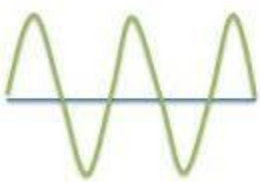
Unit-1 Review of Fundamentals

Number System

- The study of *number systems* is important from the viewpoint of understanding how data are represented before they can be processed by any digital system including a digital computer.
- It is one of the most basic topics in digital electronics.

Analog Versus Digital

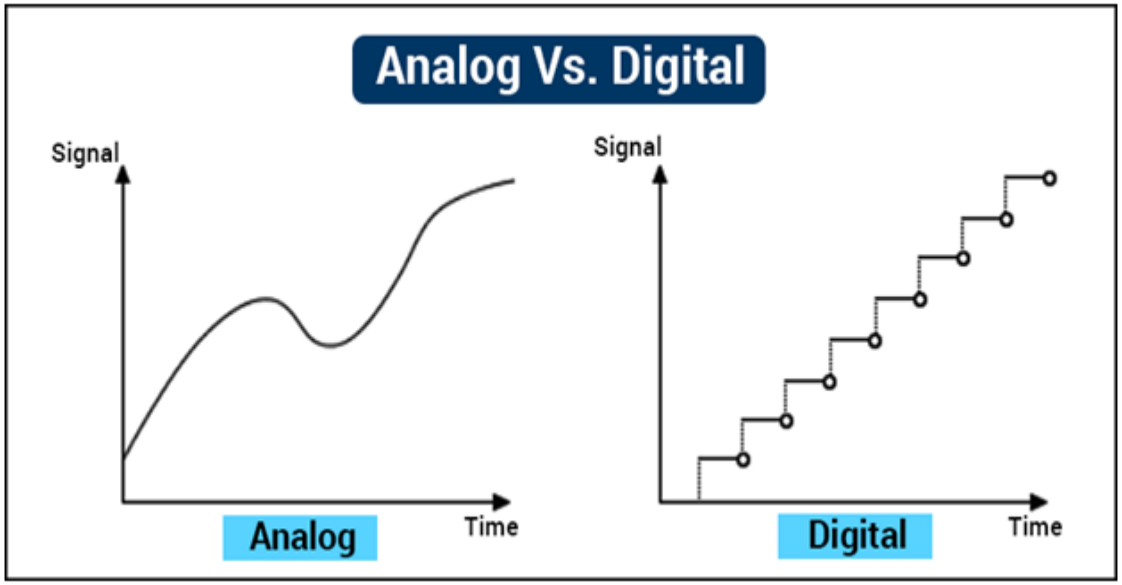


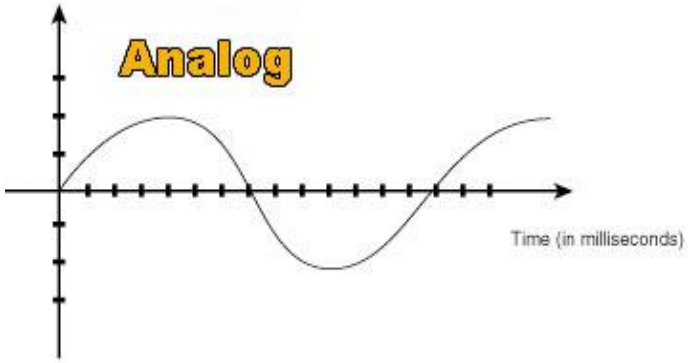
Vs

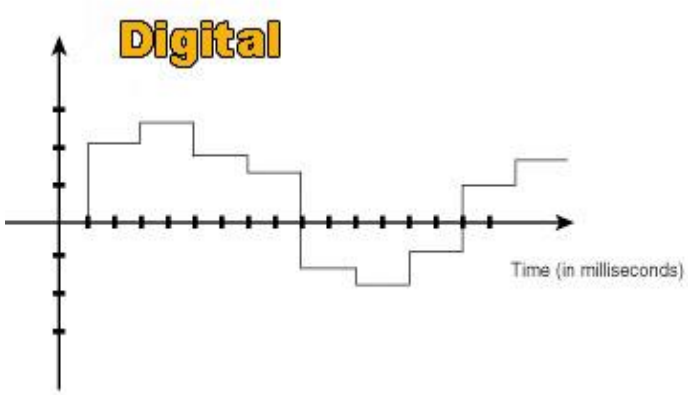
0100111101

Analog Signal

Digital Signal







Generation of Microprocessors

- **First Generation (1971 - 1973)**

The microprocessors that were introduced from 1971 -1973 were referred to as the first generation systems. First generation microprocessors processed their instructions serially. The fetched the instruction, decoded it, and then executed it. The first microprocessor, the 4004, was introduced in 1971. It was co-developed by Busicom, a Japanese manufacturers of calculators, and Intel, a US manufacturer of semiconductors. They were fabricated using p-channel metal oxide semiconductor PMOS, technology which provided low cost low speed and low output currents. They were not compatible to TTL. In 1972, Intel made the 8- bit 8008 and 8080 microprocessors.

- **Second Generation (1974 – 1978)**

As the technology evolved, the number of circuits that could be fabricated on a chip grew. Very large scale integration (VLSI) led to the chips that had speeds upto hundreds of millions of switching per second. The second generation marked the beginning of very efficient 8-bit microprocessors. Some of the popular processors were Motorola's 6800 and 6809. Intel's 8085 and zilog's Z80. The second generation devices marked a sharp contrast with the use of newer semiconductor technology to fabricate chips. They were manufactured using n- channel metal oxide semiconductor (NMOS) technology. This technology offered faster speed and higher density than PMOS. It resulted in a 5-fold increase in instruction execution speed and higher chip densities.

Third Generation (1978 – 1980)

- The third generation introduced in 1978, was dominated by Intel's 8086 and Zilog's Z8000, which were 16- bit processors with micro computer like performance. These processors had the technology of 16-bit arithmetic and pipelined instruction processor. The third generation came with IC transistor counts of about 250,000. It was designed using high density metal oxide semiconductor (HMOS) technology. HMOS provides some advantages over NMOS. Its speed – power product is 4 times better than that of NMOS. It can accommodate twice the circuit density of NMOS.

Fourth Generation (1981 - 1995)

- The microprocessors entered their fourth generation with designs containing more than a million transistors in a single packet. This era marked the beginning of 32 – bit processors. Intel introduced 80386and Motorola introduced 68020/68030. They were fabricated using high density/high speed complementary metal oxide semiconductor (HCMOS), a low-power version of the HMOS technology.

Fifth Generation (1995 – till date)

- The fifth generation microprocessors employ decoupled super scalar processing and their design contains more than 10 million transistors. This generation marks the introduction of devices that carry on-chip functionalities. It has also paved the way for high speed memory I/O devices along with the introduction 64- bit microprocessors. Intel leads the show here with Pentium, Celeron and a very recently, dual and quad core processors working with upto 3.5 GHz speed. This generation is categorised by a low margin single processors PC business, which is complemented by high volume sales. The table1 gives the comparison of major processors based on specific parameters such as a clock speed and data word size.

Different Generation of Microprocessors

General purpose processors	Transistors	CPU speed	Data length (bits)
8080	6000	2MHz	8
8085	6500	3MHz	8
8088	29000	3MHz	8
8086	30000	4MHz	16
80286	134000	6MHz	16
80386	275000	16MHz	16/32
80486	1200000	33MHz	16/32
ATHALON XP	3700000	2.8GHz	16/32/64
CELERON	7500000	1.06-2GHz	32
PENTIUM II	7500000	233-450MHz	32
PENTIUM III	9500000	450Mhz-1GHz	32
PENTIUM III XEON	28100000	500MHz-1GHz	32
PENTIUM IV	55000000	1.4-2.22GHZ	32
IBM power PC G3	6500000	233-333Mhz	32
Power PC G4	10500000	400-800MHz	32

Types of Number System

- Decimal Number System- Base 10

  (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

  Example : $(519)_{10}$

- Binary number system – Base -2

  (0, 1)

  Example : $(1010)_{2}$

- Octal Number System- Base 8

  (0, 1, 2, 3, 4, 5, 6, 7)

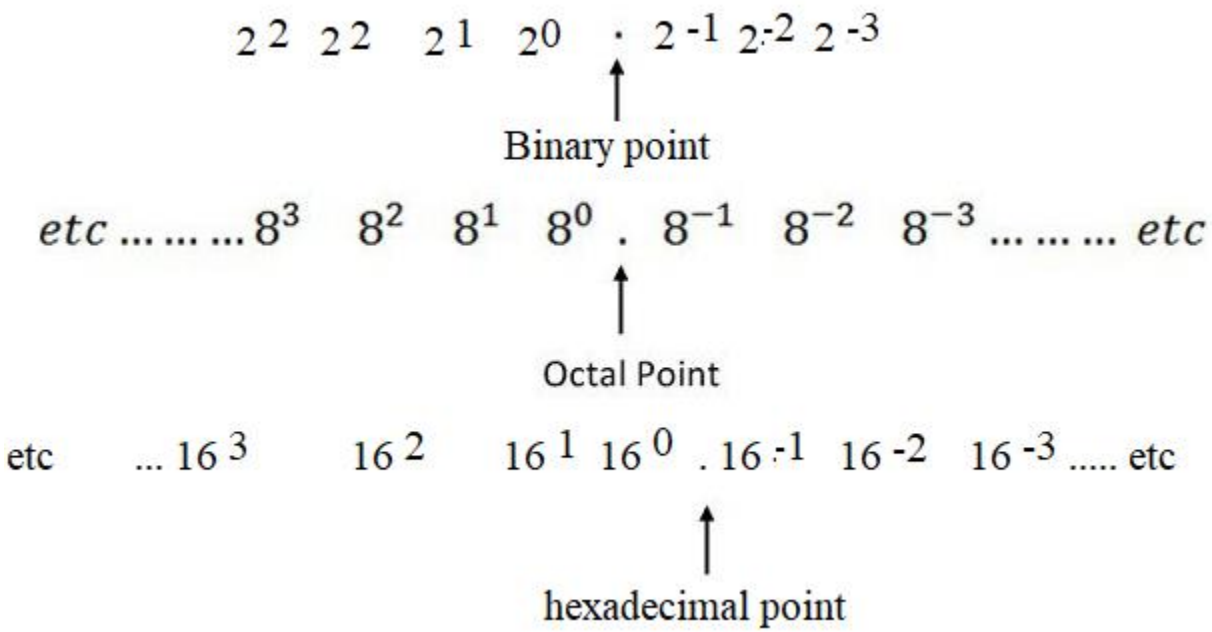
  Example : $(527)_{8}$

- Hexadecimal Number System, Base -16

  (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

  Example : $(A123)_{16}$

Important: Radix or Base is a term used to describe the **number** of digits utilized in a positional **number** system before "rolling over" to the next digit's place



- Similarly For Decimal Number

As an illustration, in the case of the decimal number 3586.265, the integer part (i.e. 3586) can be expressed as

$$3586 = 6 \times 10^0 + 8 \times 10^1 + 5 \times 10^2 + 3 \times 10^3 = 6 + 80 + 500 + 3000 = 3586$$

and the fractional part can be expressed as

$$265 = 2 \times 10^{-1} + 6 \times 10^{-2} + 5 \times 10^{-3} = 0.2 + 0.06 + 0.005 = 0.265$$

Decimal to Binary Convesion

Example 1.3

We will find the binary equivalent of $(13.375)_{10}$.

Solution

- The integer part = 13

Divisor	Dividend	Remainder
2	13	—
2	6	1
2	3	0
2	1	1
—	0	1

- The binary equivalent of $(13)_{10}$ is therefore $(1101)_2$
- The fractional part = .375
- $0.375 \times 2 = 0.75$ with a carry of 0
- $0.75 \times 2 = 0.5$ with a carry of 1
- $0.5 \times 2 = 0$ with a carry of 1
- The binary equivalent of $(0.375)_{10} = (.011)_2$
- Therefore, the binary equivalent of $(13.375)_{10} = (1101.011)_2$

Decimal to Octal Conversion

Example 1.4

We will find the octal equivalent of $(73.75)_{10}$.

Solution

- The integer part = 73

Divisor	Dividend	Remainder
8	73	—
8	9	1
8	1	1
—	0	1

- The octal equivalent of $(73)_{10} = (111)_8$
- The fractional part = 0.75
- $0.75 \times 8 = 0$ with a carry of 6
- The octal equivalent of $(0.75)_{10} = (.6)_8$
- Therefore, the octal equivalent of $(73.75)_{10} = (111.6)_8$

Decimal-to-Hexadecimal Conversion

The process of decimal-to-hexadecimal conversion is also similar. Since the hexadecimal number system has a base of 16, the progressive division and multiplication factor in this case is 16. The process is illustrated further with the help of an example.

Example 1.5

Let us determine the hexadecimal equivalent of $(82.25)_{10}$.

Solution

- The integer part = 82

Divisor	Dividend	Remainder
16	82	—
16	5	2
—	0	5

- The hexadecimal equivalent of $(82)_{10} = (52)_{16}$
- The fractional part = 0.25
- $0.25 \times 16 = 0$ with a carry of 4
- Therefore, the hexadecimal equivalent of $(82.25)_{10} = (52.4)_{16}$

Binary-to-Decimal Conversion

The decimal equivalent of the binary number $(1001.0101)_2$ is determined as follows:

- The integer part = 1001
- The decimal equivalent = $1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 = 1 + 0 + 0 + 8 = 9$
- The fractional part = .0101
- Therefore, the decimal equivalent = $0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} = 0 + 0.25 + 0 + 0.0625 = 0.3125$
- Therefore, the decimal equivalent of $(1001.0101)_2 = 9.3125$

Octal-to-Decimal Conversion

The decimal equivalent of the octal number $(137.21)_8$ is determined as follows:

- The integer part = 137
- The decimal equivalent $= 7 \times 8^0 + 3 \times 8^1 + 1 \times 8^2 = 7 + 24 + 64 = 95$
- The fractional part = .21
- The decimal equivalent $= 2 \times 8^{-1} + 1 \times 8^{-2} = 0.265$
- Therefore, the decimal equivalent of $(137.21)_8 = (95.265)_{10}$

Hexadecimal-to-Decimal Conversion

The decimal equivalent of the hexadecimal number $(1E0.2A)_{16}$ is determined as follows:

- The integer part = 1E0
- The decimal equivalent $= 0 \times 16^0 + 14 \times 16^1 + 1 \times 16^2 = 0 + 224 + 256 = 480$
- The fractional part = 2A
- The decimal equivalent $= 2 \times 16^{-1} + 10 \times 16^{-2} = 0.164$
- Therefore, the decimal equivalent of $(1E0.2A)_{16} = (480.164)_{10}$

Example 1.2

Find the decimal equivalent of the following binary numbers expressed in the 2's complement format:

(a) 00001110;
(b) 10001110.

Solution

(a) The MSB bit is '0', which indicates a plus sign.
The magnitude bits are 0001110.

$$\text{The decimal equivalent} = 0\times2^0+1\times2^1+1\times2^2+1\times2^3+0\times2^4+0\times2^5+0\times2^6$$
$$= 0+2+4+8+0+0+0 = 14$$

Therefore, 00001110 represents +14

(b) The MSB bit is '1', which indicates a minus sign
The magnitude bits are therefore given by the 2's complement of 0001110, i.e. 1110010

$$\text{The decimal equivalent} = 0\times2^0+1\times2^1+0\times2^2+0\times2^3+1\times2^4+1\times2^5$$
$$+1\times2^6$$
$$= 0+2+0+0+16+32+64 = 114$$

Therefore, 10001110 represents −114

Binary–Octal and Octal–Binary Conversions

Example 1.6

Let us find the binary equivalent of $(374.26)_8$ and the octal equivalent of $(1110100.0100111)_2$

Solution

- The given octal number = $(374.26)_8$
- The binary equivalent = $(011\ 111\ 100.010\ 110)_2 = (011111100.010110)_2$

- Any 0s on the extreme left of the integer part and extreme right of the fractional part of the equivalent binary number should be omitted. Therefore, $(011111100.010110)_2 = (11111100.01011)_2$
- The given binary number = $(1110100.0100111)_2$
- $(1110100.0100111)_2 = (1\ 110\ 100.010\ 011\ 1)_2$
  $= (001\ 110\ 100.010\ 011\ 100)_2 = (164.234)_8$

Octal Symbol	Binary equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Decimal, Binary, Octal, Hexidecimal Values

Decimal	Binary	Octal	Hexidecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Hex–Binary and Binary–Hex Conversions

Example 1.7

Let us find the binary equivalent of $(17E.F6)_{16}$ *and the hex equivalent of* $(101100110.011011101)_2$

Solution

- The given hex number $= (17E.F6)_{16}$
- The binary equivalent $= (0001\ 0111\ 1110.1111\ 0110)_2$
  $= (000101111110.11110110)_2$
  $= (101111110.1111011)_2$
- The 0s on the extreme left of the integer part and on the extreme right of the fractional part have been omitted.
- The given binary number $= (101100110.011011101)_2$
  $= (10\ 1100\ 1110.0110\ 1110\ 1)_2$
- The hex equivalent $= (0010\ 1100\ 1110.0110\ 1110\ 1000)_2 = (2CE.6E8)_{16}$

Hex–Octal and Octal–Hex Conversions

- **Example 1.8**
- *Let us find the octal equivalent of $(2F.C4)_{16}$ and the hex equivalent of $(762.013)_8$*

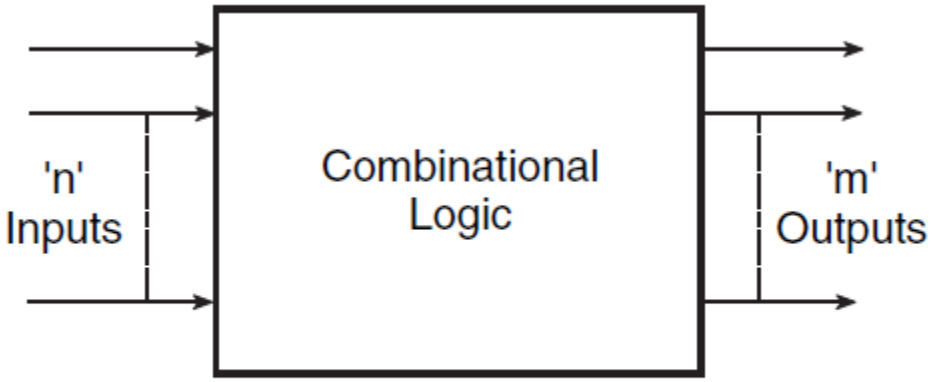
Solution

- The given hex number $= (2F.C4)_{16}$.
- The binary equivalent $= (0010\ 1111.1100\ 0100)_2 = (00101111.11000100)_2$
  $= (101111.110001)_2 = (101\ 111.110\ 001)_2 = (57.61)_8$.
- The given octal number $= (762.013)_8$.
- The octal number $= (762.013)_8 = (111\ 110\ 010.000\ 001\ 011)_2$
  $= (111110010.000001011)_2$
  $= (0001\ 1111\ 0010.0000\ 0101\ 1000)_2 = (1F2.058)_{16}$.

Combinational Circuits

- Output depends on present input only.
- A *combinational circuit* is one where the output at any time depends only on the present combination of inputs at that point of time with total disregard to the past state of the inputs.
- The logic gate is the most basic building block of combinational logic. The logical function performed by a combinational circuit is fully defined by a set of Boolean expressions.

$$Y_1 \text{ (OR output)} = A + B + C + D \quad \text{and} \quad Y_2 \text{ (NOR output)} = \overline{A + B + C + D}$$



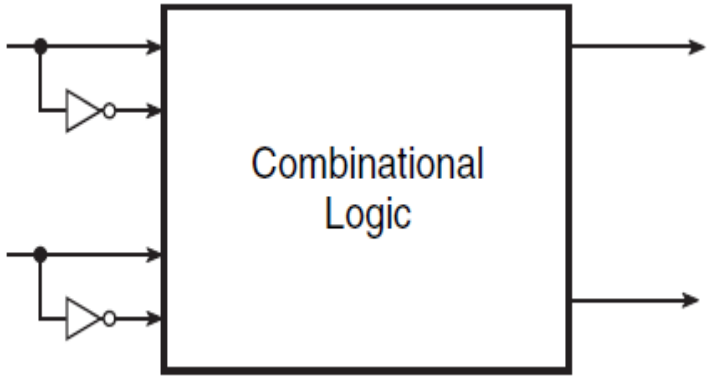
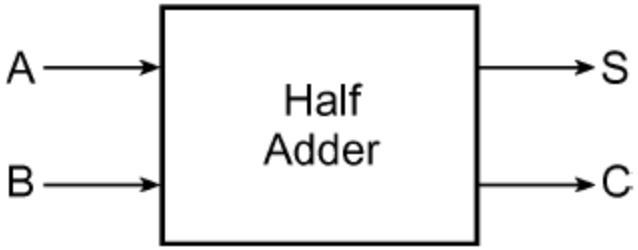


Figure Generalized combinational circuit.

Half Adder

- A *half-adder* is an arithmetic circuit block that can be used to add two bits.
- Such a circuit thus has two inputs that represent the two bits to be added and two outputs, with one producing the SUM output and the other producing the CARRY..



$$\text{SUM } S = A.\overline{B} + \overline{A}.B$$

$$\text{CARRY } C = A.B$$

Truth Table of Half Adder

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

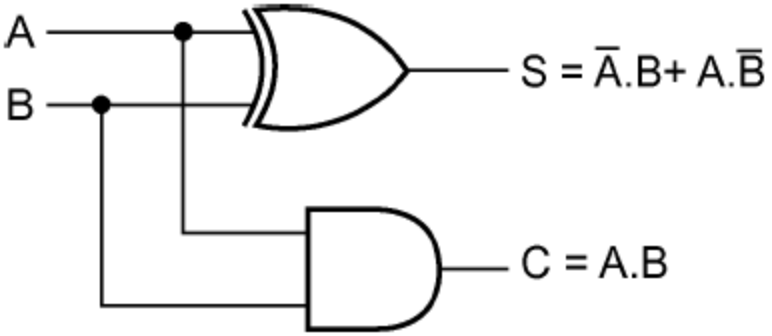
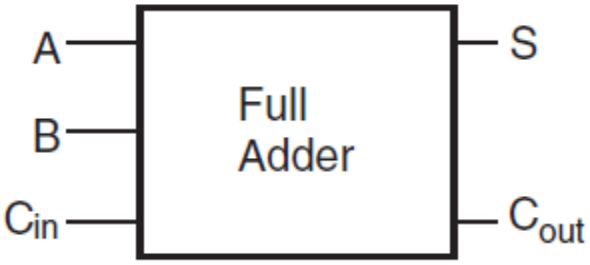


Figure Logic implementation of a half-adder

Full Adder

- A *full adder* circuit is an arithmetic circuit block that can be used to add three bits to produce a SUM and a CARRY output.
- Such a building block becomes a necessity when it comes to adding binary numbers with a large number of bits.
- The full adder circuit overcomes the limitation of the half-adder, which can be used to add two bits only.



Truth table of a full adder.

A	B	C_{in}	SUM (S)	C_{out}
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 expression and logic diagram

K-map simplification for carry and sum

For Carry (C_{out})

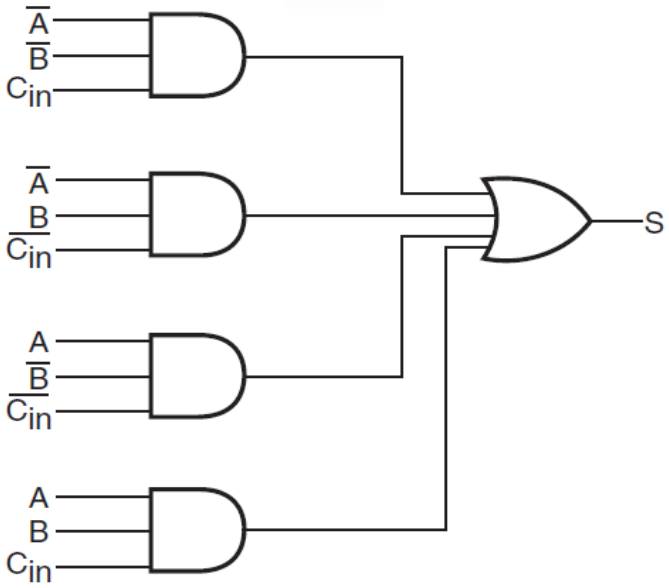
A \ BC_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

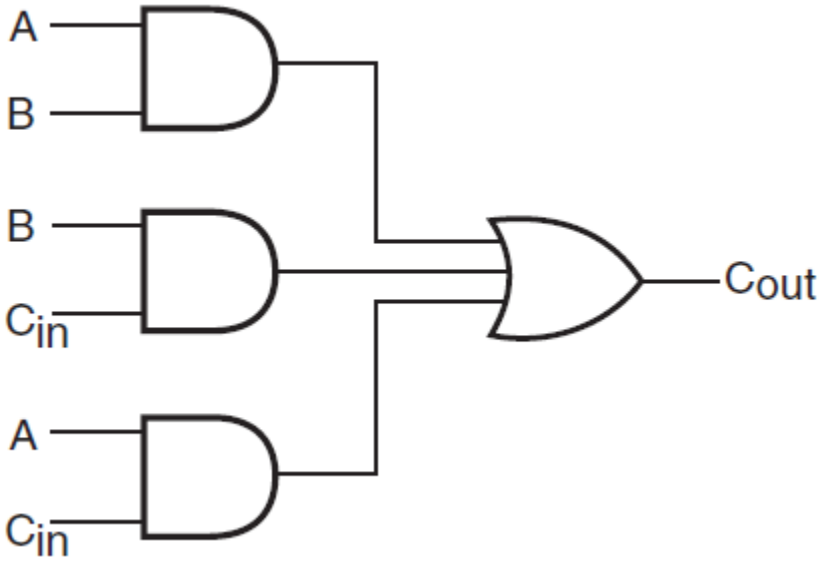
$C_{out} = AB + A\,C_{in} + B\,C_{in}$

For Sum

A \ BC_{in}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

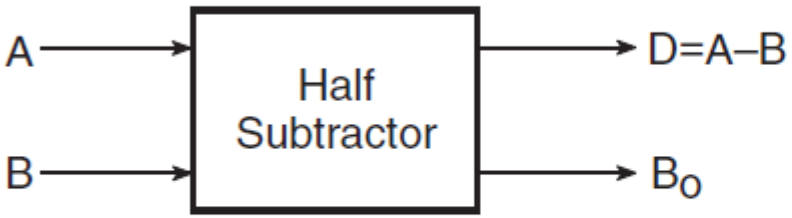
$Sum = \overline{A}\,\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + A\overline{B}\,\overline{C}_{in} + ABC_{in}$





Half-Subtractor

- A *half-subtractor* is a combinational circuit that can be used to subtract one binary digit from another to produce a DIFFERENCE output and a BORROW output.
- The BORROW output here specifies whether a ‘1’ has been borrowed to perform the subtraction.

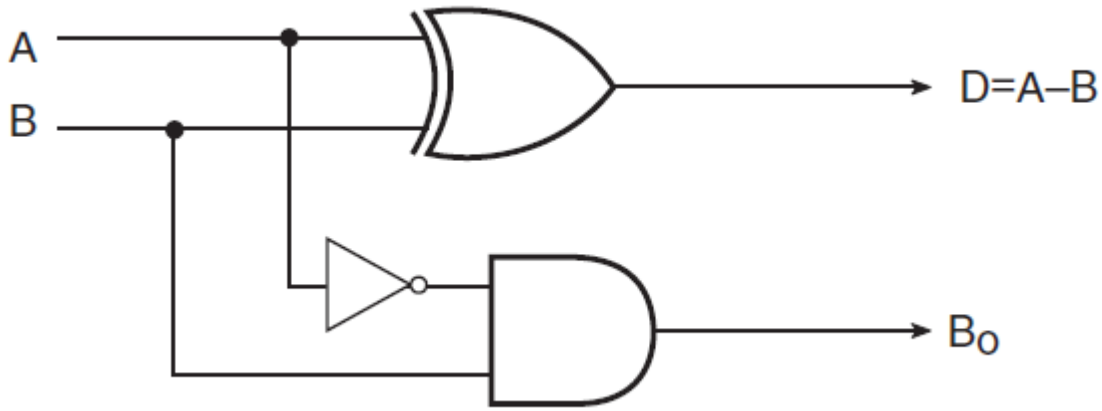


A	B	D	B_O
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

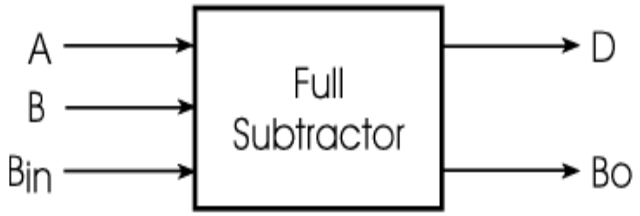
Figure Half-subtractor.

$$D = \overline{A}.B + A.\overline{B}$$

$$B_o = \overline{A}.B$$



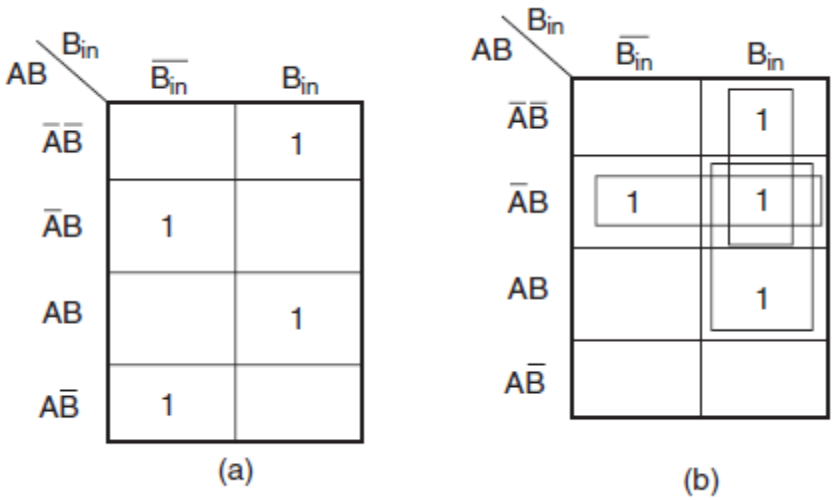
- A *full subtractor* performs subtraction operation on two bits, a minuend and a subtrahend, and also takes into consideration whether a ‘1’ has already been borrowed by the previous adjacent lower minuend bit or not.
- As a result, there are three bits to be handled at the input of a full subtractor, namely the two bits to be subtracted and a borrow bit designated as Bin .
- There are two outputs, namely the DIFFERENCE output D and the BORROW output Bo.
- The BORROW output bit tells whether the minuend bit needs to borrow a ‘1’ from the next possible higher minuend bit.



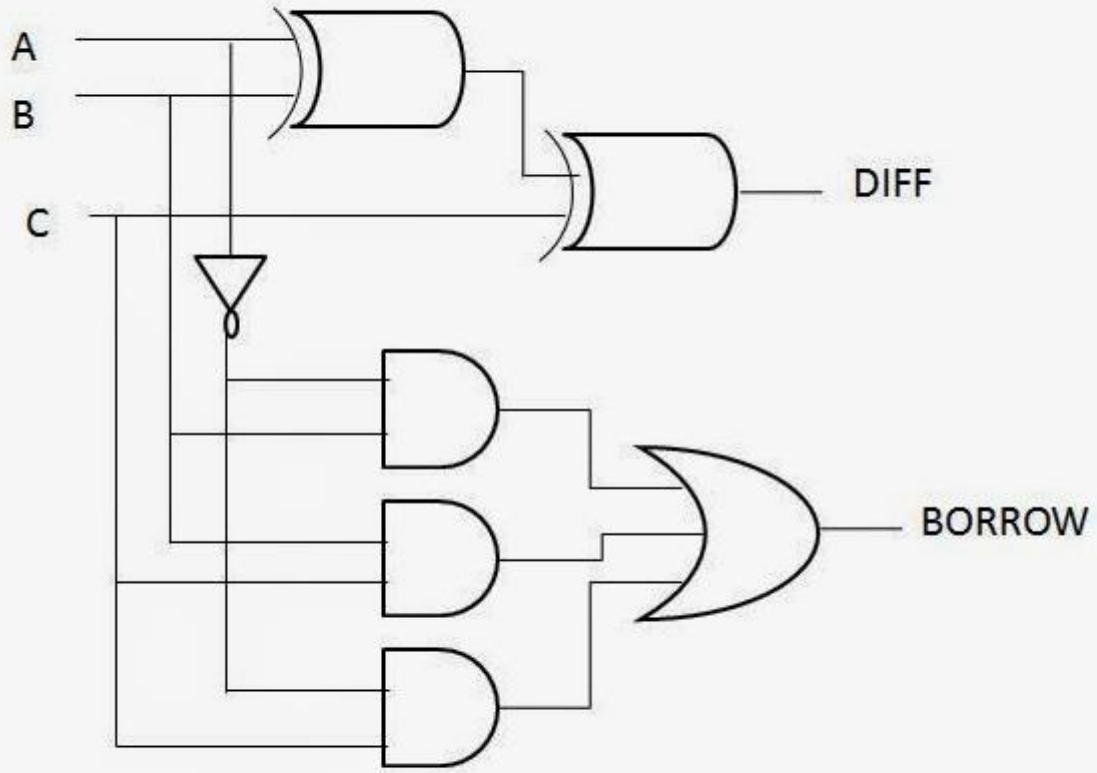
Minuend (A)	Subtrahend (B)	Borrow In (B_{in})	Difference (D)	Borrow Out (B_O)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$D = \overline{A}.\overline{B}.B_{in} + \overline{A}.B.\overline{B}_{in} + A.\overline{B}.\overline{B}_{in} + A.B.B_{in}$$

$$B_o = \overline{A}.B + \overline{A}.B_{in} + B.B_{in}$$


AB
B_{in}
$\overline{B_{in}}$
B_{in}
$\overline{A}\overline{B}$
$\overline{A}B$
AB
$A\overline{B}$
1
1
1
1
(a)
AB
B_{in}
$\overline{B_{in}}$
B_{in}
$\overline{A}\overline{B}$
$\overline{A}B$
AB
$A\overline{B}$
1
1
1
1
(b)

FULL SUBTRACTOR



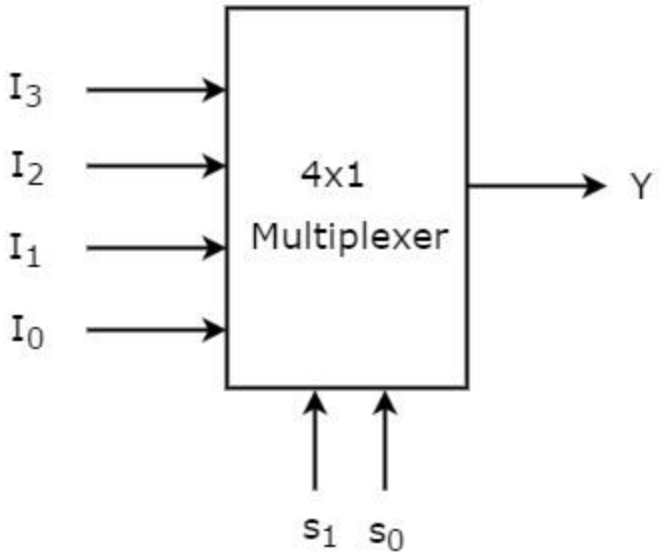
A	B	C	D	BO
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$DIFF = A \oplus B \oplus C$

$BORROW = A'.B + B.C + A'.C$

Multiplexer

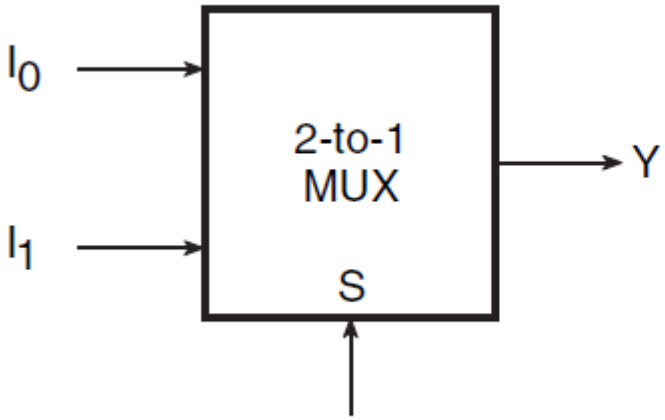
- A *multiplexer* or *MUX*, also called a *data selector*, is a combinational circuit with more than one input line, one output line and more than one selection line
- 2 x 1 Mux
- 4 x 1 Mux
- 8 x 1 Mux
- 16 x 1 Mux



2^n inputs and 1 output and

n = No if slection lines

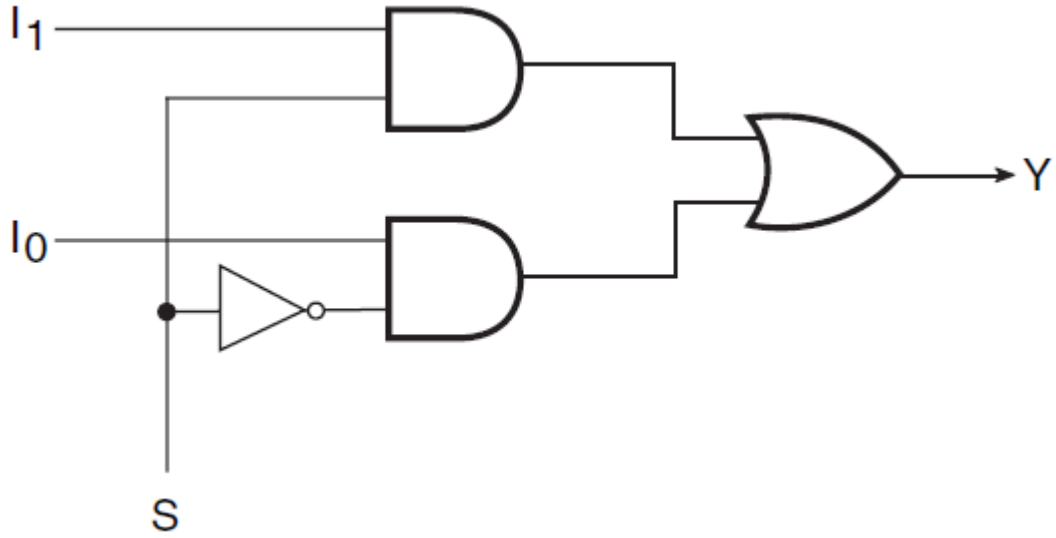
2 x 1 Mux



Truth Table

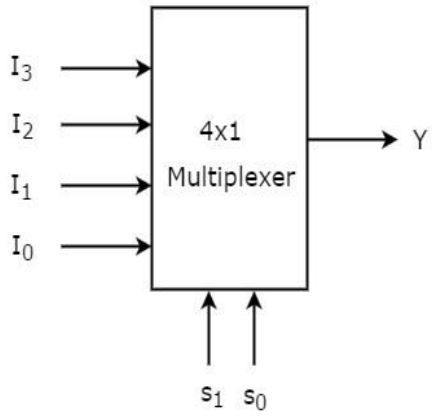
S	Y
0	I_0
1	I_1

$$Y = I0\,\overline{S} + I1\,S$$



4 x1 Mux

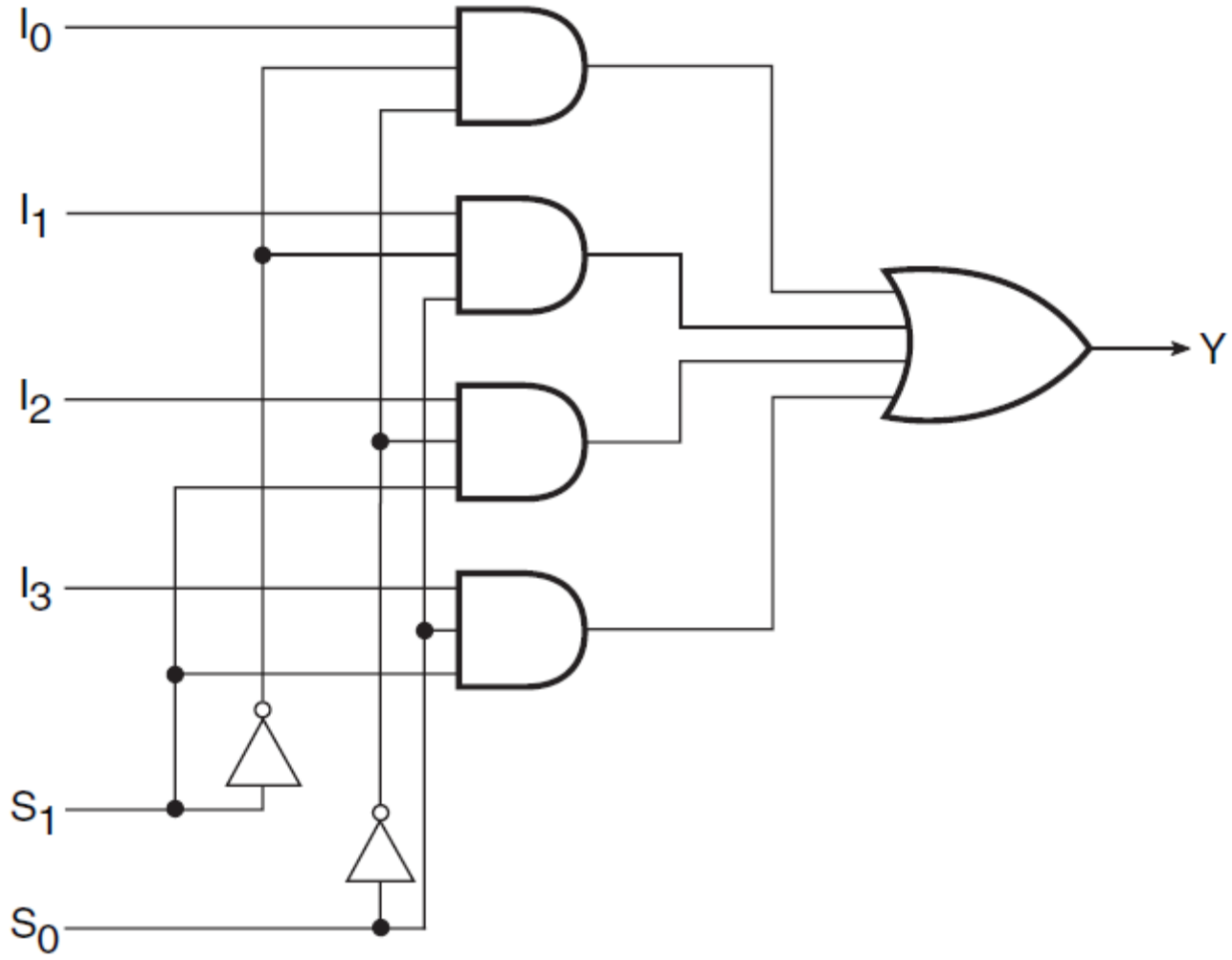
One of these 4 inputs will be connected to the output based on the combination of inputs present at these two selection lines. **Truth table** of 4x1 Multiplexer is shown below.



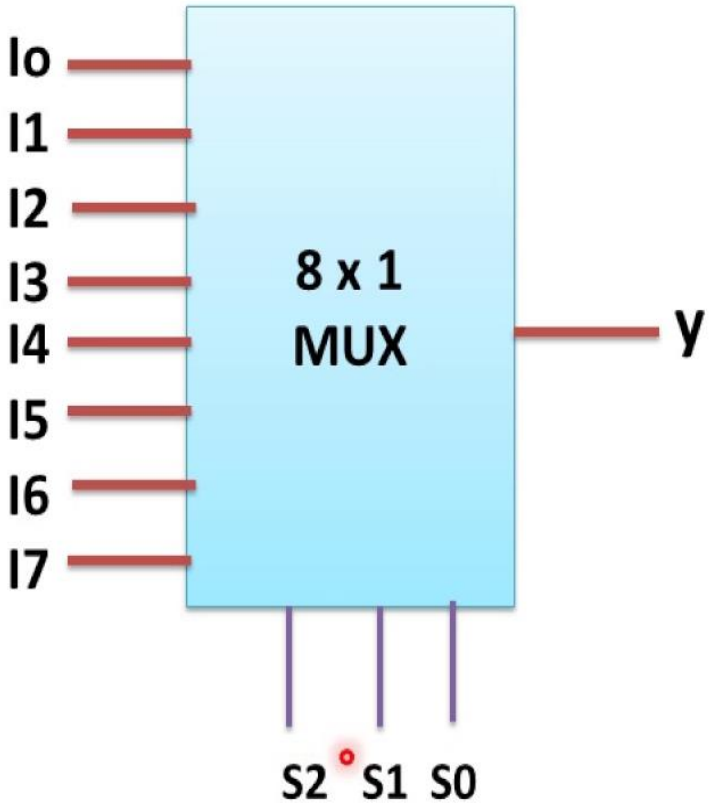
Selection Lines		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

From Truth table, we can directly write the **Boolean function** for output, Y as

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



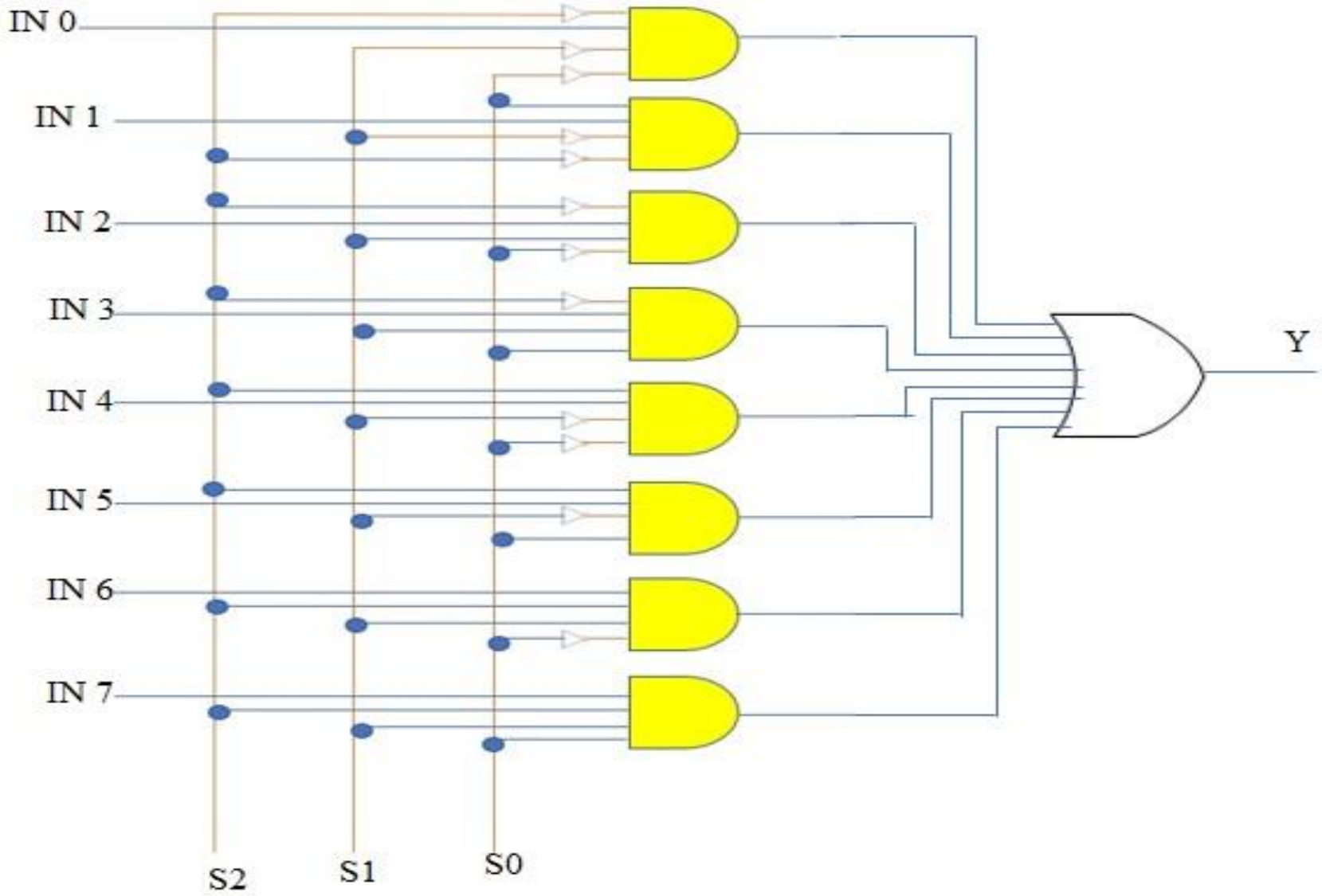
8 x 1 Mux



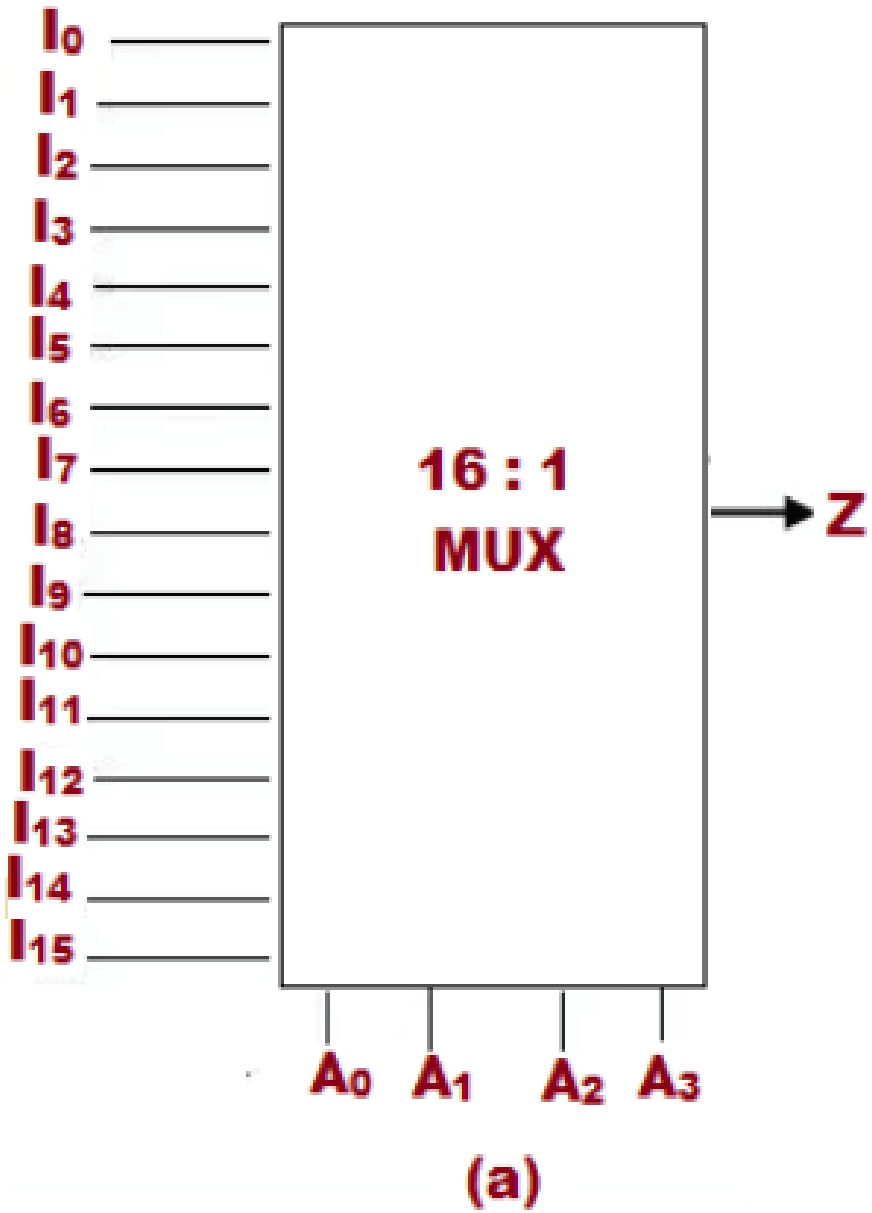
Selection Inputs			Output
S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

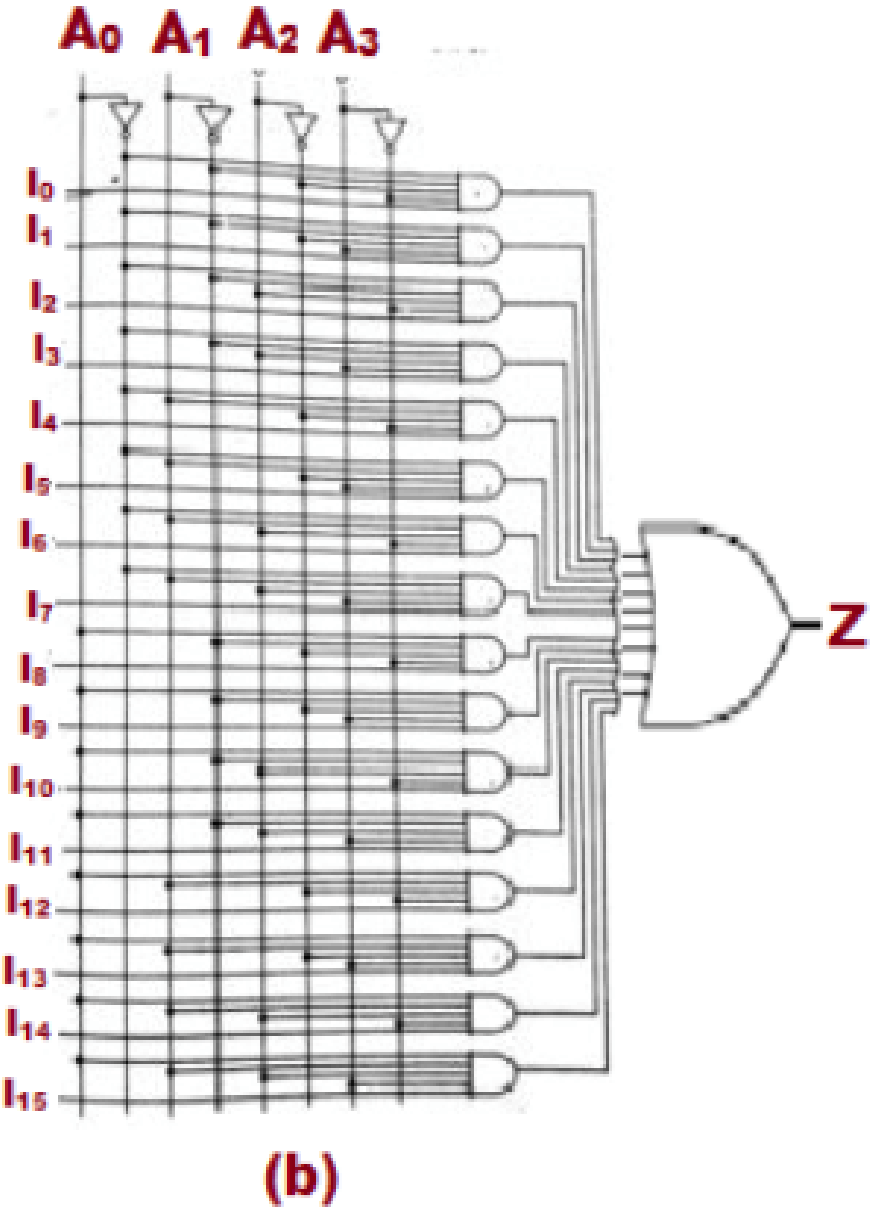
$$Y = I_0.\overline{S_2}.\overline{S_1}.\overline{S_0} + I_1.\overline{S_2}.\overline{S_1}.S_0 + I_2.\overline{S_2}.S_1.\overline{S_0} + I_3.\overline{S_2}.S_1.S_0 + I_4.S_2.\overline{S_1}.\overline{S_0}$$
$$+ I_5.S_2.\overline{S_1}.S_0 + I_6.S_2.S_1.\overline{S_0} + I_7.S_2.S_1.S_0$$

Logic Diagram



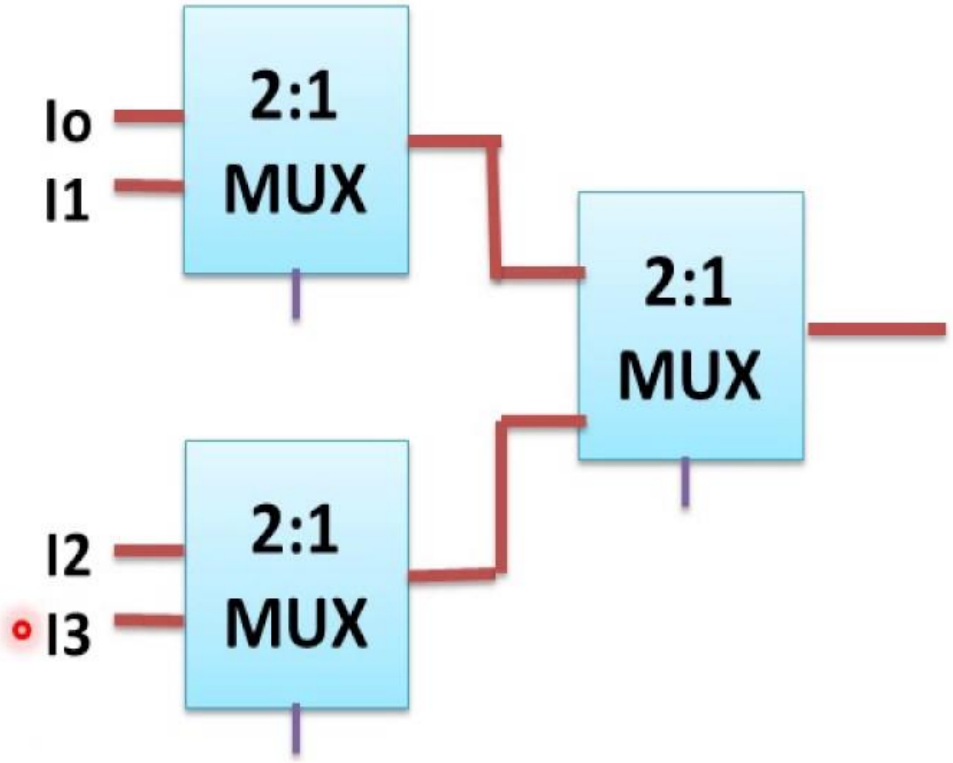
16 x 1 Mux





4x 1 Mux Using 2 x 1 Mux

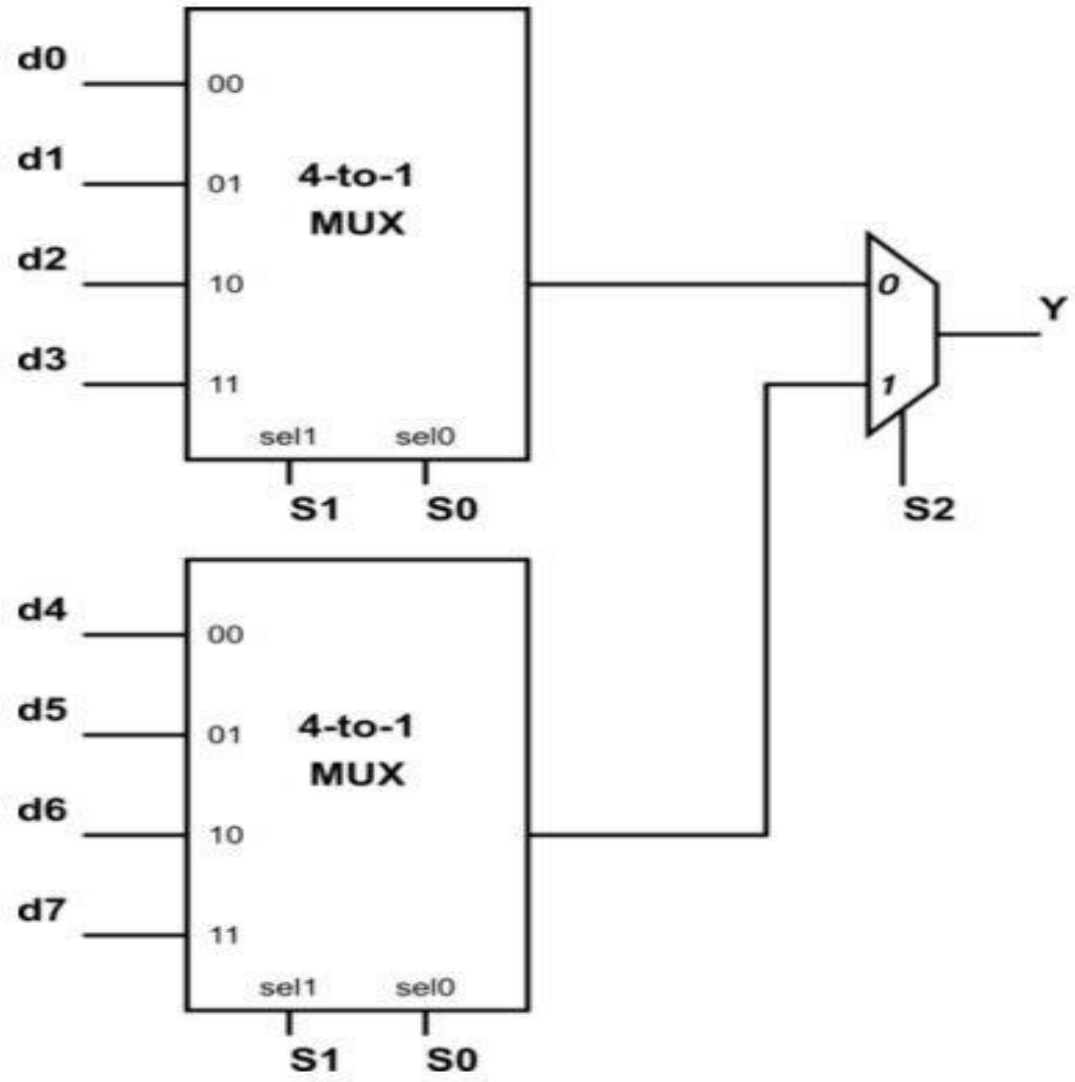
GROW



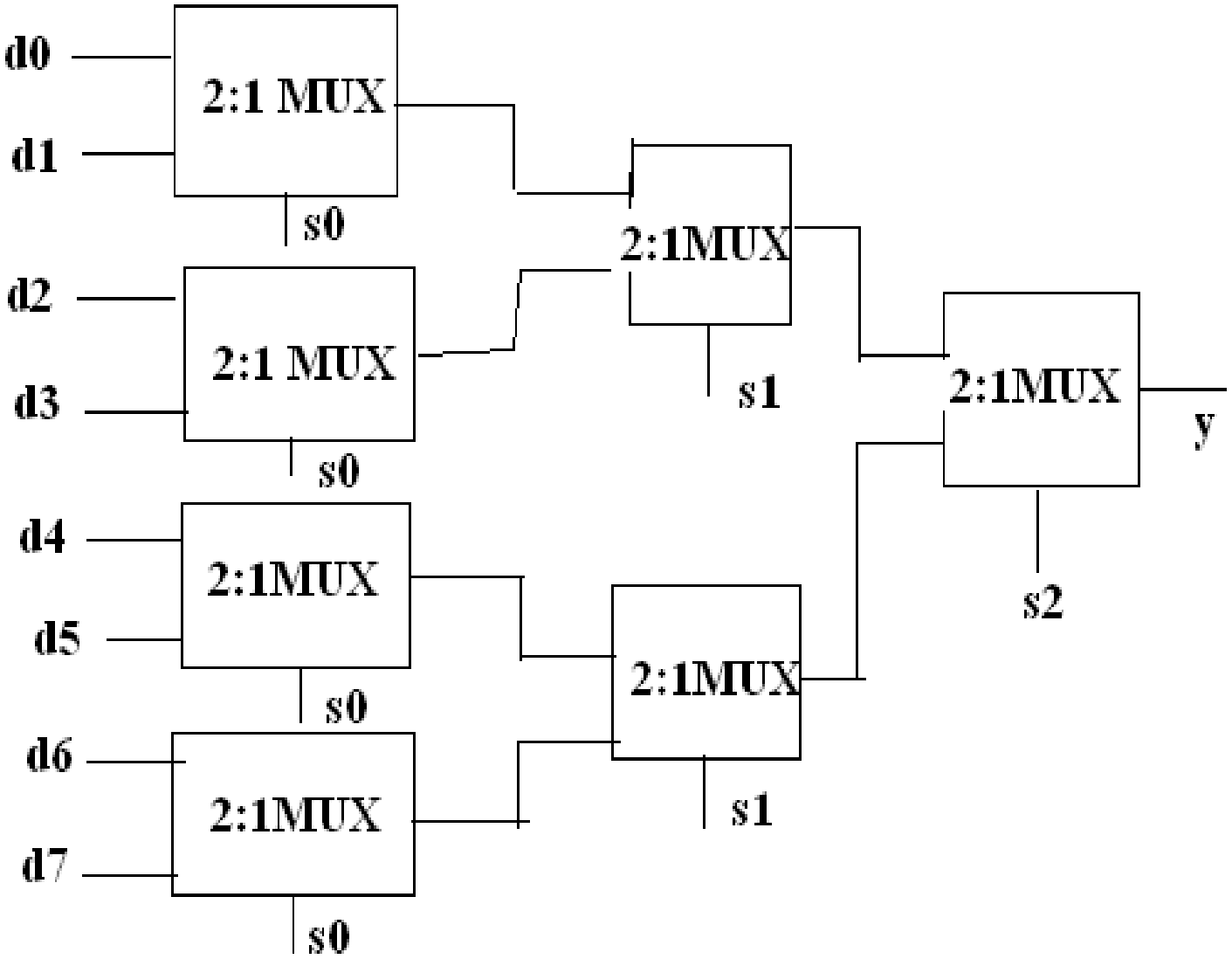
S1	S0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

LEARN

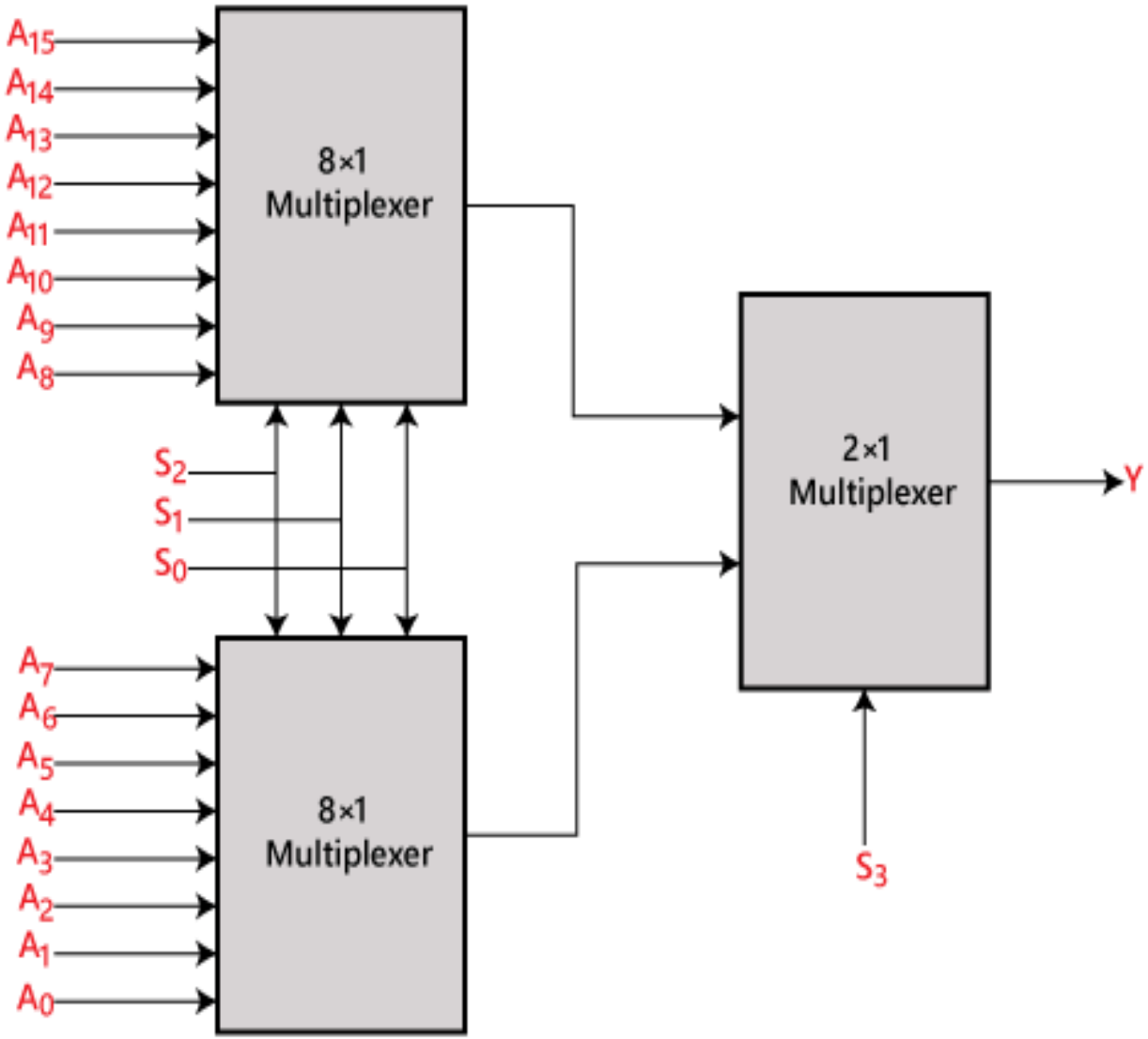
8 x 1 Mux Using 4 x 1



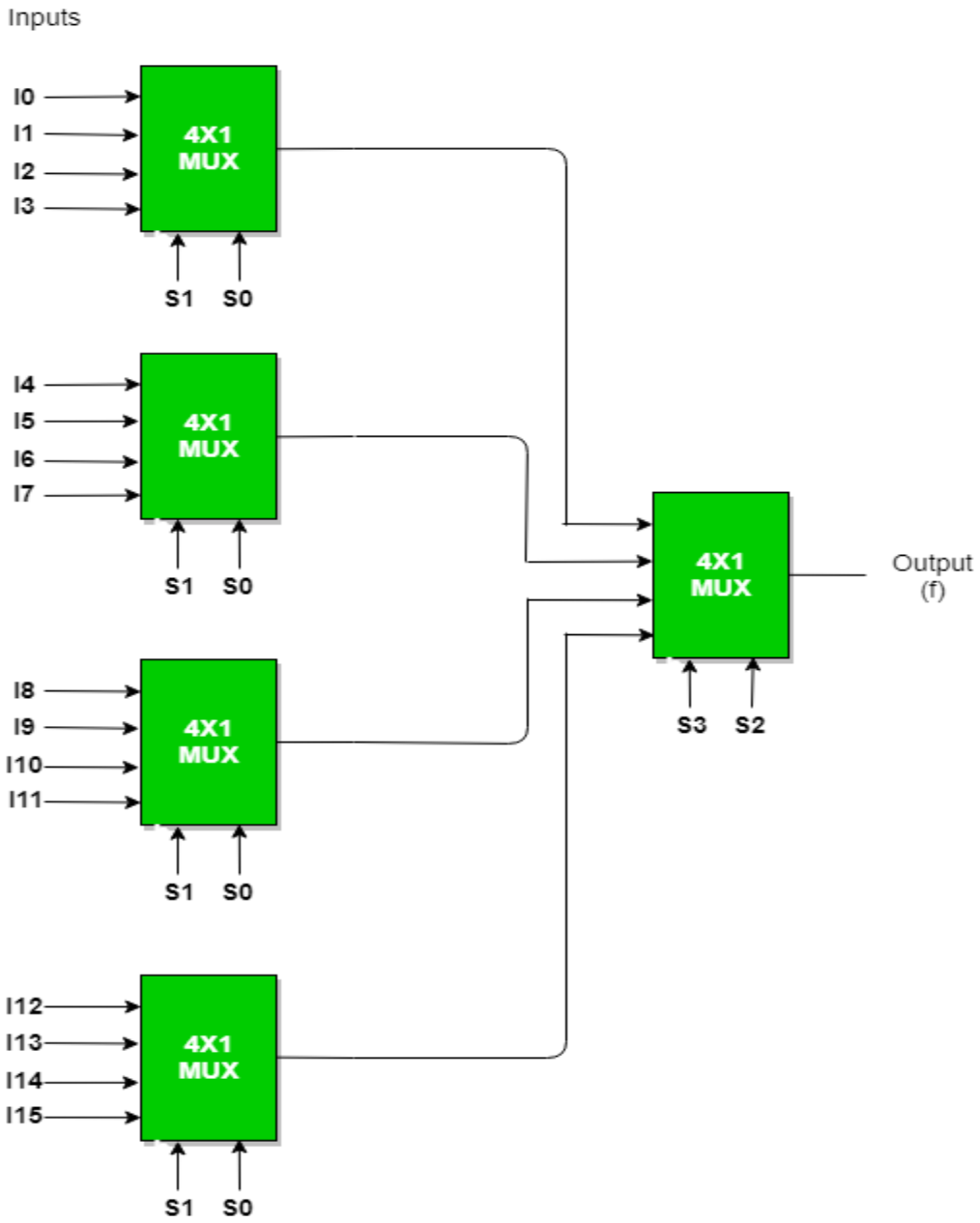
8 x 1 Mux using 2 x 1



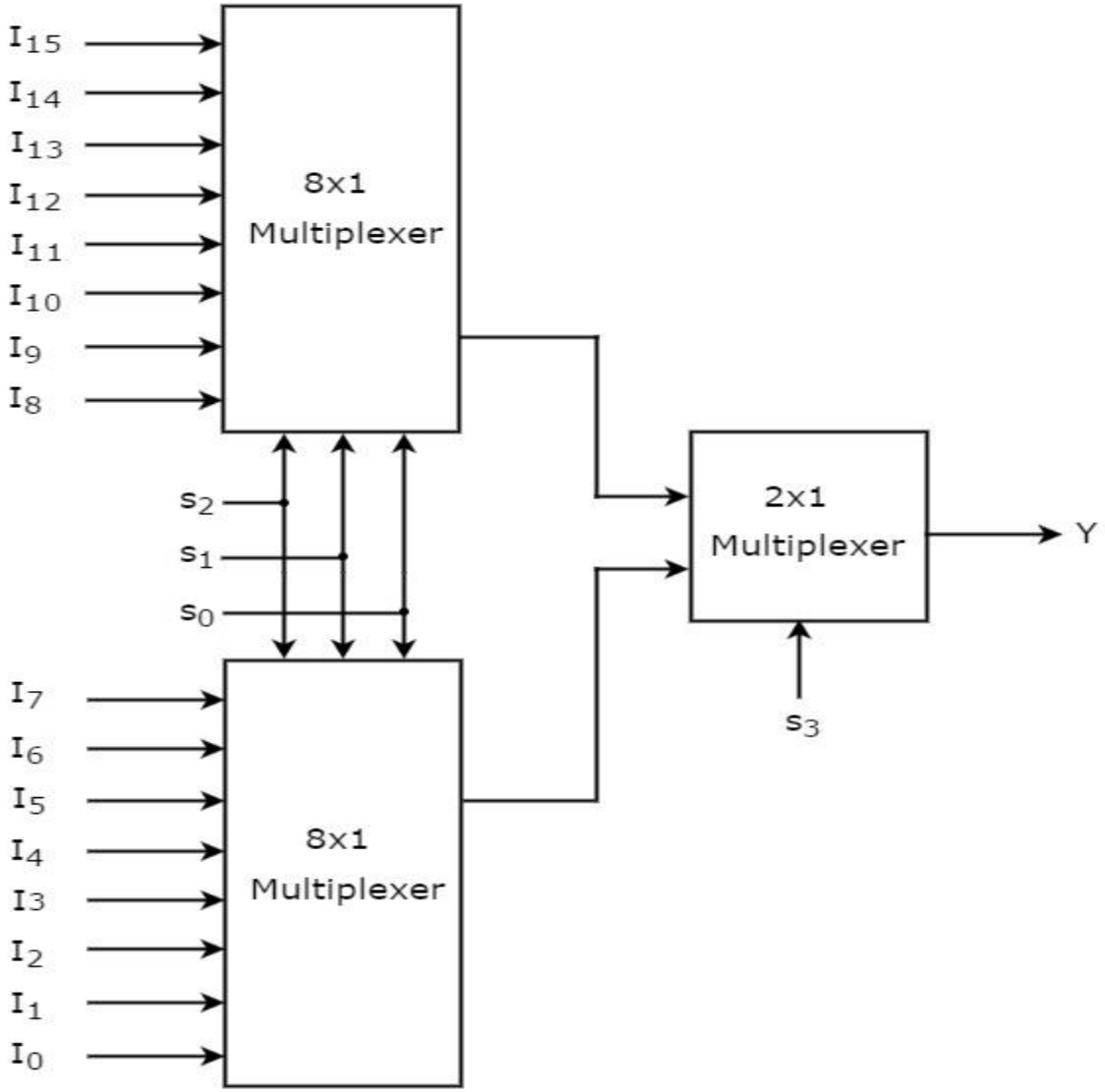
16 x 1 Mux using 8 x1



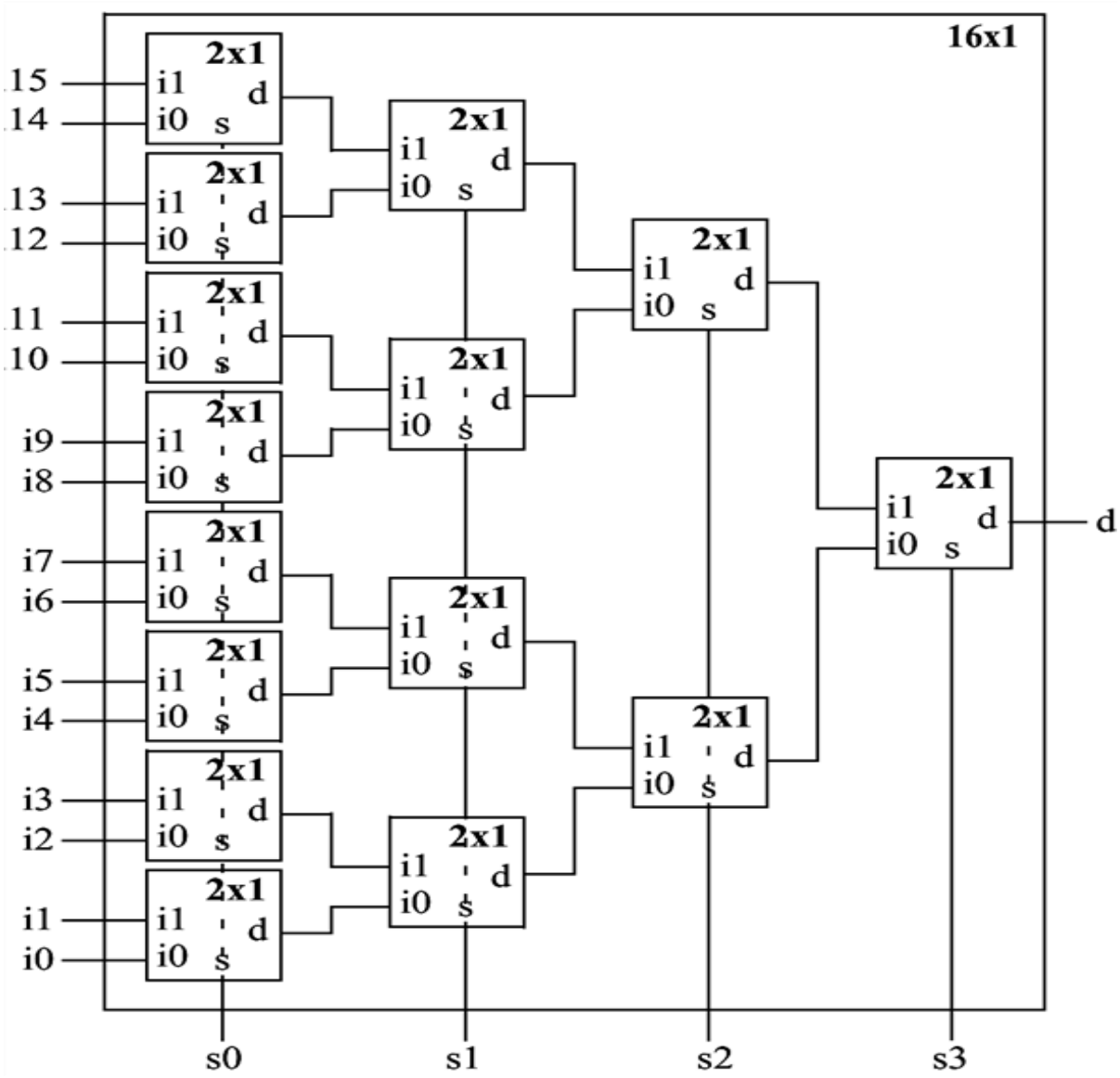
16x 1 Mux using 4 x 1 Mux Only



16 x 1 Mux using 8 x 1 Mux and 2 x 1 Mux



16 x 1 Mux using 2 x 1 Mux



Demultiplexer (1 x 4)

- A *demultiplexer* is a combinational logic circuit with an input line, 2n output lines and n select lines.
- It routes the information present on the input line to any of the output lines.

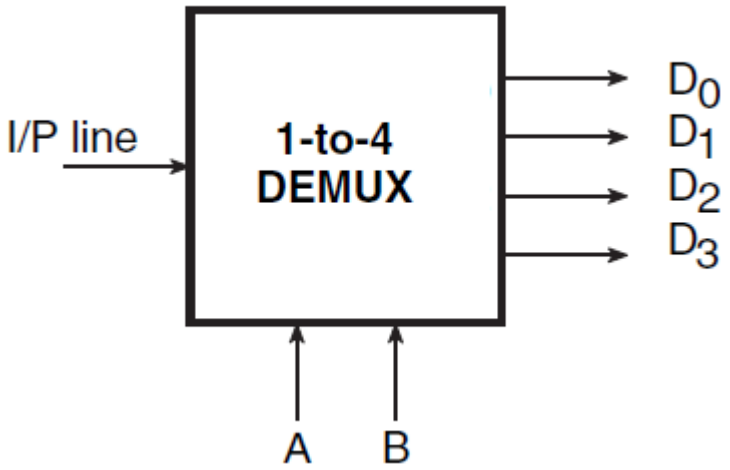
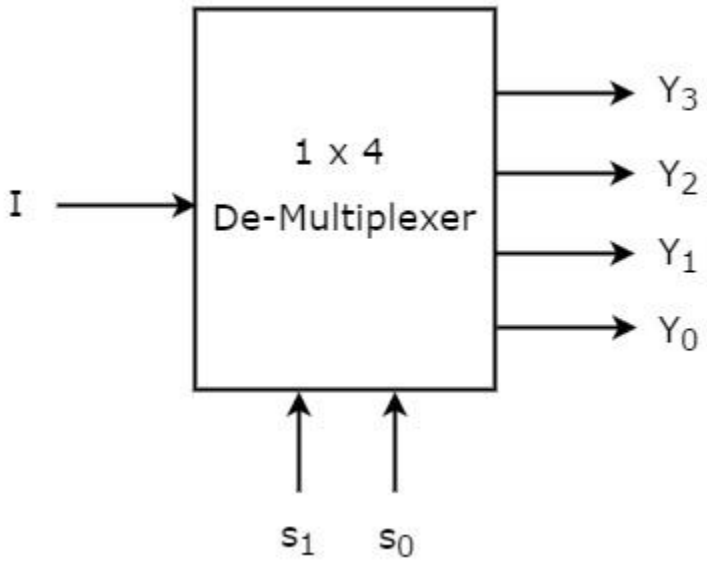


Figure 1-to-4 demultiplexer.

I/P	Select		O/P			
	A	B	D_0	D_1	D_2	D_3
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

Another Example Demultiplexer (1 x

- 1x4 De-Multiplexer has one input I, two selection lines, s_1 & s_0 and four outputs Y_3, Y_2, Y_1 &Y_0. The **block diagram** of 1x4 De-Multiplexer is shown in the following figure.



The single input 'I' will be connected to one of the four outputs, Y_3 to Y_0 based on the values of selection lines s_1 & s0. The **Truth table** of 1x4 De-Multiplexer is shown below.

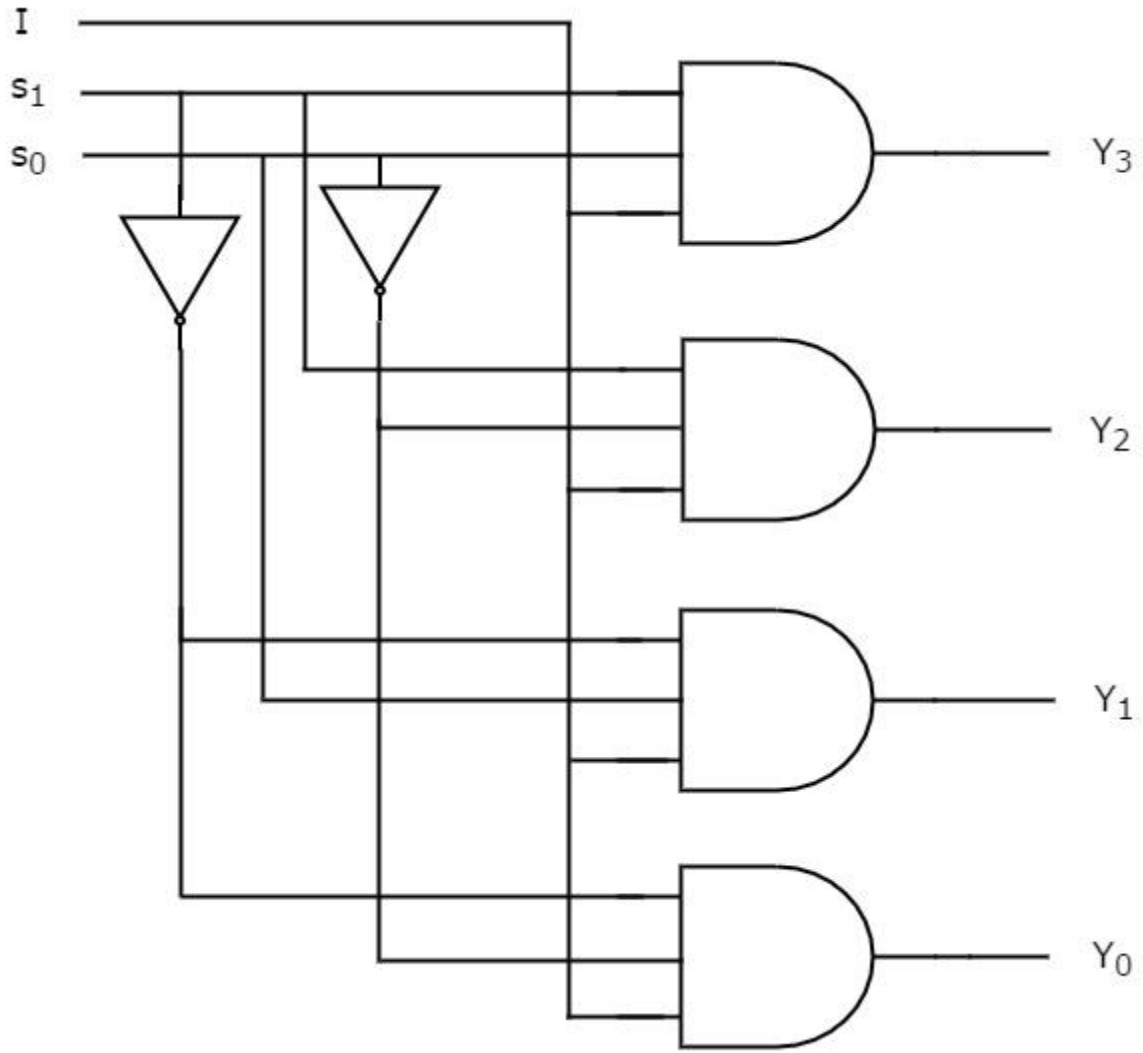
Selection Inputs		Outputs			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

From the above Truth table, we can directly write the **Boolean functions** for each output as

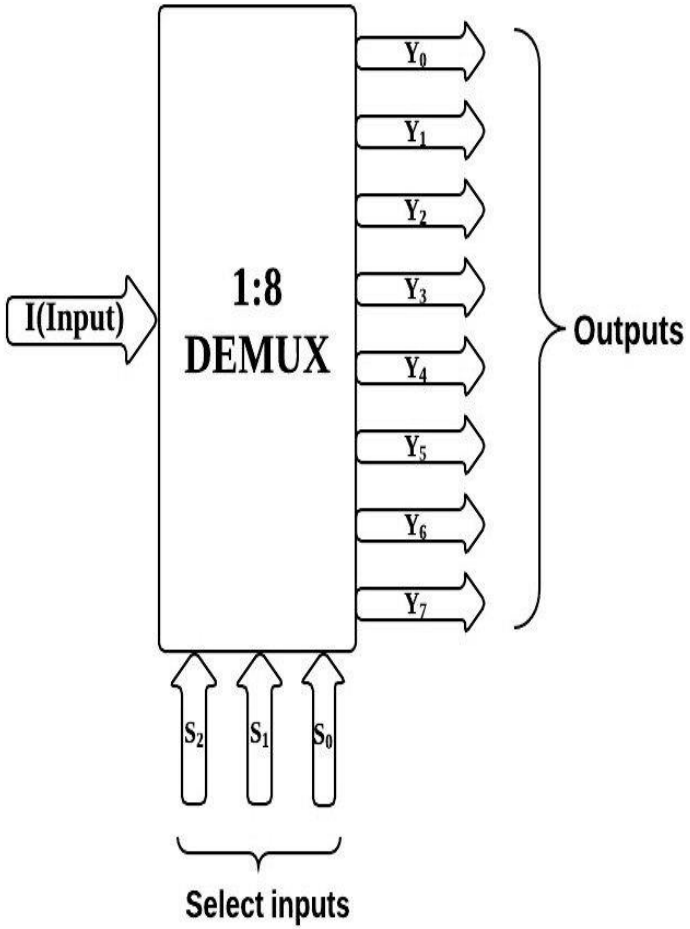
$$Y_3 = s_1 s_0 I \qquad Y_1 = s_1' s_0 I$$

$$Y_2 = s_1 s_0' I \qquad Y_0 = s_1' s_0' I$$

Logic Diagram



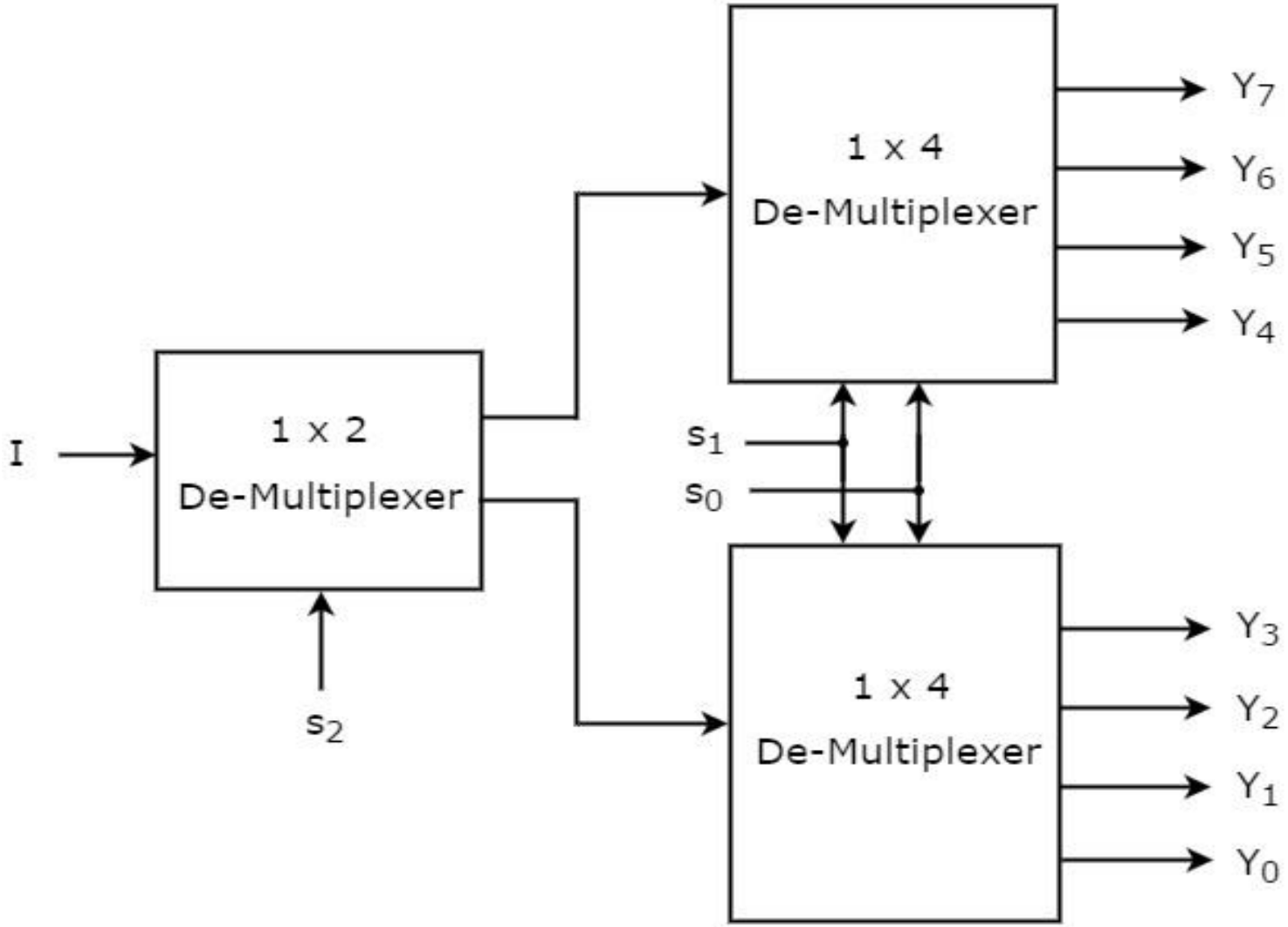
Demultiplexer (1x 8)



Selection Inputs			Outputs							
s_2	s_1	s_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

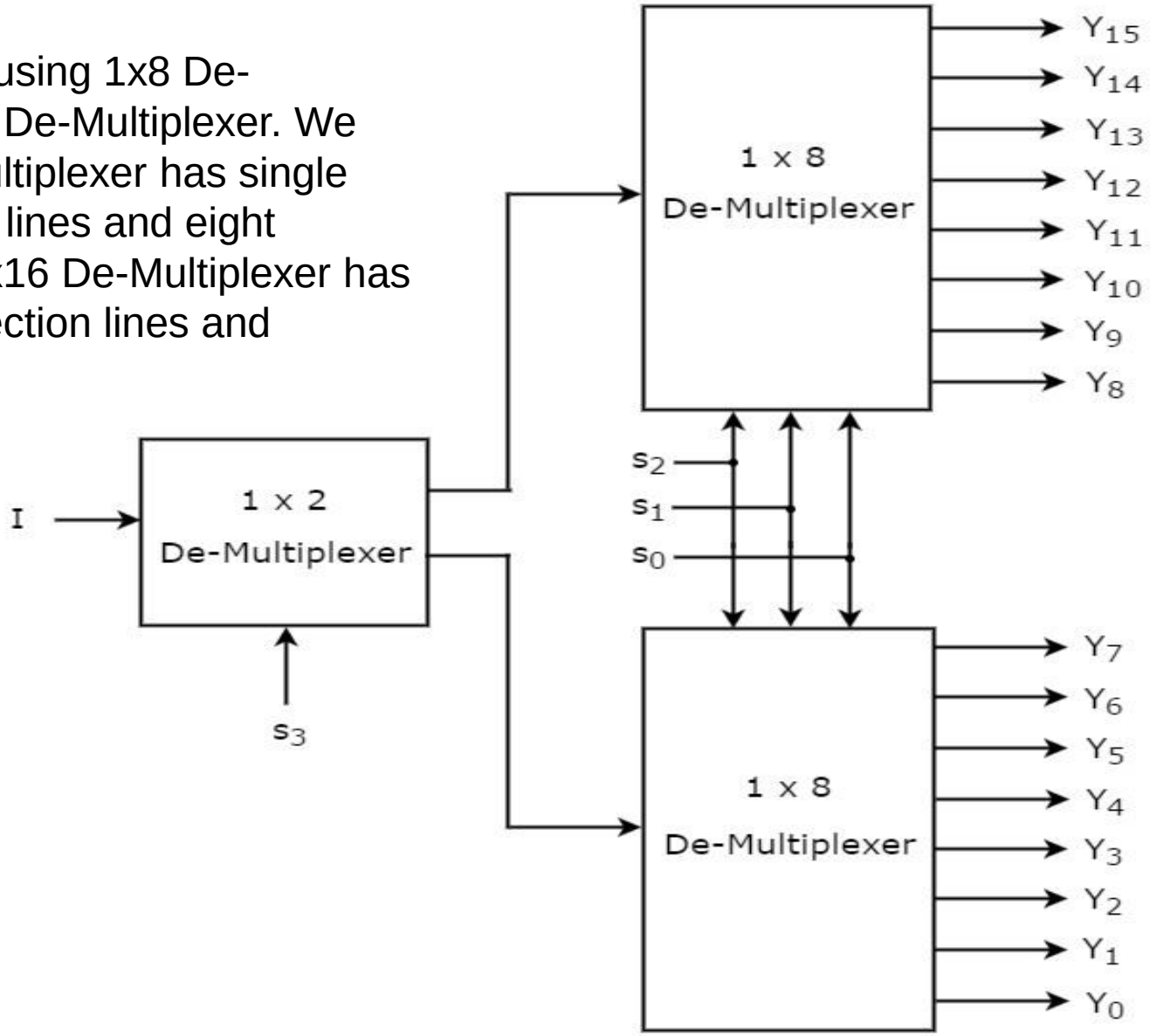
Realization of Higher Order Demultiplexers

- 1x8 De-Multiplexer using 1x4 De-Multiplexers and 1x2 De-Multiplexer. We know that 1x4 De-Multiplexer has single input, two selection lines and four outputs.
- Whereas, 1x8 De-Multiplexer has single input, three selection lines and eight outputs.
- So, we require two **1x4 De-Multiplexers** in second stage in order to get the final eight outputs. Since, the number of inputs in second stage is two, we require **1x2 DeMultiplexer** in first stage so that the outputs of first stage will be the inputs of second stage.
- Input of this 1x2 De-Multiplexer will be the overall input of 1x8 De-Multiplexer.
- Let the 1x8 De-Multiplexer has one input I, three selection lines s_2, s_1 & s_0 and outputs Y_7 to Y_0. The **Truth table** of 1x8 De-Multiplexer is shown below.


I
1 x 2
De-Multiplexer
s_2
1 x 4
De-Multiplexer
s_1
s_0
Y_7
Y_6
Y_5
Y_4
1 x 4
De-Multiplexer
Y_3
Y_2
Y_1
Y_0

1x16 De-Multiplexer

1x16 De-Multiplexer using 1x8 De-Multiplexers and 1x2 De-Multiplexer. We know that 1x8 De-Multiplexer has single input, three selection lines and eight outputs. Whereas, 1x16 De-Multiplexer has single input, four selection lines and sixteen outputs.



Decoder

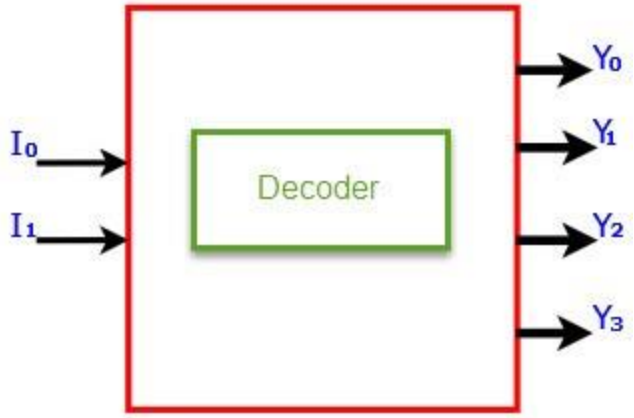
A combinational Circuit which accepts n inputs

and 2^n outputs

2 to 4 line decoder

3 to 8 line decoder

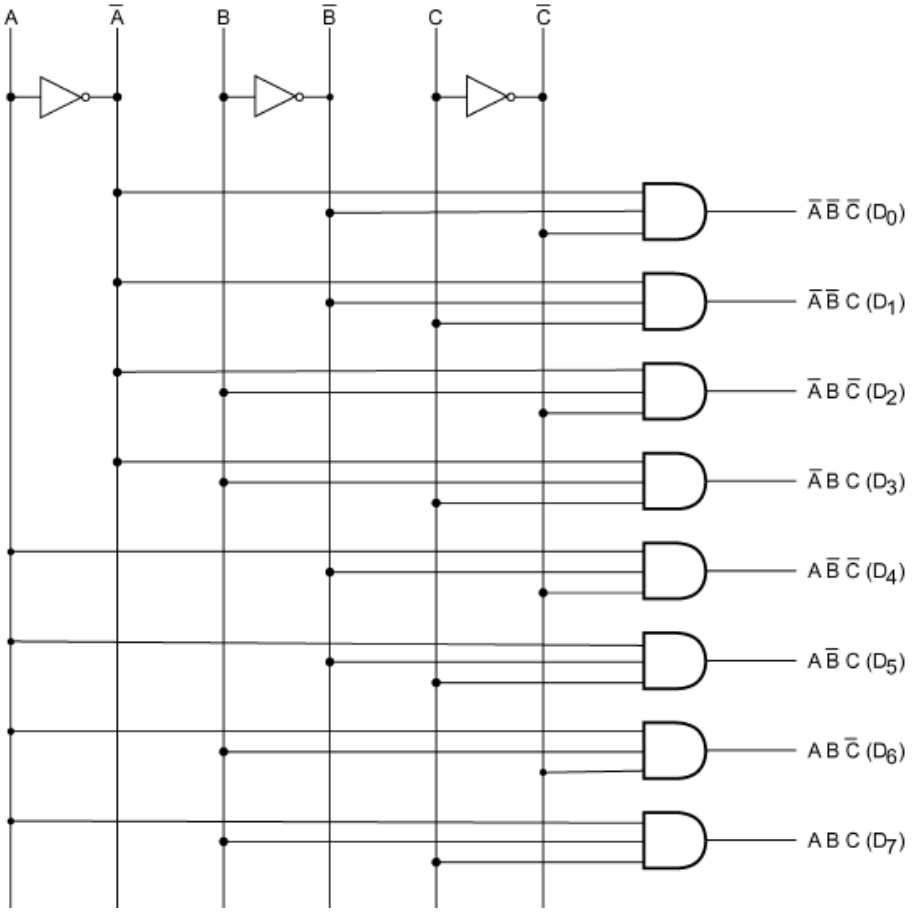
4 to 16 line decoder



Decimal Digit	Binary Inputs			Logic Function	Outputs							
					D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	0	$\overline{A_2}\,\overline{A_1}\,\overline{A_0}$	1	0	0	0	0	0	0	0
1	0	0	1	$\overline{A_2}\,\overline{A_1}\,A_0$	0	1	0	0	0	0	0	0
2	0	1	0	$\overline{A_2}\,A_1\,\overline{A_0}$	0	0	1	0	0	0	0	0
3	0	1	1	$\overline{A_2}\,A_1\,A_0$	0	0	0	1	0	0	0	0
4	1	0	0	$A_2\,\overline{A_1}\,\overline{A_0}$	0	0	0	0	1	0	0	0
5	1	0	1	$A_2\,\overline{A_1}\,A_0$	0	0	0	0	0	1	0	0
6	1	1	0	$A_2\,A_1\,\overline{A_0}$	0	0	0	0	0	0	1	0
7	1	1	1	$A_2\,A_1\,A_0$	0	0	0	0	0	0	0	1

- Logic Diagram

INPUTS			OUTPUTS							
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1



In terms of X, Y, Z

$$D_0 = x'y'z'$$

$$D_1 = x'y'z$$

$$D_2 = x'yz'$$

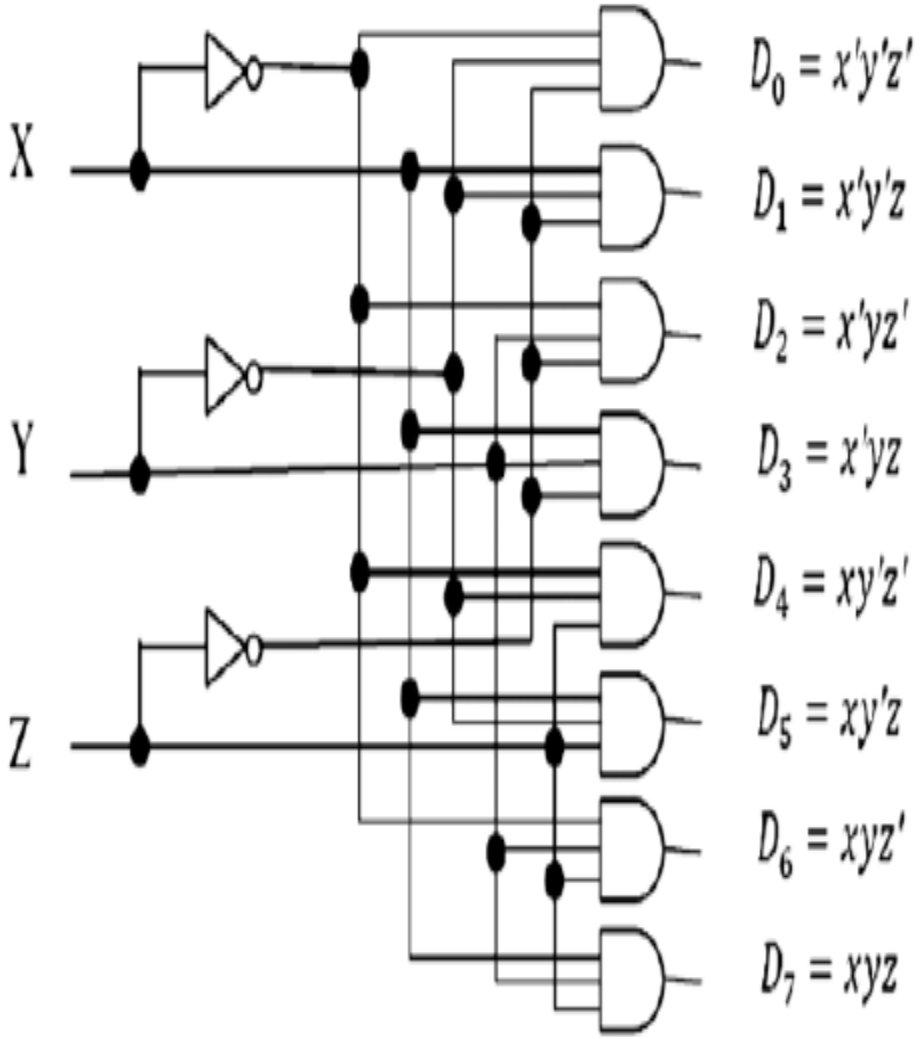
$$D_3 = x'yz$$

$$D_4 = xy'z'$$

$$D_5 = xy'z$$

$$D_6 = xyz'$$

$$D_7 = xyz$$



Encoder

- Opposite of Decoder
- 2^n inputs and n outputs
- Example 8 to 3 line decoder

Input									Output	
I7	I6	I5	I4	I3	I2	I1	I0	O2	O1	O0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

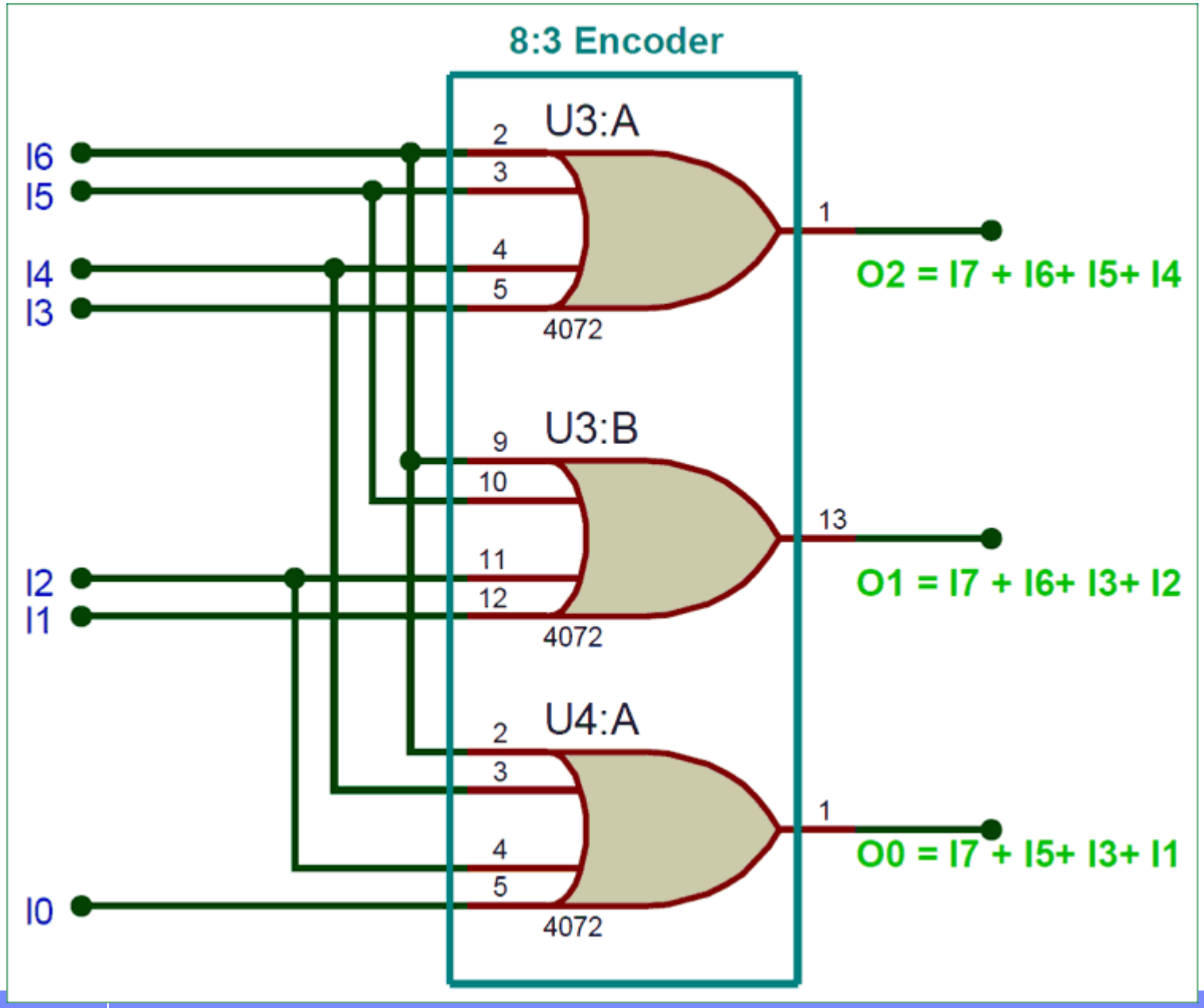
As seen from the truth table, the output is 000 when I0 is active; 001 when I1 is active; 010 when I2 is active and so on.

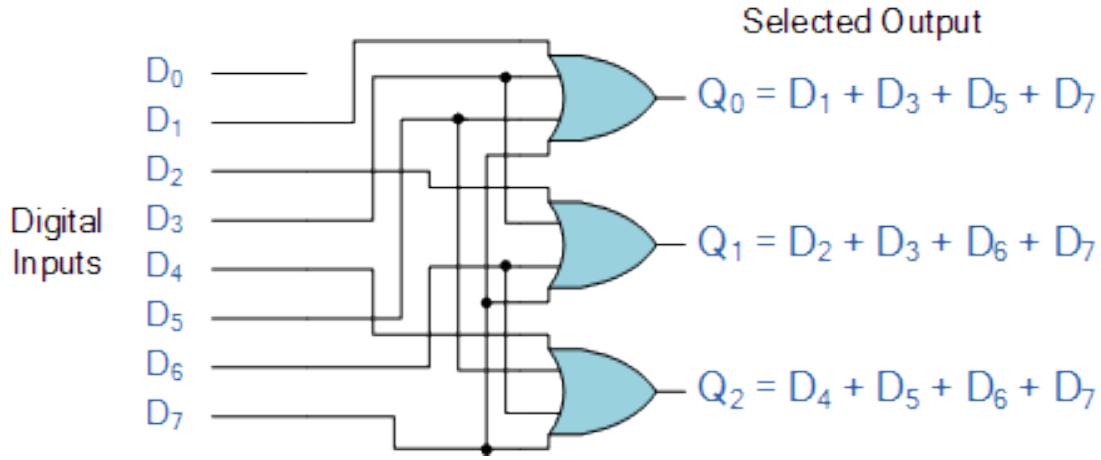
Implementation –

From the truth table, the output line Z is active when the input octal digit is 1, 3, 5 or 7. Similarly, Y is 1 when input octal digit is 2, 3, 6 or 7 and X is 1 for input octal digits 4, 5, 6 or 7. Hence, the Boolean functions would be:

X Y Z are the outputs and D0, D1, D2,D3,D4,D5,D6 and D7 are the inputs.

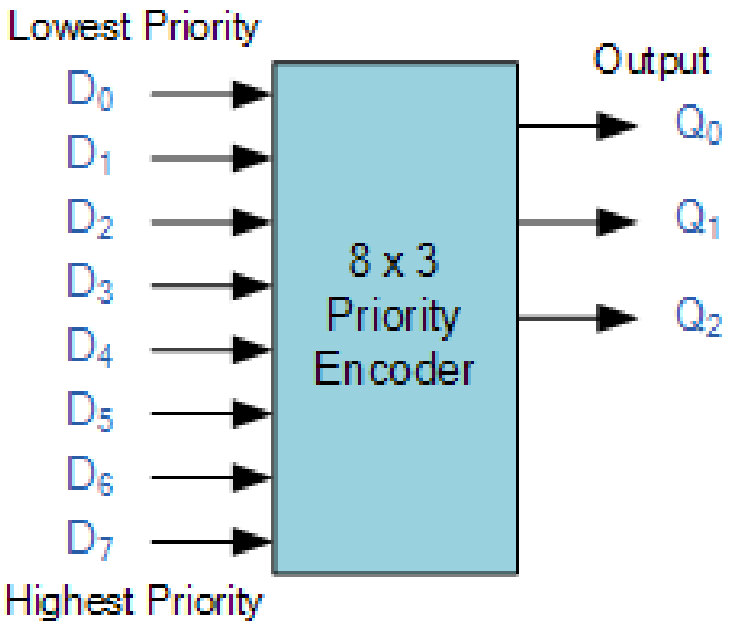
$X = D4 + D5 + D6 + D7$
$Y = D2 + D3 + D6 + D7$
$Z = D1 + D3 + D5 + D7$


8:3 Encoder
U3:A
U3:B
U4:A
4072
4072
4072
I6
I5
I4
I3
I2
I1
I0
O2 = I7 + I6+ I5+ I4
O1 = I7 + I6+ I3+ I2
O0 = I7 + I5+ I3+ I1


Selected Output
Digital Inputs
D0
D1
D2
D3
D4
D5
D6
D7
Q0 = D1 + D3 + D5 + D7
Q1 = D2 + D3 + D6 + D7
Q2 = D4 + D5 + D6 + D7

Priority Encoder –

A priority encoder is an encoder circuit in which inputs are given priorities. When more than one inputs are active at the same time, the input with higher priority takes precedence and the output corresponding to that is generated.



Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

X = dont care

Priority encoders output the highest order input first for example, if input lines “D2“, “D3” and “D5” are applied simultaneously the output code would be for input “D5” (“101”) as this has the highest order out of the 3 inputs. Once input “D5” had been removed the next highest output code would be for input “D3” (“011”), and so on.

Digital Inputs								Binary Output		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	**1**	0	0	0
0	0	0	0	0	0	**1**	X	0	0	1
0	0	0	0	0	**1**	X	X	0	1	0
0	0	0	0	**1**	X	X	X	0	1	1
0	0	0	**1**	X	X	X	X	1	0	0
0	0	**1**	X	X	X	X	X	1	0	1
0	**1**	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1