Combinational Logic circuit

Samia Sultana

Lecturer

Dept. of CSE

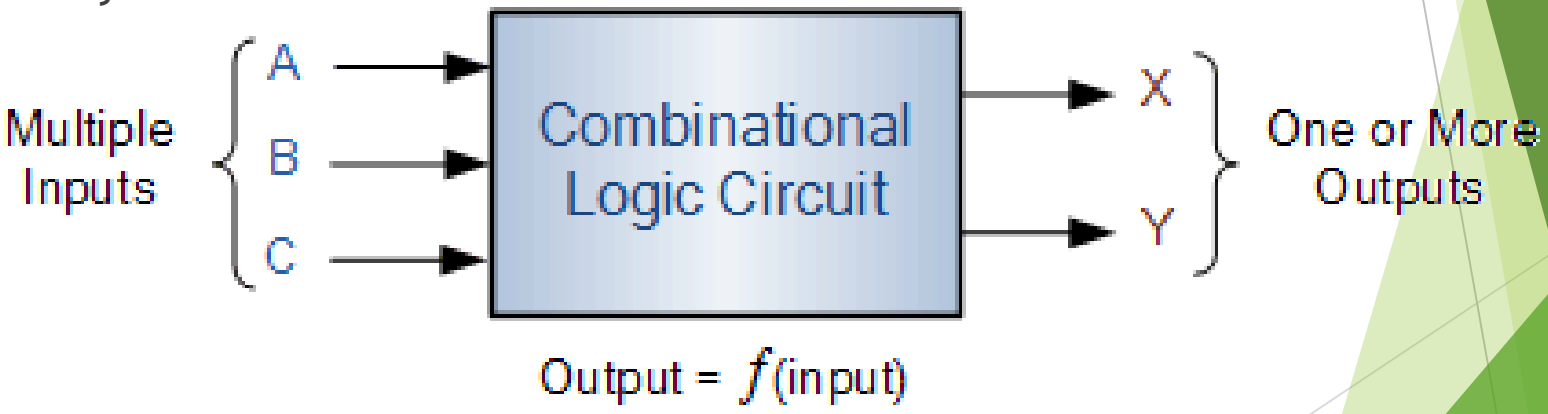
Varendra University

http://www.electronics-tutorials.ws

www.electronicshub.org

Combinational Logic circuit

- In digital **circuit** theory, **combinational logic**(sometimes also referred to as time-independent **logic**) is a type of digital **logic** which is implemented by Boolean **circuits**, where the output is a pure function of the present input only.



Combinational circuit

- **Combinational Logic Circuits** are made up from basic logic **NAND**, **NOR** or **NOT** gates that are “combined” or connected together to produce more complicated switching circuits.
- An example of a combinational circuit is a decoder, which converts the binary code data present at its input into a number of different output lines, one at a time producing an equivalent decimal code at its output.

- The three main ways of specifying the function of a combinational logic circuit are:
- 1. **Boolean Algebra** – This forms the algebraic expression showing the operation of the logic circuit for each input variable either True or False that results in a logic “1” output.
- 2. **Truth Table** – A truth table defines the function of a logic gate by providing a concise list that shows all the output states in tabular form for each possible combination of input variable that the gate could encounter.
- 3. **Logic Diagram** – This is a graphical representation of a logic circuit that shows the wiring and connections of each individual logic gate, represented by a specific graphical symbol, that implements the logic circuit.

Difference between Combinational Logic circuit and sequential Logic Circuit

- **Unlike Sequential Logic Circuits whose outputs are dependent on both their present inputs and their previous output state giving them some form of Memory, the outputs of Combinational Logic Circuits are only determined by the logical function of their current input state, logic "0" or logic "1", at any given instant in time.**
- In a **Combinational Logic Circuit**, the output is dependent at all times on the combination of its inputs. So if one of its inputs condition changes state, from 0-1 or 1-0, so too will the resulting output as by default combinational logic circuits have "no memory", "timing" or "feedback loops" within their design.

The Binary Adder

- **Another common and very useful combinational logic circuit which can be constructed using just a few basic logic gates allowing it to add together two or more binary numbers is the Binary Adder.**

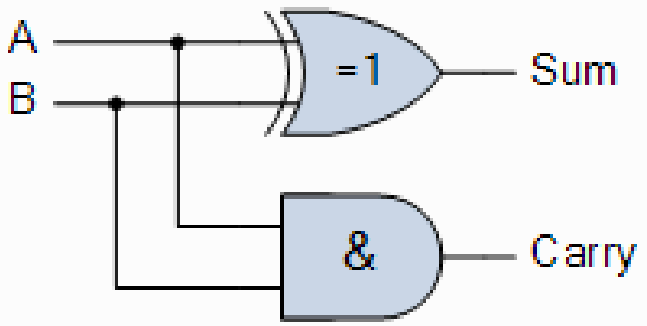
The Binary Adder

- When each column is added together a ***carry*** is generated if the result is ***greater or equal to 10***, the ***base*** number. This carry is then added to the result of the addition of the next column to the left and so on, simple school math's addition, add the numbers and carry.
- The adding of binary numbers is exactly the same idea as that for adding together decimal numbers but this time a carry is only generated when the result in any column is greater or equal to "2", the base number of binary. In other words 1 + 1 creates a carry.

Half Adder

- A half adder is a logical circuit that performs an addition operation on two binary digits. The half adder

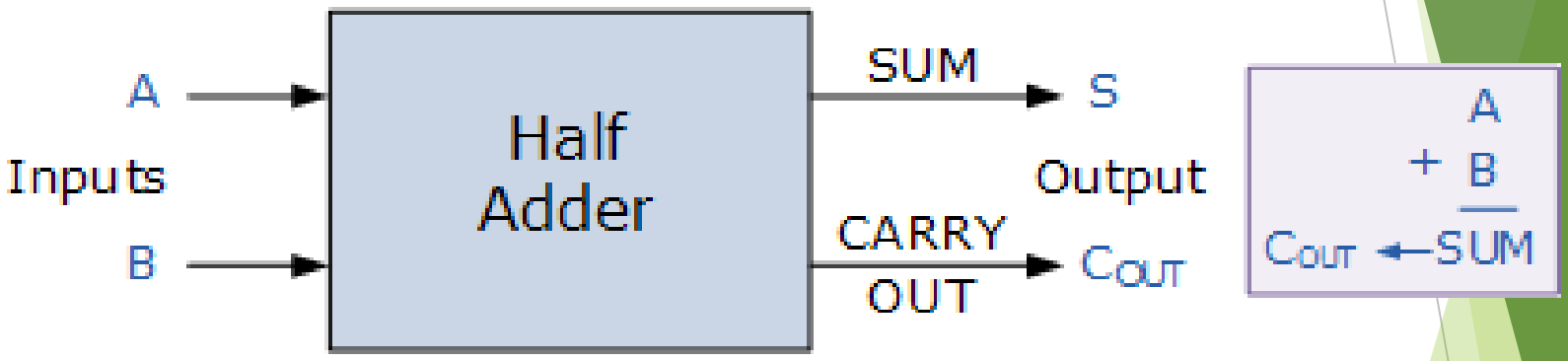
Circuit Diagram



Truth Table			
B	A	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half Adder Truth Table with Carry-Out

Block diagram



Boolean expression

- The Boolean expression for a half adder is as follows:
- For the **SUM** bit: SUM = A XOR B = A $\oplus$ B
- For the **CARRY** bit: CARRY = A AND B = A.B

Disadvantage of Half adder

- One major disadvantage of the *Half Adder* circuit when used as a binary adder, is that there is no provision for a "Carry-in" from the previous circuit when adding together multiple data bits.
- For example, suppose we want to add together two 8-bit bytes of data, any resulting carry bit would need to be able to "ripple" or move across the bit patterns starting from the least significant bit (LSB).
- One simple way to overcome this problem is to use a **Full Adder** type binary adder circuit.

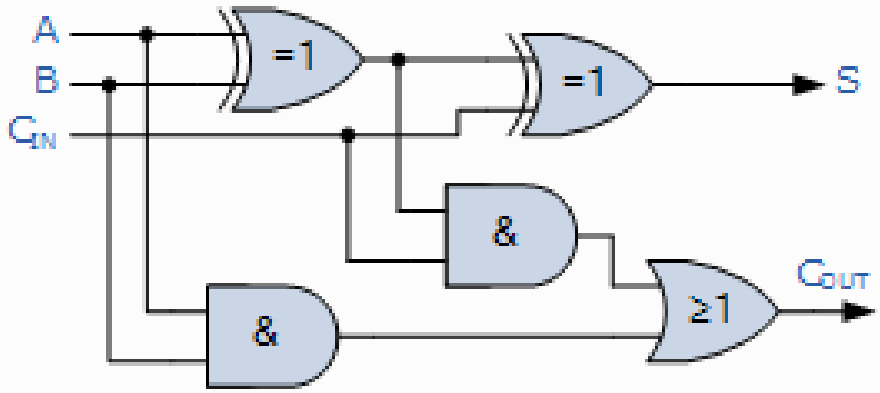
A Full Adder Circuit

- The main difference between the **Full Adder** and the previous **Half Adder** is that a full adder has three inputs. The same two single bit data inputs A and B as before plus an additional *Carry-in* (C-in) input to receive the carry from a previous stage as shown below.

- Then the **full adder** is a logical circuit that performs an addition operation on three binary digits and just like the half adder, it also generates a carry out to the next addition column. Then a *Carry-in* is a possible carry from a less significant digit, while a *Carry-out* represents a carry to a more significant digit.

Full Adder Truth Table with Carry

Circuit Diagram



Truth Table				
C-in	B	A	Sum	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

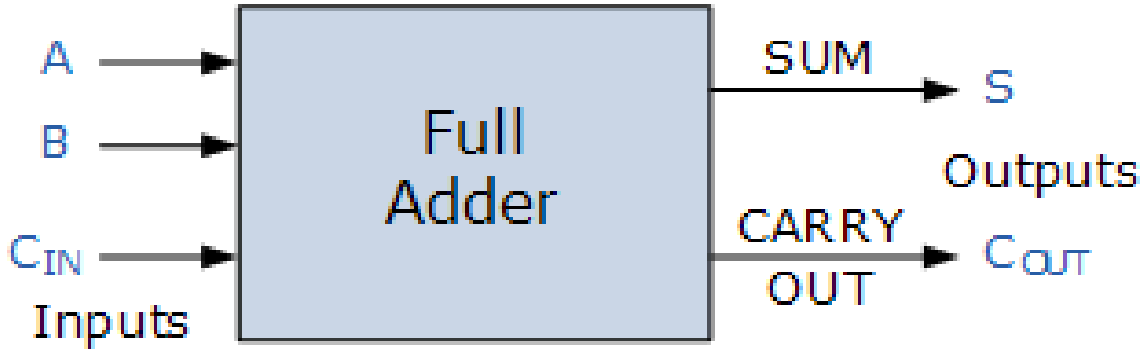
Boolean Expression

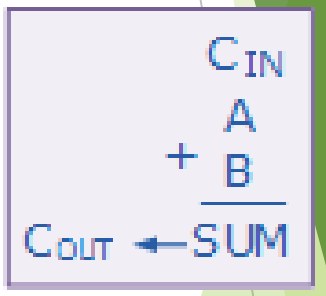
- **For the SUM (S) bit:**

SUM = (A XOR B) XOR Cin = $(A \oplus B) \oplus Cin$

- **For the CARRY-OUT (Cout) bit:**
- CARRY-
  OUT = A AND B OR Cin(A XOR B) = $A.B + Cin(A \oplus B)$

Full Adder Block Diagram



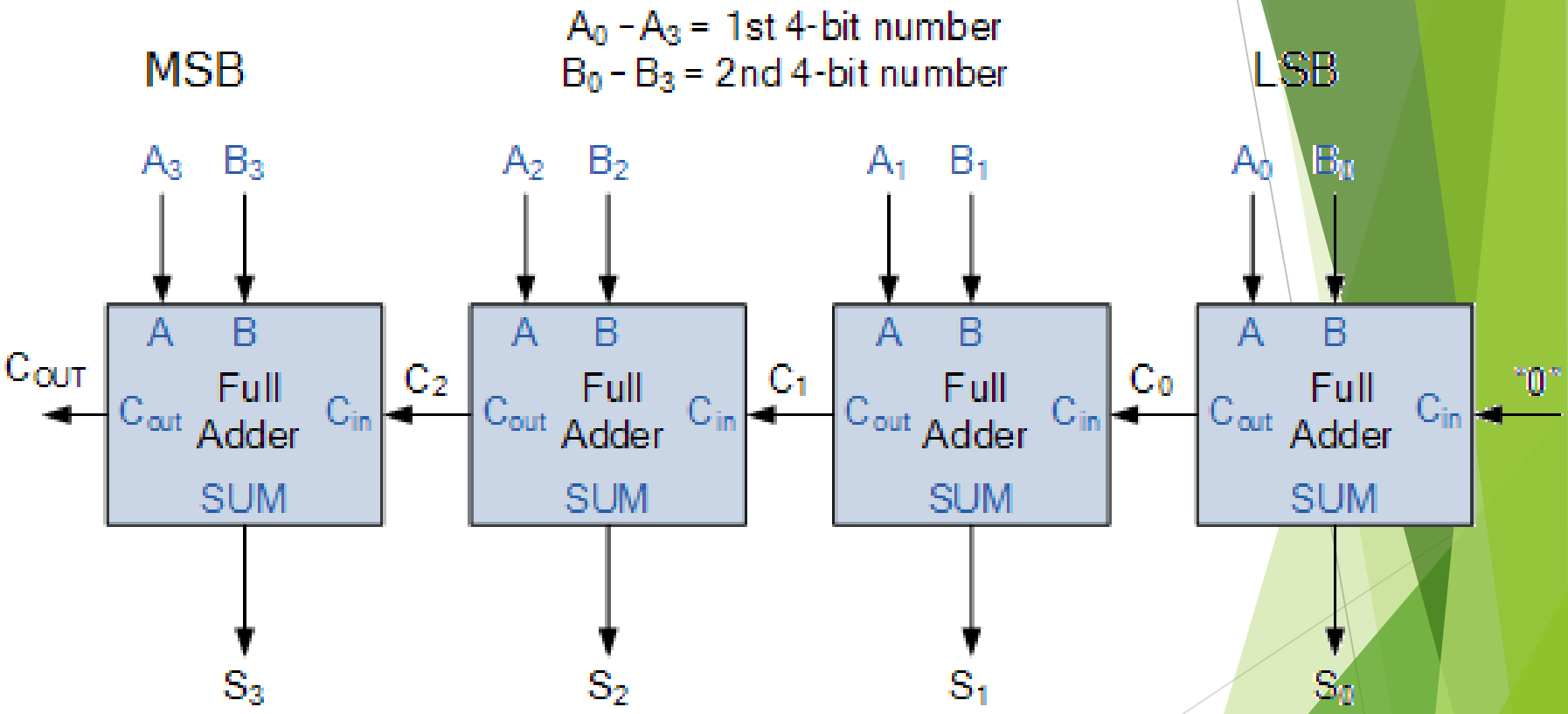


An n-bit Parallel Binary Adder

- We have seen above that single 1-bit binary adders can be constructed from basic logic gates. But what if we wanted to add together two n-bit numbers, then n number of 1-bit full adders need to be connected or “cascaded” together to produce what is known as a **Ripple Carry Adder**.
- A “ripple carry adder” is simply “n”, 1-bit full adders cascaded together with each full adder representing a single weighted column in a long binary addition. It is called a ripple carry adder because the carry signals produce a “ripple” effect through the binary adder from right to left, (LSB to MSB).

- For example, suppose we want to “add” together two 4-bit numbers, the two outputs of the first full adder will provide the first place digit sum (S) of the addition plus a carry-out bit that acts as the carry-in digit of the next binary adder.
- The second binary adder in the chain also produces a summed output (the 2nd bit) plus another carry-out bit and we can keep adding more full adders to the combination to add larger numbers, linking the carry bit output from the first full binary adder to the next full adder, and so forth. An example of a 4-bit adder is given below.

4-bit parallel Adder



The Binary Subtractor

- **The Binary Subtractor is another type of combinational arithmetic circuit that is the opposite of the Binary Adder we looked at in a previous tutorial. As their name implies, a Binary Subtractor is a decision making circuit that subtracts two binary numbers from each other, for example, X – Y to find the resulting difference between the two numbers.**
- Unlike the **Binary Adder** which produces a SUM and a CARRY bit when two binary numbers are added together, the *binary subtractor* produces a DIFFERENCE, D by using a BORROW bit, B from the previous column. Then obviously, the operation of subtraction is the opposite to that of addition.

Binary Subtraction

- We can not directly subtract 8 from 3 in the first column as 8 is greater than 3, so we have to borrow a 10, the base number, from the next column and add it to the minuend to produce 13 minus 8. This "borrowed" 10 is then return back to the subtrahend of the next column once the difference is found. Simple school math's, borrow a 10 if needed, find the difference and return the borrow.
- The subtraction of one binary number from another is exactly the same idea as that for subtracting two decimal numbers but as the *binary number system* is a Base-2 numbering system which uses "0" and "1" as its two independent digits, large binary numbers which are to be subtracted from each other are therefore represented in terms of "0's" and "1's".

Binary Subtraction of Two Bits

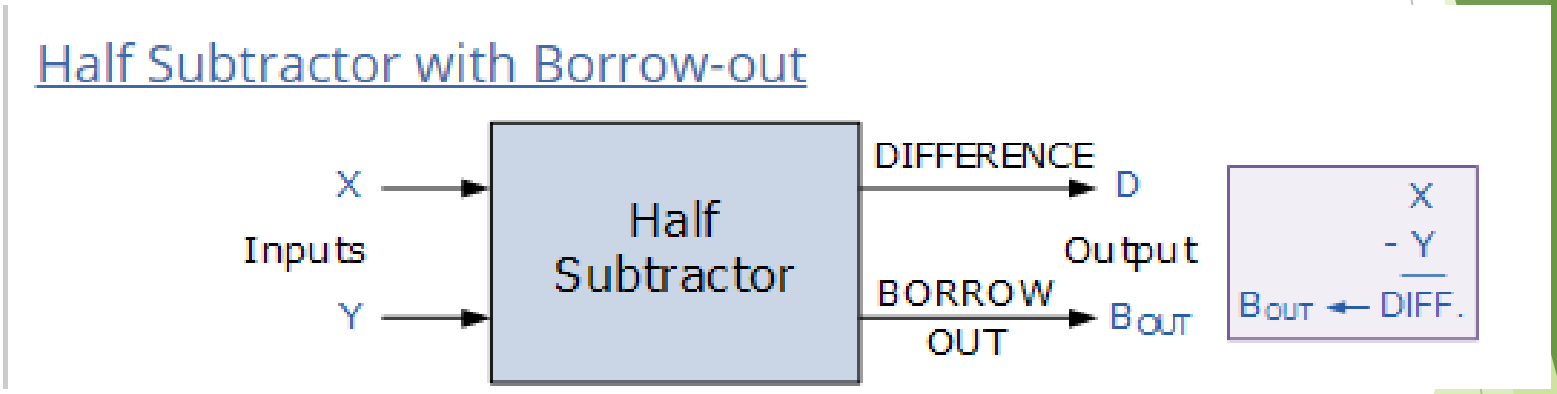
0	1	1	(borrow)1→ 0
− 0	− 0	− 1	− 1
0	1	0	1

123	X	(Minuend)
− 78	Y	(Subtrahend)
45	DIFFERENCE	

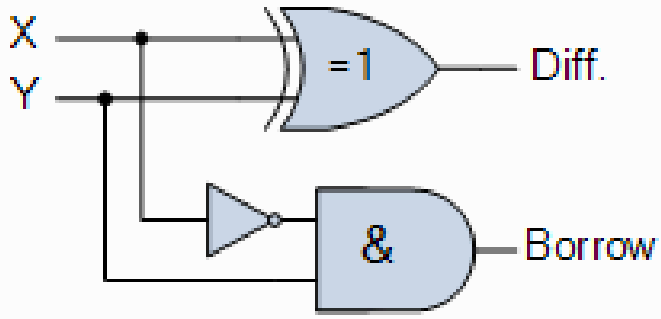
A Half Subtractor Circuit

- A half subtractor is a logical circuit that performs a subtraction operation on two binary digits. The half subtractor produces a sum and a borrow bit for the next stage.
- If we compare the Boolean expressions of the half subtractor with a half adder, we can see that the two expressions for the SUM (adder) and DIFFERENCE (subtractor) are exactly the same and so they should be because of the Exclusive-OR gate function. The two Boolean expressions for the binary subtractor BORROW is also very similar to that for the adders CARRY. Then all that is needed to convert a half adder to a half subtractor is the inversion of the minuend input X.

The block diagram



Circuit Diagram

Symbol	Truth Table			
	Y	X	DIFFERENCE	BORROW
	0	0	0	0
	0	1	1	0
	1	0	1	1
	1	1	0	0

Boolean Expression

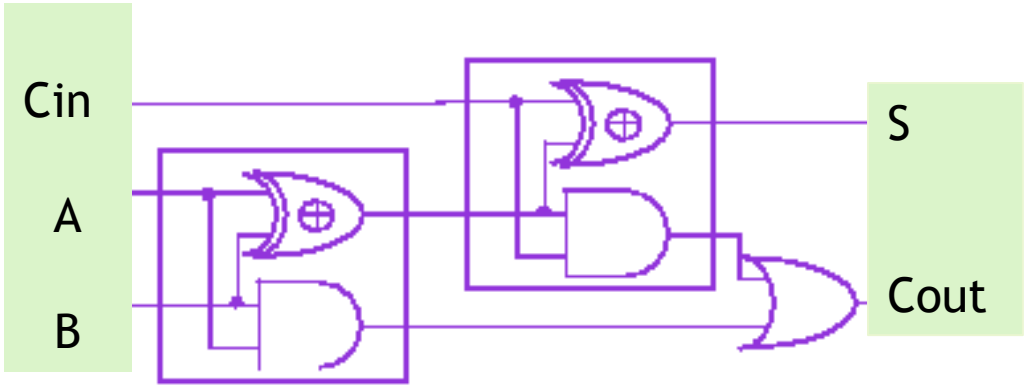
- For the **DIFFERENCE** bit

D = X XOR Y = $X \oplus Y$

- For the **BORROW** bit

B = not-X AND Y = $\overline{X}.Y$

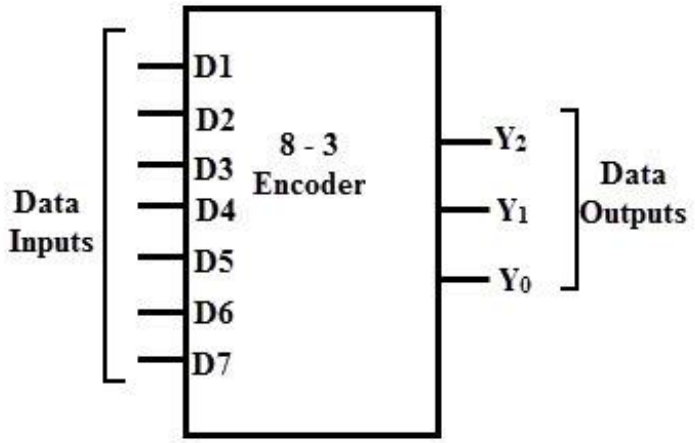
Is this possible to design a full adder from half adder? How?



Encoder

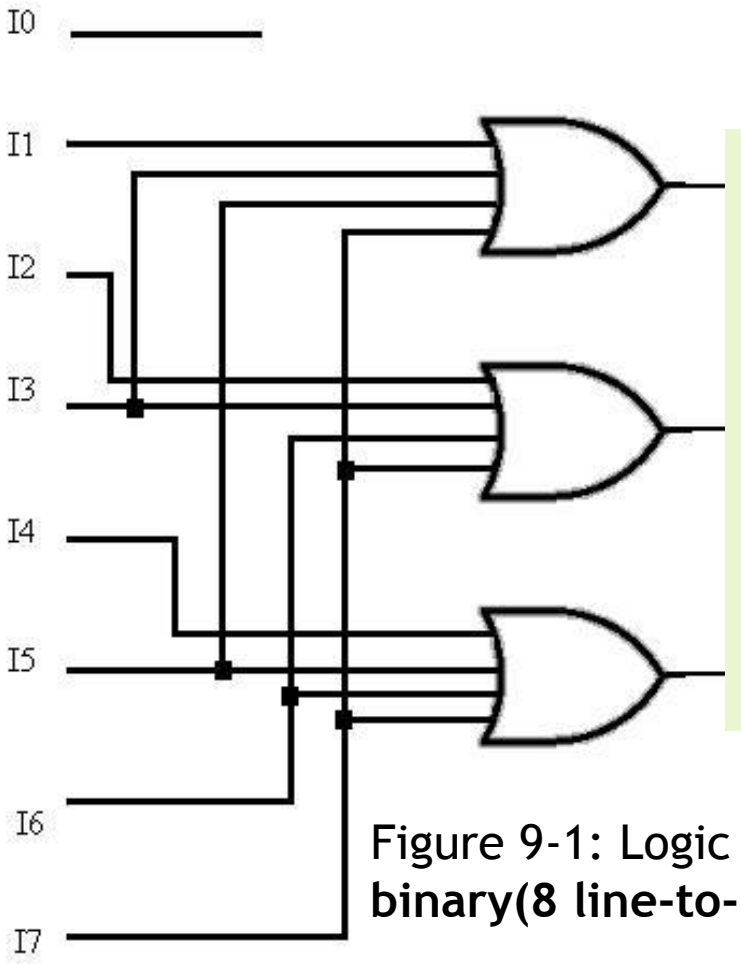
- **Encoder** is a combinational circuit that generates a specific code at its outputs such as binary or BCD in response to one or more active inputs.
- Generally, digital encoders produce outputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines. An “n-bit” binary encoder has 2^n input lines and n-bit output lines with common types that include 4-to-2, 8-to-3 and 16-to-4 line configurations.

Figure: Block diagram of encoder



Logic Circuit

- Notice that I_0 is not connected because the encoder outputs will normally be at 000 when none of the inputs will high.



Y0=I1+I3+I5+I7

Y1=I2+I3+I6+I7

Y2=I4+I5+I6+I7

Figure 9-1: Logic circuit for **an octal to binary(8 line-to-3-line) encoder**

Truth table

Octal Numbar	INPUTS								OUTPUTS		
	D0	D1	D2	D3	D4	D5	D6	D7	a	b	c
0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0	0	1	0	0
5	0	0	0	0	0	1	0	0	1	0	1
6	0	0	0	0	0	0	1	0	1	1	0
7	0	0	0	0	0	0	0	1	1	1	1

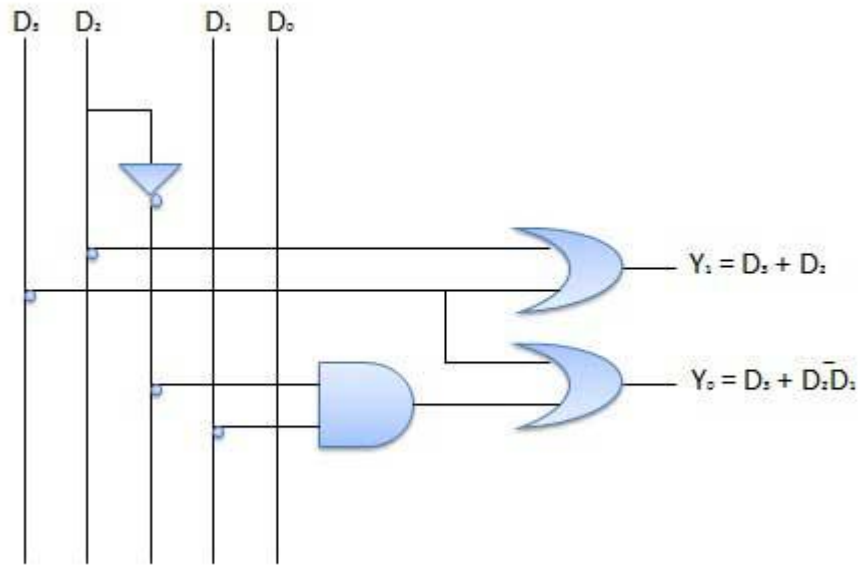
Figure: truth table for **an octal to binary(8 line-to-3-line) encoder**

- Example: 9-5
- **Determine the outputs of the encoder in Figure 9-1 when A3 and A5 are simultaneously HIGH.**
- ***Solution***
- Following through the logic gates, we see that the HIGHs at these two inputs will produce HIGHs at each output, in other words, the binary code 111. Clearly, this is not the code for either activated input.

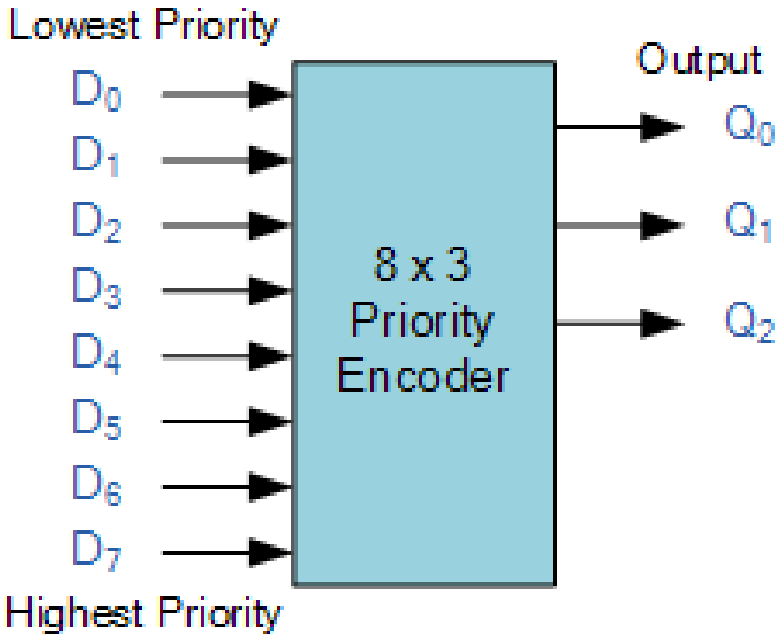
Priority Encoder

- 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:

Simple 4-Input Priority Encoder



Highest	Inputs		Lowest	Outputs	
D_3	D_2	D_1	D_0	Y_0	Y_1
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1



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

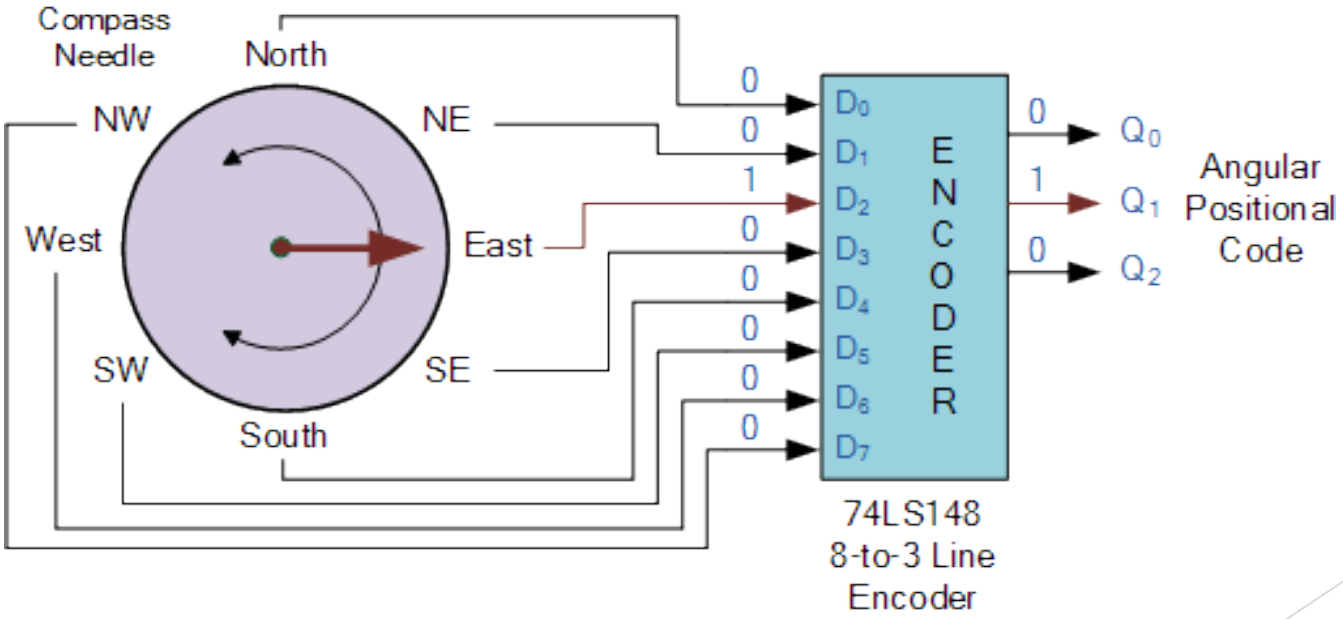
Digital Encoder Applications

- **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 individual keys on the keyboard directly to the computers input but this would be impractical for a small home PC. Another alternative and better way would be to interface the keyboard to the PC using 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 input 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.

Digital Encoder Applications

- **Positional Encoders**
- Another more common application is in magnetic positional control as used on ships navigation or for robotic arm positioning etc. Here for example, the angular or rotary position of a compass is converted into a digital code by a 74LS148 8-to-3 line priority encoder and input to the systems computer to provide navigational data and an example of a simple 8 position to 3-bit output compass encoder is shown below. Magnets and reed switches could be used at each compass point to indicate the needles angular position.

Priority Encoder Navigation

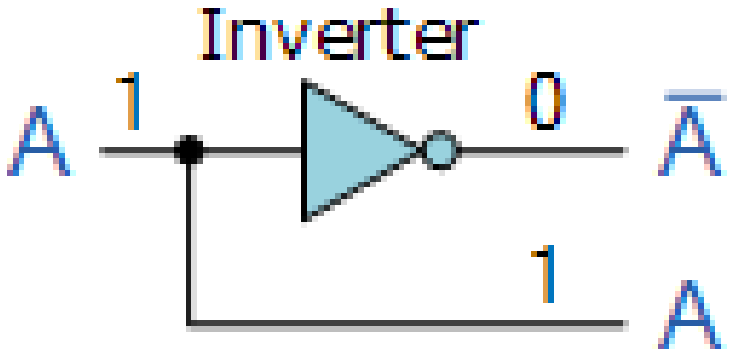


Compass Direction	Binary Output		
	Q_0	Q_1	Q_2
North	0	0	0
North-East	0	0	1
East	0	1	0
South-East	0	1	1
South	1	0	0
South-West	1	0	1
West	1	1	0
North-West	1	1	1

Decoder

- The **Binary Decoder** is another combinational logic circuit constructed from individual logic gates and is the exact opposite to that of an “Encoder” we looked at in the last tutorial. The name “Decoder” means to translate or decode coded information from one format into another, so a digital decoder transforms a set of digital input signals into an equivalent decimal code at its output.
- **Binary Decoders** are another type of digital logic device that has inputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines, so a decoder that has a set of two or more bits will be defined as having an *n*-bit code, and therefore it will be possible to represent 2^n possible values. Thus, a decoder generally decodes a binary value into a non-binary one by setting exactly one of its *n* outputs to logic “1”.

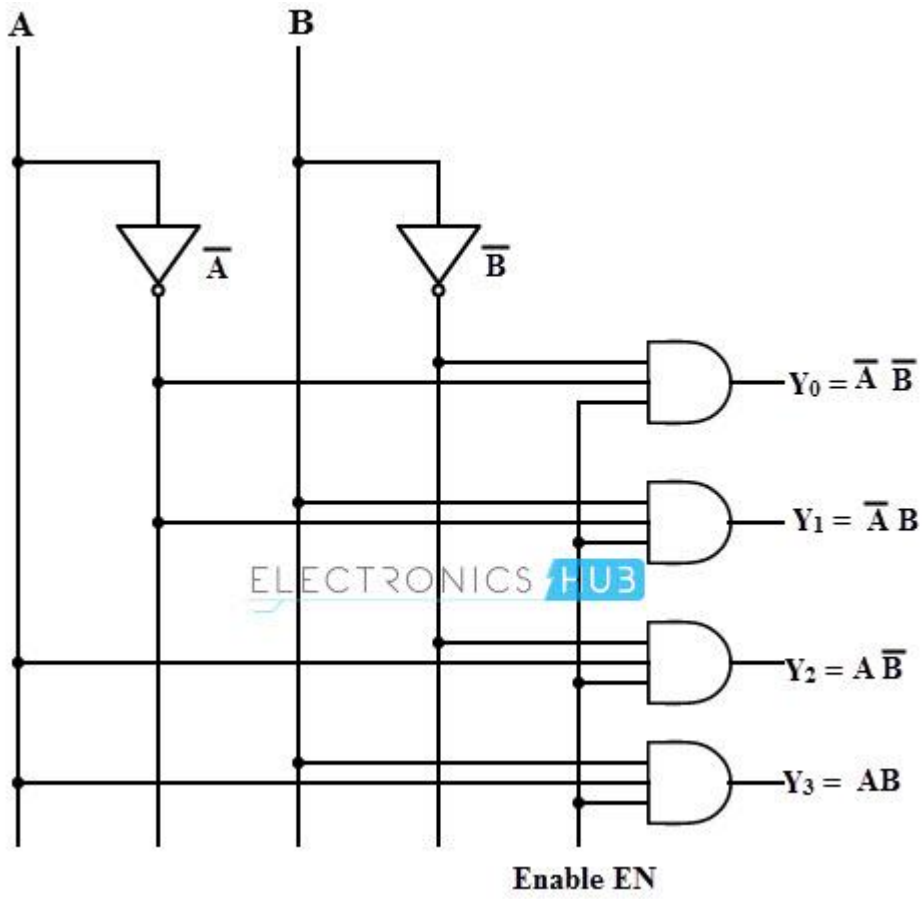
- For example, an inverter (*NOT-gate*) can be classed as a 1-to-2 binary decoder as 1-input and 2-outputs (2^1) is possible because with an input A it can produce two outputs A and A (not-A) as shown.



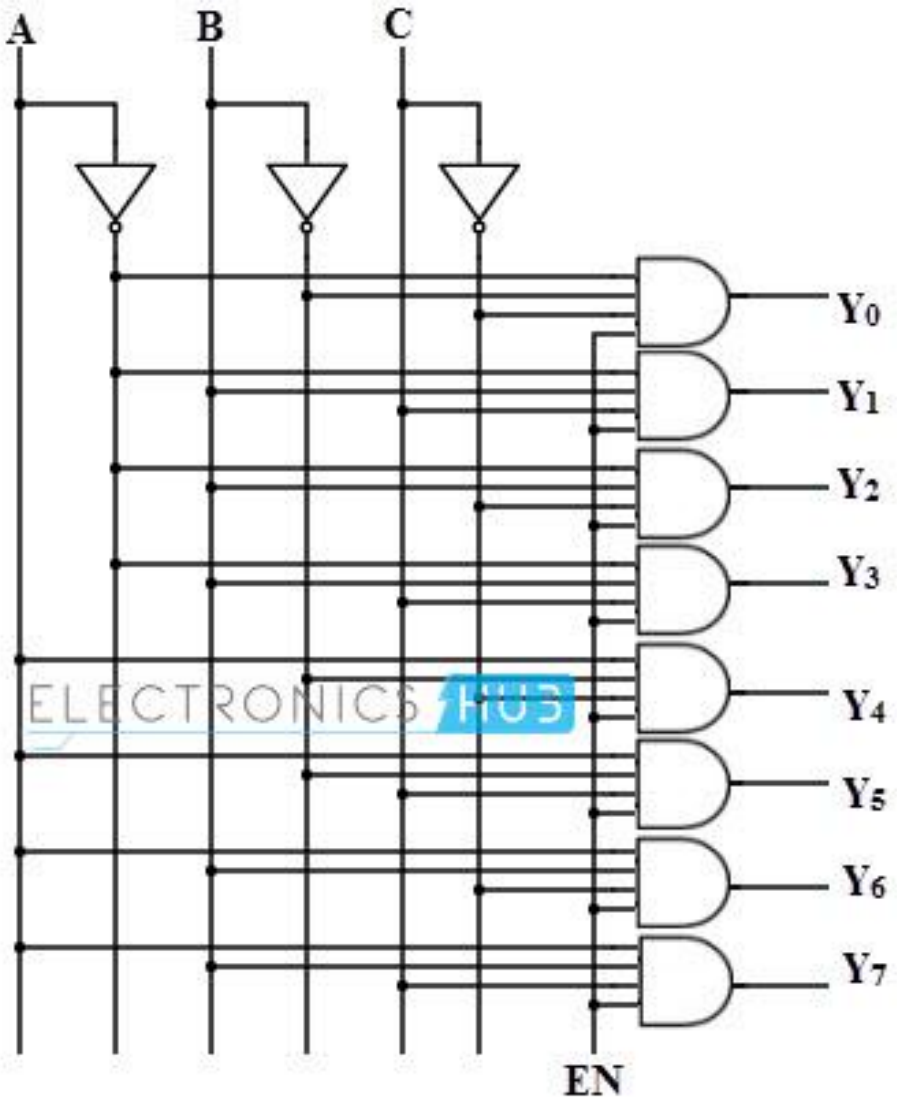
2-to-4 line Decoder

- truth table and logic circuit

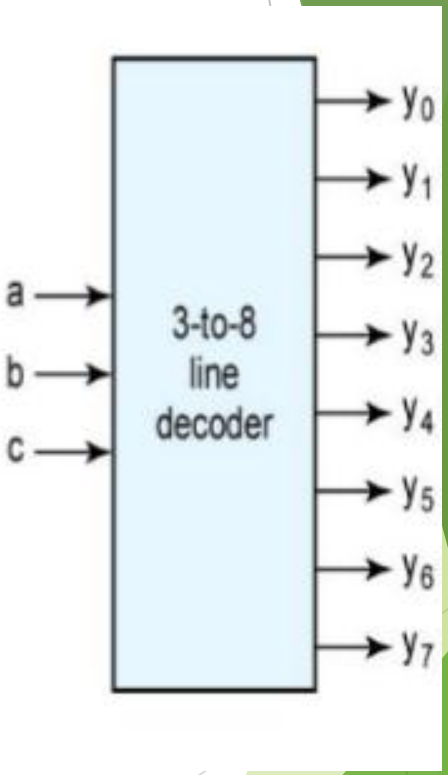
Inputs			Outputs			
EN	A	B	Y_3	Y_2	Y_1	Y_0
0	×	×	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0


A
B
$\overline{A}$
$\overline{B}$
$Y_0 = \overline{A}\ \overline{B}$
$Y_1 = \overline{A}\ B$
$Y_2 = A\ \overline{B}$
$Y_3 = AB$
Enable EN

3-to-8 line Decoder

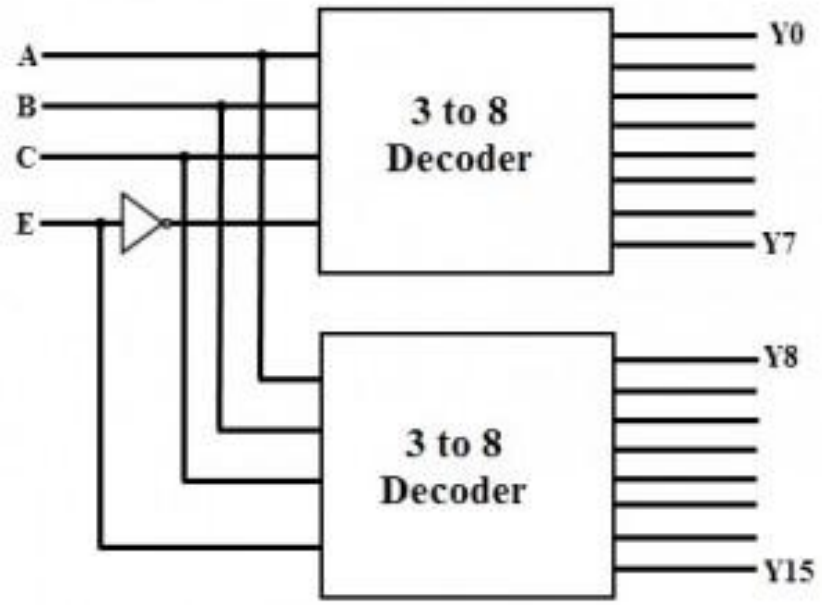


Inputs			Outputs							
X	Y	Z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	**1**	0	0	0	0	0	0	0
0	0	1	0	**1**	0	0	0	0	0	0
0	1	0	0	0	**1**	0	0	0	0	0
0	1	1	0	0	0	**1**	0	0	0	0
1	0	0	0	0	0	0	**1**	0	0	0
1	0	1	0	0	0	0	0	**1**	0	0
1	1	0	0	0	0	0	0	0	**1**	0
1	1	1	0	0	0	0	0	0	0	**1**



Cascading Decoders

- It is possible to combine or cascade two or more decoders to produce a decoder with larger number of input bits with the use of enable input of decoder. The cascade combination of two 3-to-8 line decoder is given below figure. It consists of four inputs A, B, C and Enable E and 16 outputs Y0 to Y7.
- One of the input variable is used as enable input of the first 3-to-4 decoder and this same input is complemented and connected as enable input of the second decoder. The decoder to be enabled is decided by the most significant input variable and other input variables are fed to each decoder.
- When enable input is zero then the top decoder is enabled while the other is disabled. Then the top decoder eight outputs generate the minterms 0000 to 0111. Likewise, when enable is 1, the lower decoder is enabled and top one is disabled. Thus the bottom decoder outputs generate minterms from 1000 to 1111.


A
B
C
E
3 to 8
Decoder
Y0
Y7
3 to 8
Decoder
Y8
Y15

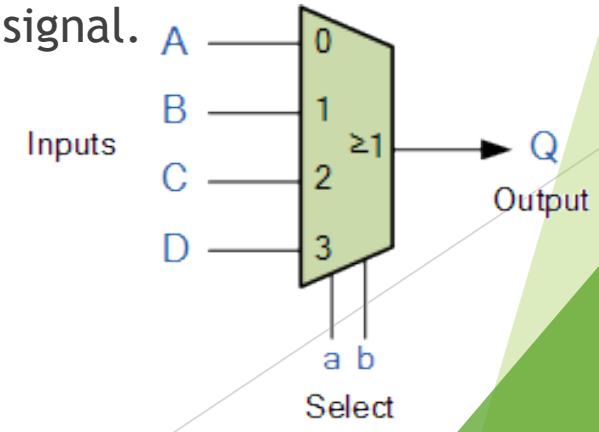
Applications of Decoders

- Decoders are greatly used in applications where the particular output or group of outputs to be activated only on the occurrence of a specific combination of input levels. Very often these input levels are provided by the outputs of a register or counter.
- When the counter or register continuously pulse the decoder inputs, the outputs will be activated sequentially. And these outputs can be used as sequencing signals or timing signals to switch the devices at particular times.
- **Binary to Decimal Decoder**
- Decoders are used to get the decimal digit corresponding to a specific input combination. A BCD number needs 4 binary digits to represent the 0 to 9 decimal digits, thus it consists of 4 input lines. It consists of 10 output lines corresponding to 0 to 9 decimal digits. T
- This type of decoder is also called as a 1 to 10 decoder. For a specific input combination, the output will be activated corresponding to the decimal equivalent of the input combination.

- **Instruction Decoder**
- Another application of the decoder can be found in the control unit of the central processing unit. This decoder is used to decode the program instructions in order to activate the specific control lines such that different operations in the ALU of the CPU are carried out.

MULTIPLEXER(MUX)

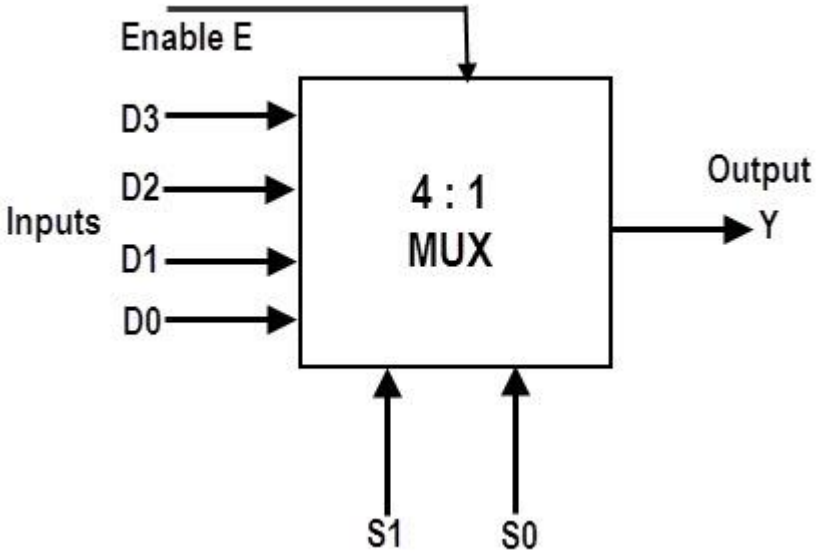
- Multiplexing is the generic term used to describe the operation of **sending one or more analogue or digital signals** over a common transmission line at different times or speeds and as such, the device we use to do just that is called a **Multiplexer**.
- The *multiplexer*, shortened to “MUX” or “MPX”, is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal.



4 to 1 channel Mux

- In this example at any one instant in time only ONE of the four analogue switches is closed, connecting only one of the input lines A to D to the single output at Q. As to which switch is closed depends upon the addressing input code on lines “S0” and “S1“, so for this example to select input D2 to the output at Q, the binary input address would need to be “S1” = logic “1” and “S0” = logic “0”.

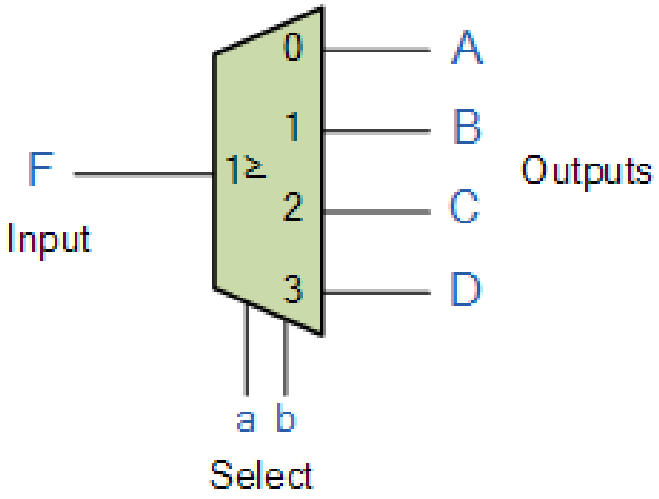
Block diagram & truth table



Select Data Inputs		Output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

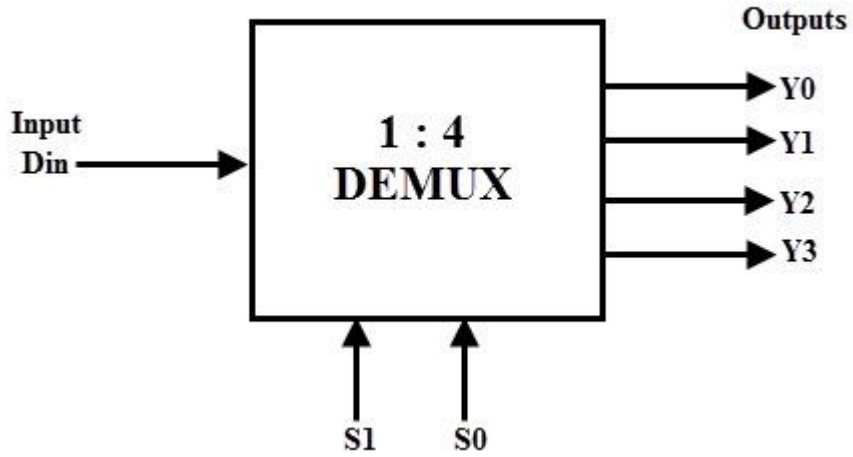
Demultiplexer

- The demultiplexer takes one single input data line and then switches it to any one of a number of individual output lines one at a time.
- demux is a one-to-many circuit. With the use of a demultiplexer , the binary data can be bypassed to one of its many output data lines.



1 to 4 Channel DeMUX

- **Block diagram & truth 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