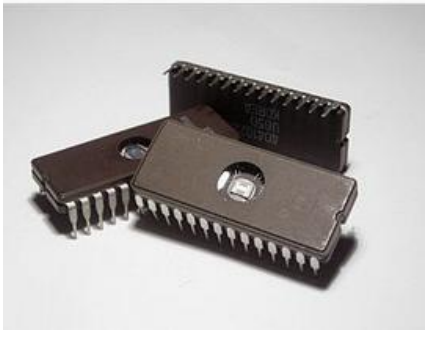
Philadelphia University
Faculty of Information Technology
Department of Computer Science

Computer Logic Design

By
Dareen Hamoudeh

Integrated circuits

- An **integrated circuit** (also referred to as an **IC**, a **chip**, or a **microchip**) is a set of electronic circuits on one small plate ("chip") of semiconductor material, normally silicon. This can be made much smaller than a discrete circuit made from independent components.



IC's Generations

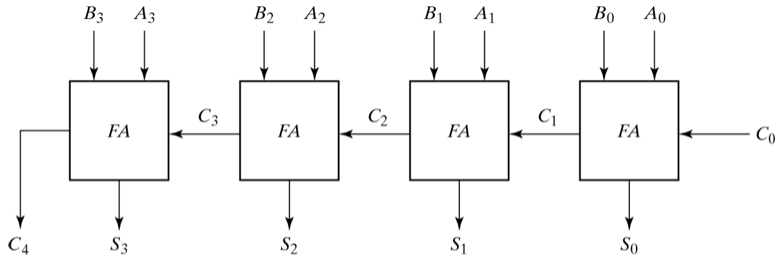
- **SSI, MSI and LSI :**
  - The first integrated circuits contained only a few transistors. Called "**small-scale integration**" (**SSI**), digital circuits containing transistors numbering in the tens provided a few logic gates.
  - The next step in the development of integrated circuits, introduced devices which contained hundreds of transistors on each chip, called "**medium-scale integration**" (**MSI**).
  - Further development, driven by the same economic factors, led to "**large-scale integration**" (**LSI**, with tens of thousands of transistors per chip.
  - The final step in the development process, was **"very large-scale integration" (VLSI).** The development started with hundreds of thousands of transistors, and continues beyond several billion transistors .

4-bit Adder Circuit

- A Full adder is used to add two (1-bit) numbers.

Like 1+ 0

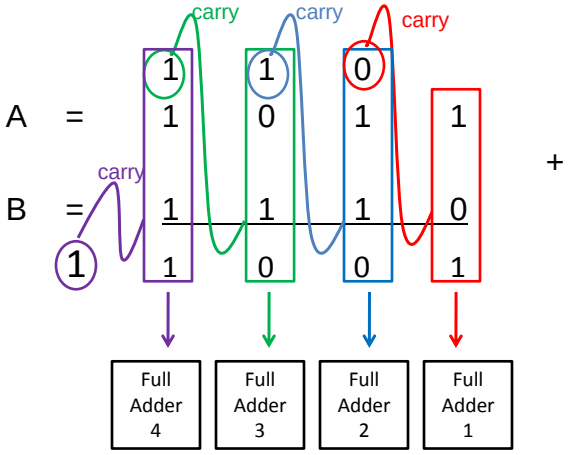
- If we want to sum two (4-bit) numbers: like 1110+ 1011
  - We use 4 (1-bit)adders.



4-Bit Adder

Example: 4-bit addition

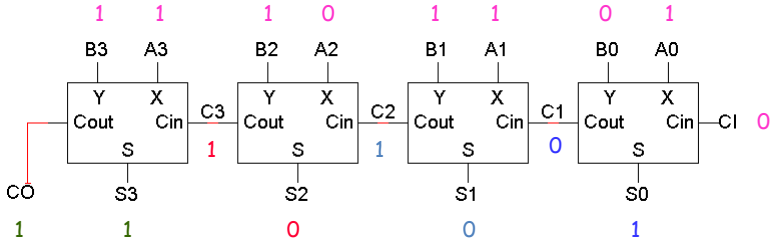
- If we want to find A+B, where A=1011 , B=1110



To implement this We use:

Example: 4-bit addition

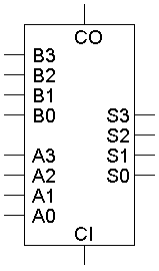
– Find A+B, where A=1011 (eleven), B=1110 (fourteen).



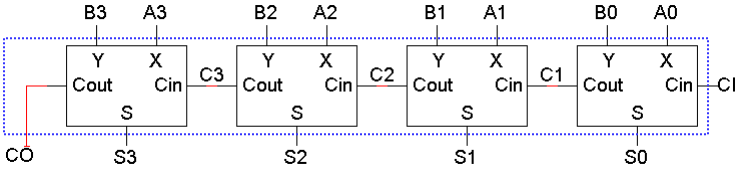
1. Fill in all the inputs, including CI=0
2. The circuit produces C1 and S0 (1 + 0 + 0 = 01)
3. Use C1 to find C2 and S1 (1 + 1 + 0 = 10)
4. Use C2 to compute C3 and S2 (0 + 1 + 1 = 10)
5. Use C3 to compute CO and S3 (1 + 1 + 1 = 11)

Woohoo! The final answer is 11001 (twenty-five).

A 4-bit adder



- Four full adders together make a 4-bit adder.
- There are nine total inputs:
  - Two 4-bit numbers, A3 A2 A1 A0 and B3 B2 B1 B0
  - An initial carry in, C_i
- The five outputs are:
  - A 4-bit sum, S3 S2 S1 S0
  - A carry out, C_o



Adder / Subtractor

Binary Subtraction
"review"

- Subtraction is done by using complements:
  - Where:

    2's Complement of (B)= 1's Complement of (B)+1
- To find A-B we convert the subtraction by addition:
  - A-B= A + (B'+1)

Adder / Subtractor

A four-bit **adder/subtractor** demonstration:

- While it is perfectly possible to design a custom circuit for the subtraction operation, it is much more common to re-use an existing adder and to replace a subtraction by a two-complement's addition.
- When the *Sub/Add* input is low (0):
  - XOR-gates act as non-inverting buffers, and the carry-input to the adder is 0.
  - Therefore, the adder calculates a four-bit **sum** plus carry-out:

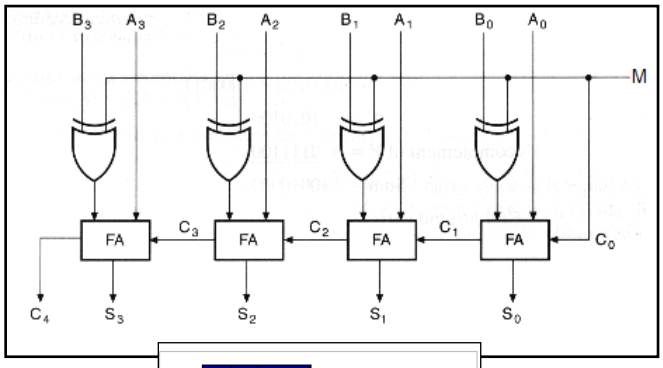
    (Cout,S3,S2,S1,S0) = (A3,A2,A1,A0) + (B3,B2,B1,B0)
- If the *Sub/Add* input is high (1):
  - the XOR-gates act as inverting buffers, and the carry-input to the adder is 1.
  - Therefore, the adder performs as a four-bit **subtraction:**

    (Cout,S3,S2,S1,S0) = (A3,A2,A1,A0) - (B3,B2,B1,B0)

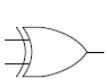
Adder / Subtractor

- The circuit has a mode control signal M which determines if the circuit is to operate as an adder or a subtractor.
- Each XOR gate receives input M and one of the inputs of B, i.e., Bi. To understand the behavior of XOR gate consider its truth table given below. If one input of XOR gate is zero then the output of XOR will be same as the second input. While if one input of XOR gate is one then the output of XOR will be complement of the second input.
- when M = 0, the output of XOR gate will be Bi $\oplus$ 0 = Bi. If the full adders receive the value of B, and the input carry C0 is 0, the circuit performs **A plus B.**
- When M = 1, the output of XOR gate will be Bi $\oplus$ 1 = Bi'. If the full adders receive the value of B', and the input carry C0 is 1, the circuit performs A plus 1's complement of B plus 1, which is equal to **A minus B**

Adder / Subtractor

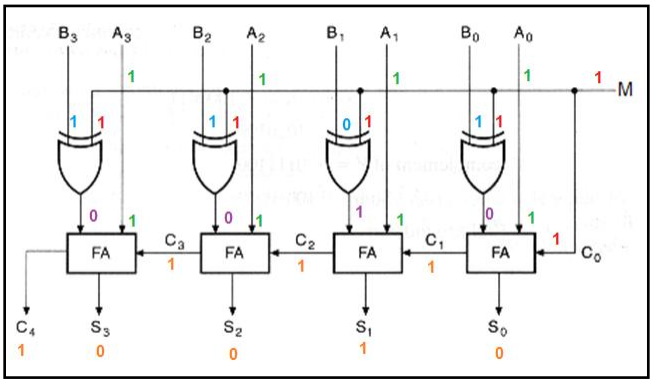


A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0



Adder / Subtractor

- Find A-B, where A=1111 , B=1101
  - First M must be 1



  - C_4=1, we discard it, the result is (1111)-(1101)= 0010

Adder / Subtractor

- remember
  - The circuit for subtracting *A* - *B* consists of an adder with inverters placed *between* each data input B and the corresponding input of the full adder.
  - The input carry Co must be equal to 1 when subtraction is performed.
  - The operation thus performed becomes A, plus the 1's complement of *B, plus* 1. this is equal to A plus the 2's complement of *B.*
  - For unsigned numbers, that gives *A* - B if *A>=b or the 2's* complement of *(B - A)* if A < B.
  - For signed numbers, the result is *A* - B, provided that there is no overflow.

4-bit adder subtractor "Signed Numbers"

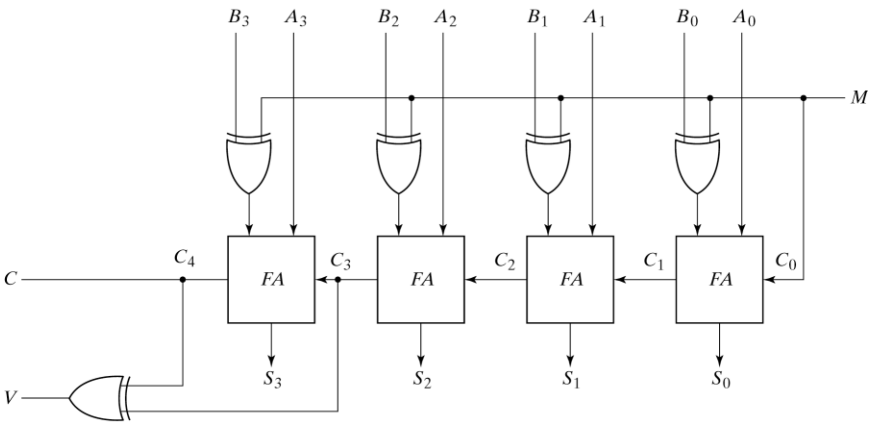


Fig. 4-13 4-Bit Adder Subtractor

Questions

5-3 The adder-subtractor of Fig. 5-2(b) is used to subtract the following unsigned 4-bit numbers: 0110 − 1001 (6 − 9).
(a) What are the binary values in the nine inputs of the circuit?
(b) What are the binary values of the five outputs of the circuit? Explain how the output is related to the operation of 6 − 9.

5-4 The adder-subtractor circuit of Fig. 5-2(b) has the following values for mode input M and data inputs A and B. In each case, determine the values of the outputs: S_4, S_3, S_2, S_1, and C_5.

	M	A	B
(a)	0	0111	0110
(b)	0	1000	1001
(c)	1	1100	1000
(d)	1	0101	1010
(e)	1	0000	0001

Philadelphia University

Decoder

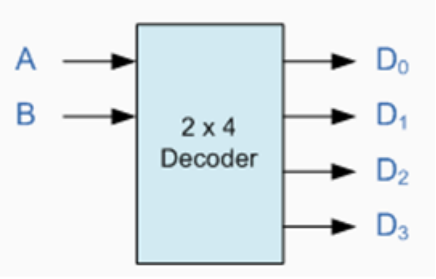
Decoder

- A **decoder** is a device which does the reverse operation of an encoder, undoing the encoding so that the original information can be retrieved.
- It is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines.
- Decoders are sometimes called minterm generators.

Decoder

- **Binary Decoders** have inputs of 2-bit, 3-bit or 4-bit or 5-bit codes depending upon the number of data input lines, and a n-bit decoder has 2^n output lines.
- It is called n-to-2^n line decoder.
- if it receives n inputs (usually grouped as a binary or Boolean number) it activates one and only one of its 2^n outputs based on that input with all other outputs deactivated.
- A decoders output code normally has more bits than its input code and practical binary decoder circuits include, 2-to-4, 3-to-8 and 4-to-16 line configurations.
- decoder circuit would be an AND gate because the output of an AND gate is "High" (1) only when all its inputs are "High." Such output is called as "active High output".

2-to-4 Binary Decoders



2-to-4 Binary Decoders

- **Truth Table:**

Input		Output			
A	B	D0	D1	D2	D3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- **Simplified Function for each output:**

$$D_0 = \overline{A} \cdot \overline{B}$$

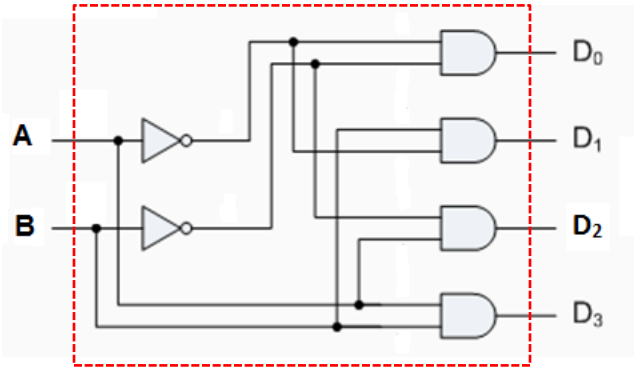
$$D_1 = \overline{A} \cdot B$$

$$D_2 = A \cdot \overline{B}$$

$$D_3 = A \cdot B$$

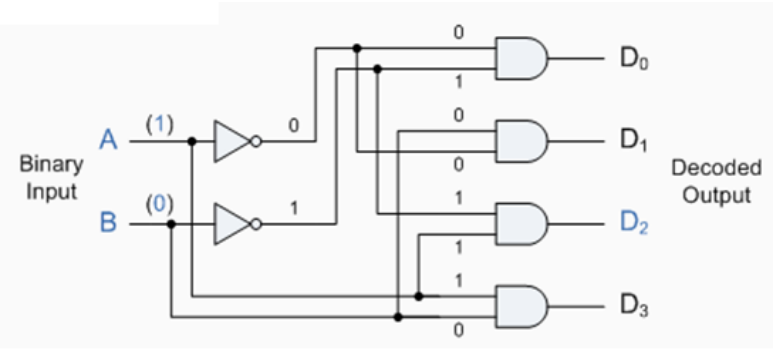
2-to-4 Binary Decoders

- **Combinational Logic Circuit:**



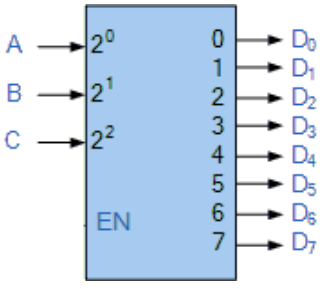
Example

- If the input is 10, the output is D2



3-to-8 Binary Decoders

- In the three-to-eight decoder circuit, the three inputs are decoded into eight outputs, each representing one of the minterms of the three input variables.
- The three inverters provide the complement of the inputs, and each one of the eight *AND* gates generates one of the minterms.
- A particular application of this decoder is binary - to - octal conversion.



3-to-8 Binary Decoders

- **Truth Table:**

Input			Output							
X	Y	Z	D0	D1	D2	D3	D4	D5	D6	D7
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

3-to-8 Binary Decoders

- **Simplified functions:**

$D0 = x'y'z'$

$D1 = x'y'z$

$D2 = x'yz'$

$D3 = x'yz$

$D4 = xy'z'$

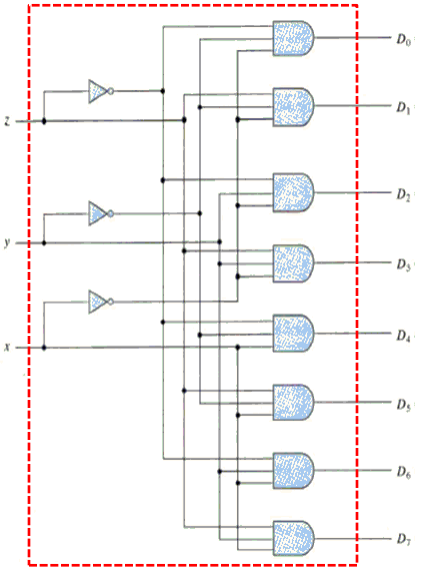
$D5 = xy'z$

$D6 = xyz'$

$D7 = xyz$

3-to-8 Binary Decoders

- **Combinational logic circuit:**



Example: binary to Octal

- The input variables represent a binary number. and the outputs represent the eight digits of a number in the octal number system.
- However, a three to eight line decoder can be used for decoding any three bit code to provide eight outputs, one for each element of the code.
- The operation of the decoder may be clarified by the truth table.
- For each possible input combination, there are seven outputs that *are* equal to 0 and only one that is equal to I.
- The output whose value is equal to 1 represents the minterms equivalent of the binary number currently available the input lines.

Decoder Enable Input

- Enable inputs must be on for the decoder to function, otherwise its outputs assume a single "disabled" output code word.
- Decoding is necessary in applications such as data multiplexing, 7 segment display and memory address decoding.

Decoder Enable Input

- Some binary decoders have an additional input labeled "Enable" that controls the outputs from the device. This allows the decoders outputs to be turned "ON" or "OFF".
- An alternative way of looking at the decoder circuit is to
  - inputs A, B and C as address signals. Each combination of A, B or C defines a unique address which can access a location having that address.
  - Enable Line:
    - If enable=0 , decoder is off. It means all output lines are zero
    - If **enable=1**, decoder **is on** and depending on input, the corresponding output line is 1, all other lines are 0.

Decoder Enable Input

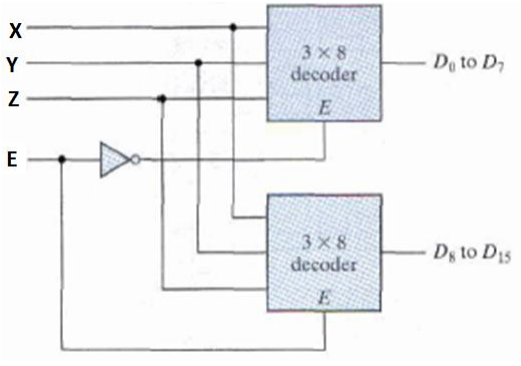
EN	A	B	D0	D1	D2	D3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

• In this table, note that whenever EN=0, the outputs are always 0, **regardless of inputs S1 and S0**
We can abbreviate the table by writing x's in the input columns for S1 and S0.

EN	A	B	D0	D1	D2	D3
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

4-to-16 Binary Decoders

- Decoders with enable inputs can be connected together to form a larger decoder circuit.
- We can connect two 3-to-8 Binary Decoders to form 4-to-16 Binary Decoders using the Enable Line as follows:



4-to-16 Binary Decoders

- When *E* = 0
  - the top decoder is enabled and the other is disabled.
  - The bottom decoder outputs all 0's, and the top eight outputs generate minterm *0000* to 0111.
- When *E* = 1
  - the enable conditions are reversed: The bottom decoder outputs generate minterm I000 to 1111,
  - the outputs of *the* top decoder *are* all 0s.
- This example demonstrates the usefulness of enable inputs in decoders and other combinational logic components.
- In *general,* enable inputs are a convenient feature for interconnecting two or more standard components for the purpose of combining them into a similar function with more inputs and outputs.

Decoder With NAND

- The Circuit operates with complemented outputs and a complement enable input.
- The decoder is enabled when *E* is equal to *0* (i.e., active-low enable). As indicated by the truth table, only one output *can* be *equal* to 0 *at* any given time; all other outputs are equal to 1.
- The output whose value is equal to 0 represents the minterm selected by inputs *A* and *B.*
- The circuit is disabled when *E* is equal to 1, regardless *of* the values of the other two inputs. When the circuit is disabled, none of the outputs are *equal to 0* and none of the minterms are selected.

2-to-4 Binary Decoders with NAND

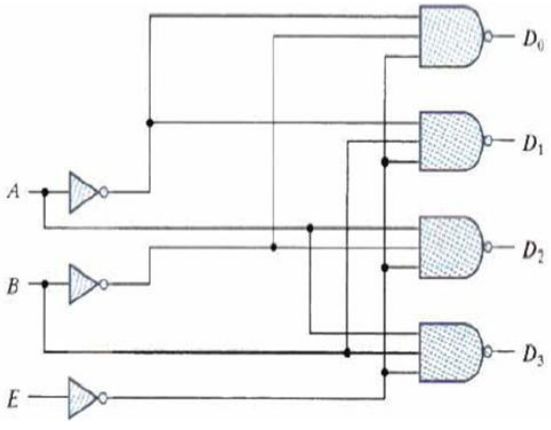
- A two-to-four decoder with an enable input constructed with NAND gates:

- **Truth Table:**

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Decoder

- **Combinational Circuit:**



Decoder Block

- A decoder block provides abstraction:
  - You can use the decoder as long as you know its truth table or equations, without knowing exactly what's inside.
  - It makes diagrams simpler by hiding the internal circuitry.
  - It simplifies hardware reuse. You don't have to keep rebuilding the decoder from scratch every time you need it.
- These blocks are like functions in programming!

Addition using Decoders Decoder-based adder

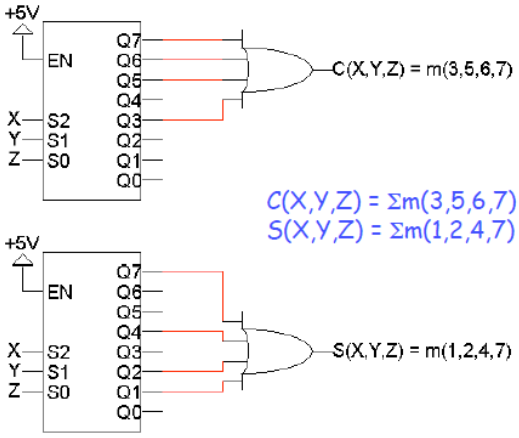
- Let's make a circuit that adds three 1-bit inputs X, Y and Z.
- We will need two bits to represent the total; let's call them C and S, for "carry" and "sum." Note that C and S are two separate functions of the same inputs X, Y and Z.
- Here is a truth table and sum-of-minterms equations for C and S.

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$C(X,Y,Z) = \Sigma m(3,5,6,7)$
$S(X,Y,Z) = \Sigma m(1,2,4,7)$

Addition using Decoders
Decoder-based adder

- Here, two 3-to-8 decoders implement C and S as sums of minterms.

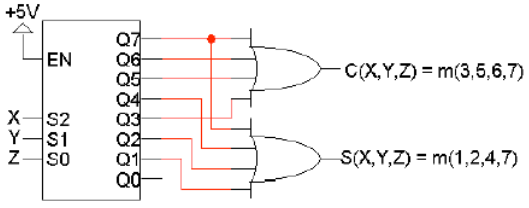


$C(X,Y,Z) = \Sigma m(3,5,6,7)$
$S(X,Y,Z) = \Sigma m(1,2,4,7)$

+5v means that the decoder always active.

Addition using Decoders
Decoder-based adder

- the two functions C and S both have the same inputs, we could use just one decoder instead of two.



$C(X,Y,Z) = \Sigma m(3,5,6,7)$
$S(X,Y,Z) = \Sigma m(1,2,4,7)$

Decoder Application

Decoder Application
Seven-segment display

BCD

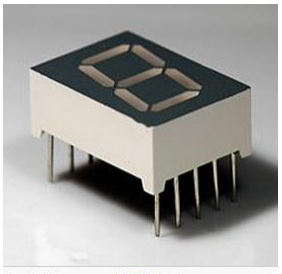
Binary Coded Decimal (BCD):

- When dealing with decimal numbers BCD code is used.
- It is a class of binary encodings of decimal numbers where each decimal digit is represented by a fixed number of bits, usually four or eight, although other sizes (such as six bits) have been used historically. Special bit patterns are sometimes used for a sign or for other indications (e.g., error or overflow).
- BCD's main advantage is its more accurate representation and rounding of decimal quantities as well as an ease of conversion into human-readable representations.
- How are decimal numbers presented in BCD?

Decimal	Binary	BCD
9	1001	1001
19	10011	(0001)(1001)

Seven-segment display

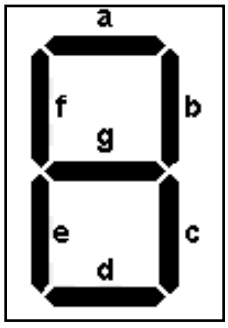
- A **seven-segment display** (**SSD**), or **seven-segment indicator**, is a form of electronic display device for displaying decimal numerals.
- Seven-segment displays are widely used in digital clocks, electronic meters, and other electronic devices for displaying numerical information.



typical 7-segment LED display component

Seven-segment display

- The seven segments are arranged as a rectangle of two vertical segments on each side with one horizontal segment on the top, middle, and bottom. Additionally, the seventh segment bisects the rectangle horizontally.

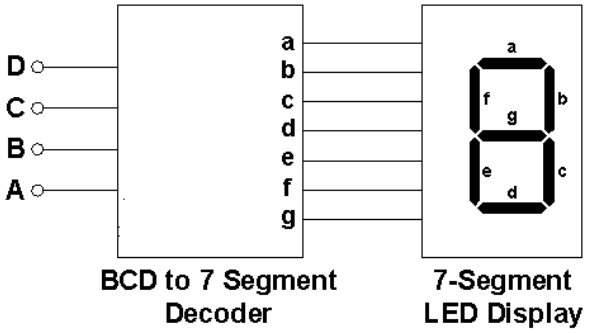


Seven-segment display

- BCD to 7-Segment Display Decoder
  - A **Decoder** IC, is a device which converts one digital format into another and the most commonly used device for doing this is the Binary Coded Decimal (BCD) to 7-Segment Display Decoder. 7-segment **LED** (Light Emitting Diode) or **LCD** (Liquid Crystal) displays, provide a very convenient way of displaying information or digital data in the form of numbers, letters or even alpha-numerical characters and they consist of 7 individual LED's (the segments), within one single display package.

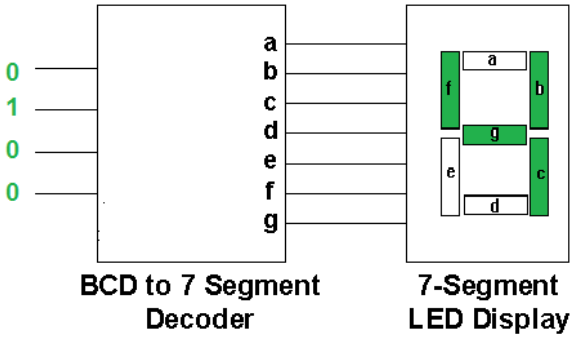
Seven-segment display

- Diagram of a display module with a BCD to 7-Segment Decoder



Seven-segment display

- For example, if the input is 0100 the output should be ***4.***



Seven-segment display

- **Truth Table**

BCD				Decoder Outputs							7-Segment Display Outputs
A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9

Seven-segment display

- **Simplify each output function using k-map**
  - Note that :

  the numbers from 10 To 15 will be don't care conditions.
  - Let simplify the **"e"** function:

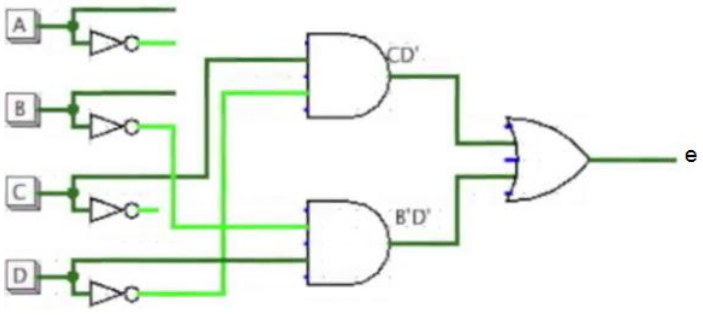
AB \ CD	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	x	x	x	x
10	1	0	x	x

e= B'D' + CD'

A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x

Seven-segment display

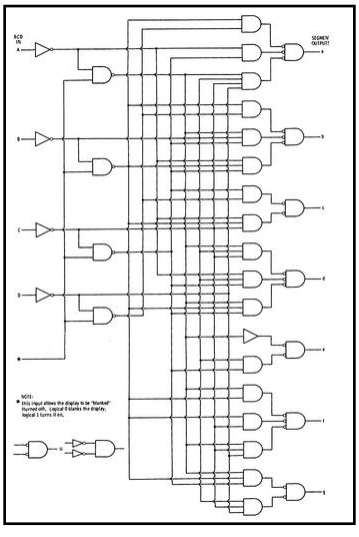
- We draw the logic circuit for “e” output:



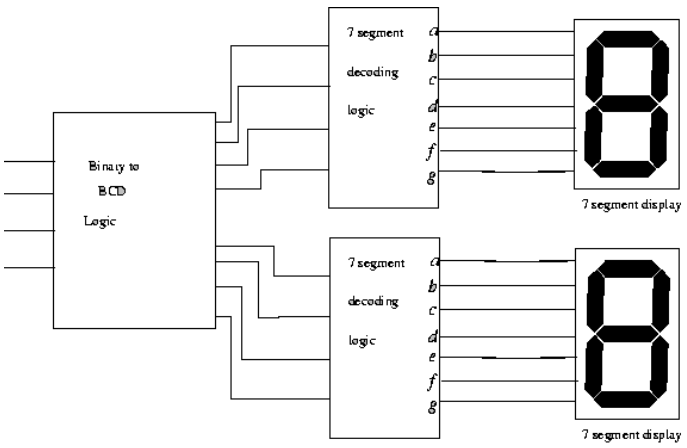
Seven-segment display

- After simplifying all output minterms:
- To see larger image Please visit:

http://thediagram.com/7_2/bcd_decoder.html



Seven-segment display



Block diagram of how to display a binary number

Decoder Application

Memory Address Decoder

Memory Address Decoder

- **Memory Address Decoder.**
  - **Binary Decoders** are most often used in more complex digital systems to access a particular memory location based on an "address" produced by a computing device. In modern microprocessor systems the amount of memory required can be quite high and is generally more than one single memory chip alone. One method of overcoming this problem is to connect lots of individual memory chips together and to read the data on a common "Data Bus". In order to prevent the data being "read" from each memory chip at the same time, each memory chip is selected individually one at time and this process is known as **Address Decoding**.

Helpful Sites

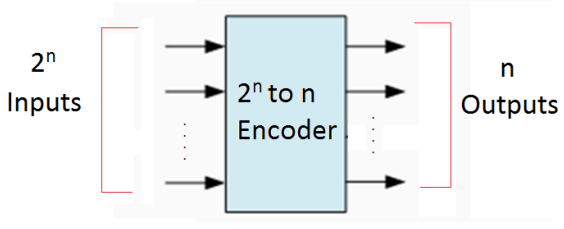
- Half adder:
  - http://www.ustudy.in/node/3035
- Adder/subtractor:
  - http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/20-arithmetic/40-addsub/add-sub.html
- Decoder:
  - https://filebox.ece.vt.edu/~jgtront/introcomp/decoder.swf

Encoders

By
Dareen Hamoudeh
From
www.asic-world.com

Simple Encoder

- An encoder is a combinational circuit that performs the inverse operation of a decoder.
- If a device output code has fewer bits than the input code has, the device is usually called an encoder. e.g. 2^n-to-n, priority encoders.
- The simplest encoder is a 2^n-to-n binary encoder, where it has only one of 2^n inputs = 1 and the output is the n-bit binary number corresponding to the active input.



Octal-to-Binary Encoder

- Octal-to-Binary take 8 inputs and provides 3 outputs, thus doing the opposite of what the 3-to-8 decoder does.
- At any one time, only one input line has a value of 1.
- **Truth Table→**

Input								Output		
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_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

Octal-to-Binary Encoder

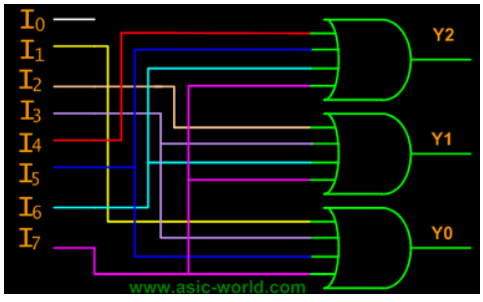
- For an 8-to-3 binary encoder with inputs I0-I7 the **logic expressions** of the outputs Y0-Y2 are:

$Y0 = I_1 + I_3 + I_5 + I_7$

$Y1 = I_2 + I_3 + I_6 + I_7$

$Y2 = I_4 + I_5 + I_6 + I_7$

Circuit



Decimal-to-Binary Encoder

- Decimal-to-Binary take 10 inputs and provides 4 outputs, thus doing the opposite of what the 4-to-10 decoder does.
- At any one time, only one input line has a value of 1.

Truth Table

I0	I1	I2	I3	I4	I5	I6	I7	I8	I9	Y3	Y2	Y1	Y0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

functions Y3, Y2, Y1 and Y0 as given below.

- Y3 = I8 + I9
- Y2 = I4 + I5 + I6 + I7
- Y1 = I2 + I3 + I6 + I7
- Y0 = I1 + I3 + I5 + I7 + I9

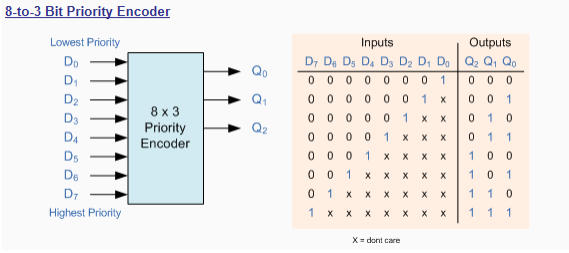
Priority Encoder

If we look carefully at the Encoder circuits that we got, we see the following limitations:

- If more than two inputs are active simultaneously, the output is unpredictable or rather it is not what we expect it to be.
- This ambiguity is resolved if priority is established so that only one input is encoded, no matter how many inputs are active at a given point of time.
- The priority encoder includes a priority function. The operation of the priority encoder is such that if two or more inputs are active at the same time, the input having the highest priority will take precedence.

Priority Encoder

- The **Priority Encoder** solves the problems mentioned above by allocating a priority level to each input. The *priority encoders* output corresponds to the currently active input which has the highest priority. So when an input with a higher priority is present, all other inputs with a lower priority will be ignored. The priority encoder comes in many different forms with an example of an 8-input priority encoder along with its truth table shown below.



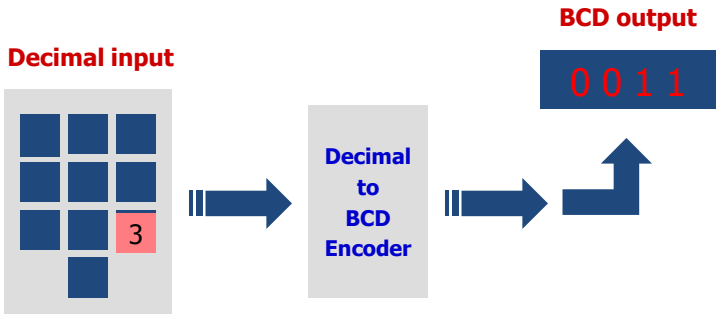
Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

X = dont care

Encoder Applications

- **Keyboard Encoder**
  - Priority encoders can be used to reduce the number of wires needed in a particular circuits or application that have multiple inputs. For example, assume that a microcomputer needs to read the 104 keys of a standard QWERTY keyboard where only one key would be pressed either "HIGH" or "LOW" at any one time. One way would be to connect all 104 wires from the keys directly to the computer but this would be impractical for a small home PC, but another better way would be to use a priority encoder. The 104 individual buttons or keys could be encoded into a standard ASCII code of only 7-bits (0 to 127 decimal) to represent each key or character of the keyboard and then inputted as a much smaller 7-bit B.C.D code directly to the computer. Keypad encoders such as the 74C923 20-key encoder are available to do just that.

Encoder Applications



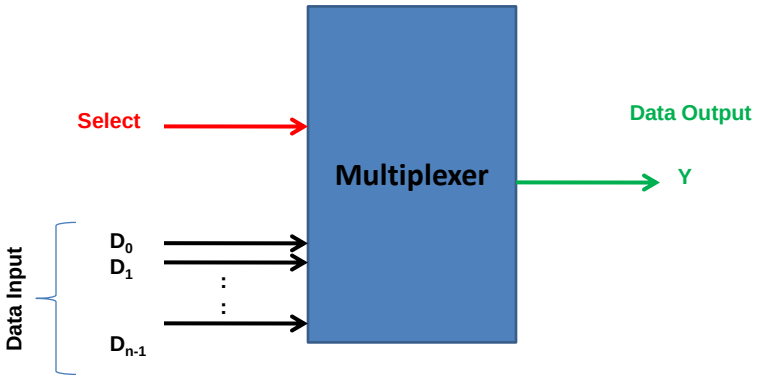
- **Encoders are available in IC form.**
- **This encoder translates from decimal input to BCD output.**

Multiplexer

Multiplexer

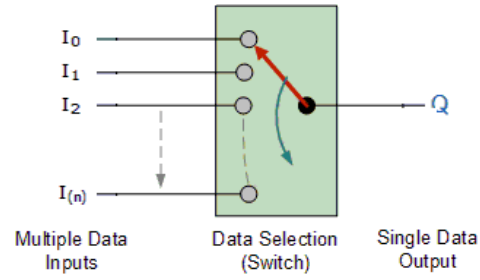
- A multiplexer (MUX) is a digital **switch** which connects data from one of n sources to the output.
- A number of select inputs determine which data source is connected to the output.
- The next slide shows a block diagram of MUX with n data sources of b bits wide and s bits wide select line.
- MUX acts like a digitally controlled multi-position switch where the binary code applied to the select inputs controls the input source that will be switched on to the output.
- At any given point of time only one input gets selected and is connected to output, based on the select input signal.
- A multiplexer is also called a **data selector**.

Multiplexer



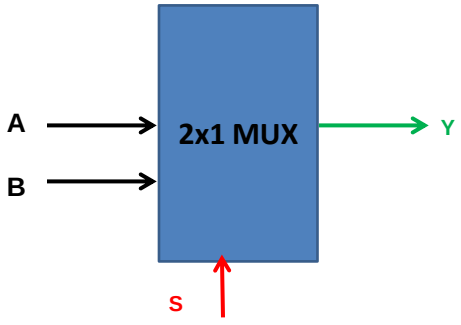
Multiplexer

- The operation of a multiplexer can be better explained using a mechanical switch.
- This **rotary** switch can touch any of the inputs, which is connected to the output. As you can see at any given point of time only one input gets transferred to output

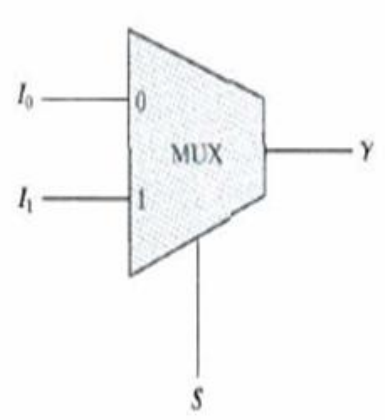


2x1 Multiplexer

- Each 2 input lines A to B is applied to one input of an AND gate.
- Selection lines S are decoded to select a particular AND gate.



Block Diagram



2x1 Multiplexer

First you have to Know selection Truth table:

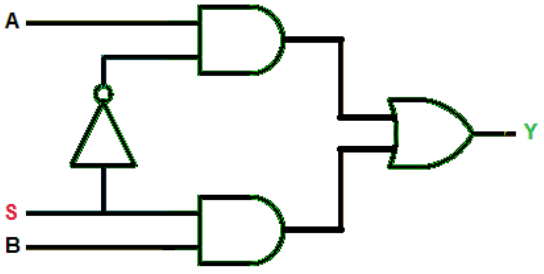
S	Y
0	A
1	B

Truth Table:

S	A	B	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

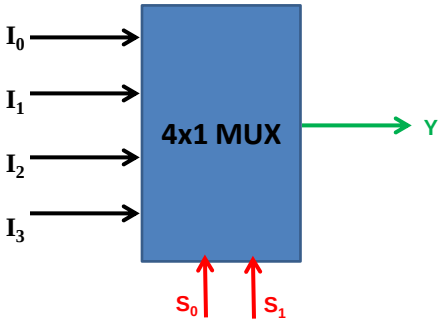
2x1 Multiplexer

- Simplified output function after drawing k-map: Y= AS' + BS
- Then we draw the Circuit:



4x1 Multiplexer

- each of 4 input lines I_0 to I_3 is applied to one input of an AND gate.
- Selection lines S_0 and S_1 are decoded to select a particular AND gate.



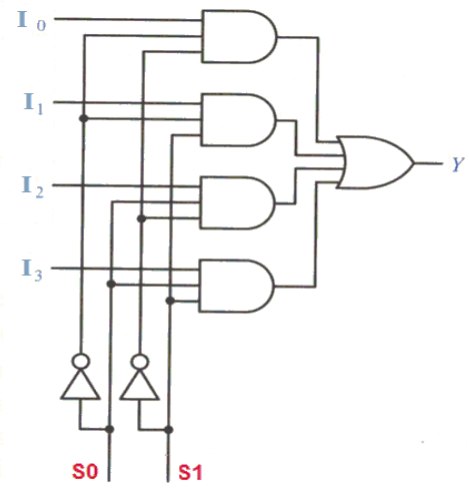
4x1 Multiplexer

First you have to Know selection Truth table:

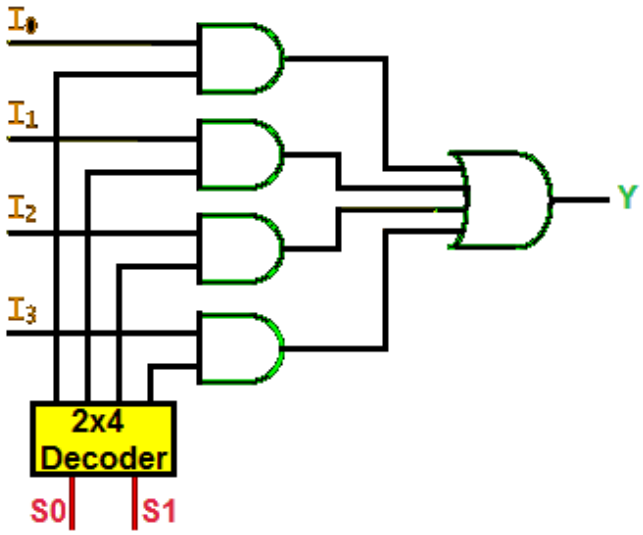
S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Without finding simplified function for the output, we notice that the output is only one for each of the selection combination, the circuit will be:

4x1 Multiplexer



4x1 Multiplexer



Larger Multiplexers

- Larger multiplexers can be constructed from smaller ones. An 8-to-1 multiplexer can be constructed from smaller multiplexers as shown below.

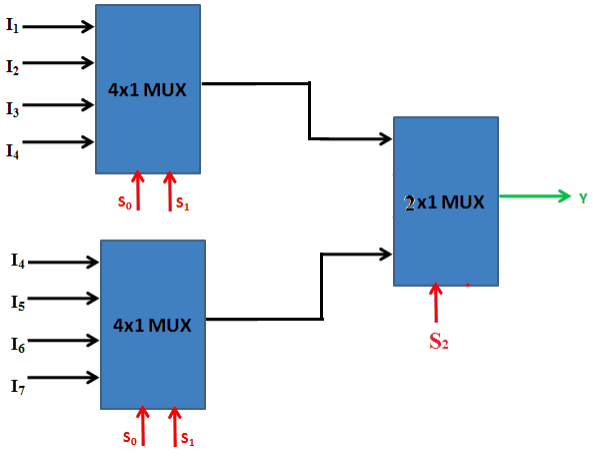
8x1 Multiplexer

Truth Table:

S_2	S_0	S_1	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

8x1 Multiplexer

Circuit:



16x1 Multiplexer