Module -2 **Hr:10**

Analysis and design of combinational logic - I:

General approach, Decoders-BCD decoders, Encoders

Recommended readings:

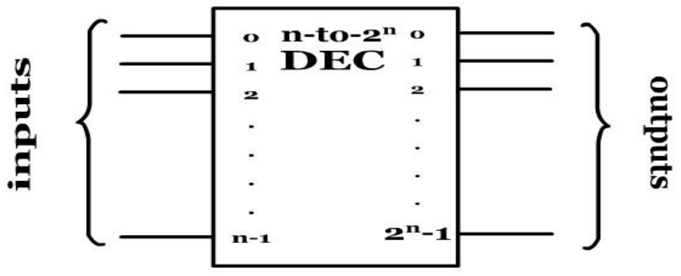
1. John M Yarbrough, “Digital Logic Applications and Design”,

Thomson Learning, 2001.

Unit- 4.1, 4.3, 4.4

Decoder

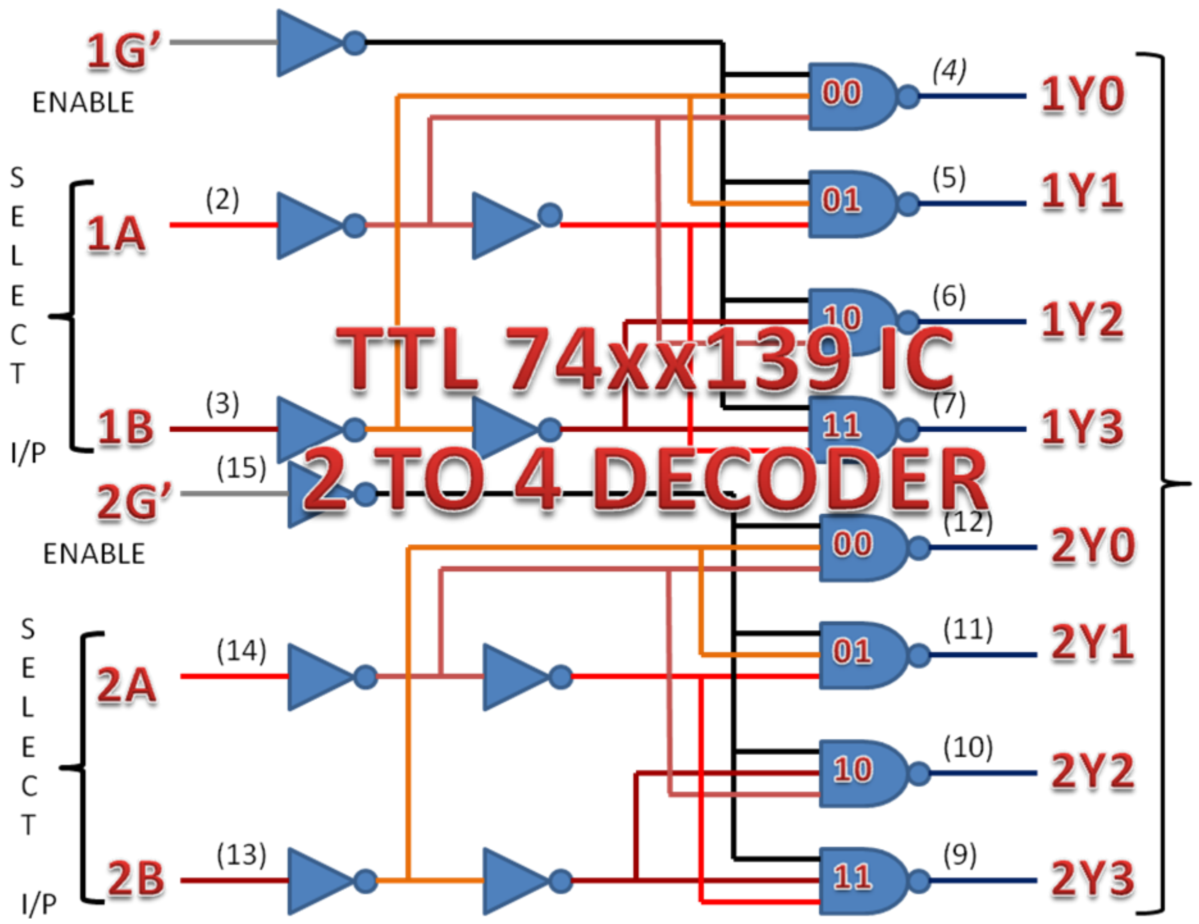
A Decoder is a multiple input ,multiple output logic circuit.The block diagram of a decoder is as shown below.



An n-to-2ⁿline decoder symbol.

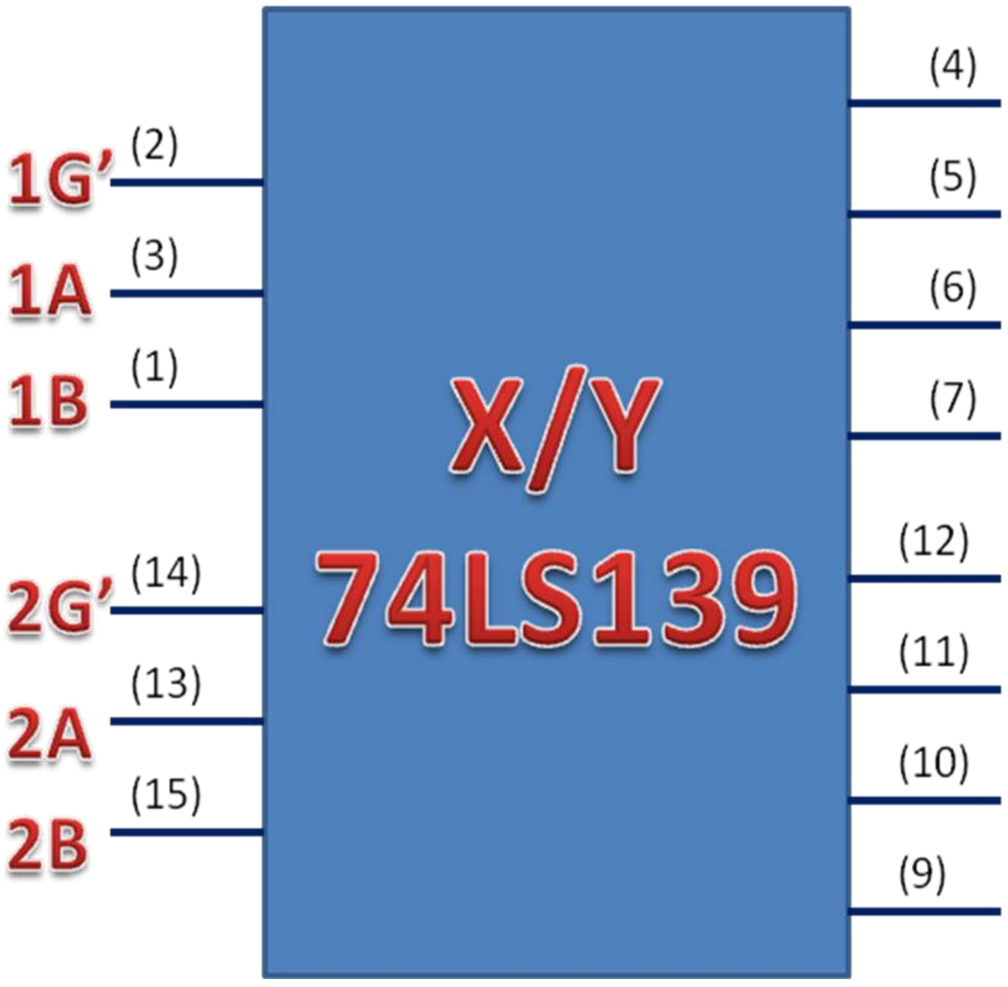
The most commonly used decoder is a n –to 2^n decoder which ha n inputs and 2^n

Output lines .

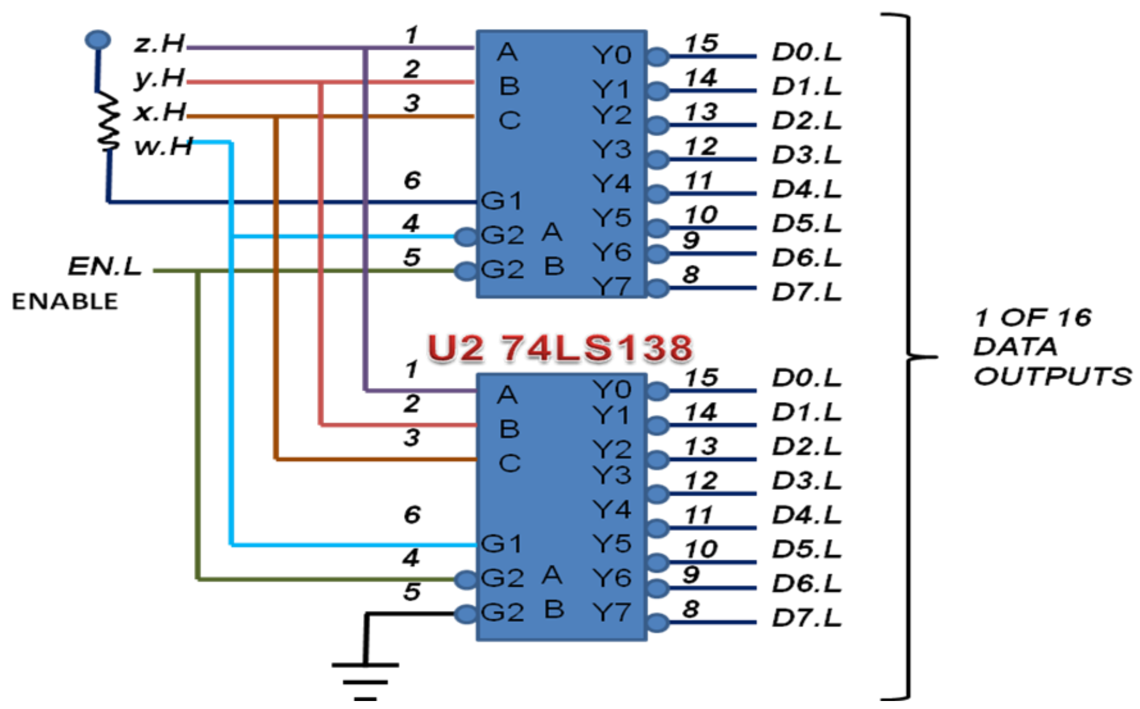


INPUTS			OUTPUTS			
ENABLE	SELECT					
G'	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H

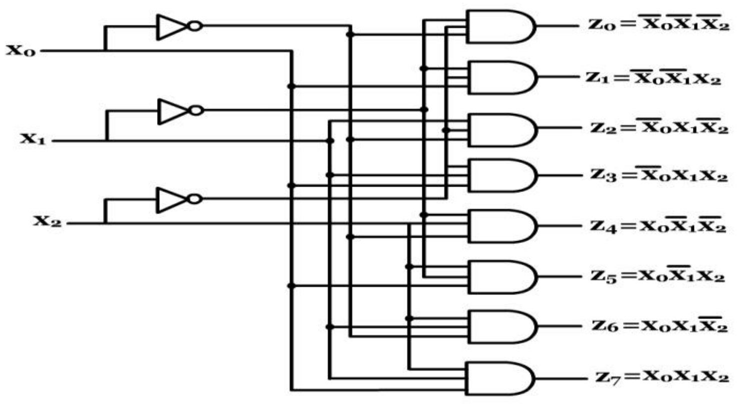
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L



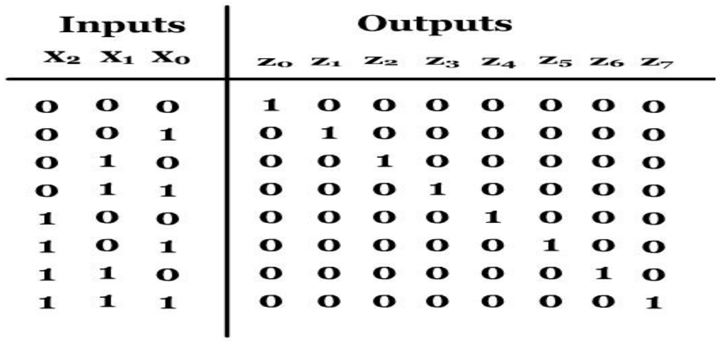
U1 74LS138



3-to-8 decoder logic diagram


X_0
X_1
X_2
$Z_0=\overline{X}_0\overline{X}_1\overline{X}_2$
$Z_1=\overline{X}_0\overline{X}_1X_2$
$Z_2=\overline{X}_0X_1\overline{X}_2$
$Z_3=\overline{X}_0X_1X_2$
$Z_4=X_0\overline{X}_1\overline{X}_2$
$Z_5=X_0\overline{X}_1X_2$
$Z_6=X_0X_1\overline{X}_2$
$Z_7=X_0X_1X_2$

A 3-to-8 decoder
Logic diagram



Inputs			Outputs							
X_2	X_1	X_0	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_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

Truth table.

In this realization shown above the three inputs are assigned x_0,x_1,and $x_{2,}$ and the eight outputs are Z_0 to $Z_{7.}$

Function specifc decoders also exist which have less than 2^n outputs . examples are 8421 code decoder also called BCD to decimal decoder. Decoders that drive seven segment displays also exist.

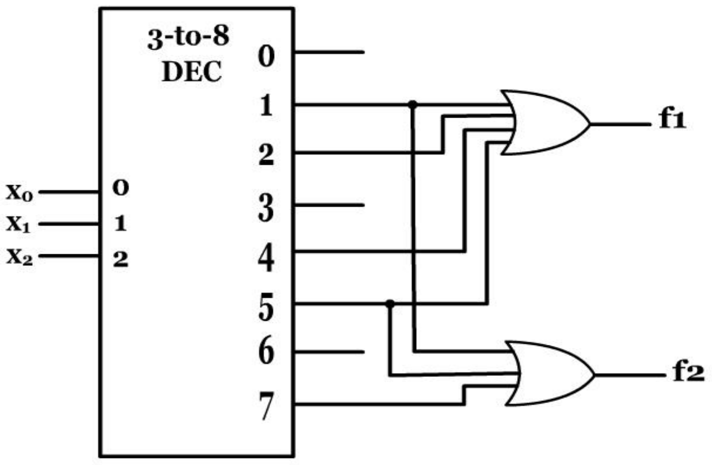
Realization of boolean expression using Decoder and OR gate

We see from the above truth table that the output expressions corrwespond to a single minterm. Hence a n –to 2^n decoder is a minterm generator. Thus by using OR gates in conjunction with a a n –to 2^n decoder boolean function realization is possible.

Ex: to realize the Boolean functions given below using decoders…

•F1=Σm(1,2,4,5)

•F2=Σm(1,5,7)

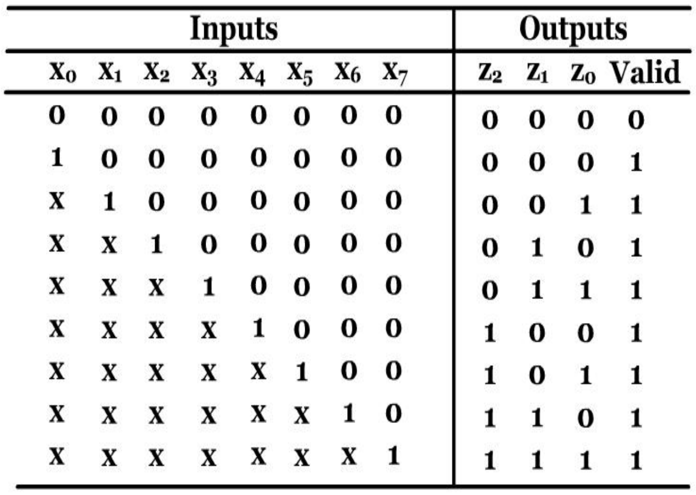


Realisation of boolean expressions

Priority encoder
8-3 line priority encoder

In priority encoder a priority scheme is assigned to the input lines so that whenever more than one input line is asserted at any time, the output is determined by the input line having the highest priority.

The Valid bit is used to indicate that atleast one inut line is asserted. This is done to distinguish the situation that no input line is asserted from when the X0 input line is asserted , since in both cases Z1Z2Z3 =000.



Inputs								Outputs			
X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	Z_2	Z_1	Z_0	Valid
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
X	1	0	0	0	0	0	0	0	0	1	1
X	X	1	0	0	0	0	0	0	1	0	1
X	X	X	1	0	0	0	0	0	1	1	1
X	X	X	X	1	0	0	0	1	0	0	1
X	X	X	X	X	1	0	0	1	0	1	1
X	X	X	X	X	X	1	0	1	1	0	1
X	X	X	X	X	X	X	1	1	1	1	1

Condensed truth table for an 8-to-3 line priority encoder

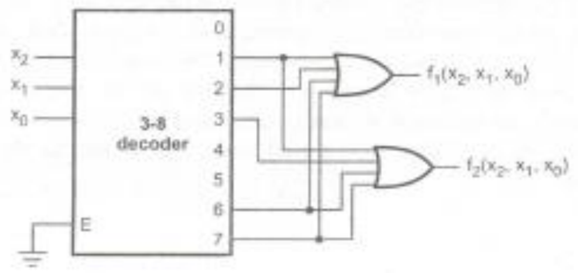
Recommended question and answer –unit-3

Jan 2009

Q.3 a) *Realize the following functions expressed in maxterm canonical form in two possible ways using* 3-8 *line and decoder :*

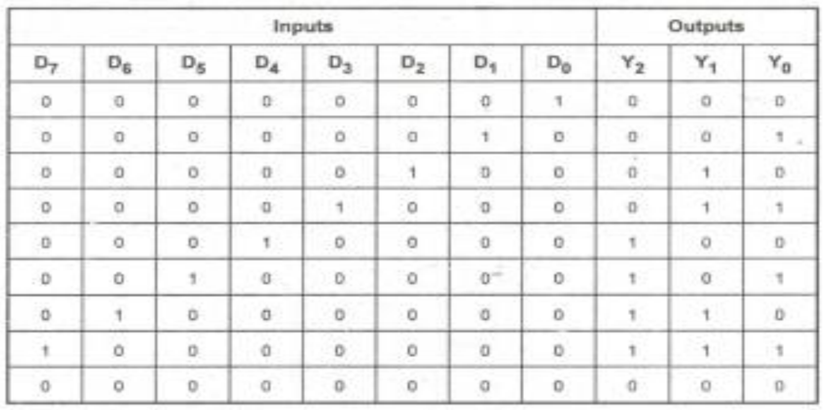
$f_1(x_2, x_1, x_0) = \pi M\,(1, 2, 6, 7),\ f_2(x_2, x_1, x_0) = \pi M\,(1, 3, 6, 7)$ (10)

Ans. :



b) *What are the problems associated with the basic encoder? Explain, how can these problems be overcome by priority encoder, considering* 8 *input lines.* (10)

Ans. : The basic encoder has ambiguity that when all inputs are Osth~ outputs are Os. The zero output can also be generated when Do = 1. This amb:guity can be resolved by providing an additional output that specifies the valid condition.



Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Y_2	Y_1	Y_0
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
0	0	0	0	0	0	0	0	0	0	0

A priority encoder is an encoder circuit that indicates the priority function. In priority encoder, if two or more inputs are equal to 1 at the same time, the inputs having the highest priority will take precedence. Also, the output V (valid output indicator) indicates one or more of the inputs are equal to 1. If all inputs are '0', V is equal to b and other two outputs of the circuit are not used.

Jan 2008

Final expression

$$Y = \overline{A}\,B\,\overline{C} + B\,\overline{C}\,D + \overline{B}\,C + A\,\overline{B} + A\,\overline{D}$$

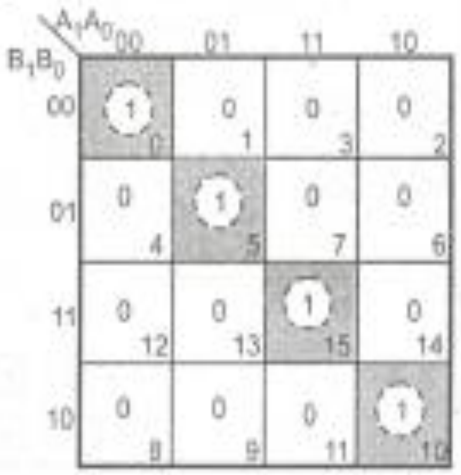
Q.3 a) *Design a combinational circuit that will multiply two two-bit binary values.*

Ans. :

A		B		Output		
A_1	A_0	B_1	B_0	A = B	A > B	A < B
0	0	0	0	1	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	1	0
1	1	0	1	0	1	0
1	1	1	0	0	1	0
1	1	1	1	1	0	0

$$y = \overline{A}_0\overline{A}_1\overline{B}_0\overline{B}_1 + B_0B_1\overline{A}_1A_0 + B_0B_1A_1A_0 + B_1\overline{B}_0A_1\overline{A}_0$$

A = B



$y_{A>B} = B_1\overline{A}_1 + B_1\overline{B}_0A_0 + B_1B_0\overline{A}_0$

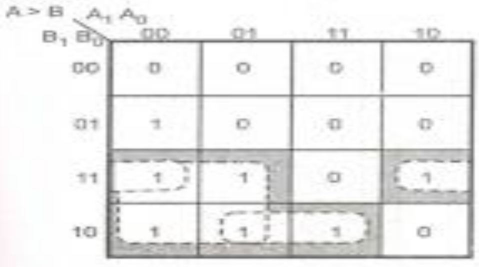


Fig. 7

$y_{A<B} = A_0\overline{B}_1\overline{B}_0 + A_1\overline{B}_0\overline{B}_1 + A_1B_0B_1$

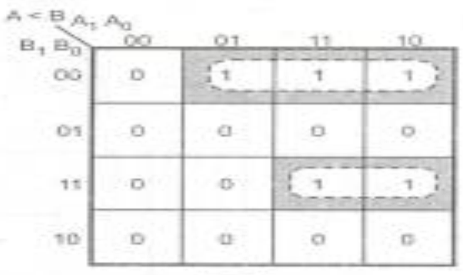
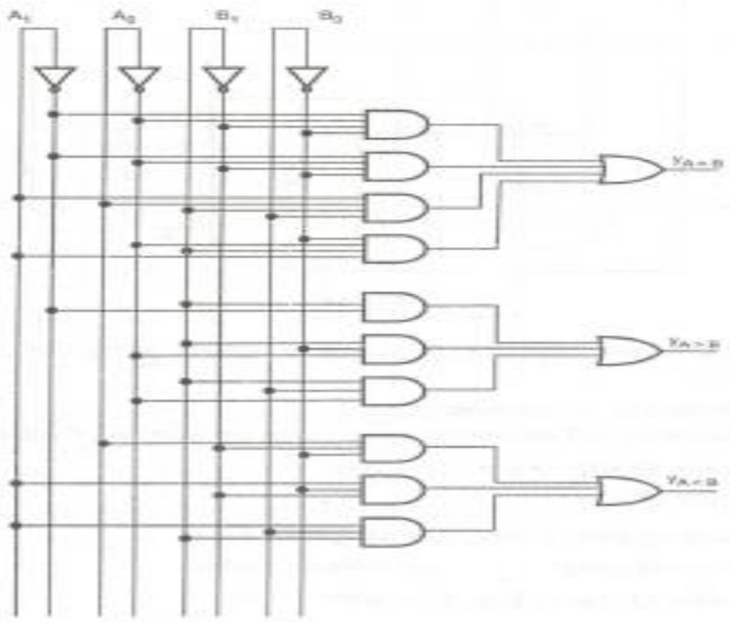


Fig. 8

Design



b) *Design a 4 to 16 decoder using two 3 to 8 decoder (74LS138).*

Ans. :

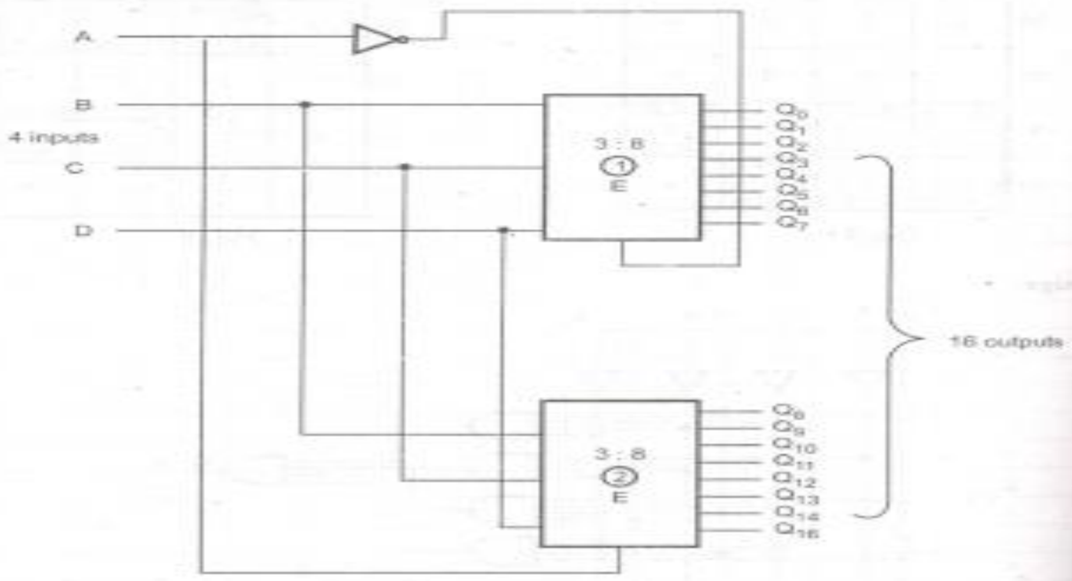


Fig. 10

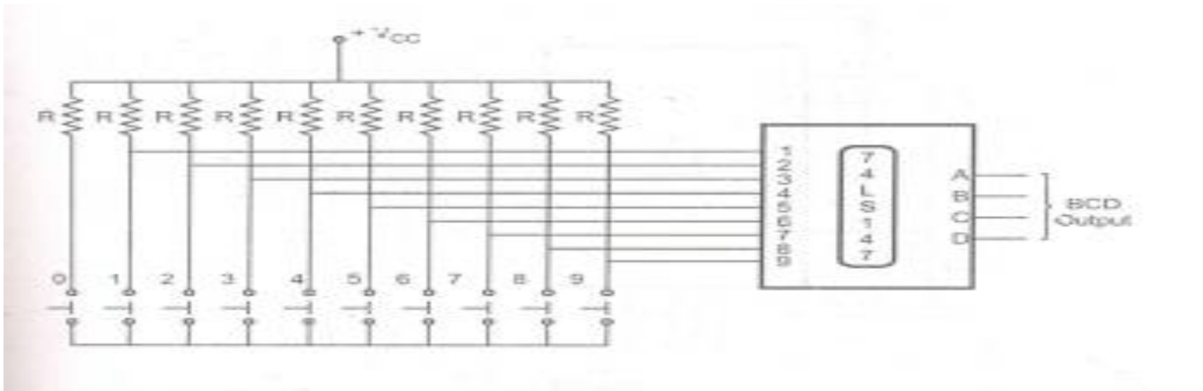


Fig. 11 Design of keypad interface to a digital system using BCD encoder IC 74147

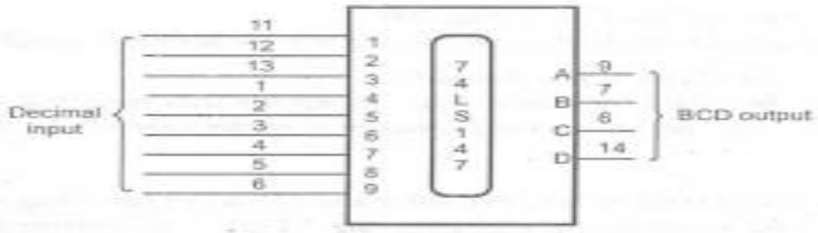


Fig. 12 Pin configuration of 74LS147

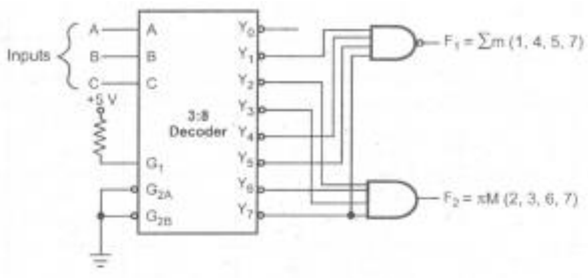
<u>Aug 2009</u>

Q.3 a) *Implement following multiple output function using 74LS138 and extend gates.*

fl (A, B, C) = *I,m* (1,4,5, 7)

f2 (A, B, C) = 1tM (2, 3, 6, 7)

Ans. : In this example, we use Ie 74LS138, 3 8 decoder to implement multiple output function. The outputs of 74LS138 are active low, therefore, SOP function (function F1) can be implemented using NAr-..-rn gate and POS function (function F2) can be implemented using AND gate, as shown in Fig. 6.

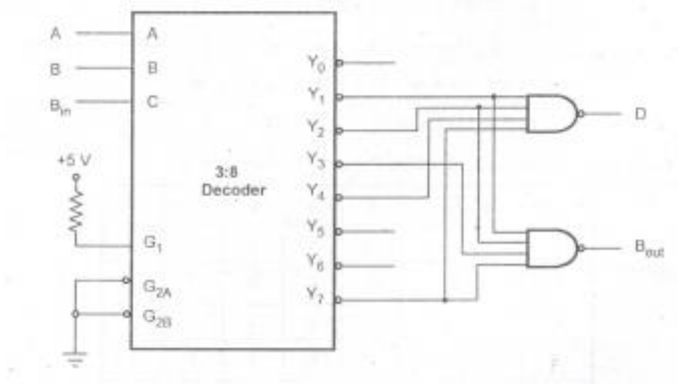


b) *Implement full subtractor using decoder and write a truth table.*

Ans. : The truth table for full subtractor is as shown in Table 1.

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Table 1 Truth table for full subtractor



c) *Write a note on encoders.* (6)

Ans. : An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2n (or fewer) input lines and n output lines: In encoder the output lines generate the binary code corresponding to the input value. The Fig. 8 shows the general structure of the encoder circuit. As shown in the Fig. 8, the decoded information is presented as 2n inputs producing n possible outputs.

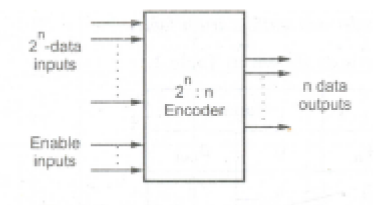


Fig. 8 General structure of encoder

Aug-2007

Q.5 a) *What is a magnitude comparator? Design a 2 bit digital comparator by writing truth table, the relevant expression, and logic diagram.*

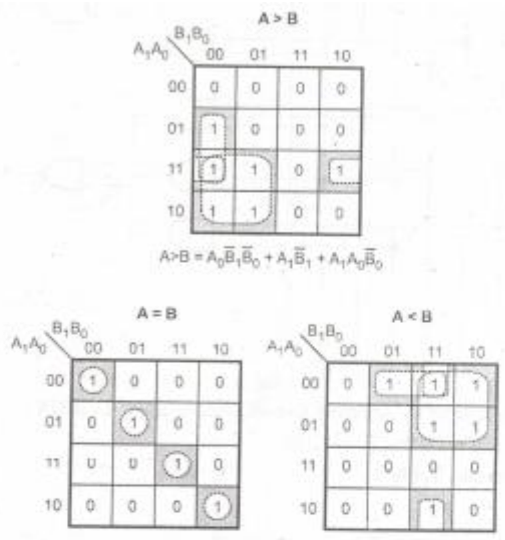
Sol. : The truth table for 2-bit comparator is given in Table 1.

Inputs				Outputs		
A_1	A_0	B_1	B_0	A > B	A = B	A < B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
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	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Table 1

K-map simplification

$$\begin{aligned}(A = B) &= \bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0 + \bar{A}_1A_0\bar{B}_1B_0 \\ &\quad + A_1A_0B_1B_0 + A_1\bar{A}_0B_1\bar{B}_0 \\ &= \bar{A}_1\bar{B}_1(\bar{A}_0\bar{B}_0 + A_0B_0) \\ &\quad + A_1B_1(A_0B_0 + \bar{A}_0\bar{B}_0) \\ &= (A_0 \odot B_0)(A_1 \odot B_1) \\ (A < B) &= \bar{A}_1\bar{A}_0B_0 + \bar{A}_0B_1B_0 + \bar{A}_1B_1\end{aligned}$$



$A>B = A_0\bar{B}_1\bar{B}_0 + A_1\bar{B}_1 + A_1A_0\bar{B}_0$

Fig. 8

Logic Diagram

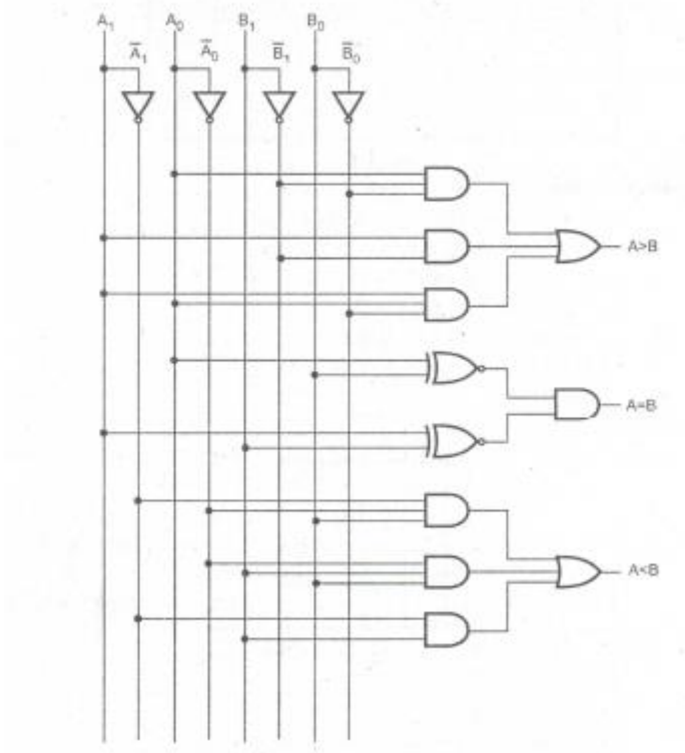


Fig. 9

Analysis and design of combinational logic - II:

Digital multiplexers-Using multiplexers as Boolean function generators. Adders and subtractors - Cascading full adders, Look ahead carry, Binary comparators

Recommended readings:

1. John M Yarbrough, "Digital Logic Applications and Design",

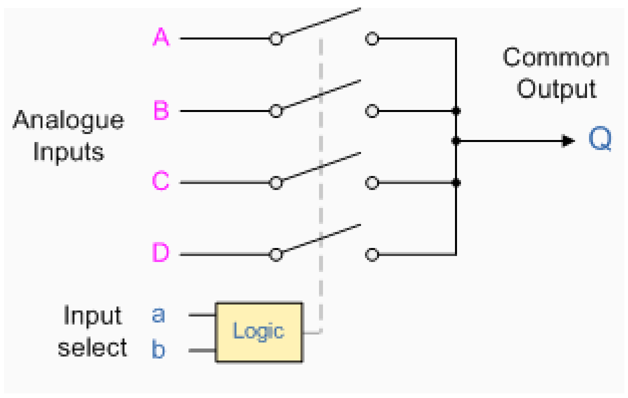
Thomson Learning, 2001.

Units- 4.5, 4.6 - 4.6.1, 4.6.2, 4.7

The Multiplexer

The Multiplexer which sometimes are simply called "Mux" or "Muxes", are devices that act like avery fast acting rotary switch. They connect multiple input lines 2, 4, 8, 16 etc one at a time to a common output line and are used as one method of reducing the number of logic gates required in a circuit. Multiplexers are individual Analogue Switches as opposed to the "mechanical" types such as normal conventional switches and relays. They are usually made from MOSFETs devices encased in a single package and are controlled using standard logic gates. An example of a Multiplexer is shown below.

4-to-1 Channel Multiplexer



Addressing		Input Selected
b	a	
0	0	A
0	1	B
1	0	C
1	1	D

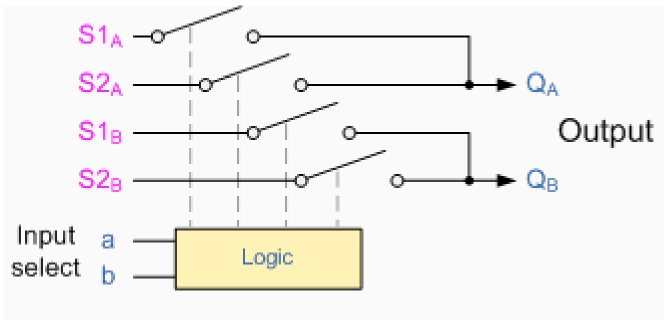
The Boolean expression for this 4 to 1 **Multiplexer** is given as:

Q = abA + abB + abC + abD

In this example at any 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 "a" and "b", so for this example to select input B to the output at Q, the binary input address would need to be "a" = logic "1" and "b" = logic "0". Adding more control address lines will allow the multiplexer to control more inputs. Multiplexers can also be used to switch either analogue, digital or video signals, with the switching current in analogue circuits limited to below 10mA to 20mA per channel in order to reduce heat dissipation.

Multiplexers are not limited to just switching a number of different input lines or channels to one common single output. There are also types that can switch their inputs to multiple outputs and have arrangements or 4 to 2, 8 to 3 or even 16 to 4 etc configurations and an example of a simple Dual channel 4 input multiplexer (4 to 2) is given below:

4-to-2 Channel Multiplexer

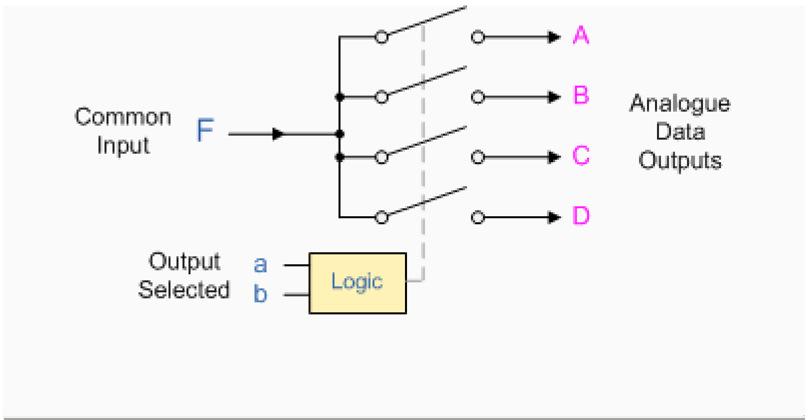


Here in this example the 4 input channels are switched to 2 individual output lines but larger arrangements are also possible. This simple 4 to 2 configuration could be used for example, to switch audio signals for stereo pre-amplifiers or mixers.

The De-multiplexer

De-multiplexers or "De-muxes", are the exact opposite of the **Multiplexers** we saw in the previous tutorial in that they have one single input data line and then switch it to any one of their individual multiple output lines one at a time. **The De-multiplexer** converts the serial data signal at the input to a parallel data at its output lines as shown below.

1-to-4 Channel De-multiplexer



Addressing	Output

b	a	Selected
0	0	A
0	1	B
1	0	C
1	1	D

The Boolean expression for this De-multiplexer is given as:

$$F = ab A + abB + abC + abD$$

The function of the De-multiplexer is to switch one common data input line to any one of the 4 output data lines A to D in our example above. As with the multiplexer the individual solid state switches are selected by the binary input address code on the output select pins "a" and "b" and by adding more address line inputs it is possible to switch more outputs giving a 1-to-2n data lines output. Some standard De-multiplexer IC´s also have an "enable output" input pin which disables or prevents the input from being passed to the selected output. Also some have latches built into their outputs to maintain the output logic level after the address inputs have been changed. However, in standard decoder type circuits the address input will determine which single data output will have the same value as the data input with all other data outputs having the value of logic "0".

Standard De-multiplexer IC packages available are the TTL 74LS138 1 to 8-output De-multiplexer, theTTL 74LS139 Dual 1 to 4-output De-multiplexer or the CMOS CD4514 1 to 16-output De-multiplexer. Another type of De-multiplexer is the 24-pin, 74LS154 which is a 4-bit to 16-line De-multiplexer/decoder. Here the output positions are selected using the 4-bit binary coded input.

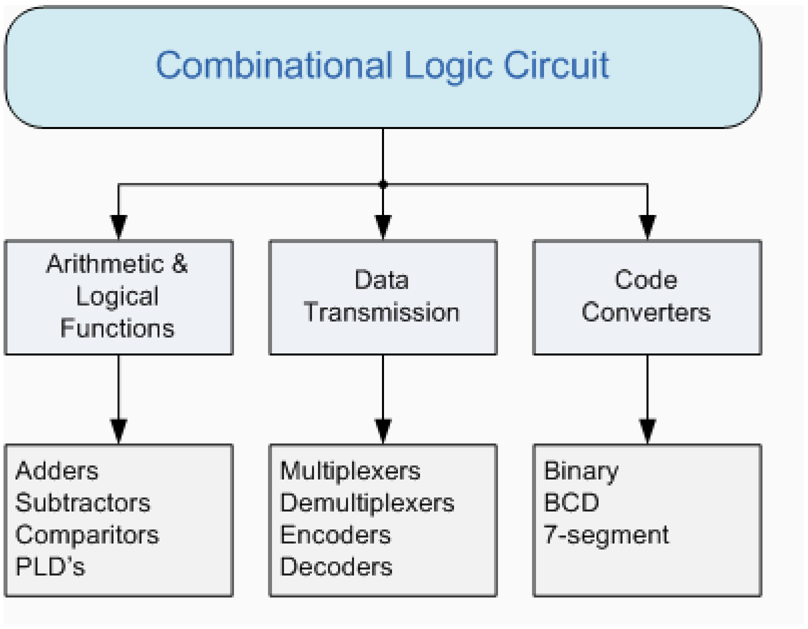
Combination Logic

Unlike Sequential Logic circuits whose outputs are dependant on both the present input and their previous output state giving them some form of Memory, the outputs of Combinational Logic circuits are only determined by their current input state as they have no feedback, and any changes to the signals being applied to their inputs will immediately have an effect at the output. In other words, in a Combination Logic circuit, if the input condition changes state so too does the output as combinational circuits have No Memory.

Combination Logic circuits are made up from basic logic AND, OR or NOT gates that are "combined" or connected together to produce more complicated switching circuits.

As combination logic circuits are made up from individual logic gates they can also be considered as "decision making circuits" and combinational logic is about combining logic gates together to process two or more signals in order to produce at least one output signal according to the logical function of each logic gate. Common combinational circuits made up from individual logic gates include Multiplexers, Decoders and De-multiplexers, Full and Half Adders etc.

Classification of Combinational Logic

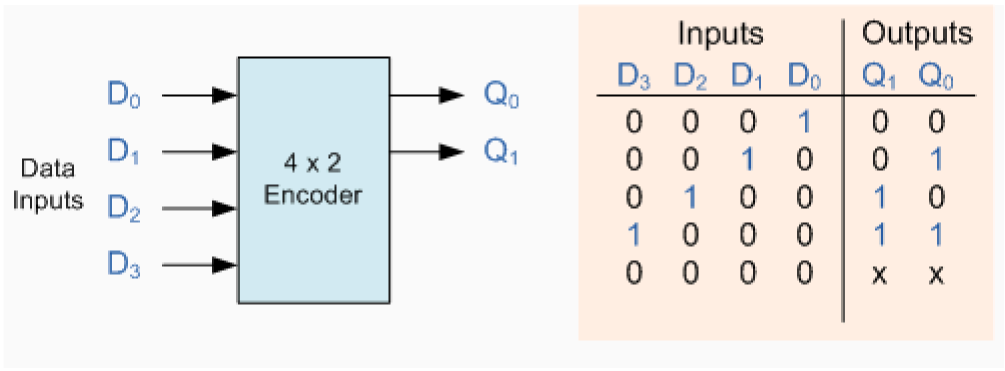


One of the most common uses of combination logic is in Multiplexer and De-multiplexer type circuits. Here, multiple inputs or outputs are connected to a common signal line and logic gates are used to decode an address to select a single data input or output switch. A multiplexer consist of two separate components, a logic decoder and some solid state switches, but before we can discuss multiplexers, decoders and de-multiplexers in more detail we first need to understand how these devices use these "solid state switches" in their design.

The Encoder

Unlike a multiplexer that selects one individual data input line and then sends that data to a single output line or switch, an Encoder takes all the data inputs one at a time and converts them to a single encoded output. Then, it is a multi-input data line, combinational logic circuit that converts the logic level "1" data at its inputs to an equivalent binary code at its output. Generally encoders produce outputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines and a "n-bit" encoder has 2n input lines with common types that include 4-to-2, 8-to-3 and 16-to-4 line configurations. Encoders are available to encode either a decimal or hexadecimal input pattern to typically a binary or B.C.D. output code.

4-to-2 Bit Encoder



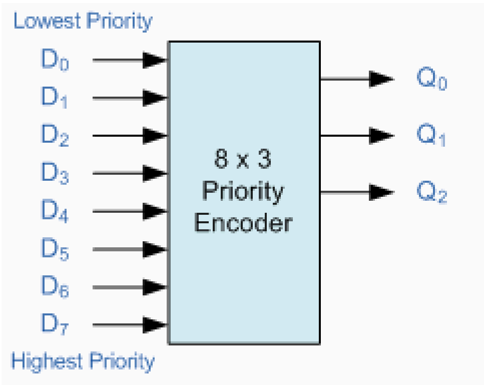
Inputs				Outputs	
D_3	D_2	D_1	D_0	Q_1	Q_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
0	0	0	0	x	x

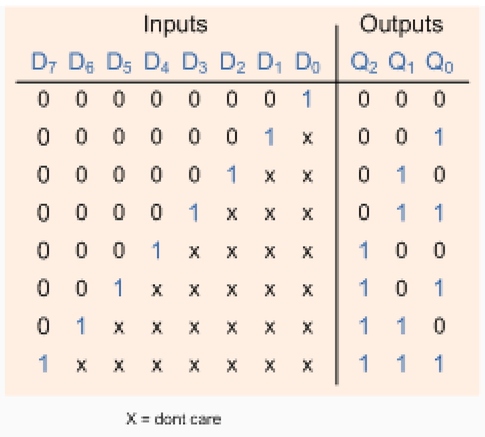
One of the main disadvantages of standard encoders is that they can generate the wrong output code when there is more than one input present at logic level "1". For example, if we make inputs D1 and D2 HIGH at logic "1" at the same time, the resulting output is neither at "01" or at "10" but will be at "11" which is an output code that is different to the actual input present. One simple way to overcome this problem is to "Prioritize" the level of each input pin and if there was more than one input at logic level "1" the actual output code would only correspond to the input with the highest designated priority. Then this type of encoder are known as Priority Encoders or P-encoder.

Priority Encoders

Priority Encoders come in many forma and an example of an 8-input Priority Encoder along with its truth table is as shown below.

8-to-3 Bit Priority Encoder





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 are available in standard IC form and the TTL 74LS148 is an 8 to 3 bit priority encoder which has eight active LOW (logic "0") inputs and provides a 3-bit code of the highest ranked input at its output. 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.

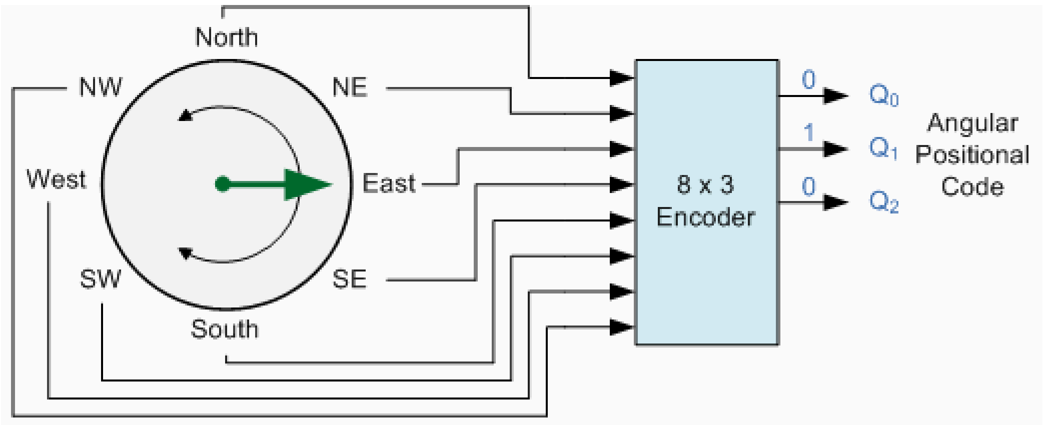
Encoder Applications

Keyboard Encoders

Priority encoders can be used to reduce the number of wires needed in circuits or applications that has 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 or HIGH 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 an 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 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.

Positional Encoders

Another more common application is in magnetic positional control as used on ships or robots etc. Here the angular or rotary position of a compass is converted into a digital code by an encoder and inputted 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.



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

Interrupt Requests

Other applications for Priority Encoders may include detecting interrupts in microprocessor applications. Here the microprocessor uses interrupts to allow peripheral devices such as the disk drive, scanner, mouse, or printer etc, to communicate with it, but the microprocessor can only "talk" to one peripheral device at a time. The processor uses "Interrupt Requests" or "IRQ" signals to assign priority to the devices to ensure that the most important peripheral device is serviced first. The order of importance of the devices will depend upon their connection to the priority encoder.

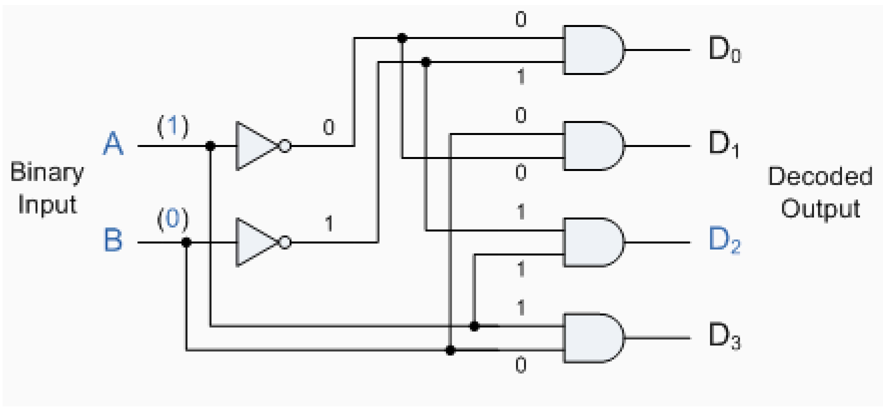
Because implementing such a system using priority encoders such as the standard 74LS148 priority encoder IC involves additional logic circuits, purpose built integrated circuits such as the 8259 Programmable Priority Interrupt Controller is available.

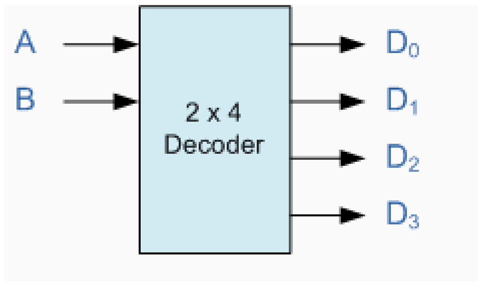
IRQ Number	Typical Use	Description
IRQ 0	System timer	Internal System Timer.
IRQ 1	Keyboard	Keyboard Controller.
IRQ 3	COM2 & COM4	Second and Fourth Serial Port.
IRQ 4	COM1 & COM3	First and Third Serial Port.
IRQ 5	Sound	Sound Card.
IRQ 6	Floppy disk	Floppy Disk Controller.
IRQ 7	Parallel port	Parallel Printer.
IRQ 12	Mouse	PS/2 Mouse.
IRQ 14	Primary IDE	Primary Hard Disk Controller.
IRQ 15	Secondary IDE	Secondary Hard Disk Controller.

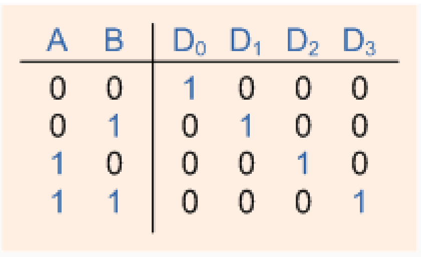
Binary Decoders

A Decoder is the exact opposite to that of an "Encoder" we looked at in the last tutorial. It is basically, a combinational type logic circuit that converts the binary code data at its input into one of a number of different output lines, one at a time producing an equivalent decimal code at its output. Binary Decoders have inputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines, and a "n-bit" decoder has 2n output lines. Typical combinations of decoders include, 2-to-4, 3-to-8 and 4-to-16 line configurations. Binary Decoders are available to "decode" either a Binary or BCD input pattern to typically a Decimal output code.

A 2-to-4 Binary Decoders.







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

In this simple example of a 2-to-4 line binary decoder, the binary inputs A and B determine which output line from D0 to D3 is "HIGH" at logic level "1" while the remaining outputs are held "LOW" at logic "0". Therefore, whichever output line is "HIGH" identifies the binary code present at the input, in other words it "de-codes" the binary input and these types of binary decoders are commonly used as Address Decoders in microprocessor memory applications.

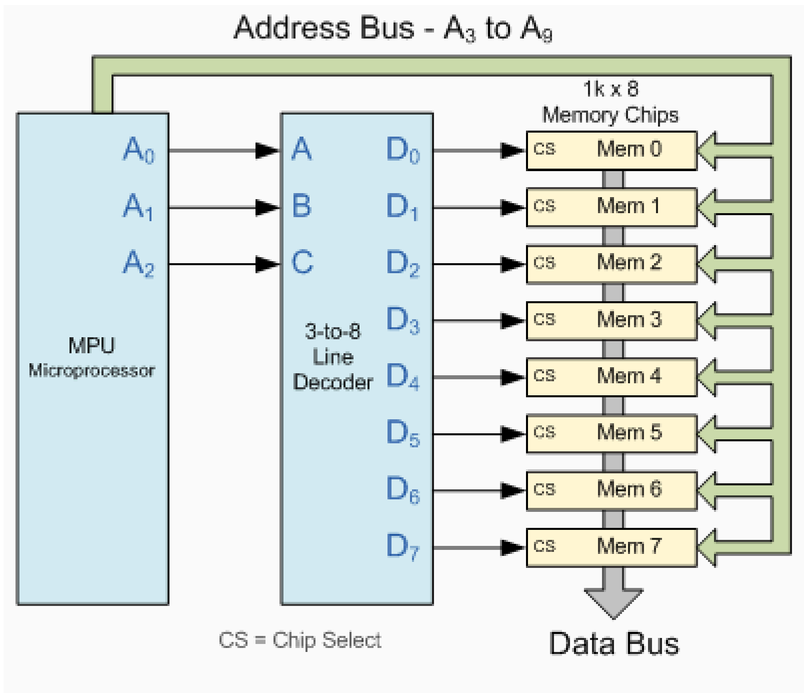
Memory Address Decoder.

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.

Each memory chip has an input called Chip Select or CS which is used by the MPU to select the appropriate memory chip and a logic "1" on this input selects the device and a logic "0" on the input de-selects it. By selecting or de-selecting each chip, allows us to select the correct memory device for a particular address and when we specify a particular memory address, the corresponding memory location exists ONLY in one of the chips.

For example, Lets assume we have a very simple microprocessor system with only 1Kb of RAM memory and 10 address lines. The memory consists of 128x8-bit (128x8 = 1024 bytes) devices and for 1Kb we will need 8 individual memory devices but in order to select the correct memory chip we will also require a 3-to-8 line binary decoder as shown below.

Memory Address Decoding.



The binary decoder requires 3 address lines, (A0 to A2) to select each one of the 8 chips (the lower part of the address), while the remaining 7 address lines (A3 to A9) select the correct memory location on that chip (the upper part of the address). Having selected a memory location using the address bus, the information at the particular internal memory location is sent to the "Data Bus" for use by the microprocessor. This is of course a simple example but the principals remain the same for all types of memory chips or modules.

Binary Decoders are very useful devices for converting one digital format to another, such as binary or BCD type data into decimal or octal etc and commonly available decoder IC's are the TTL 74LS138 3-to-8 line binary decoder or the 74ALS154 4-to-16 line decoder. They are also very useful for interfacing to 7-segment displays such as the TTL 74LS47 which we will look at in the next tutorial.

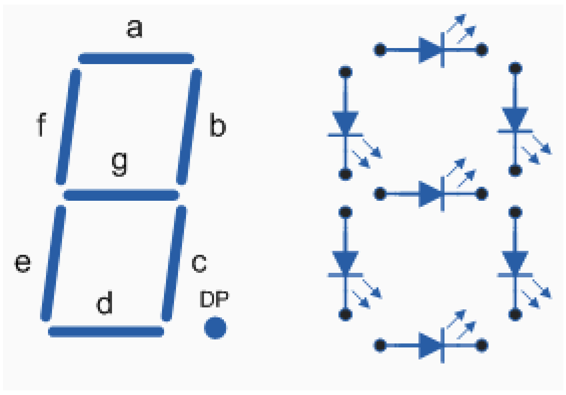
Display Decoders

A Decoder IC, is a device which converts one digital format into another and the most commonly used device for doing this is the BCD (Binary Coded Decimal) 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 LEDs (the segments), within one single display package. In order to produce the required numbers or characters from 0 to 9 and A to F respectively, on the display the correct combination of LED segments need to be illuminated and Display Decoders do just that. A standard 7-segment LED or LCD display generally has 8 input connections, one for each LED segment and one that acts as a common terminal or connection for all the internal segments. Some single displays have an additional input pin for the decimal point in their lower right or left hand corner.

There are two important types of 7-segment LED digital display.

- The Common Cathode Display (CCD) - In the common cathode display, all the cathode connections of the LEDs are joined together to logic "0" and the individual segments are illuminated by application of a "HIGH", logic "1" signal to the individual Anode terminals.
- The Common Anode Display (CAD) - In the common anode display, all the anode connections of the LEDs are joined together to logic "1" and the individual segments are illuminated by connecting the individual Cathode terminals to a "LOW", logic "0" signal.

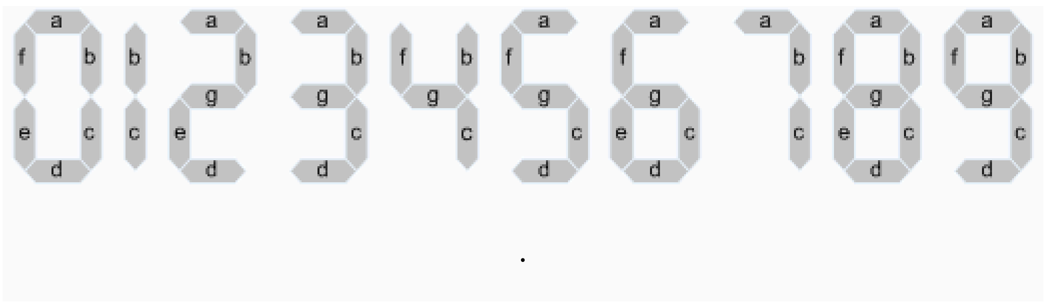
7-Segment Display Format



Truth Table for a 7-segment display

Individual Segments							Display
a	b	c	d	e	f	g	
×	×	×	×	×	×		0
	×	×					1
×	×		×	×		×	2
×	×	×	×			×	3
	×	×			×	×	4
×		×	×		×	×	5
×		×	×	×	×	×	6
×	×	×					7

Individual Segments							Display
a	b	c	d	e	f	g	
×	×	×	×	×	×	×	8
×	×	×			×	×	9
×	×	×		×	×	×	A
		×	×	×	×	×	b
×			×	×	×		C
	×	×	×	×		×	d
×			×	×	×	×	E
×				×	×	×	F



It can be seen that to display any single digit number from 0 to 9 or letter from A to F, we would need 7 separate segment connections plus one additional connection for the LED's "common" connection. Also as the segments are basically a standard light emitting diode, the driving circuit would need to produce up to 20mA of current to illuminate each individual segment and to display the number 8, all 7 segments would need to be lit resulting a total current of nearly 140mA, (8 x 20mA). Obviously, the use of so many connections and power consumption is impractical for some electronic or microprocessor based circuits and so in order to reduce the number of signal lines required to drive just one single display, display decoders such as the BCD to 7-Segment Display Decoder and Driver IC's are used instead.

Binary Coded Decimal

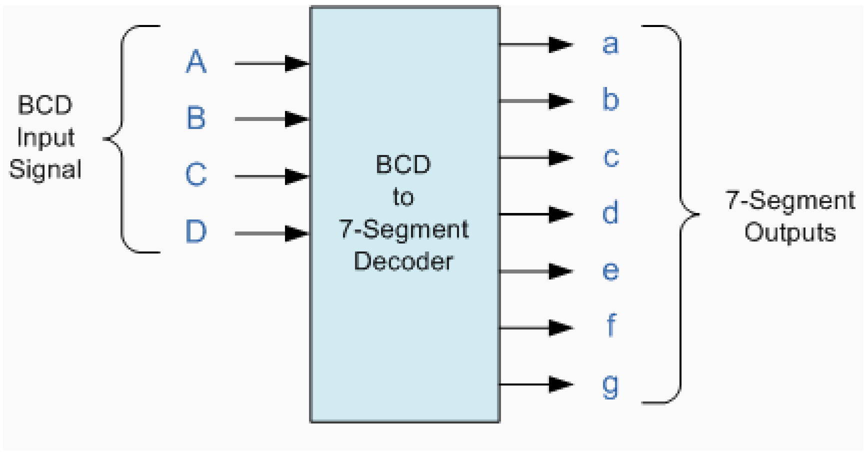
Binary Coded Decimal (BCD or "8421" BCD) numbers are made up using just 4 data bits (a nibble or half a byte) similar to the Hexadecimal numbers we saw in the binary tutorial, but unlike hexadecimal numbers that range in full from 0 through to F, BCD numbers only range from 0 to 9, with the binary number patterns of 1010 through to 1111 (A to F) being invalid inputs for this type of display and so are not used as shown below.

Decimal	Binary Pattern				BCD	Decimal	Binary Pattern				BCD
	8	4	2	1			8	4	2	1	
0	0	0	0	0	0	8	1	0	0	0	8
1	0	0	0	1	1	9	1	0	0	1	9
2	0	0	1	0	2	10	1	0	1	0	Invalid
3	0	0	1	1	3	11	1	0	1	1	Invalid
4	0	1	0	0	4	12	1	1	0	0	Invalid
5	0	1	0	1	5	13	1	1	0	1	Invalid
6	0	1	1	0	6	14	1	1	1	0	Invalid
7	0	1	1	1	7	15	1	1	1	1	Invalid

BCD to 7-Segment Display Decoders

A binary coded decimal (BCD) to 7-segment display decoder such as the TTL 74LS47 or 74LS48, have 4 BCD inputs and 7 output lines, one for each LED segment. This allows a smaller 4-bit binary number (half a byte) to be used to display all the denary numbers from 0 to 9 and by adding two displays together, a full range of numbers from 00 to 99 can be displayed with just a single byte of 8 data bits.

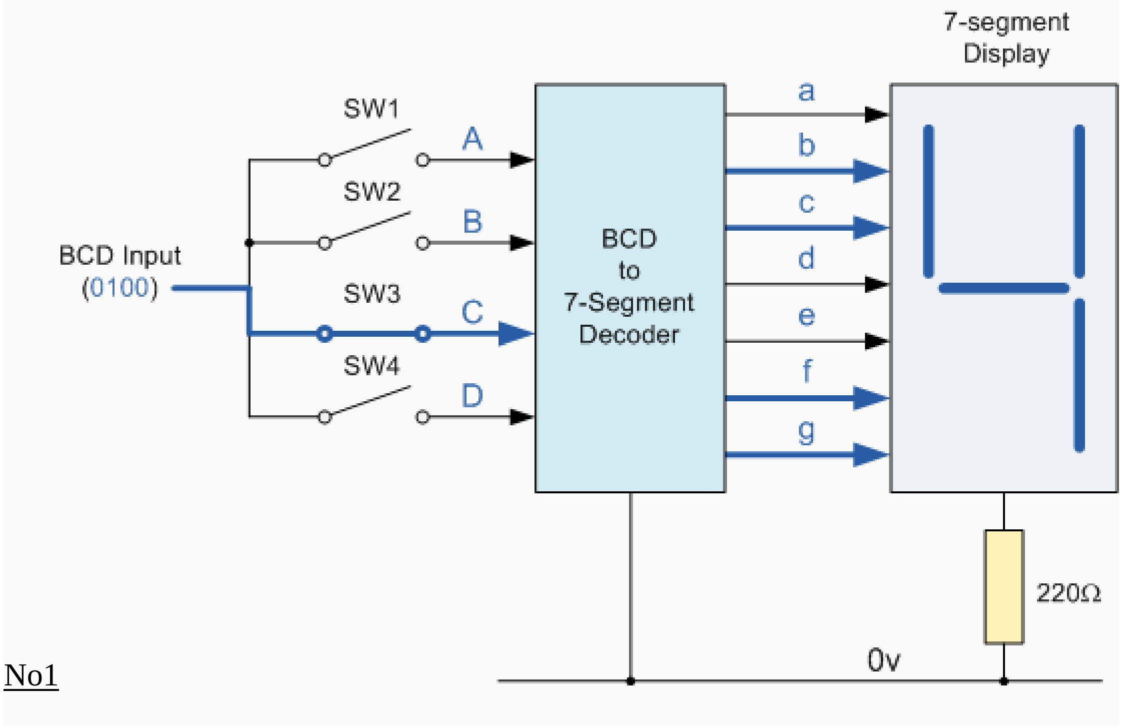
BCD to 7-Segment Decoder



The use of packed BCD allows two BCD digits to be stored within a single byte (8-bits) of data, allowing a single data byte to hold a BCD number in the range of 00 to 99.

An example of the 4-bit BCD input (0100) representing the number 4 is given below.

Example



No1

In practice current limiting resistors of about 150Ω to 220Ω would be connected in series between the decoder/driver chip and each LED display segment to limit the maximum current flow. Different display decoders or drivers are available for the different types of display available, e.g. 74LS48 for common-cathode LED types, 74LS47 for common-anode LED types, or the CMOS CD4543 for liquid crystal display (LCD) types.

Liquid crystal displays (LCD´s) have one major advantage over similar LED types in that they consume much less power and nowadays, both LCD and LED displays are combined together to form larger Dot-Matrix Alphanumeric type displays which can show letters and characters as well as numbers in standard Red or Tri-colour outputs.

The Binary Adder

Another common and very useful combinational logic circuit is that of the Binary Adder circuit. The Binary Adder is made up from standard AND and Ex-OR gates and allow us to "add" single bits of data together to produce two outputs, the SUM ("S") of the addition and a CARRY ("C"). One of the main uses for the Binary Adder is in arithmetic and counting circuits.

Consider the addition of two denary (base 10) numbers below.

123	A	(Augend)
+ 789	B	(Addend)
912	SUM	

Each column is added together starting from the right hand side. As each column is added together a carry is generated if the result is greater or equal to ten, 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. Binary addition is based on similar principals but a carry is only generated when the result in any column is greater or equal to "2", the base number of binary.

Binary Addition

Binary Addition follows the same basic rules as for the denary addition above except in binary there are only two digits and the largest digit is "1", so any "SUM" greater than 1 will result in a "CARRY". This carry 1 is passed over to the next column for addition and so on. Consider the single bit addition below.

0	0	1	1
+ 0	+ 1	+ 0	+ 1
0	1	1	10

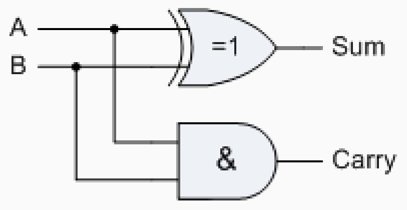
The single bits are added together and "0 + 0", "0 + 1", or "1 + 0" results in a sum of "0" or "1" until you get to "1 + 1" then the sum is equal to "2". For a simple 1-bit addition problem like this, the resulting carry bit could be ignored which would result in an output truth table resembling that of an Ex-OR Gate as seen in the Logic Gates section and whose result is the sum of the two bits but without the carry. An Ex-OR gate only produces an output "1" when either input is at logic "1", but not both. However, all microprocessors and electronic calculators require the carry bit to correctly calculate the equations so we need to rewrite them to include 2 bits of output data as shown below.

00	00	01	01
+ 00	+ 01	+ 00	+ 01
00	01	01	10

From the above equations we know that an Ex-OR gate will only produce an output "1" when "EITHER" input is at logic "1", so we need an additional output to produce a carry output, "1" when "BOTH" inputs "A" and "B" are at logic "1" and a standard AND Gate fits the bill nicely. By combining the Ex-OR gate with the AND gate results in a simple digital binary adder circuit known commonly as the "Half-Adder" circuit.

The Half-Adder Circuit

1-bit Adder with Carry-Out

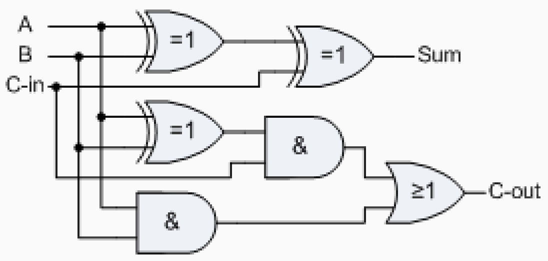
Symbol	Truth Table			
	A	B	SUM	CARRY
	0	0	0	0
	0	1	1	0
	1	0	1	0
	1	1	0	1
Boolean Expression: Sum = $A \oplus B$ Carry = $A . B$				

From the truth table we can see that the SUM (S) output is the result of the Ex-OR gate and the Carry-out (CO) is the result of the AND gate. 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 thebit patterns starting from the least significant bit (LSB). As the Half-Adder has no carry input the resultant added value would be incorrect. One simple way to overcome this problem is to use a "Full-Adder" type binary adder circuit.

The Full-Adder Circuit

The main difference between the "Full-Adder" and the previous seen "Half-Adder" is that a Full-Adder has 3-inputs, the two same data inputs "A" and "B" as before plus an additional "Carry-In" (C-in) input as shown below.

Full-Adder with Carry-In

Symbol	Truth Table				
 A B C-in =1 =1 Sum =1 & & ≥1 C-out	A	B	C-in	Sum	C-out
	0	0	0	0	0
	0	1	0	1	0
	1	0	0	1	0
	1	1	0	0	1
	0	0	1	1	0
	0	1	1	0	1
	1	0	1	0	1
	1	1	1	1	1
Boolean Expression: Sum = A ⊕ B ⊕ C-in					

The Full-Adder circuit above consists of three Ex-OR gates, two AND gates and an OR gate. The truth table for the Full-Adder includes an additional column to take into account the Carry-in input as well as the summed output and Carry-out. 4-bit Full-Adder circuits are available as standard IC packages in the form of the TTL 74LS83 or the 74LS283 which can add together two 4-bit binary numbers and generate a SUM and a CARRY output.

The Digital Comparator

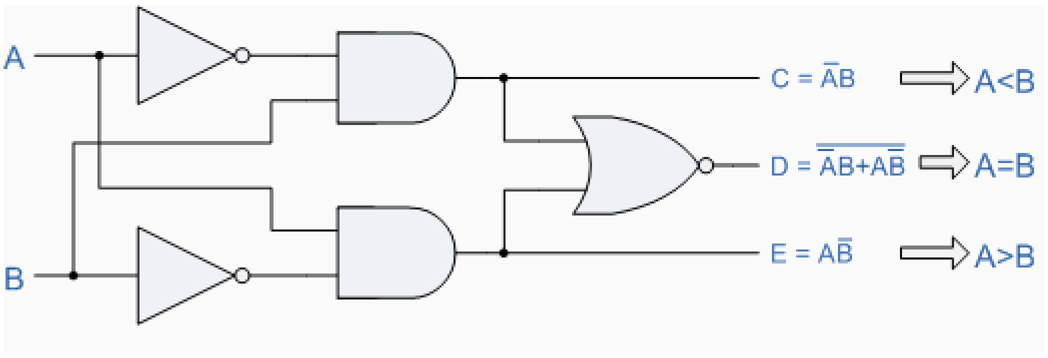
Another common and very useful combinational logic circuit is that of the Digital Comparator circuit. Digital or Binary Comparators are made up from standard AND, NOR and NOT gates that compare the digital signals at their input terminals and produces an output depending upon the condition of the inputs. For example, whether input A is greater than, smaller than or equal to input B etc.

Digital Comparators can compare a variable or unknown number for example A (A1, A2, A3, ... An, etc) against that of a constant or known value such as B (B1, B2, B3, ... Bn, etc) and produce an output depending upon the result. For example, a comparator of 1-bit, (A and B) would produce the following three output conditions.

$$A > B, \quad A = B, \quad A < B$$

This is useful if we want to compare two values and produce an output when the condition is achieved. For example, produce an output from a counter when a certain count number is reached. Consider the simple 1-bit comparator below.

1-bit Comparator



Then the operation of a 1-bit digital comparator is given in the following Truth Table.

Truth Table

Inputs		Outputs		
B	A	A > B	A = B	A < B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

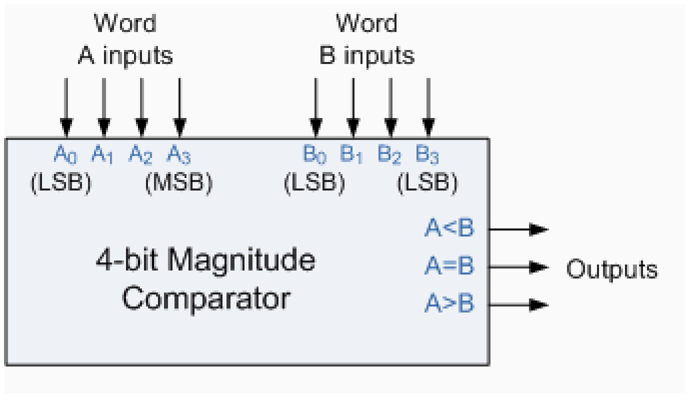
You may notice two distinct features about the comparator from the above truth table. Firstly, the circuit does not distinguish between either two "0" or two "1"'s as an output A = B is produced when they are both equal, either A = B = "0" or A = B = "1". Secondly, the output condition for A = B resembles that of a commonly available logic gate, the Exclusive-NOR or Ex-NOR gate giving $Q = A \oplus B$

Digital comparators actually use Exclusive-NOR gates within their design for comparing the respective pairs of bits in each of the two words with single bit comparators cascaded together to produce Multi-bit comparators so that larger words can be compared.

Magnitude Comparators

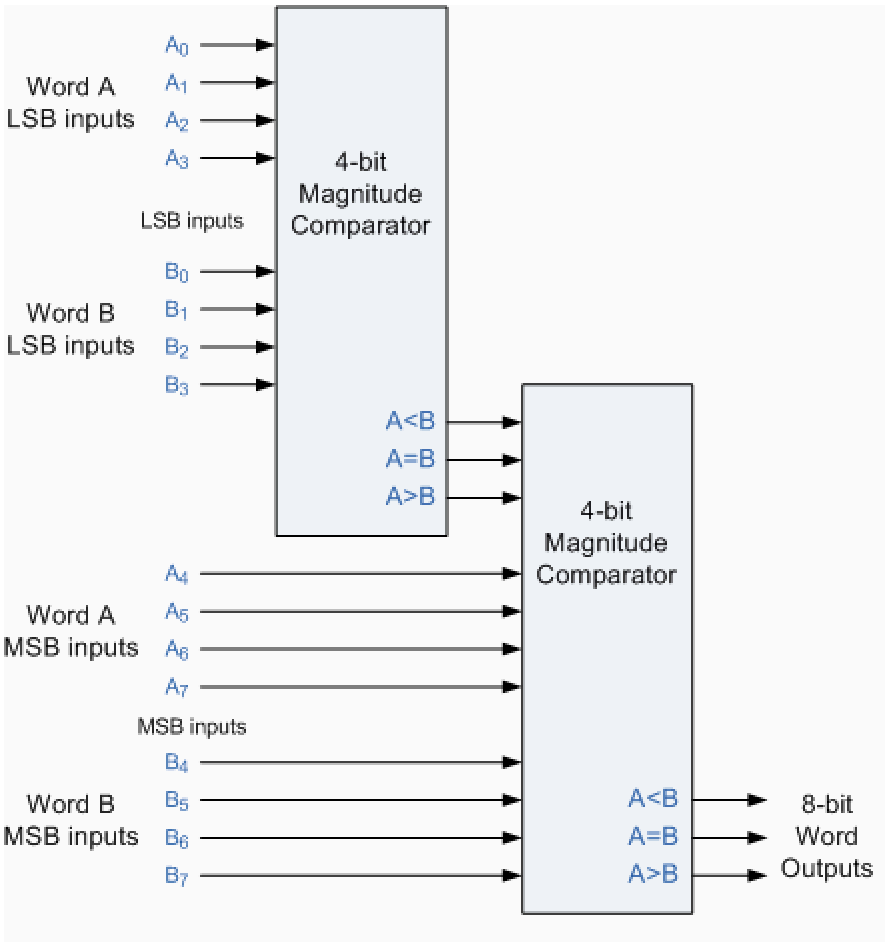
As well as comparing individual bits, multi-bit comparators can be constructed to compare whole binary or BCD words to produce an output if one word is larger, equal to or less than the other. A very good example of this is the 4-bit Magnitude Comparator. Here, two 4-bit words ("nibbles") are compared to produce the relevant output with one word connected to inputs A and the other to be compared against connected to input B as shown below.

4-bit Magnitude Comparator



Some commercially available Magnitude Comparators such as the 7485 have additional input terminals that allow more individual comparators to be "cascaded" together to compare words larger than 4-bits with magnitude comparators of "n"-bits being produced. These cascading inputs are connected directly to the corresponding outputs of the previous comparator as shown to compare 8, 16 or even 32-bit words.

8-bit Word Comparator



Designing a circuit that adds three 4-bit numbers

Recall that a 4-bit binary adder adds two binary numbers, where each number is of 4 bits.

For adding three 4-bit numbers we have:

Inputs

_ First 4-bit number X = X3X2X1X0

_ Second 4-bit number Y = Y3Y2Y1Y0

_ Third 4-bit number Z = Z3Z2Z1Z0

Outputs

The summation of X, Y, and Z. How many output lines are exactly needed will be discussed as we proceed.

To design a circuit using MSI devices that adds three 4-bit numbers, we first have to understand how the addition is done. In this case, the addition will take place in two steps, that is, we will first add the first two numbers, and the resulting sum will be added to the third number, thus giving us the complete addition.

Apparently it seems that we will have to use two 4-bit adders, and probably some extra hardware as well. Let us analyze the steps involved in adding three 4-bit numbers.

Step 1: Addition of X and Y

A 4-bit adder is required. This addition will result in a sum and a possible carry, as follows:

```
X3X2X1X0
Y3Y2Y1Y0
-----------------
C4 S3S2S1S0
```

Note that the input carry Cin = 0 in this 4-bit adder

Step 2: Addition of S and Z

This resulting partial sum (i.e. S3S2S1S0) will be added to the third 4-bit number Z3Z2Z1Z0 by using another 4-bit adder as follows, resulting in a final sum and a possible carry:

```
S3S2S1S0
Z3Z2Z1Z0
-----------------
D4 F3F2F1F0
```

where F3F2F1F0 represents the final sum of the three inputs X, Y, and Z. Again, in this step, the input carry to this second adder will also be zero.

Notice that in Step 1, a carry C4 was generated in bit position 4, while in Step 2, another carry D4 was generated also in bit position 4. These two carries must be added together to generate the final Sum bits of positions 4 and 5 (**F4** and **F5**).

Adding C4 and D4 requires a half adder. Thus, the output from this circuit will be six bits, namely F5 F4 F3F2F1F0 (See Figure)

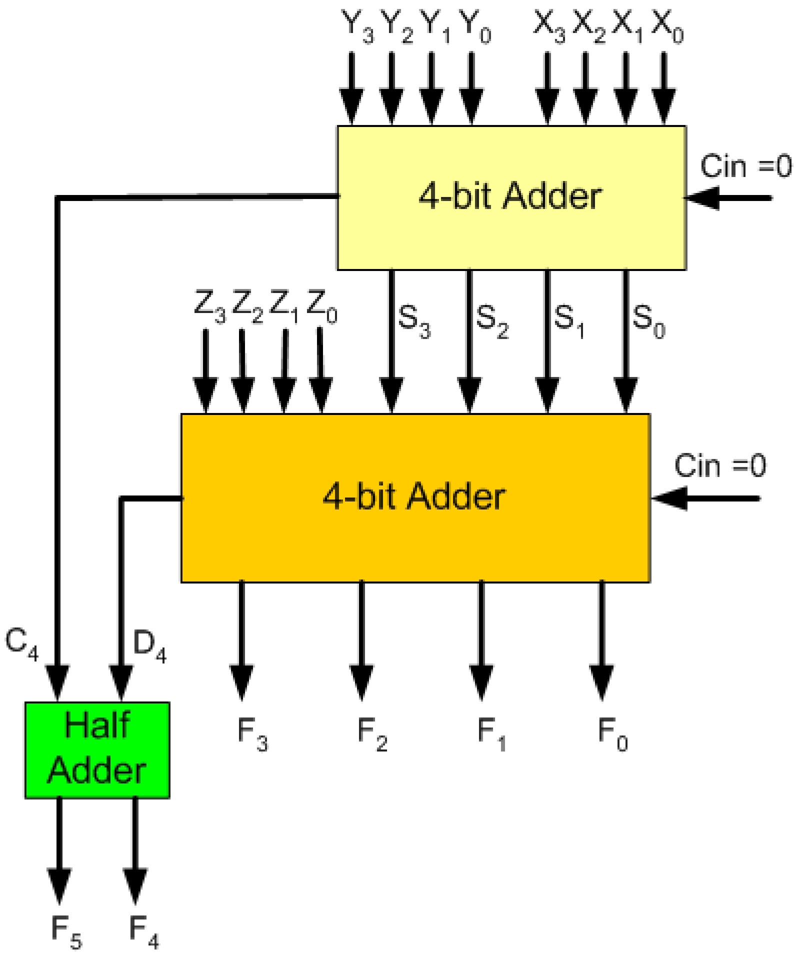
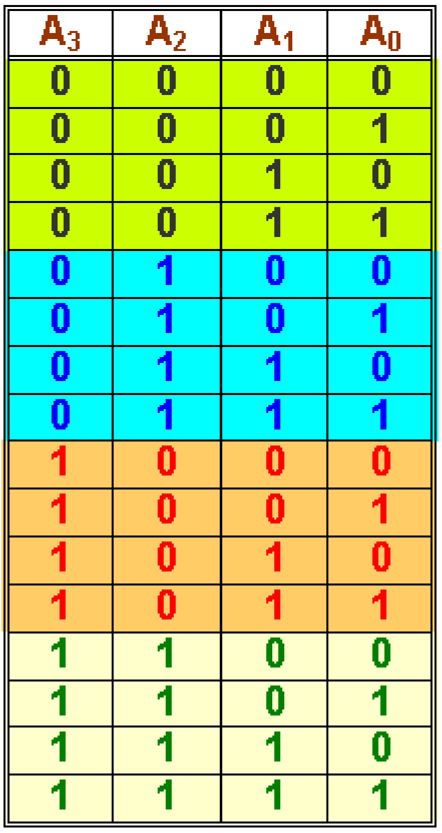


Figure : Circuit for adding three 4-bit numbers

Design a 4-to-16 Decoder using five 2-to-4 Decoders with enable inputs

We have seen how can we construct a bigger decoder using smaller decoders, by taking the specific example of designing a 3-to-8 decoder using two 2-to-4 decoders. Now we will design a 4-to-16 decoder using five 2-to-4 decoders. There are a total of sixteen possible input combinations, as shown in the table (Figure). These sixteen combinations can be divided into four groups, each group containing four combinations. Within each group, A3 and A2 remain constant, while A1 and A0 change their values. Also, in each group, same combination is repeated for A1 and A0

(i.e.00→01→10→11)



A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Figure : Combinations with 4 variables

Thus we can use a 2-to-4 decoder for each of the groups, giving us a total of four decoders (since we have sixteen outputs; each decoder would give four outputs). To each decoder, A1 and A0 will go as the input. A fifth decoder will be used to select which of the four other decoders should be activated. The inputs to this fifth decoder will be A3 and A2. Each of the four outputs of this decoder will go to each enable of the other four decoders in the "proper order". This means that line 0 (representing A3A2 = 00) of decoder '5' will go to the enable of decoder '1'. Line 1 (representing A3A2 = 01) of decoder '5' will go to the enable of de coder '2' and so on. Thus a combination of A3 and A2 will decide which "group" (decoder) to select, while the combination of A1 and A0 will decide which output line of that particular decoder is to be selected. Moreover, the enable input of decoder '5' will be connected to logic switch, which will provide logic 1 value to activate the decoder.

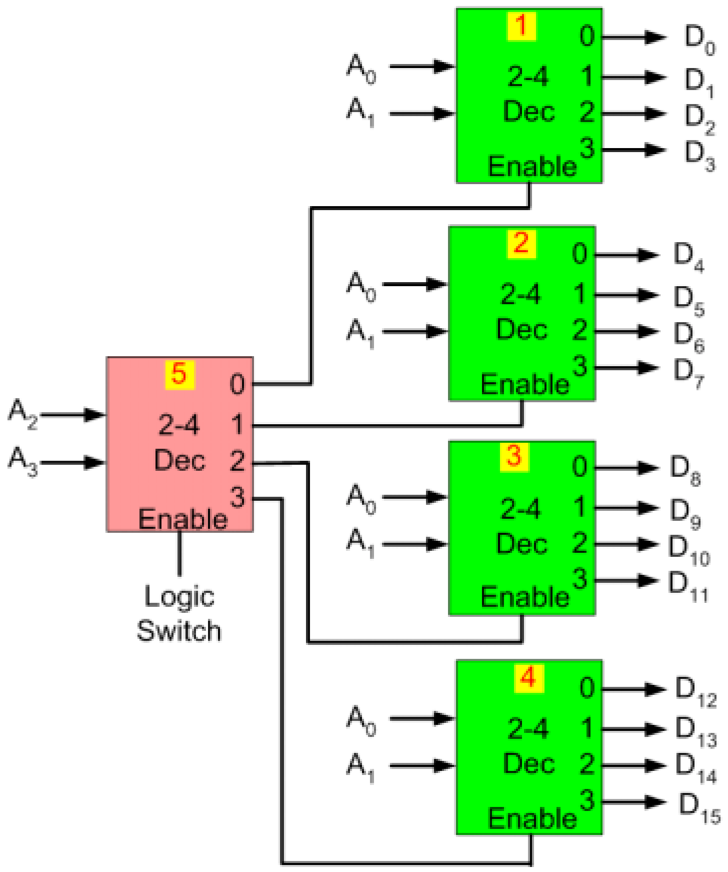


Figure: Constructing 4-to-16 decoder using 2-to-4 decoders

Decoder example: “Activate” line D2. The corresponding input combination that would activate this line is 0010. Now apply 00 at input of decoder ‘5’. This activates line ‘0’ connected to enable of decoder ‘1’. Once decoder ‘1’ is activated, inputs at A1A0 = 10 activate line D2.

Thus we get the effect of a 4-16 decoder using this design, by applying input

combinations in two steps.

As another example, to “activate” the line D10: The corresponding input combination is 1010. Apply 10 at the input of decoder ‘5’. This activates line ‘2’ connected to enable of decoder ‘3’. Once decoder ‘3’ is activated, the inputs at A1A0 = 10 activate line D10.

Given two 4-bit unsigned numbers A and B, design a circuit which outputs

the larger of the 2 numbers.

Here we will use Quad 2-1 Mux, and a 4-bit magnitude comparator. Both of these devices have been discussed earlier. The circuit is given in the figure Since we are to select one of the two 4-bit numbers A (A3A2A1A0) and B (B3B2B1B0), it is obvious that we will need a quad 2-1 Mux. The inputs to this Mux are the two 4-bit numbers A and B. The select input of the Mux must be a signal which indicates the relative magnitude of the two numbers A and B. This signal may be True if A<B or if A>B. Such signal is easily obtained from a 4-bit magnitude comparator.

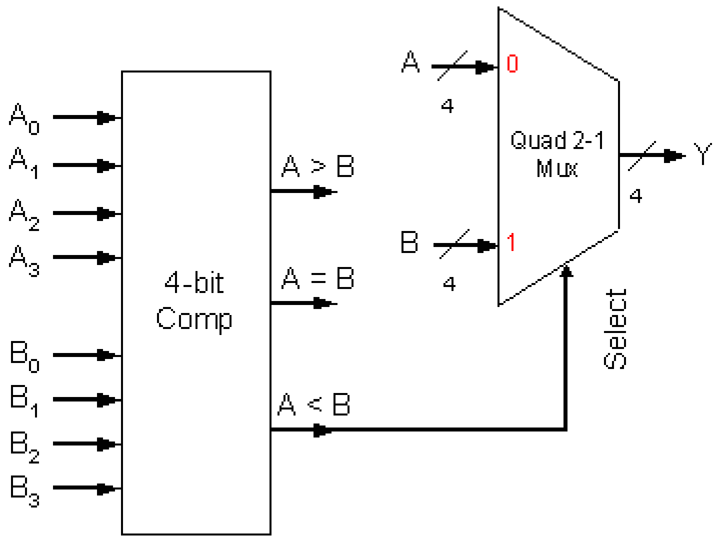


Figure : Circuit that outputs the larger of two numbers

By connecting the select input to the A<B output of the magnitude comparator, we must connect A to the 0 input of the Mux and B to the 1 input of the Mux . Alternatively, if we connect the select input to the A>B output of the magnitude comparator, we must connect A to the 1 input of the Mux and B the 0 input of the Mux . In either case, the Mux output will be the larger of the two numbers

Designing a 16-bit adder using four 4-bit adders

Adds two 16-bit numbers X (X0 to X15), and Y (Y0 to Y15) producing a 16-bit Sum S (S0 to S15) and a carry out C16 as the most significant position. Thus, four 4-bit adders are connected in cascade. Each adder takes four bits of each input (X and Y) and generates a 4-bit sum and a carry that is fed into the next 4-bit adder as shown in Figure .

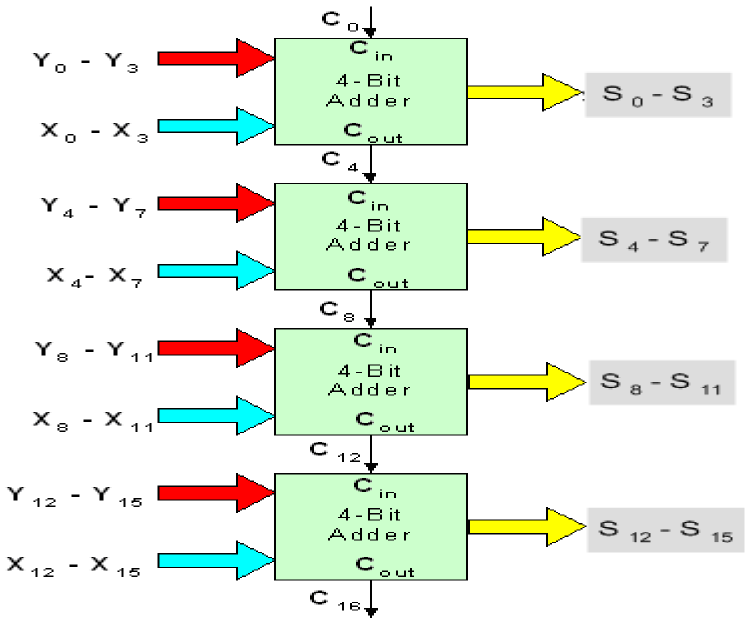


Figure : A 16-bit adder

Designing an Excess-3 code converter using a Decoder and an Encoder

In this example, the circuit takes a BCD number as input and generates the corresponding

Ex-3 code. The truth table for this circuit is given in figure 6. The outputs 0000, 0001, 0010, 1101, 1110, and 1111 are never generated To design this circuit, a 4-to-16 decoder and a 16-to-4 encoder are required. The design is given in figure 7. In this circuit, the decoder takes 4 bits as inputs, represented by variables w, x, y, and z. Based on these four bits, the corresponding min term output is activated. This decoder output then goes to the input of encoder which is three greater than the value generated by the decoder. The encoder then encodes the value and sends the output bits at A, B, C, and D. For example, suppose 0011 is sent as input. This will activate min term 3 of the decoder. This output is connected to input 6 of encoder. Thus the encoder will generate the corresponding bit combination, which is 0110.

w	x	y	z	A	B	C	D
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	0
1	0	0	1	1	1	0	0

Figure : table for BCD to Ex-3 conversion

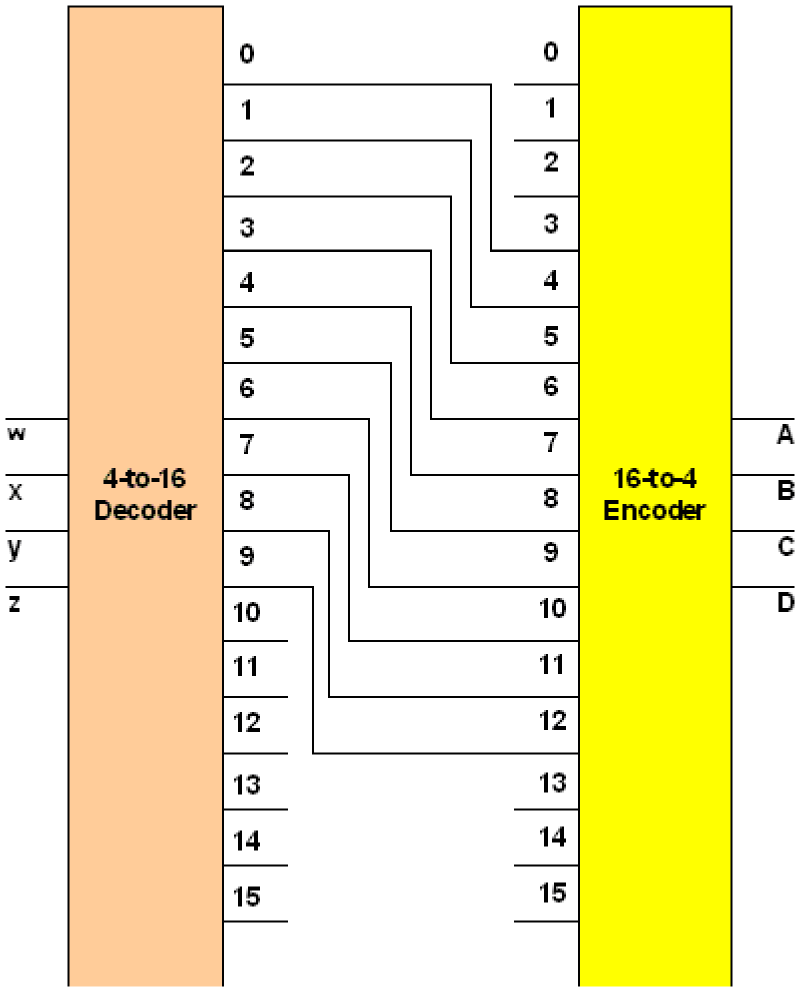


Figure : Circuit for BCD to Ex-3 conversion

Recommended question and answer –unit-4

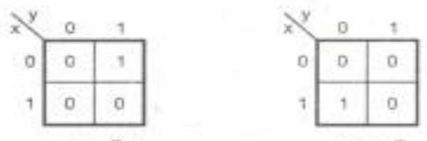
Jan 2009

Q.4 b) The 1-bit comparator had 3 outputs corresponding ,to x > y, x = y and x < y. It is possible to code these three outputs using two bits s1 s0 such as s1' s0 = 00, 10, 01 for x = y, x > y and x < y respectively. This implies that only two-output lines occur from each 1-bit comparator. However at the output of the last 1-bit comparator, an additional network must be designed to convert the end results back to three outputs. Design such a 1-bit comparator as well as the output converter network.

Ans. :

x	y	S_1	S_0	G	E	L
0	0	0	0	0	1	0
0	1	0	1	0	0	1
1	0	1	0	1	0	0
1	1	0	0	0	1	0

K-map simplification :



$S_0 = \bar{x}y$ $S_1 = x\bar{y}$

Fig. 4 (a)

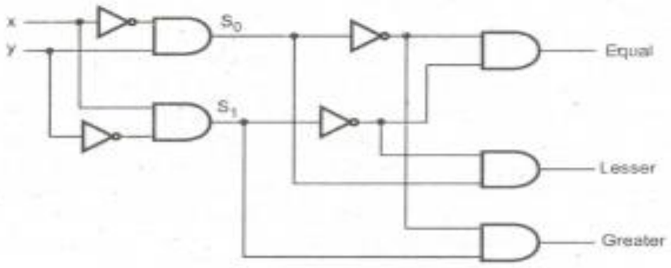


Fig. 4 (b)

PART B

Jan-2008

c) Realize F = !(x, y, z) = L (1, 2, 4, 5, 7) using 8 - to - 1 multiplexer (74L5151).
(4)

Ans.: Equation = f(x, y, z) = L (I, 2, 4, 5, 7)

Design table : We implement given equation SOP form using 8 : 1 MUX.

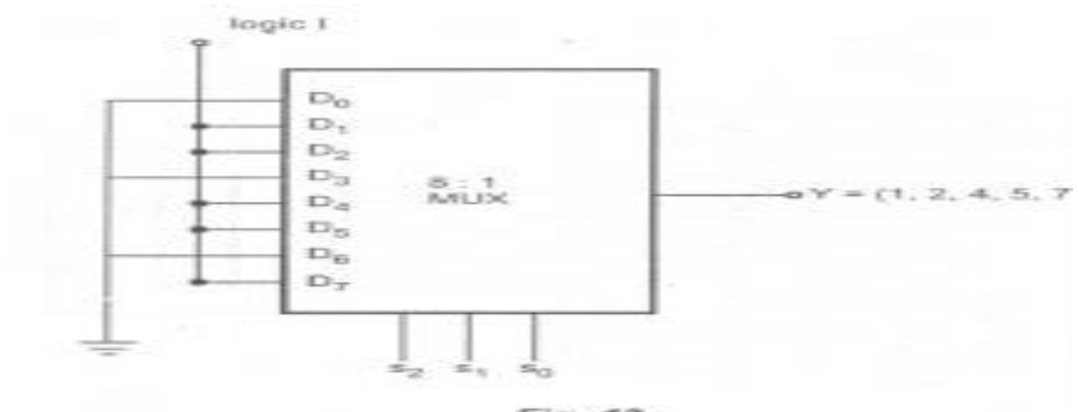


Fig. 13

Aug 2009

b) *Implement the following Boolean function using* 8 : 1 *multiplexer.*

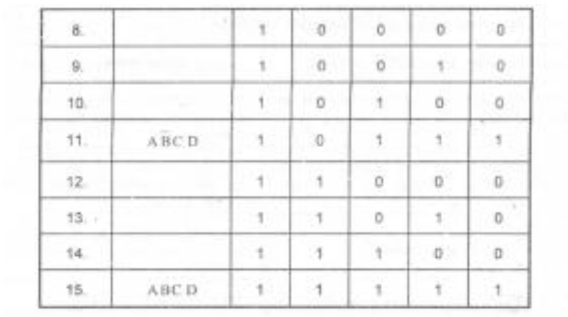
$$f(A, B, C, D) = \overline{A}\,B\,\overline{D} + A\,C\,D + \overline{B}\,C\,D + \overline{A}\,\overline{C}\,D$$

Ans. : The given Boolean expression is not in standard SOP form. Let us first convert this in standard SOP form

$$\begin{aligned}
F(A, B, C, D) &= \overline{A}\,B\,\overline{D}\,(C+\overline{C}) + A\,C\,D(B+\overline{B}) \\
&\quad + \overline{B}\,C\,D\,(A+\overline{A}) + \overline{A}\,\overline{C}\,D(B+\overline{B}) \\
&= \overline{A}\,B\,C\,\overline{D} + \overline{A}\,B\,\overline{C}\,\overline{D} + A\,B\,C\,D + A\,\overline{B}\,C\,D \\
&\quad + A\,\overline{B}\,C\,D + \overline{A}\,\overline{B}\,C\,D + \overline{A}\,B\,\overline{C}\,D + \overline{A}\,\overline{B}\,\overline{C}\,D \\
&= \overline{A}\,B\,C\,\overline{D} + \overline{A}\,B\,\overline{C}\,\overline{D} + A\,B\,C\,D + A\,\overline{B}\,C\,D \\
&\quad + \overline{A}\,\overline{B}\,C\,D + \overline{A}\,B\,\overline{C}\,D + \overline{A}\,\overline{B}\,\overline{C}\,D
\end{aligned}$$

The truth table for this standard SOP form can be given as

Sr. No.	Minterms	A	B	C	D	Y
0.		0	0	0	0	0
1.	$\overline{A}\,\overline{B}\,\overline{C}\,D$	0	0	0	1	1
2.		0	0	1	0	0
3.	$\overline{A}\,\overline{B}\,C\,D$	0	0	1	1	1
4.	$\overline{A}\,B\,\overline{C}\,\overline{D}$	0	1	0	0	1
5.	$\overline{A}\,B\,\overline{C}\,D$	0	1	0	1	1
6.	$\overline{A}\,B\,C\,\overline{D}$	0	1	1	0	1
7.		0	1	1	1	0

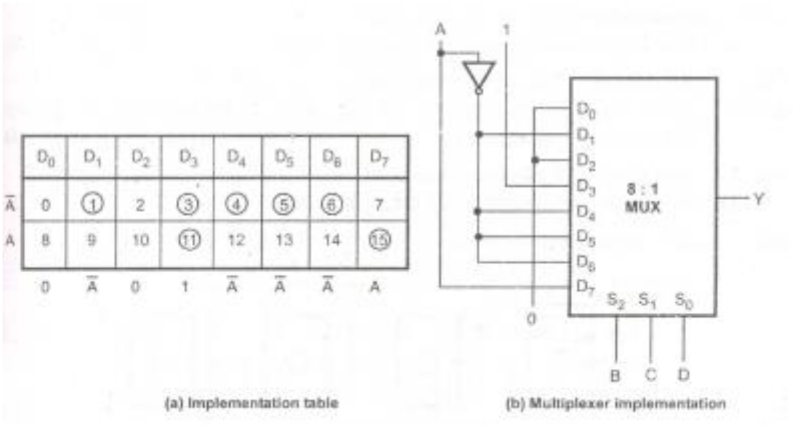


8.		1	0	0	0	0
9.		1	0	0	1	0
10.		1	0	1	0	0
11.	$A\bar{B}CD$	1	0	1	1	1
12.		1	1	0	0	0
13.		1	1	0	1	0
14.		1	1	1	0	0
15.	ABCD	1	1	1	1	1

Table 2 Truth table

From the truth table Boolean function can be implemented using 8: 1 multiplexer

as follows:



(a) Implementation table (b) Multiplexer implementation

Aug-2008

Q.3 a) *Realize the following Boolean function f (ABeD) = L (0*, 1, 3, 5, 7)

Using - i) 8 : 1 MUX (74151) ii) 4 : 1 MUX (74153).

Ans. : Design using 8 : 1 MUX :

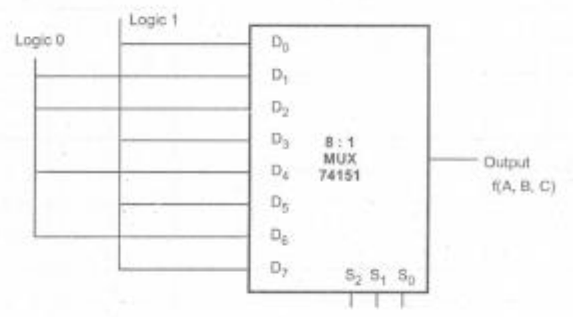
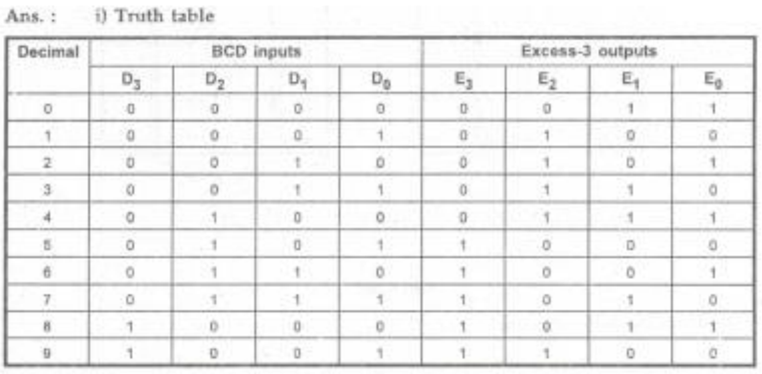


Fig. 5 (a)

b) *Design a combinational logic circuit that will convert a straight BCD digit to an Excess* - 3 *BCD digits.*

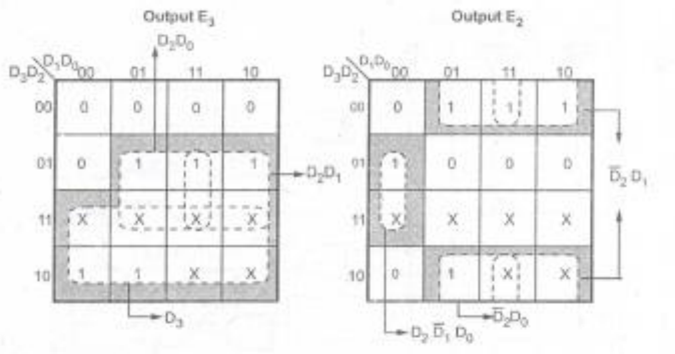
i) Construct the truth table. ii) Simplify each output function using Karnaugh map. and write the reduced equations. .iii) *Draw the resulting logic diagram.* ***(12)***



Ans. : i) Truth table

Decimal	BCD inputs				Excess-3 outputs			
	D_3	D_2	D_1	D_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

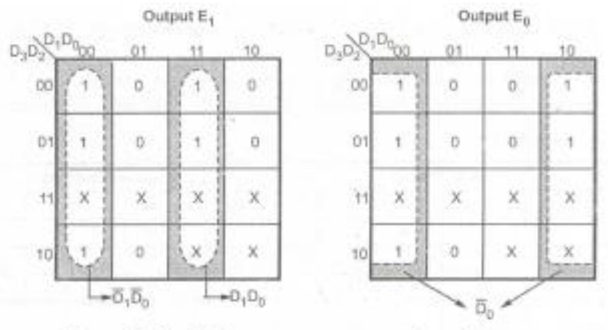
ii) Simplified equations using K-map



$E_3 = D_3 + D_2D_0 + D_2D_1$ $\qquad$ $E_2 = \overline{D}_2D_1 + \overline{D}_2D_0 + D_2\overline{D}_1\overline{D}_0$

$E_3 = D_3 + D_2(D_0 + D_1)$ $\qquad$ $E_2 = \overline{D}_2(D_1 + D_0) + D_2\overline{D}_1\overline{D}_0$

Fig. 6 (a)



$E_1 = \overline{D}_1\overline{D}_0 + D_1D_0$ $\qquad$ $E_0 = \overline{D}_0$

$= (D_1 \odot D_0)$

Fig. 6 (b)

iii) Resulting logic diagram

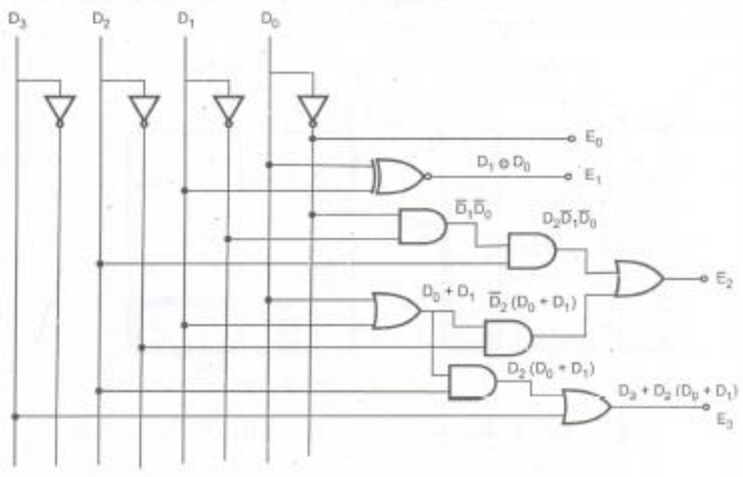


Fig. 6 (c) BCD to Excess-3 code converter