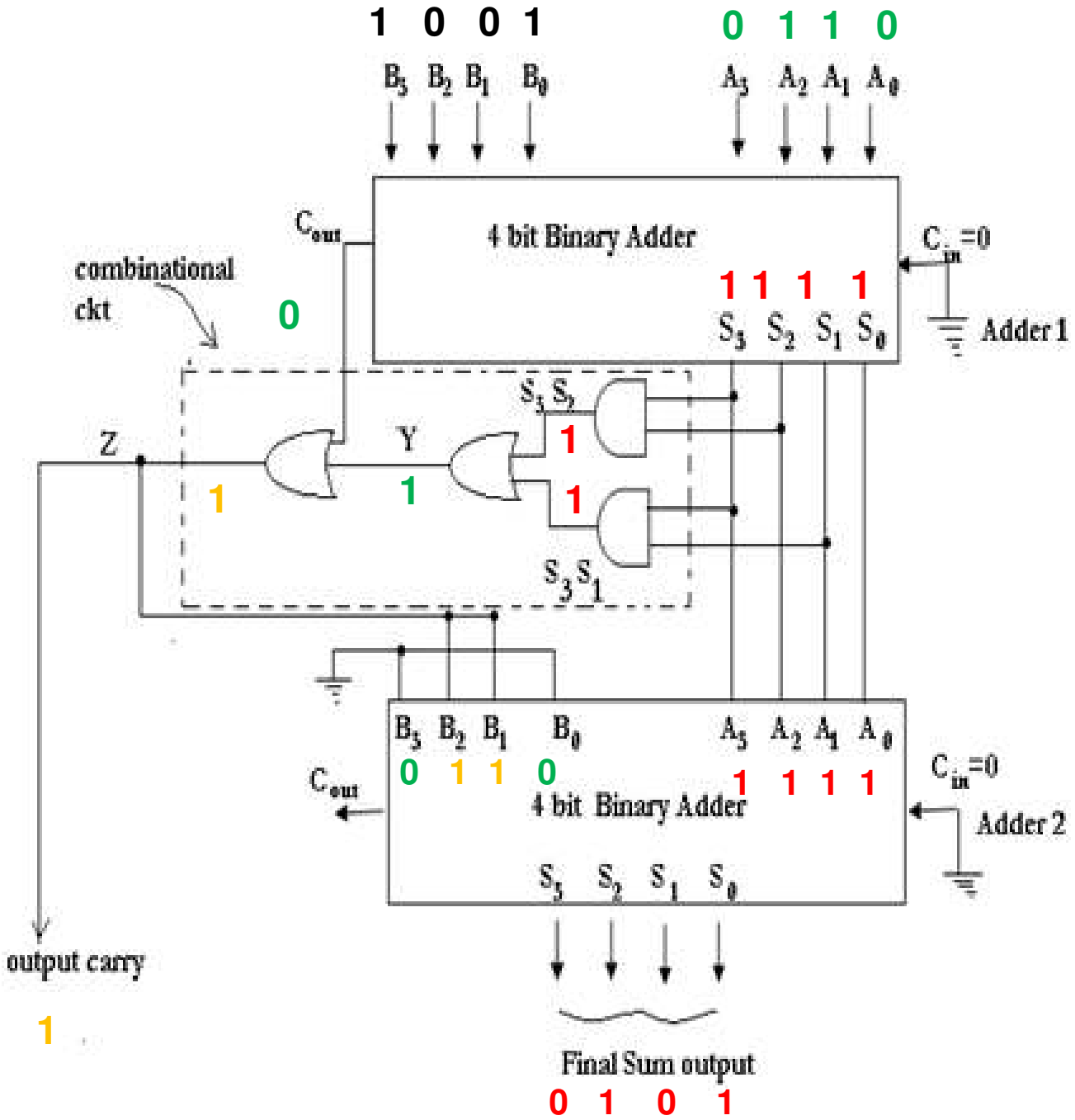
BCD Adder

1. The output of the combinational circuit should be 1 if Cout of adder-1 is high. Therefore Y is ORed with Cout of adder 1.
2. Output of Combinational circuit is connected to B1 and B2 input of adder 2; B0 and B3 connected to ground permanently. This makes B3B2B1B0 = 0110 when Z=1 otherwise B3B2B1B0 = 0000.
3. The sum output of adder 1 are applied to A3-A0 of adder 2.
4. The output of Combinational circuit is to be used as final output carry and the carry output of adder 2 is to be ignored.



Example: 6 + 9

```
A  0 1 1 0
B  1 0 0 1
   -------
    1 1 1 1        invalid BCD
+   0 1 1 0        add 0110
   -------
  1 0 1 0 1
  1     5
```



Agenda

01 Code Converter: BCD, Excess-3, Gray, Binary Code

02 Half Adder, Full Adder, Half Subtractor, Full Subtractor

03 Binary Adder (IC 7483), BCD Adder, Look ahead carry generator

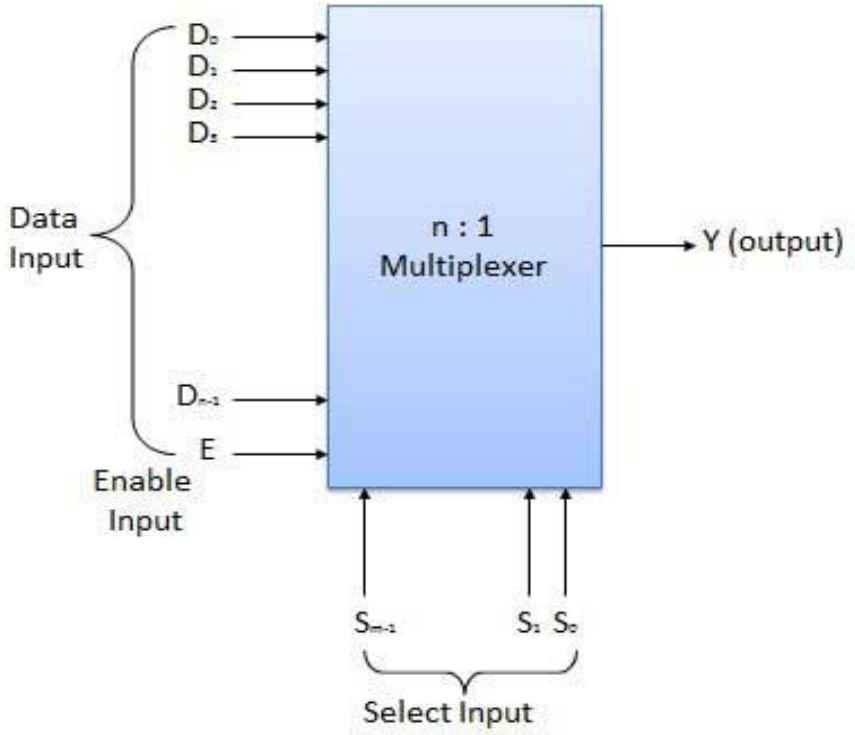
04 Multiplexer (MUX): MUX IC (74153, 74151), Cascading multiplexer

05 Demultiplexer (DEMUX)- Decoder (IC 74138, 74154), Implementation of SOP and POS using MUX , DEMUX

06 Comparators (2 bit), Parity Generators and Checkers.

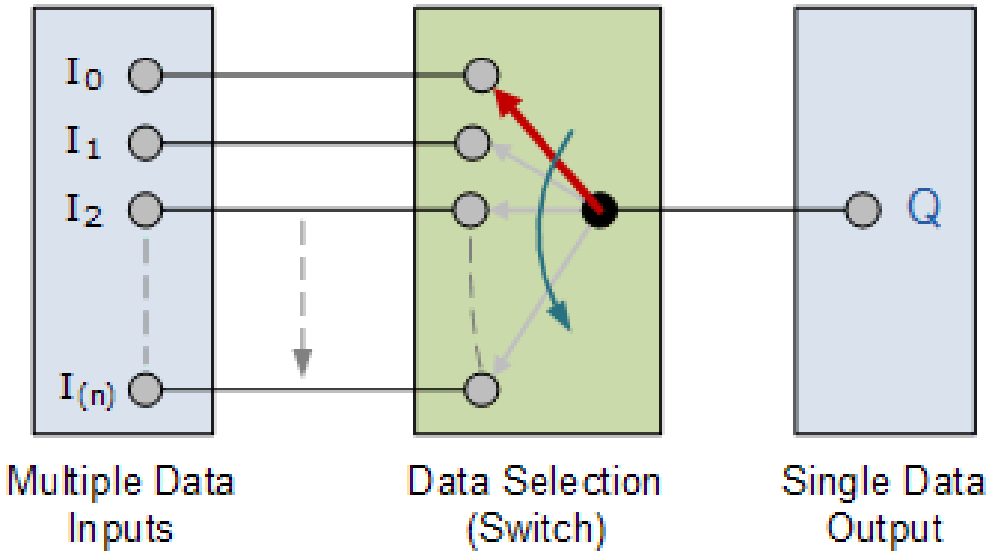
Multiplexer (MUX)

- **Multiplexer** is a combinational circuit that has maximum of n data inputs, ‘m’ selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.
- Since there are ‘m’ selection lines, there will be 2^m possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer is also called as **MUX**.
- Relation between Data inputs (n) and select inputs (m) **is 2^m = n.**



Multiplexer (MUX)

- In digital electronics, multiplexers are also known as data selectors because they can “select” each input line.
- Enable input (E) is active low input used for cascading.



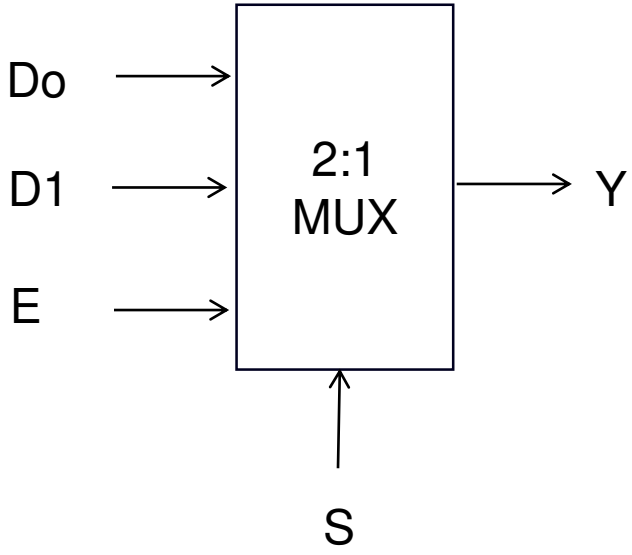
Types of multiplexer:

1. 2:1 multiplexer
2. 4:1 multiplexer
3. 8:1 multiplexer
4. 16:1 multiplexer
5. 32:1 multilexer

2:1 MUX

- A 2:1 multiplexer consists of two inputs D0 and D1, one select input S and one output Y. Depends on the select signal, the output is connected to either of the inputs.
- Since there are two input signals only two ways are possible to connect the inputs to the outputs, so one select is needed to do these operations.

Block Diagram:



Function Table:

E	S	Y
0	X	0
1	0	D0
1	1	D1

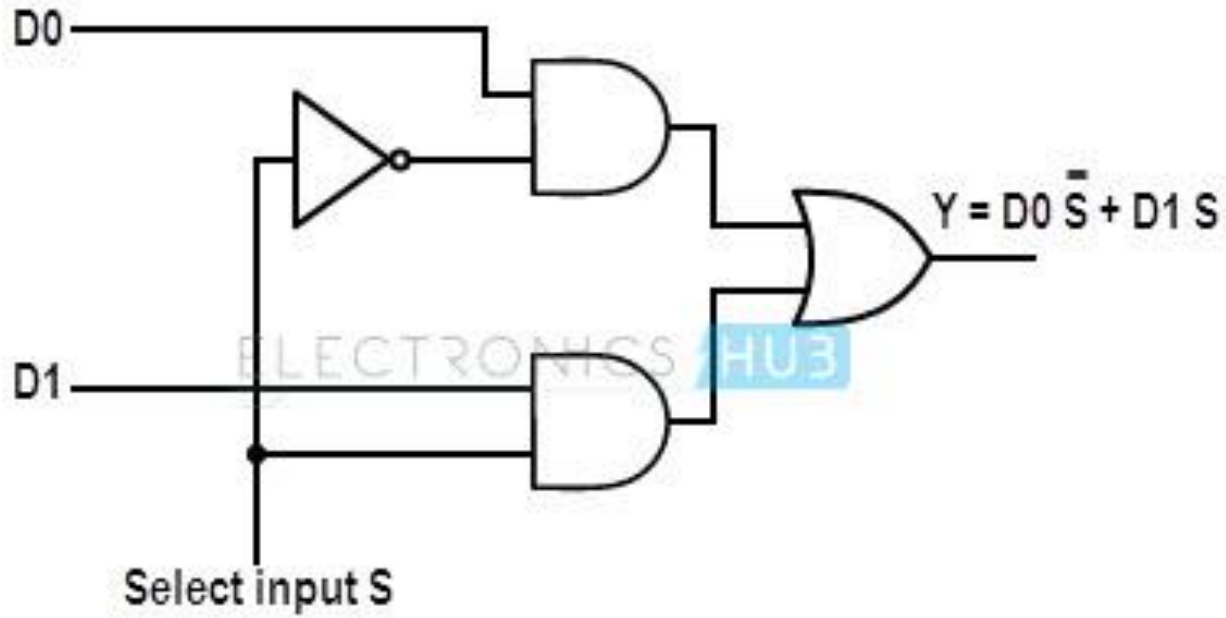
Truth Table:

E	S	D1	D0	Y	
0	X	X	X	**0**	
1	0	X	0	**0**	
1	0	X	1	**1**	⇦ ES'Do
1	1	0	X	**0**	
1	1	1	X	**1**	⇦ ESD1

$Y = ES'Do + ESD1$

2:1 MUX

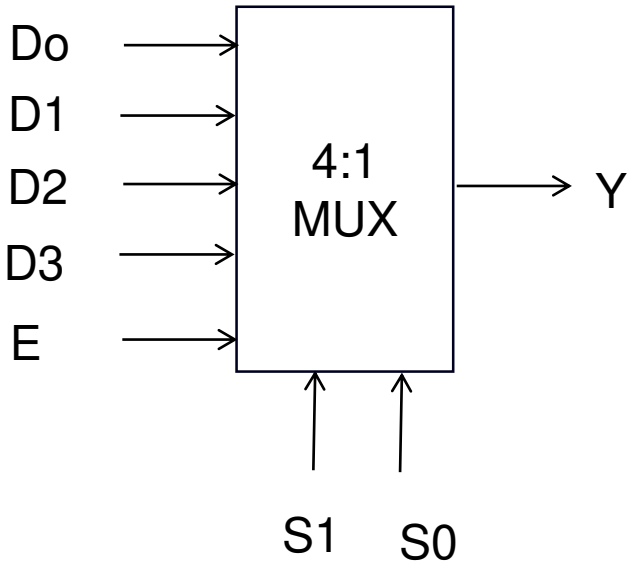
- **Logic Realization of 2:1 MUX**



4:1 MUX

• A 4:1 multiplexer consists of four inputs D0, D1, D2 and D3; two select inputs S1 and S0 and one output Y. Depends on the select signal, the output is connected to either of the inputs.

Block Diagram:



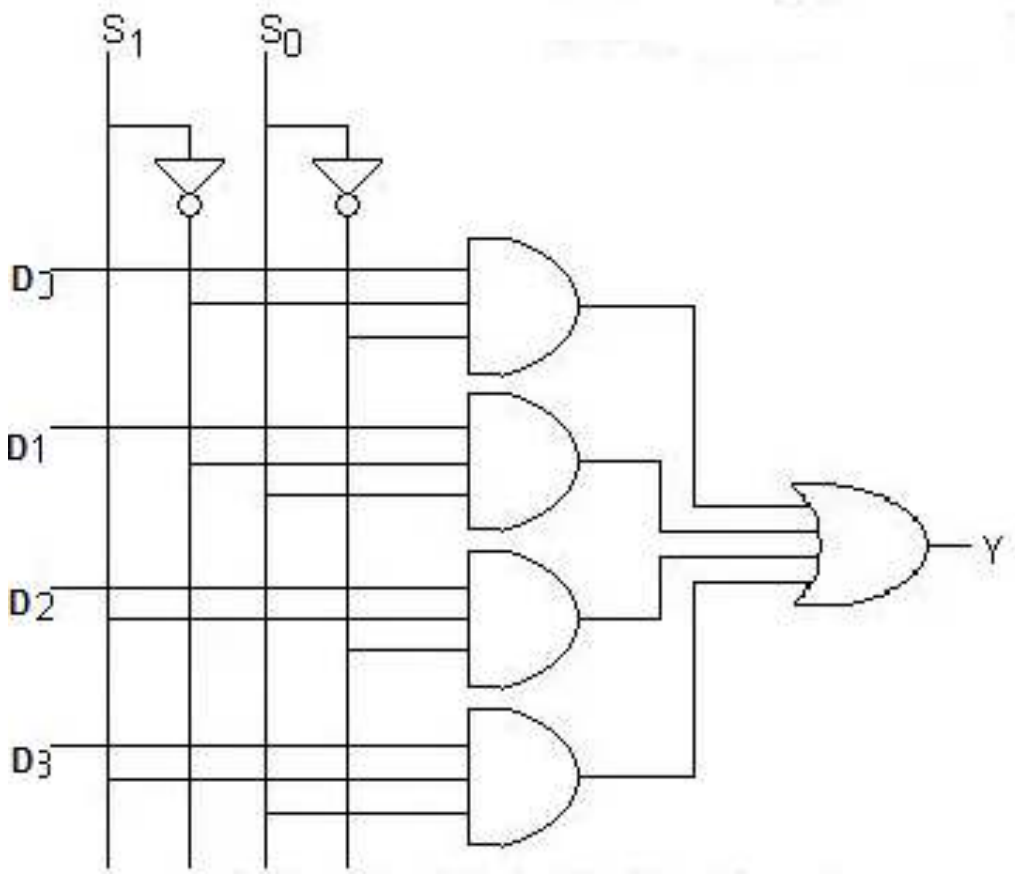
Function Table:

E	S1	S0	Y	
0	X	X	0	
1	0	0	D0	⇦ ES1'S0'D0
1	0	1	D1	⇦ ES1'S0D1
1	1	0	D2	⇦ ES1S0'D2
1	1	1	D3	⇦ ES1S0D3

$$Y = ES1'S0'D0 + ES1'S0D1 + ES1S0'D2 + ES1S0D3$$

4:1 MUX

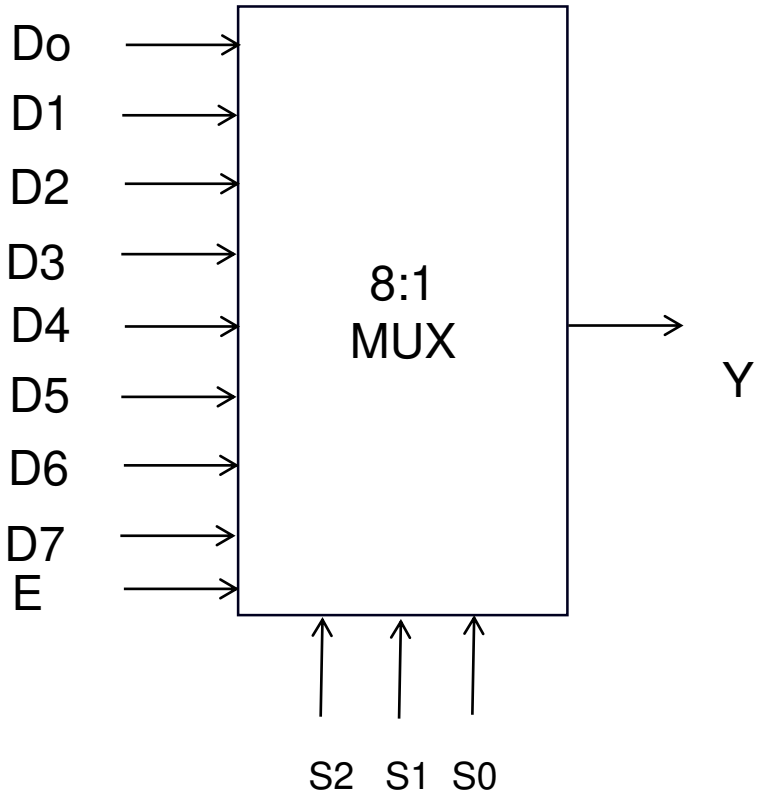
- **Logic Realization of 4:1 MUX**



8:1 MUX

• A 8:1 multiplexer consists of eight inputs D0 to D7, three select input S2, S1, S0 and one output Y. Depends on the select signal, the output is connected to either of the inputs.

Block Diagram:



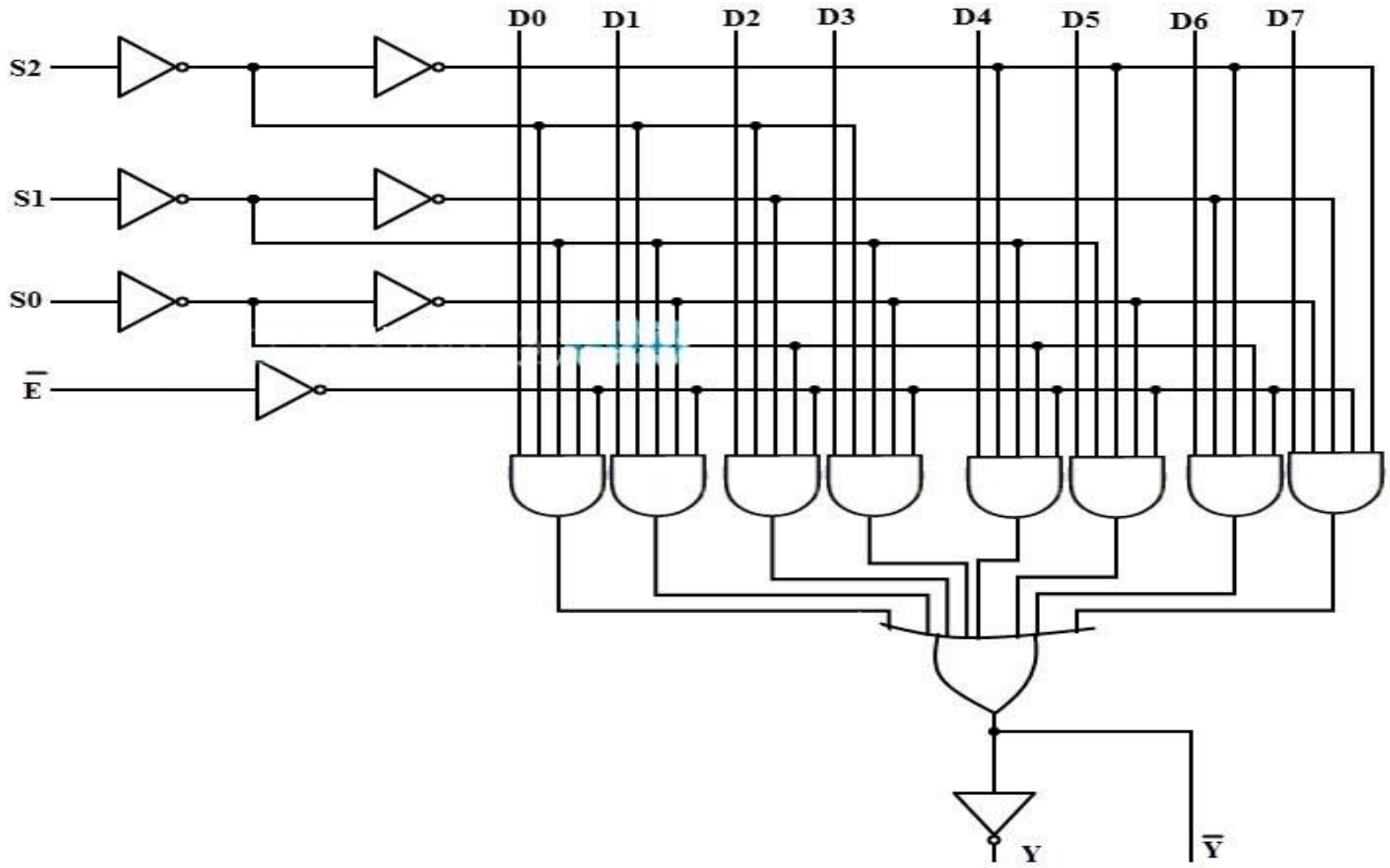
Function Table:

E	S2	S1	S0	Y	
0	X	X	X	0	
1	0	0	0	D0	⇐ ES2’S1’S0’D0
1	0	0	1	D1	⇐ ES2’S1’S0D1
1	0	1	0	D2	⇐ ES2’S1S0’D2
1	0	1	1	D3	⇐ ES2’S1S0D3
1	1	0	0	D4	⇐ ES2S1’S0’D4
1	1	0	1	D5	⇐ ES2S1’S0D5
1	1	1	0	D6	⇐ ES2S1S0’D6
1	1	1	1	D7	⇐ ES2S1S0D7

8:1 MUX

- **Logic Realization of 8:1 MUX**

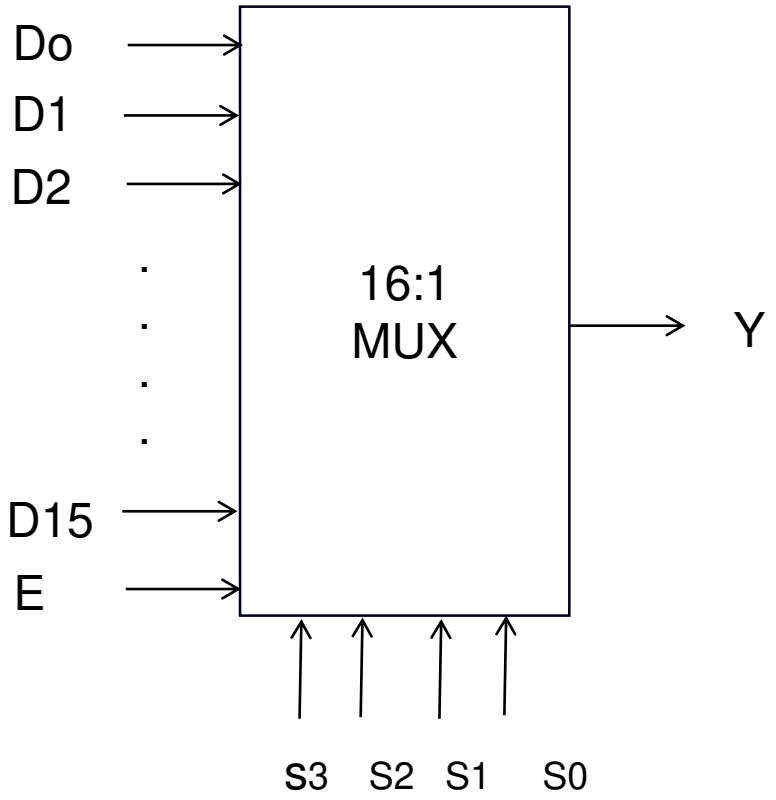
Y = ES2’S1’S0’D0 + ES2’S1’S0D1 + ES2’S1S0’D2 + ES2’S1S0D3 + ES2’S1’S0’D0 + ES2’S1’S0D1 + ES2’S1S0’D2 + ES2’S1S0D3



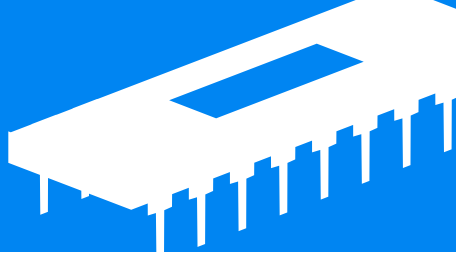
16:1 MUX

• A 16:1 multiplexer consists of sixteen inputs D0 to D15, four select input S3,S2, S1, S0 and one output Y. Depends on the select signal, the output is connected to either of the inputs.

Block Diagram:

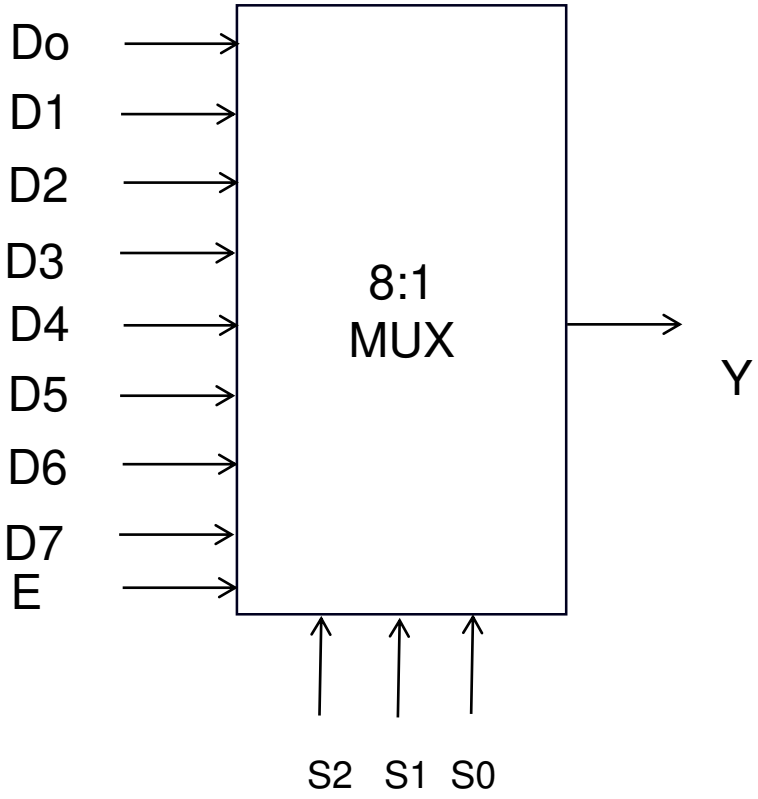


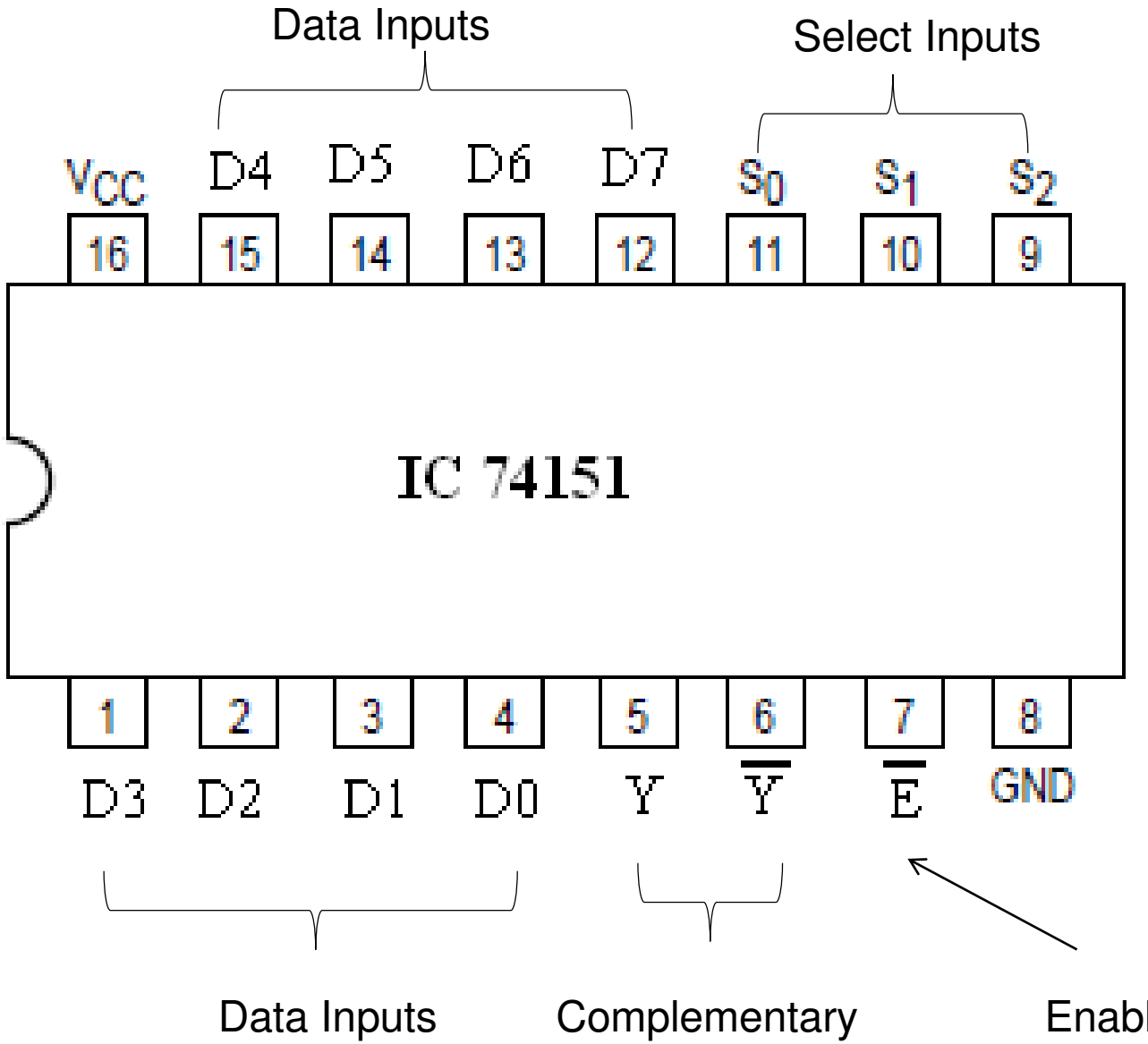
MUX IC



IC number	Description	Output
74150	16:1 multiplexer	Inverted input
74151	8:1 multiplexer	Complementary output
74151A	8:1 multiplexer	Complementary output
74152	8:1 multiplexer	Inverted input
74153	Dual 4:1 multiplexer	Same as input
74158	Quad 2:1 multiplexer	Inverted input
74157	Quad 2:1 multiplexer	Same as input

74151 IC


Do
D1
D2
D3
D4
D5
D6
D7
E
8:1
MUX
Y
S2
S1
S0

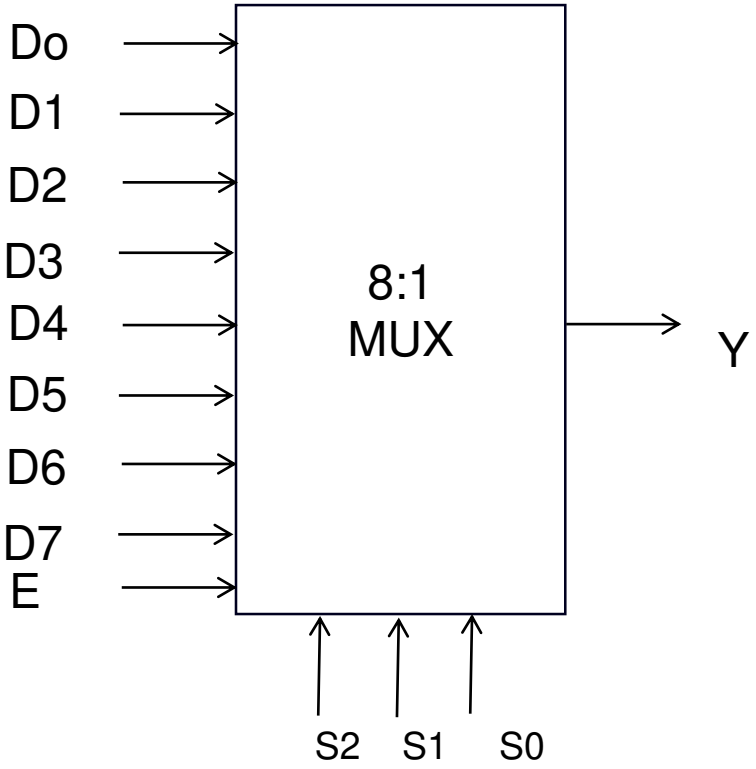

Data Inputs
Select Inputs
VCC
D4
D5
D6
D7
S0
S1
S2
16
15
14
13
12
11
10
9
IC 74151
1
2
3
4
5
6
7
8
D3
D2
D1
D0
Y
Y
E
GND
Data Inputs
Complementary
Enable

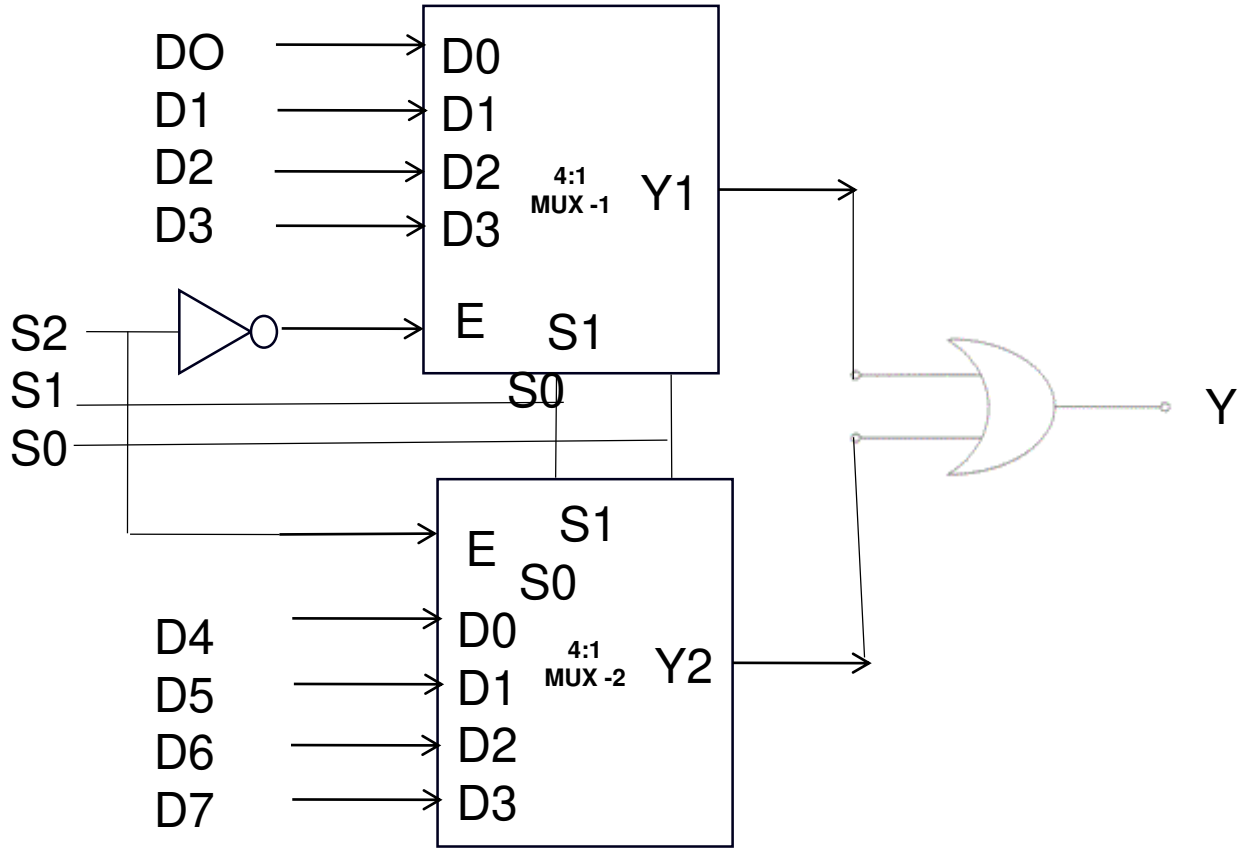
Applications of MUX

1. **Communication System –** A Multiplexer is used in communication systems, which has a transmission system and also a communication network. A Multiplexer is used to increase the efficiency of the communication system by allowing the transmission of data such as audio & video data from different channels via cables and single lines.
2. **Computer Memory –** A Multiplexer is used in computer memory to keep up a vast amount of memory in the computers, and also to decrease the number of copper lines necessary to connect the memory to other parts of the computer.
3. **Telephone Network –** A multiplexer is used in telephone networks to integrate the multiple audio signals on a single line of transmission.
4. It is used in A/D and D/A converters.
5. **P**arallel to serial converter

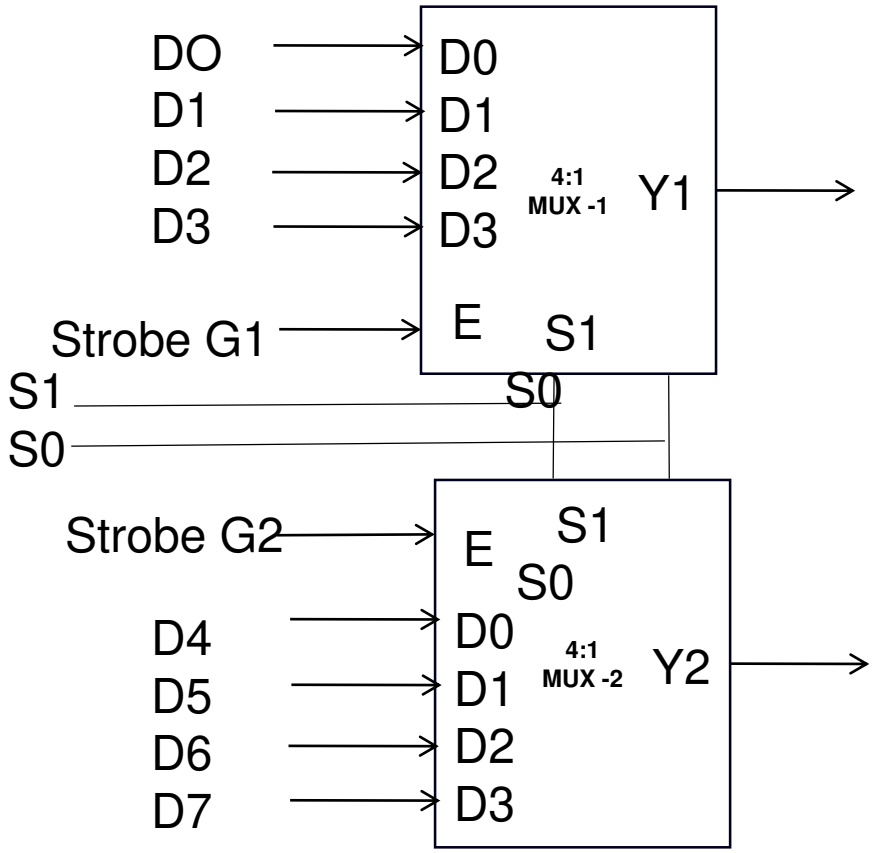
Multiplexer Tree

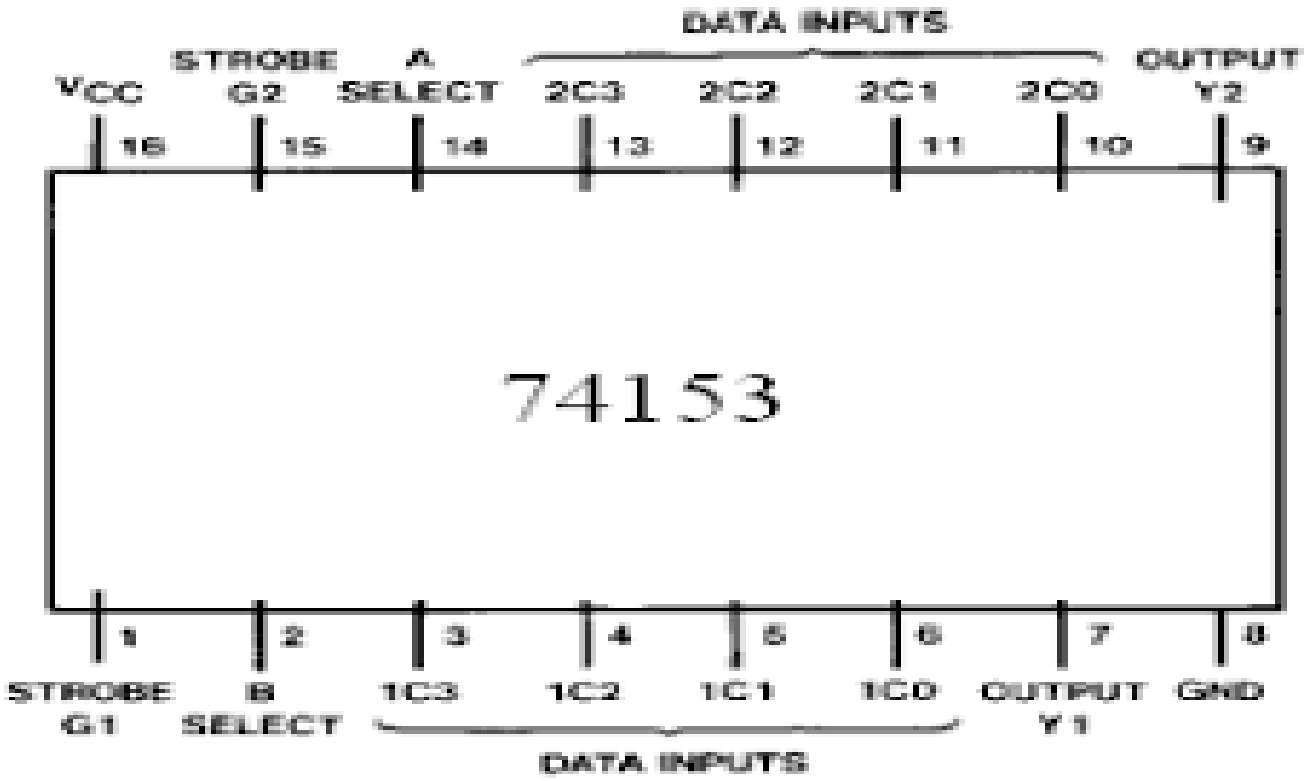
- **The multiplexers with more number of inputs can be obtained by cascading multiplexers with less number of inputs. Such a configuration is called as multiplexer.**
- **E.g. Implementation of 8:1 Mux using 4:1 mux.**



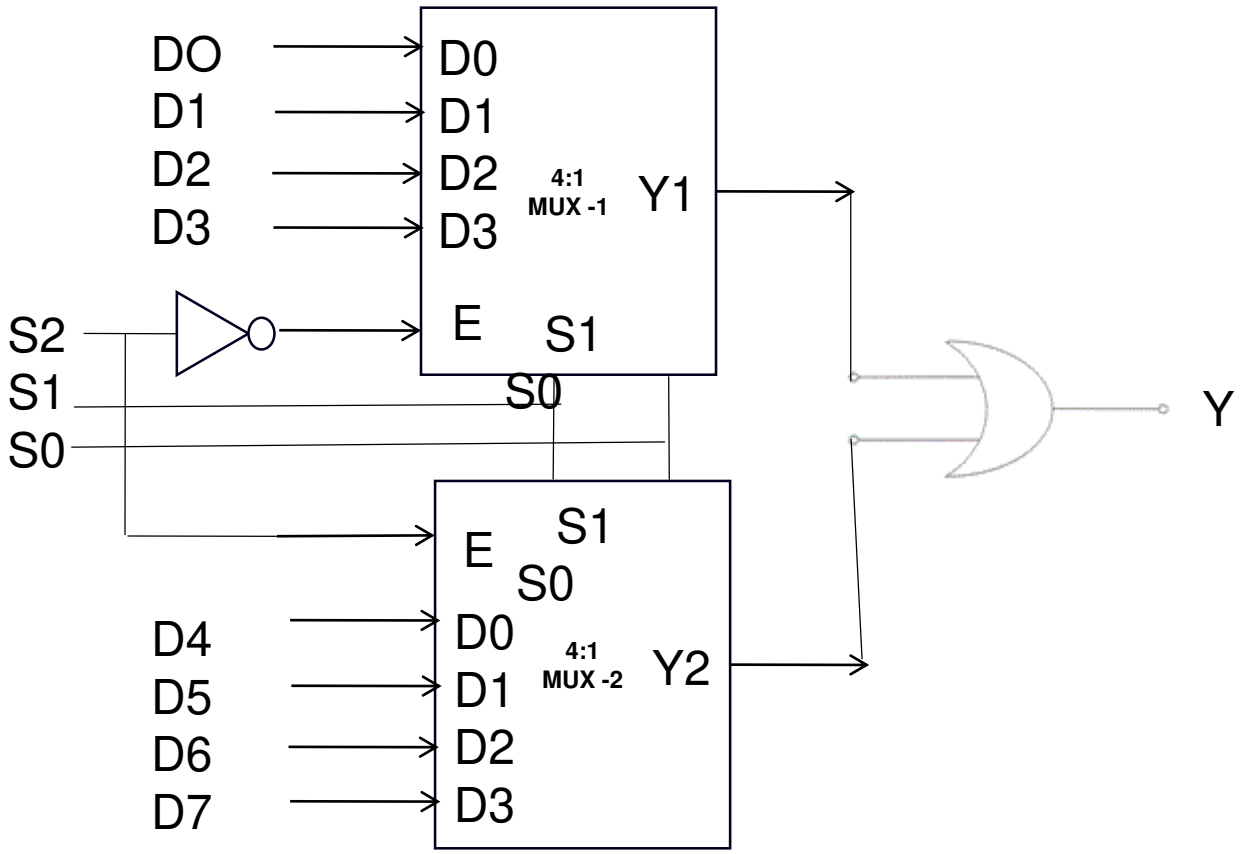


74153 IC


DO
D1
D2
D3
D0
D1
D2
D3
4:1
MUX -1
Y1
Strobe G1
E
S1
S1
S0
S0
Strobe G2
E
S1
S0
D4
D5
D6
D7
D0
D1
D2
D3
4:1
MUX -2
Y2


DATA INPUTS
STROBE G2
A SELECT
OUTPUT Y2
VCC
2C3
2C2
2C1
2C0
16
15
14
13
12
11
10
9
74153
1
2
3
4
5
6
7
8
STROBE G1
B SELECT
1C3
1C2
1C1
1C0
OUTPUT Y1
GND
DATA INPUTS

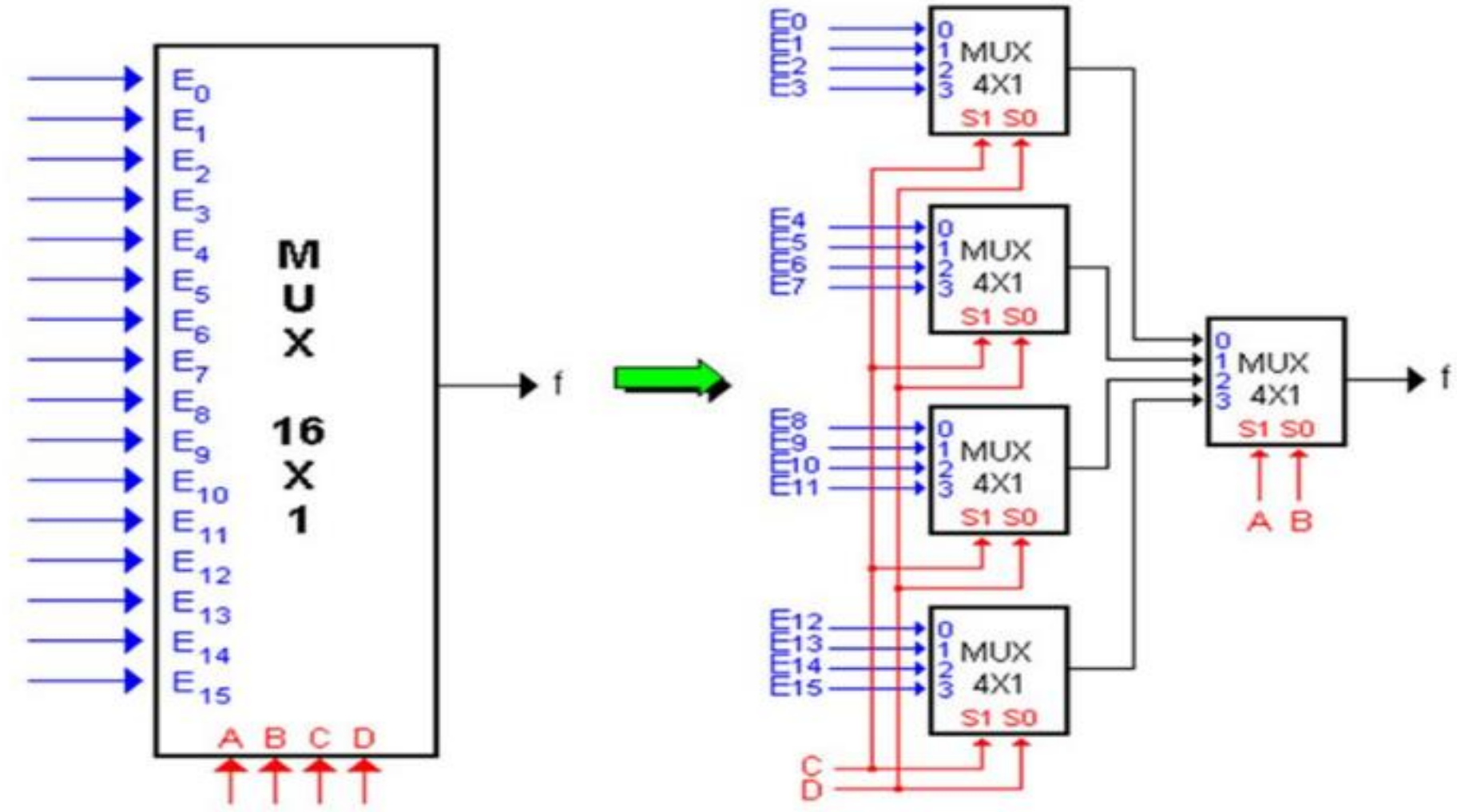
74153 MUX



S2 (E)	S1	S0	Y1	Y2	Y
0	0	0	D0	D4	D0
0	0	1	D1	D5	D1
0	1	0	D2	D6	D2
0	1	1	D3	D7	D3
1	0	0	D0	D4	D4
1	0	1	D1	D5	D5
1	1	0	D2	D6	D6
1	1	1	D3	D7	D7

Multiplexer Tree

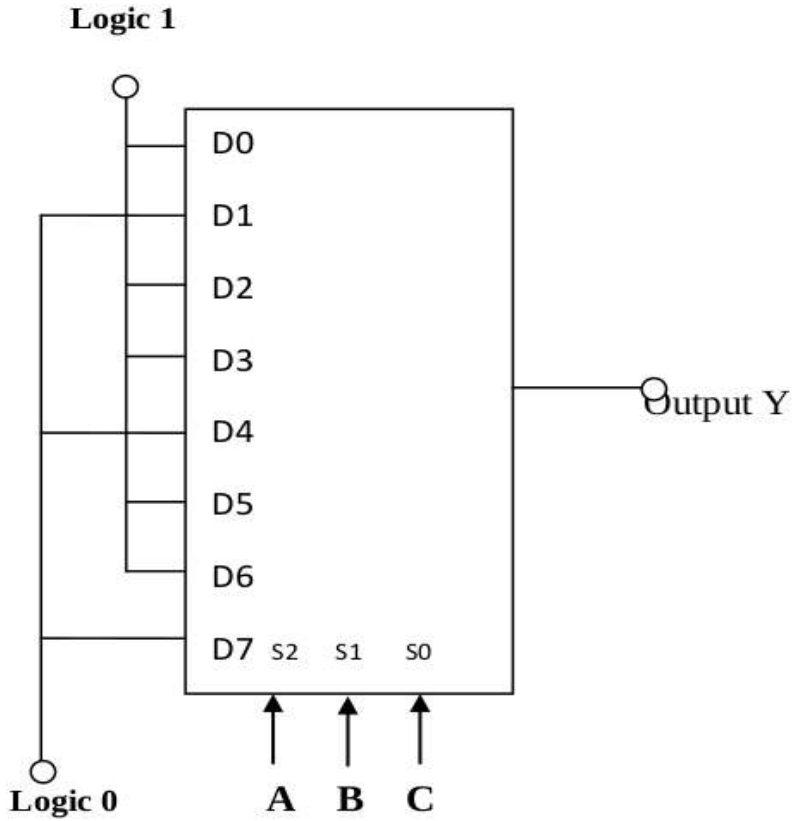
16:1 MUX using 4:1 MUX



Examples

1. **Implement the following expression using a MUX F(A,B,C) = ∑m(0,2,3,5,6)**

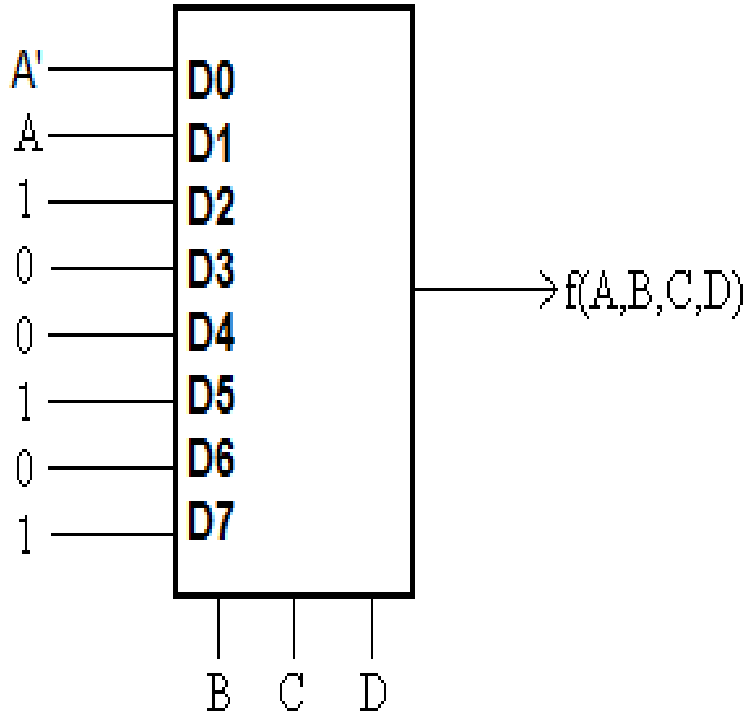
A	B	C	Y	
0	0	0	**1**	**D0**
0	0	1	**0**	**D1**
0	1	0	**1**	**D2**
0	1	1	**1**	**D3**
1	0	0	**0**	**D4**
1	0	1	**1**	**D5**
1	1	0	**1**	**D6**
1	1	1	**0**	**D7**



Examples

2. Implement the following Boolean function using 8:1 MUX F(A,B,C,D) = ∑ m(0,2,5,7,9,10,13,15)

	D0	D1	D2	D3	D4	D5	D6	D7
A’	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	A’	A	VCC	GND	GND	VCC	GND	VCC



Examples

3. Implement the Full Adder using 8:1 MUX.

4. Implement the following Boolean function using 8:1 MUX $F(A,B,C,D) = \sum m(0,12,3,6,8,9,11,12,14)$

Agenda

01 Code Converter: BCD, Excess-3, Gray, Binary Code

02 Half Adder, Full Adder, Half Subtractor, Full Subtractor

03 Binary Adder (IC 7483), BCD Adder, Look ahead carry generator

04 Multiplexer (MUX): MUX IC (74153, 74151), Cascading multiplexer

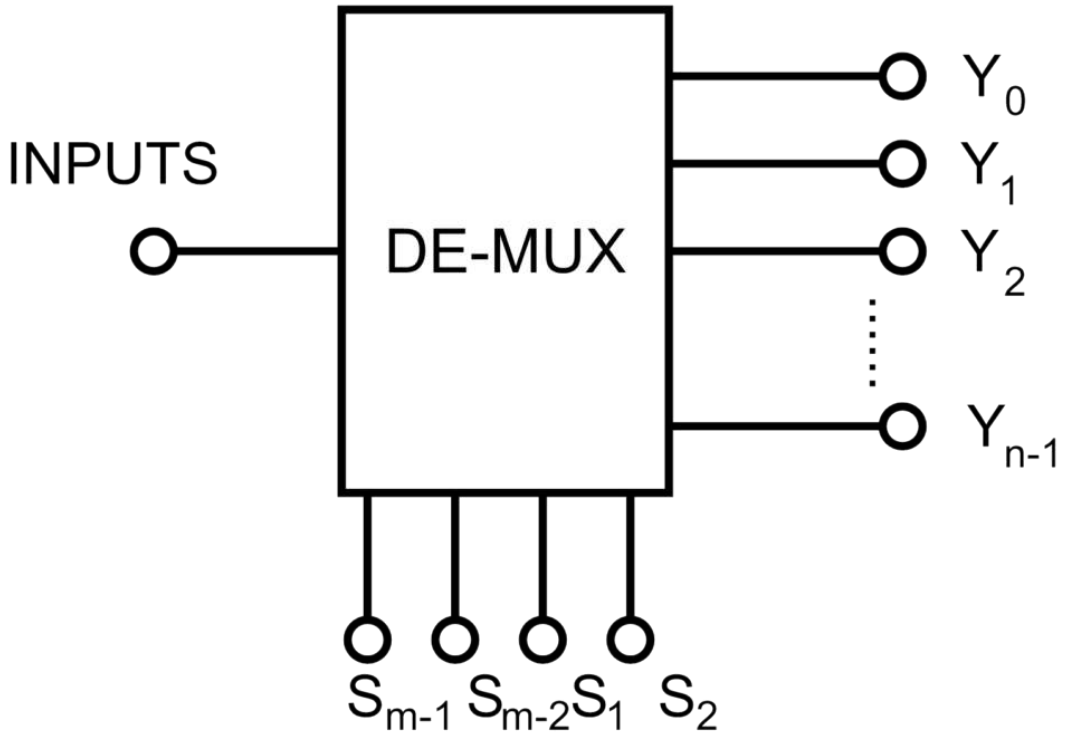
05 Demultiplexer (DEMUX)- Decoder (IC 74138, 74154), Implementation of SOP and POS using MUX , DEMUX

06 Comparators (2 bit), Parity Generators and Checkers.

Demultiplexer

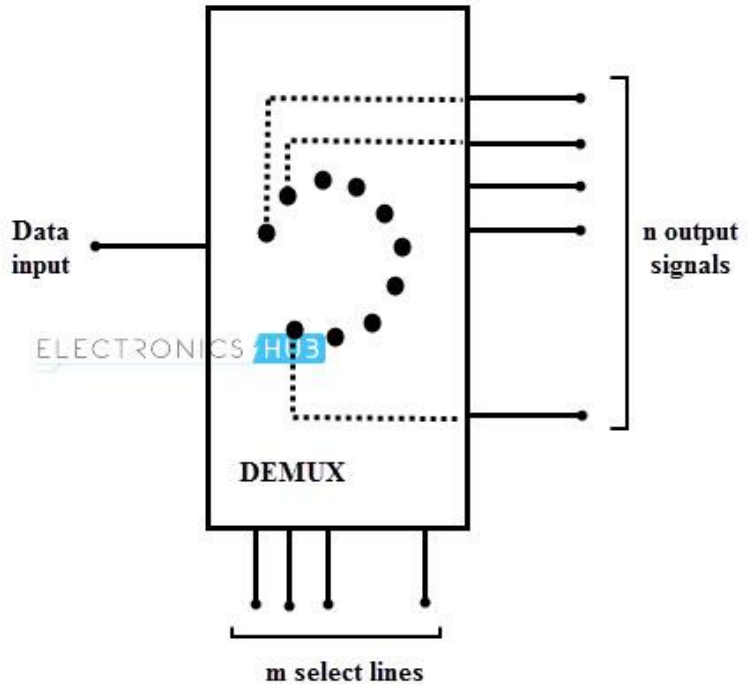
• **De-Multiplexer** is a combinational circuit that performs the reverse operation of Multiplexer. It has single input, ‘m’ selection lines and maximum of 2m outputs. The input will be connected to one of these outputs based on the values of selection lines.

• Since there are ‘m’ selection lines, there will be 2^m possible combinations of zeros and ones. So, each combination can select only one output. De-Multiplexer is also called as **De-Mux**



De-multiplexer (DEMUX)

- In digital electronics, demultiplexers are also known as data distributor because they can "distribute" a input to each output line.
- Enable input (E) is active low input used for cascading.



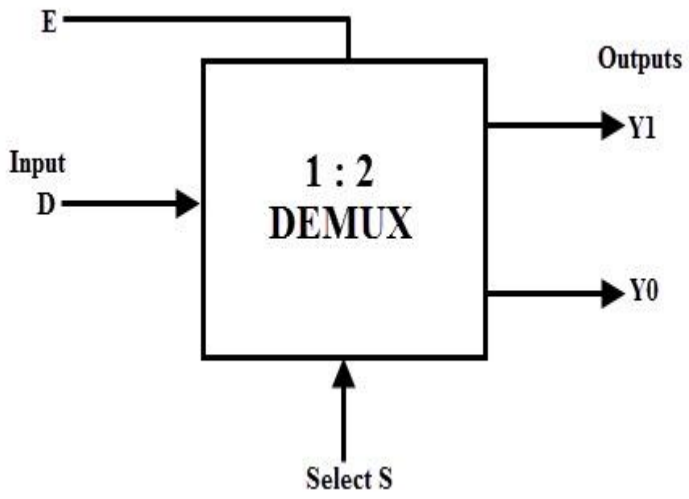
Types of demultiplexer:

1. 1:2 demultiplexer
2. 1:4 demultiplexer
3. 1:8 demultiplexer
4. 1:16 demultiplexer
5. 1:32 demultilexer

1:2 DEMUX

- A 1:2 demultiplexer consists of one input line, two output lines and one select line. The signal on the select line helps to switch the input to one of the two outputs.
- There are only two possible ways to connect the input to output lines, thus only one select signal is enough to do the demultiplexing operation. When the select input is low, then the input will be passed to Y0 and if the select input is high then the input will be passed to Y1.

Block Diagram:



Function Table:

E	S	Y1	Y0
0	X	0	0
1	0	0	D
1	1	D	0

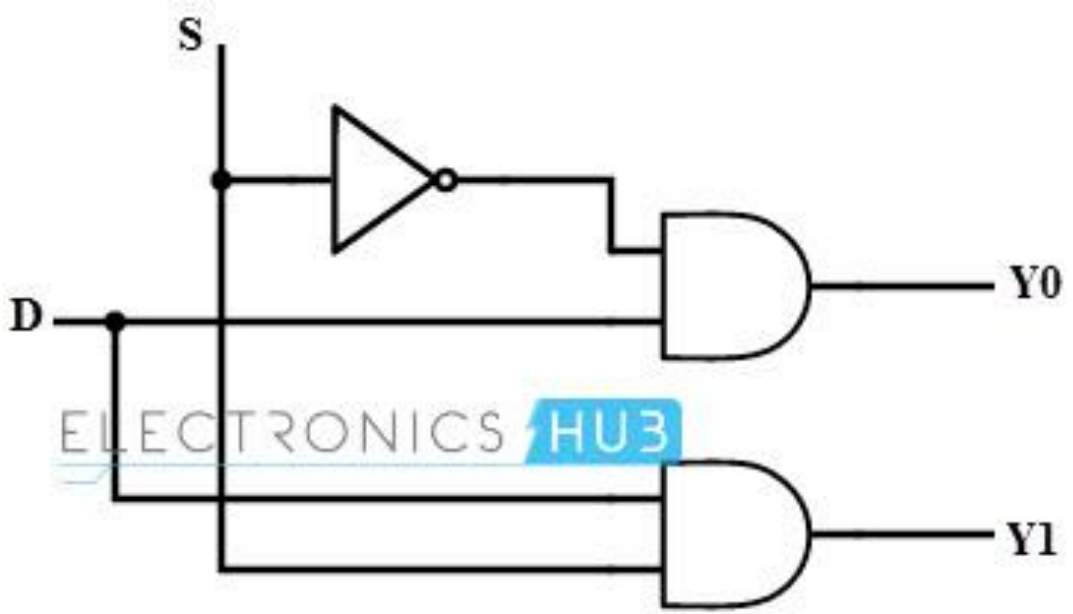
Truth Table:

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

Y0 = ES'D
Y1 = ESD

1:2 DEMUX

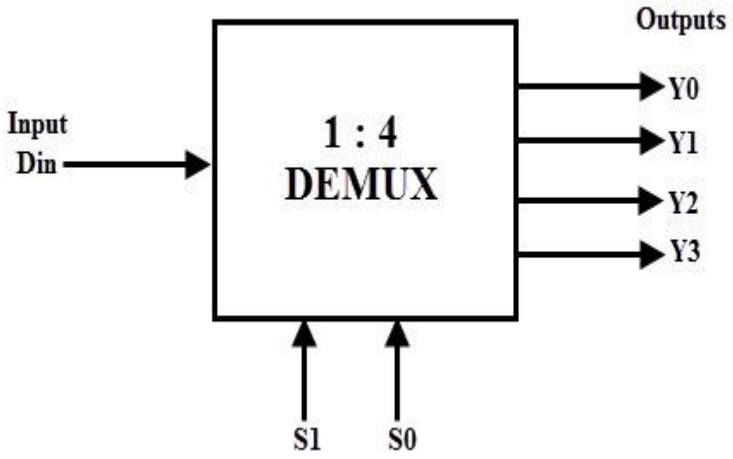
- **Logic Realization of 1:2 DEMUX**



1:4 DEMUX

- A 1-to-4 demultiplexer has a single input (D), two selection lines (S1 and S0) and four outputs (Y0 to Y3). The input data goes to any one of the four outputs at a given time for a particular combination of select lines.
- This demultiplexer is also called as a 2-to-4 demultiplexer which means that two select lines and 4 output lines.

Block Diagram:



Function Table:

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

1:4 DEMUX

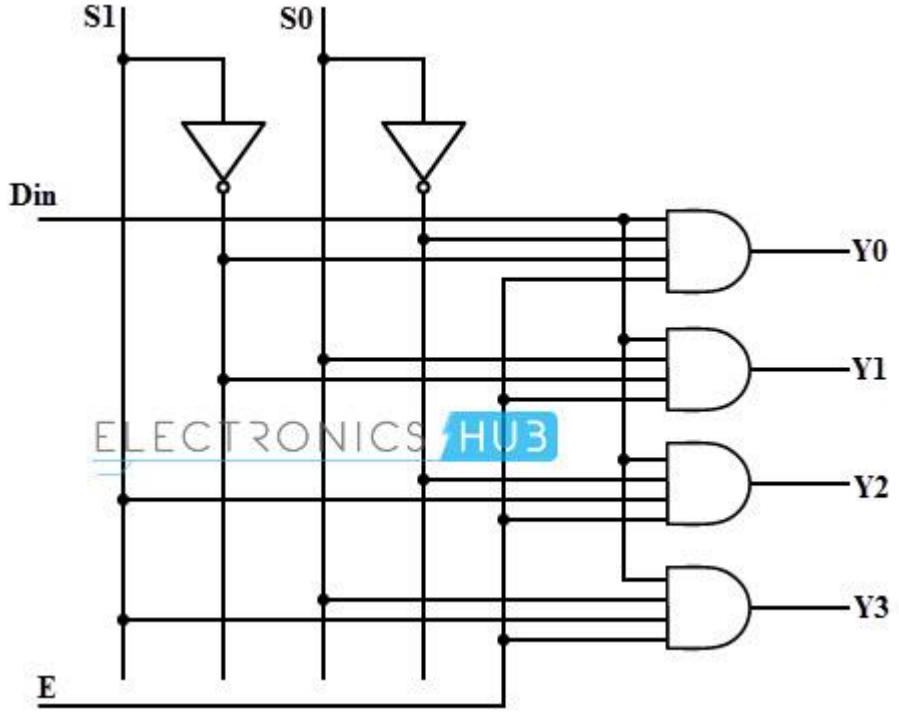
- **Logic Realization of 1:2 DEMUX**

$$Y0 = \overline{S1}\ \overline{S0}\ D$$

$$Y1 = \overline{S1}\ S0\ D$$

$$Y2 = S1\ \overline{S0}\ D$$

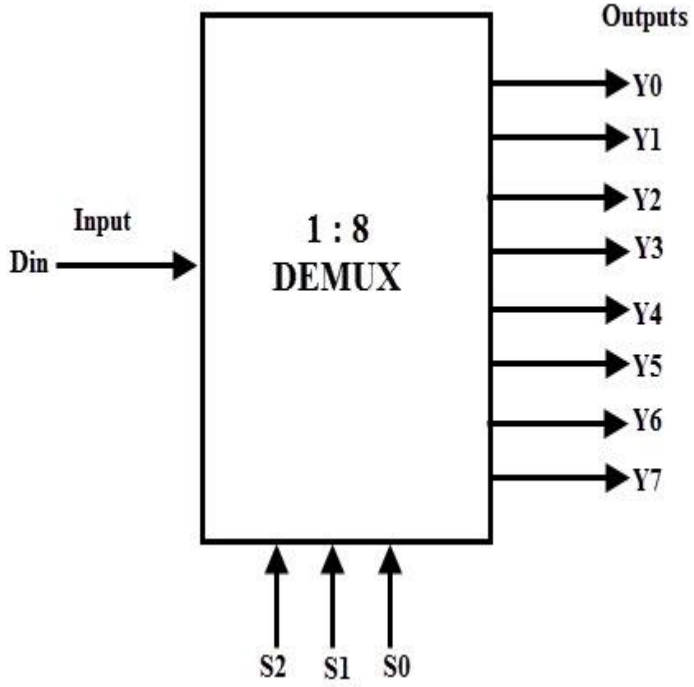
$$Y3 = S1\ S0\ D$$



1:8 DEMUX

- A1-to-8 demultiplexer that consists of single input D, three select inputs S2, S1 and S0 and eight outputs from Y0 to Y7.
- It is also called as 3-to-8 demultiplexer due to three select input lines. It distributes one input line to one of 8 output lines depending on the combination of select inputs.

Block Diagram:



Function Table:

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

1:8 DEMUX

- **Logic Realization of 1:2 DEMUX**

$Y0 = D\,\overline{S2}\,\overline{S1}\,\overline{S0}$

$Y1 = D\,\overline{S2}\,\overline{S1}\,S0$

$Y2 = D\,\overline{S2}\,S1\,\overline{S0}$

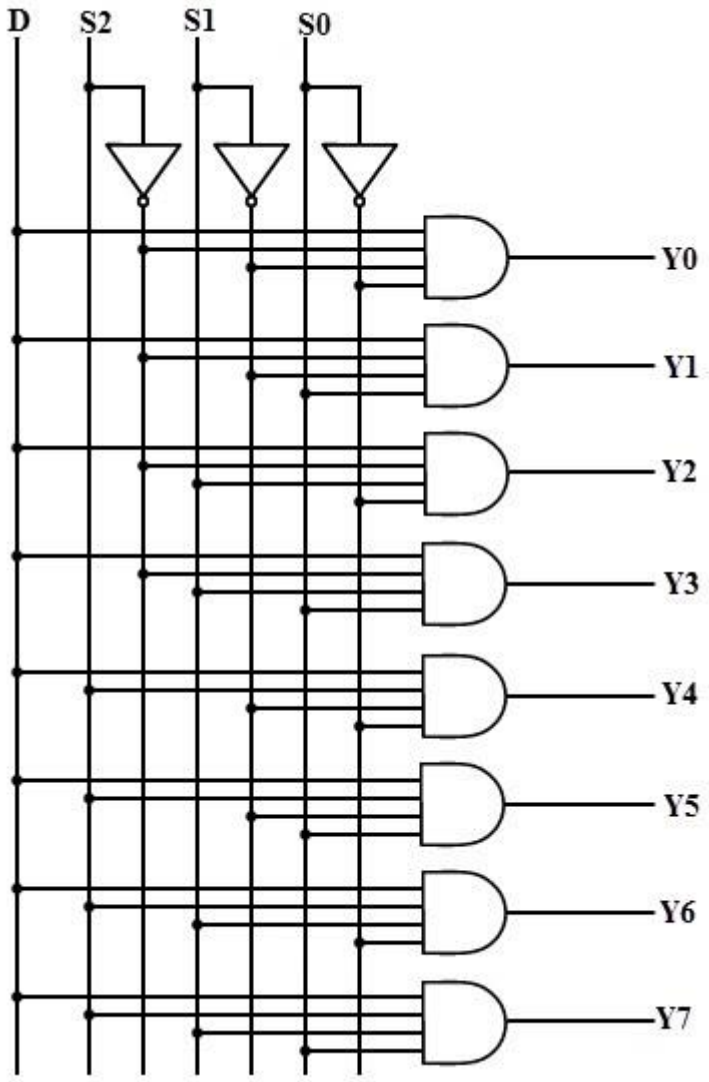
$Y3 = D\,\overline{S2}\,S1\,S0$

$Y4 = D\,S2\,\overline{S1}\,\overline{S0}$

$Y5 = D\,S2\,\overline{S1}\,S0$

$Y6 = D\,S2\,S1\,\overline{S0}$

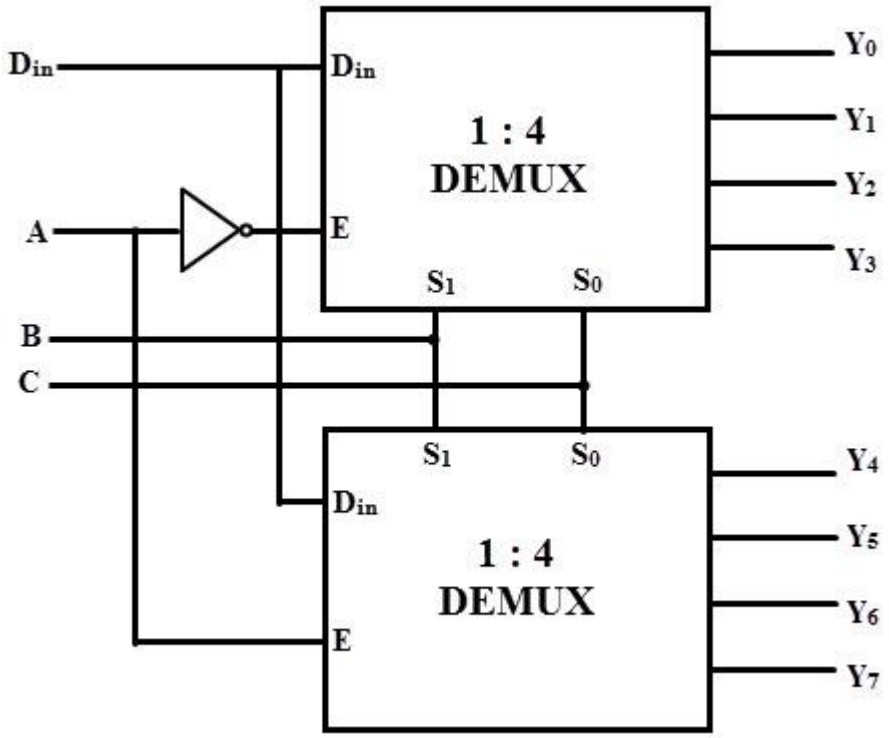
$Y7 = D\,S2\,S1\,S0$



DEMUX Tree

• When the application requires a large demultiplexer with more number of output pins, then we cannot implement by a single integrated circuit. In case if more than 16 output pins are needed, then two or more demultiplexer ICs are cascaded to fulfill the requirement.

• **E.g. 1:8 DEMUX using Two 1:4 Demultiplexers**



Example

- **Implementation of Full Subtractor Using 1:8 DEMUX**

Solution:

Consider the case for implementing a demultiplexer circuit in order to produce the full subtractor output. The truth table below shows the output of a full subtractor.

A	B	B_{in}	D	B_{out}
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

From the above table, the full subtractor output D can be written as

$$D = f(A, B, C) = \sum m(1, 2, 4, 7)$$

And the borrow output can be expressed as

$$Bout = F(A, B, C) = \sum m(1, 2, 3, 7)$$

From these Boolean functions, a demultiplexer for producing full subtractor output can be built by properly configuring the 1-to-8 DEMUX such that with input D=1 it gives the minterms at the output.

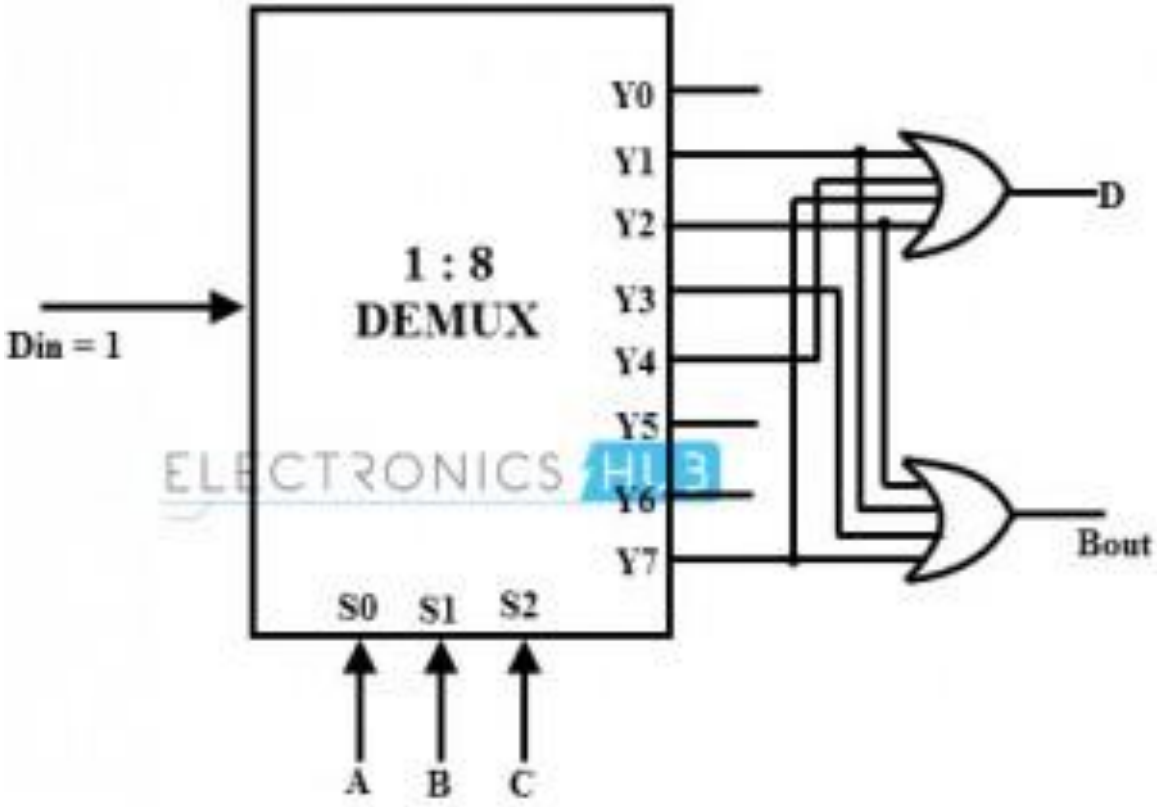
And by logically ORing these minterms, the outputs of difference and borrow can be obtained as shown in figure.

Example

D = f (A, B, C)= ∑m (1, 2, 4, 7)

And the borrow output can be expressed as

Bout = F (A, B, C) = ∑m (1, 2, 3, 7)



Applications of Demultiplexer

- Since the demultiplexers are used to select or enable the one signal out of many, these are extensively used in microprocessor or computer control systems such as

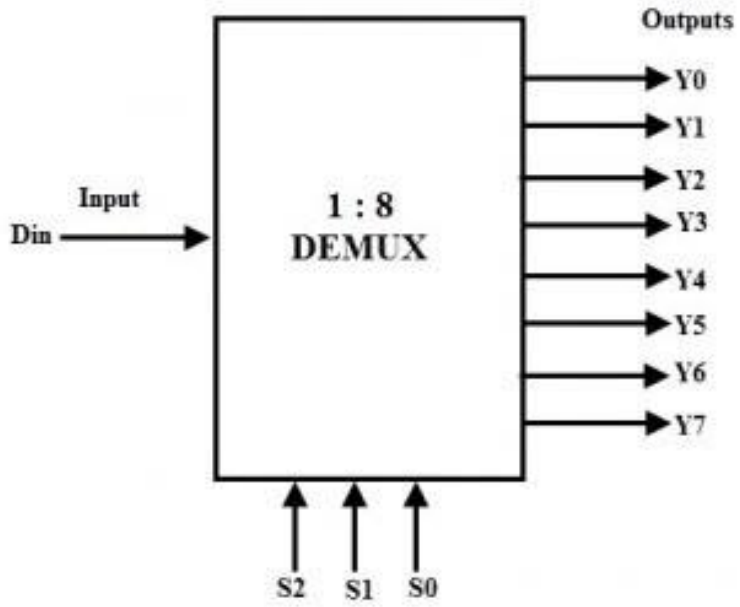
1. Selecting different IO devices for data transfer
2. Choosing different banks of memory
3. Depends on the address, enabling different rows of memory chips
4. Enabling different functional units.

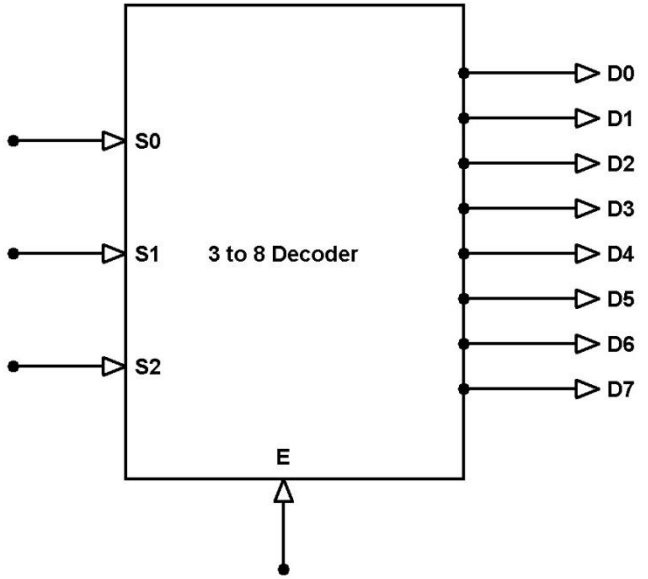
- Other than these, demultiplexers can be found in a wide variety of application such as

1. Synchronous data transmission systems
2. Boolean function implementation (as we discussed full subtractor function above)
3. Data acquisition systems
4. Combinational circuit design
5. Automatic test equipment systems
6. Security monitoring systems (for selecting a particular surveillance camera at a time), etc.

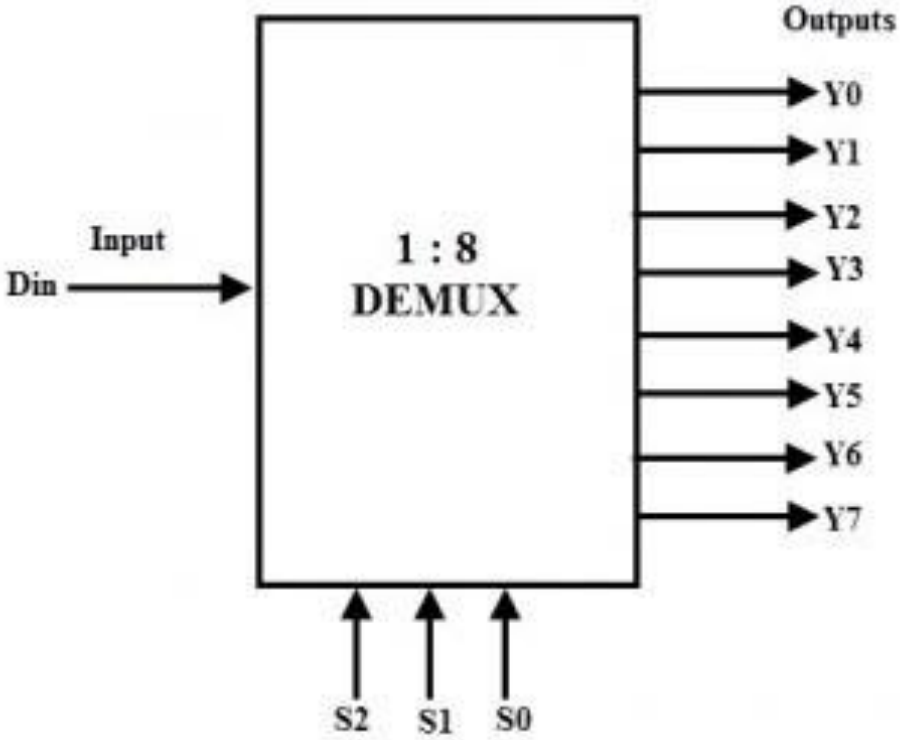
DEMUX as Decoder

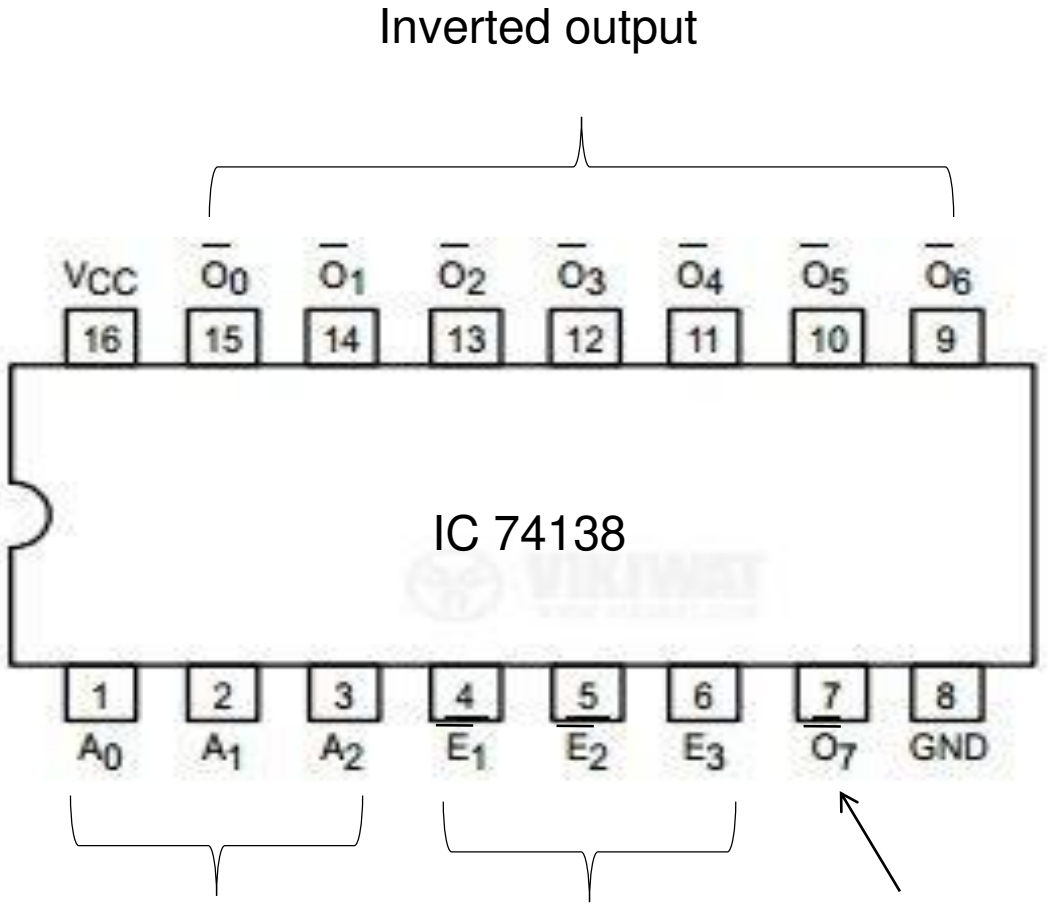
- Demultiplexer is equivalent with decoder..
- Select input consider as input for decoders.
- Keep Din open.
- **1:8 Demux is equivalent with 3 to 8 line decoder**.





DEMUX IC 74138





Agenda

01 Code Converter: BCD, Excess-3, Gray, Binary Code

02 Half Adder, Full Adder, Half Subtractor, Full Subtractor

03 Binary Adder (IC 7483), BCD Adder, Look ahead carry generator

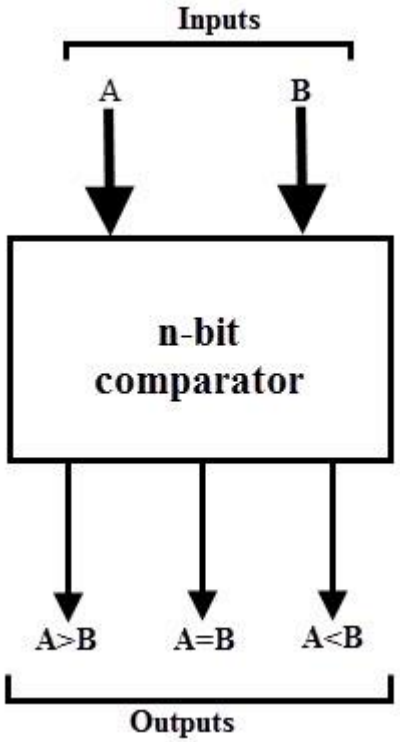
04 Multiplexer (MUX): MUX IC (74153, 74151), Cascading multiplexer

05 Demultiplexer (DEMUX)- Decoder (IC 74138, 74154), Implementation of SOP and POS using MUX , DEMUX

06 Comparators (2 bit), Parity Generators and Checkers.

Comparator

- Comparator is combinational circuit. It has 2 input A and B with three outputs as A>B, A=B and A<B.
- Depending on the magnitude of the two numbers one of the outputs will be high.



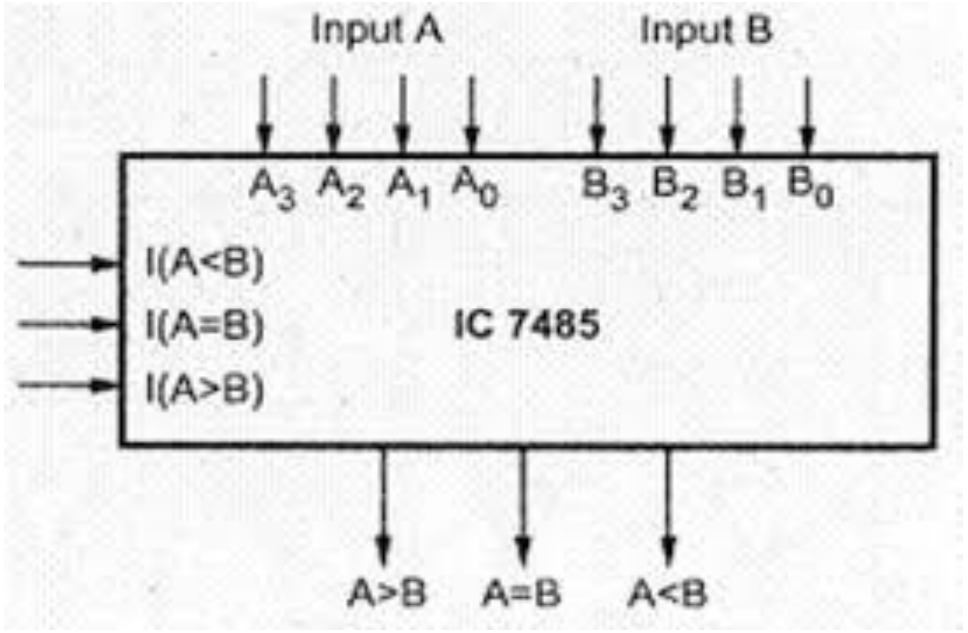
- These comparators can compare 2-bit, 4-bit and 8-bit numbers depending on the application requirement. These are available in TTL as well as CMOS logic family ICs and some of these ICs include IC 7485 (4-bit comparator), IC 4585 (4-bit comparator in CMOS family) and IC 74AS885 (8-bit comparator).

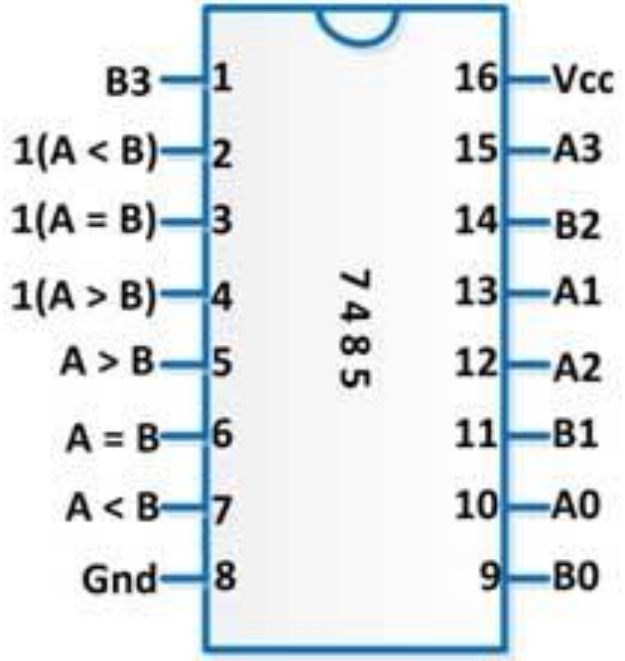
Comparator IC 7485

• **4-bit Magnitude Comparator**

The 4-bit comparator is mostly available in IC form and common type of this IC is 7485. This IC can be used to compare two 4-bit binary words by grounding I (A>B), I (A<B) and I (A=B) connector inputs to Vcc terminal.

In addition to the normal comparator, this IC is provided with cascading inputs in order to facilitate the cascading several comparators. Any number of bits can be compared by cascading several of these comparator ICs.



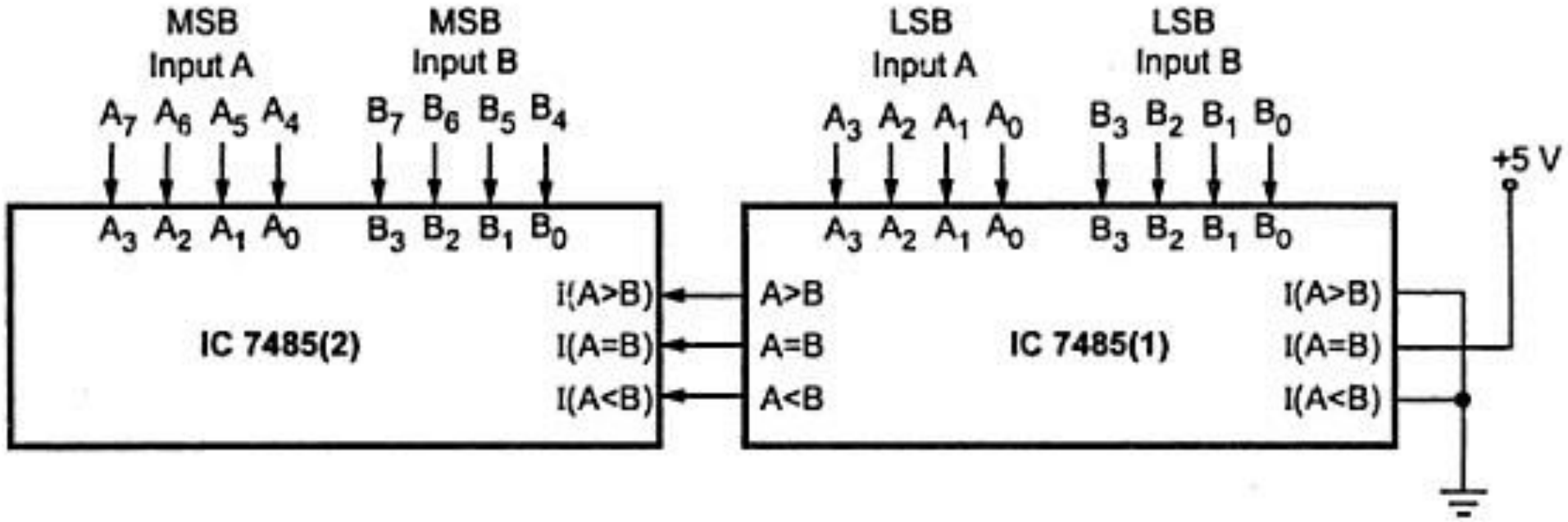


Comparator IC 7485

• **8-bit Magnitude Comparator**

• An 8-bit comparator compares the two 8-bit numbers by cascading of two 4-bit comparators. The circuit connection of this comparator is shown below in which the lower order comparator A<B, A=B and A>B outputs are connected to the respective cascade inputs of the higher order comparator.

• For the lower order comparator, the A=B cascade input must be connected High, while the other two cascading inputs A ,B must be connected to LOW. The outputs of the higher order comparator become the outputs of this eight-bit comparator.

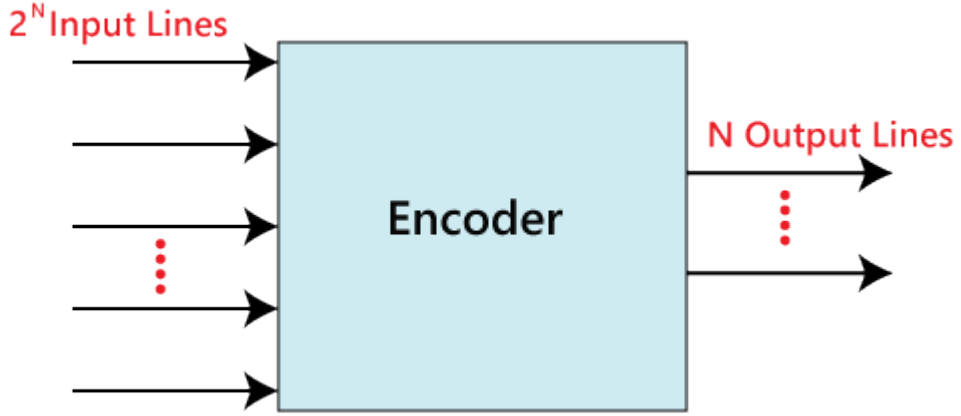


Applications of Comparators

1. Comparators are used in central processing units (CPUs) and microcontrollers (MCUs).
2. These are used in control applications in which the binary numbers representing physical variables such as temperature, position, etc. are compared with a reference value.
3. Comparators are also used as process controllers and for Servo motor control.
4. Used in password verification and biometric applications.

Encoders

1. The combinational circuits that change the binary information into N output lines are known as **Encoders**. The binary information is passed in the form of 2^N input lines. The output lines define the N-bit code for the binary information.
2. In simple words, the **Encoder** performs the reverse operation of the **Decoder**. At a time, only one input line is activated for simplicity. The produced N-bit output code is equivalent to the binary information.

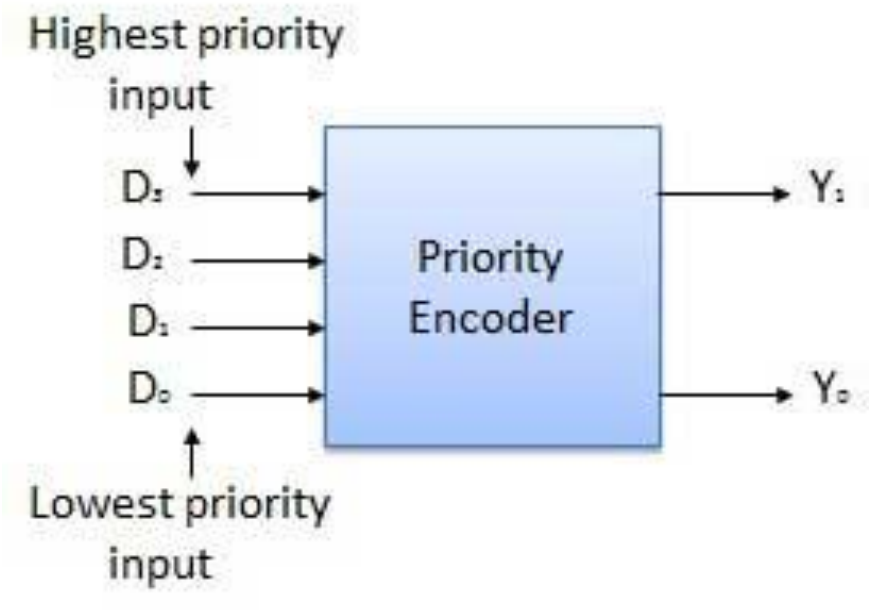


3. Types of Encoders

- Decimal to BCD Encoder
- Octal to Binary Encoder
- **Priority Encoder**
- Hexadecimal to binary encoder

Priority Encoder

- In this priority encoder, there are total of 4 inputs, i.e., D_0, D_1, D_2, and D_3, and two outputs, i.e., Y_0 and Y_1. The D_3 has high and D_0 has low priority inputs.
- When more than one input is '1' at the same time, the output will be the (binary) code corresponding to the higher priority input. Below is the truth table of the 4 to 2 line priority encoder.



Inputs				Outputs		
D_0	D_1	D_2	D_3	Y_1	Y_0	V
0	0	0	0	×	×	0
1	0	0	0	0	0	1
×	1	0	0	0	1	1
×	×	1	0	1	0	1
×	×	×	1	1	1	1

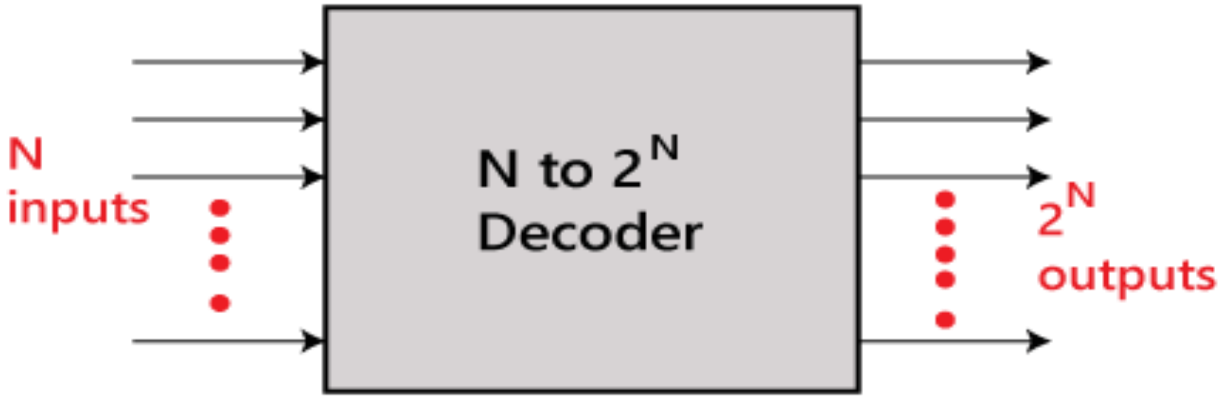
Applications of Encoders

- Uses of Encoders:

1. These systems are very easy to use in all digital systems.
2. Encoders are used to convert a decimal number into the binary number. The objective is to perform a binary operation such as addition, subtraction, multiplication, etc.
3. Email
4. Video encoders
5. Communication system and networking

Decoders

- The combinational circuit that change the binary information into 2^N output lines is known as **Decoders.** The binary information is passed in the form of N input lines.
- The output lines define the 2^N-bit code for the binary information. In simple words, the **Decoder** performs the reverse operation of the **Encoder**. At a time, only one input line is activated for simplicity. The produced 2^N-bit output code is equivalent to the binary information.

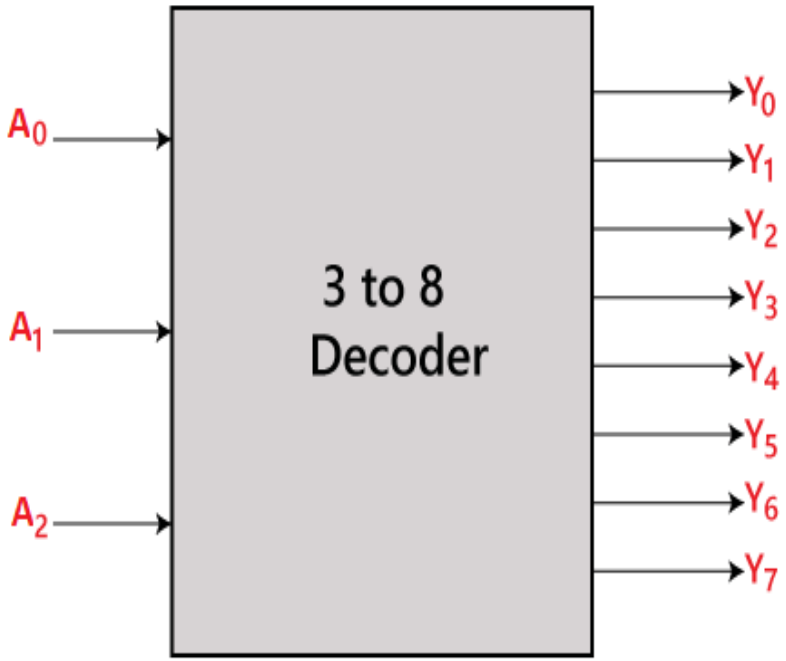


- Types of decoder

1. 2 to 4 line decoder
2. 3 to 8 line decoder
3. 4 to 16 line Decoder

3 to 8 line decoder

- The 3 to 8 line decoder is also known as Binary to Octal Decoder. In a 3 to 8 line decoder, there is a total of eight outputs, i.e., Y0, Y1, Y2, Y3, Y4, Y5, Y6, and Y7 and three outputs, i.e., A0, A1, and A2. This circuit has an enable input 'E'. Just like 2 to 4 line decoder, when enable 'E' is set to 1, one of these four outputs will be 1.



Enable	INPUTS			Outputs							
E	A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0