Combinational Logic Circuits

Many applications of digital logic require a circuit with multiple inputs and outputs in which the are uniquely determined by the current inputs. Such a circuit is called a **combinational logic circuit (CLC)**.

Not all circuits have this property. For example a circuit containing memory elements may well generate outputs that depend on the stored values as well as the input variables. Such a circuit is called a **sequential logic circuit (SLC).**

A combinational circuit may contain an arbitrary number of logic gates and inverters but **no feedback loop** .

A feedback loop is a signal path of a circuit that allows the output of a gate to propagate back to the input of that same gate.

Combinational logic circuit (CLC)

Adders

Comparators

Decoders

Encoders

Multuplexers

Demultiplexers

Sequential logic circuit (SLC)

Latches

Flip-Flops

Registers

Counters

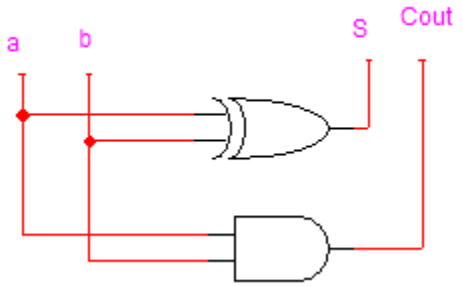
Adders

Additions are the most commonly performed arithmetic operation in digital systems. An adder combines 2 arithmetic operands using the addition rules.

An adder can perform subtraction as the addition of minuend and the complemented subtrahend.

Half adder

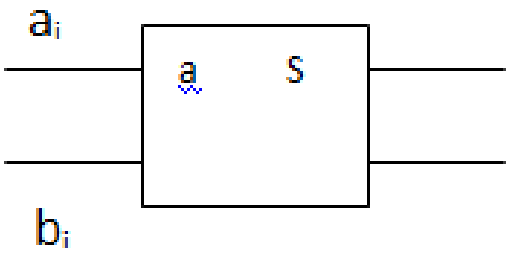
The simplest adder called a half adder adds two 1-bit operands *a* and *b*, producing a 2-bit sum: a low-adder bit is a half sum and the high-adder bit is carryout



Truth table

a_i	b_i	S_i	C_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S_i = a_i \oplus b_i$$
$$C_i = a_i \bullet b_i$$



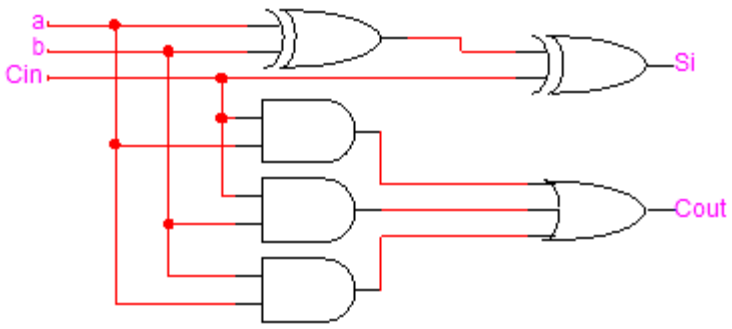
Full-adder

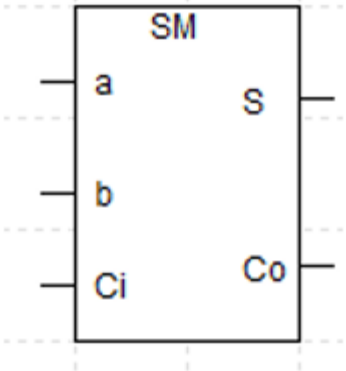
To add operands with more than one bit we most provide for carries between bit positions. This circuit is called a full-adder. A full-adder has 3 inputs a_i, b_i, and a carry-bit input.

a_i	b_i	C_{in}	S_i	Cout
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

$$S_i = a_i \oplus b_i \oplus C_{in}$$

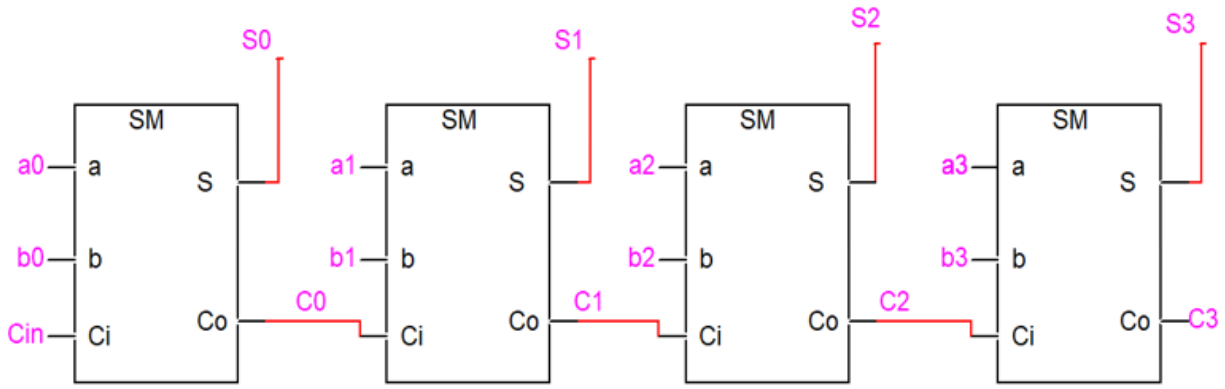
$$C_{out} = a_i b_i + a_i C_{in} + b_i C_{in}$$





Ripple Adders

Two binary words each with n bits can be added using a ripple adder which consists of n-full adders.



The carry input to the LSB C_o is normally set to 0 and the carry output of each full adder is connected to the carry input of the next most significant full adder.

A ripple adder is slow, since in the worst case a carry must propagate from the least significant full adder to the most significant one.

Carry look ahead Adders

A faster adder can be built by representing each C_{out} in terms of a_i and b_i.

$$C = a_i b_i + a_i C_i + b_i C_i = a_i b_i + C_i (a_i \vee b_i)$$

We define two functions:

$G_i = a_i b_i$ – carry generate function. A stage i unconditionally generate a carry if its addend bits are s.

$P_i = a_i + b_i$ carry propagate function. A stage i propagates carries if at least one of its addend bit is 1.

The carry output of a stage can be written in terms of the generate and propagate signals

$$C_{i+1} = G_i + P_i * C_i$$

To eliminate carry ripple we recursively expand the C_i term for each stage and multiply it to obtain a 2-level AND-OR expression.

Using this technique we can obtain the following carry equation for the first four adder stages.

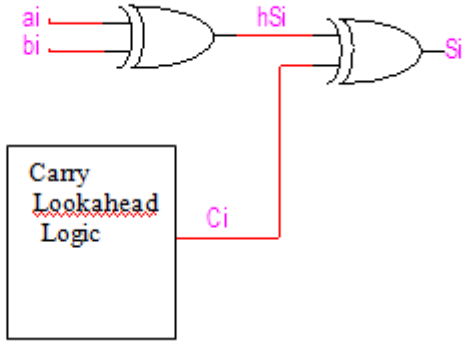
$C_i = G_0 + P_0 C_0$

$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = G_2 + P_2 C_2 = G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_0) = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 C_0$

Each equation correspond to a circuit with just 3 level of delay

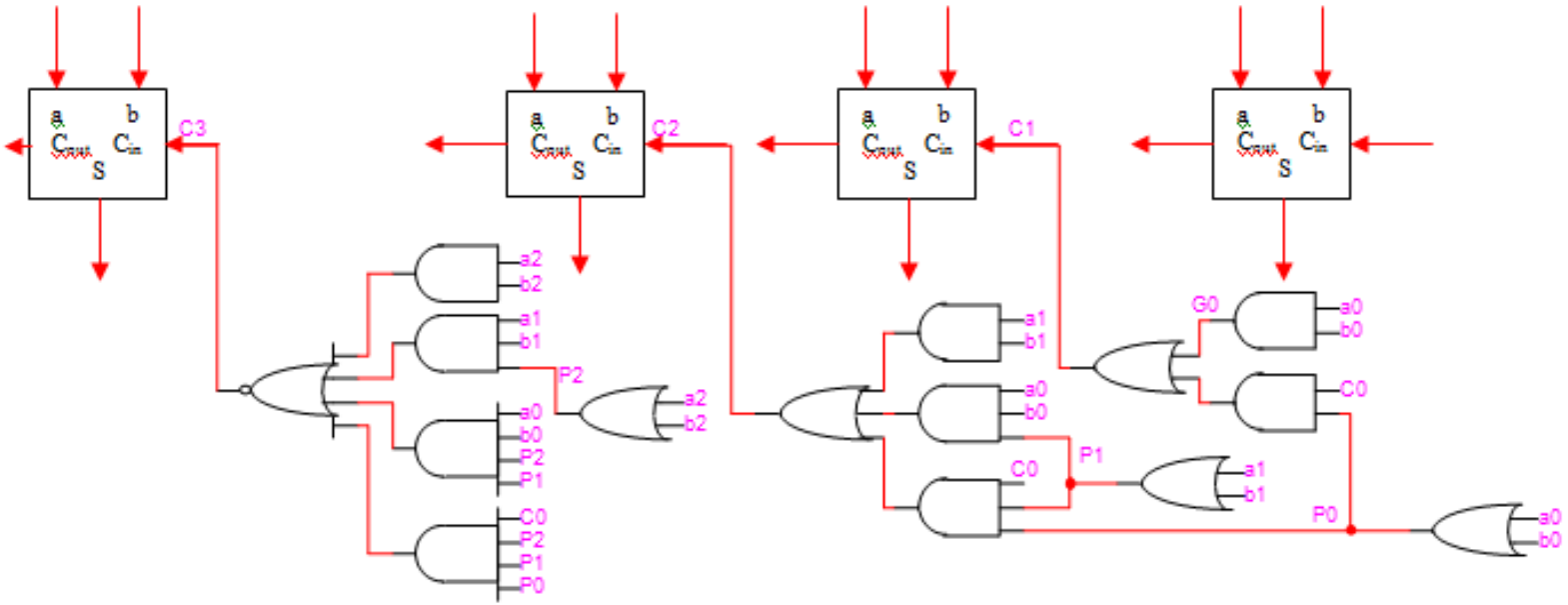
Structure of one stage of a carry look ahead adder



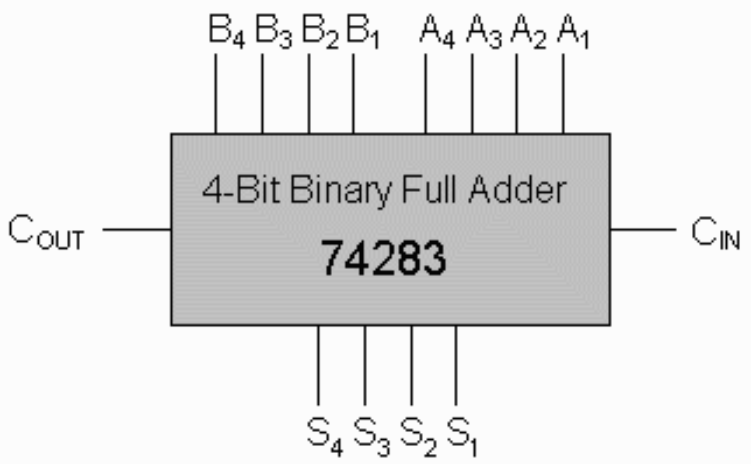
$G_i = a_i\, b_i$

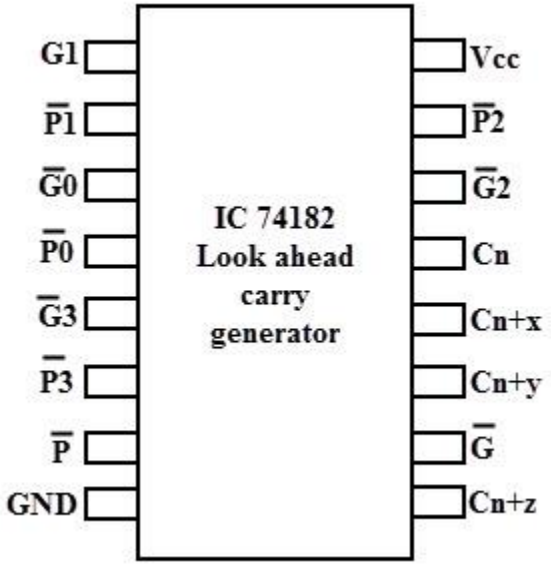
$P_i = a_i + b_i$

Structure of a n-bit carry look ahead adder



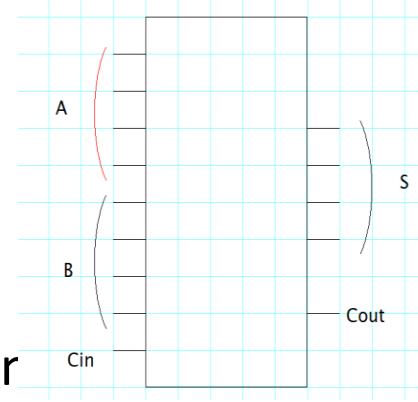
$C_3 = G_2 + P_2\, G_1 + P_2\, P_1\, G_0 + P_2\, P_1\, C_0$ $C_2 = = G_1 + P_1\, G_0 + P_1\, P_0\, C_0$ $C_i = G_0 + P_0\, C_0$





In group-carry look ahead adders is used a carry look ahead circuit (74x182) between each 4-bit adder.

BCD Adders



We can summarise the BCD addition procedure as follows :

1. Add two BCD numbers using ordinary binary additior
2. If four-bit sum is equal to or less than 9, no correction is needed. The sum is in proper BCD form.
3. If the four-bit sum is greater than 9 or if a carry is generated from the four-bit sum, the sum is invalid.

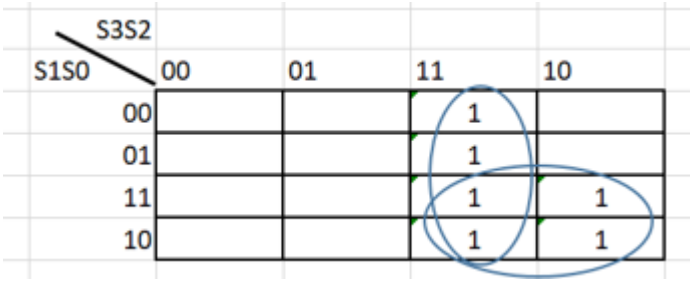
   To correct the invalid sum, add 0110_2 to the four-bit sum. If a carry results from this addition, add it to the next higher-order BCD digit.

   Thus to implement BCD adder we require :

1. 4-bit binary adder for initial addition
2. Logic circuit to detect sum greater than 9
3. One more 4-bit adder to add 0110 in the sum if sum is greater than 9 or carry is 1.

The logic circuit to detect sum greater than 9 can be determined by simplifying the boolean expression of given truth table.

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

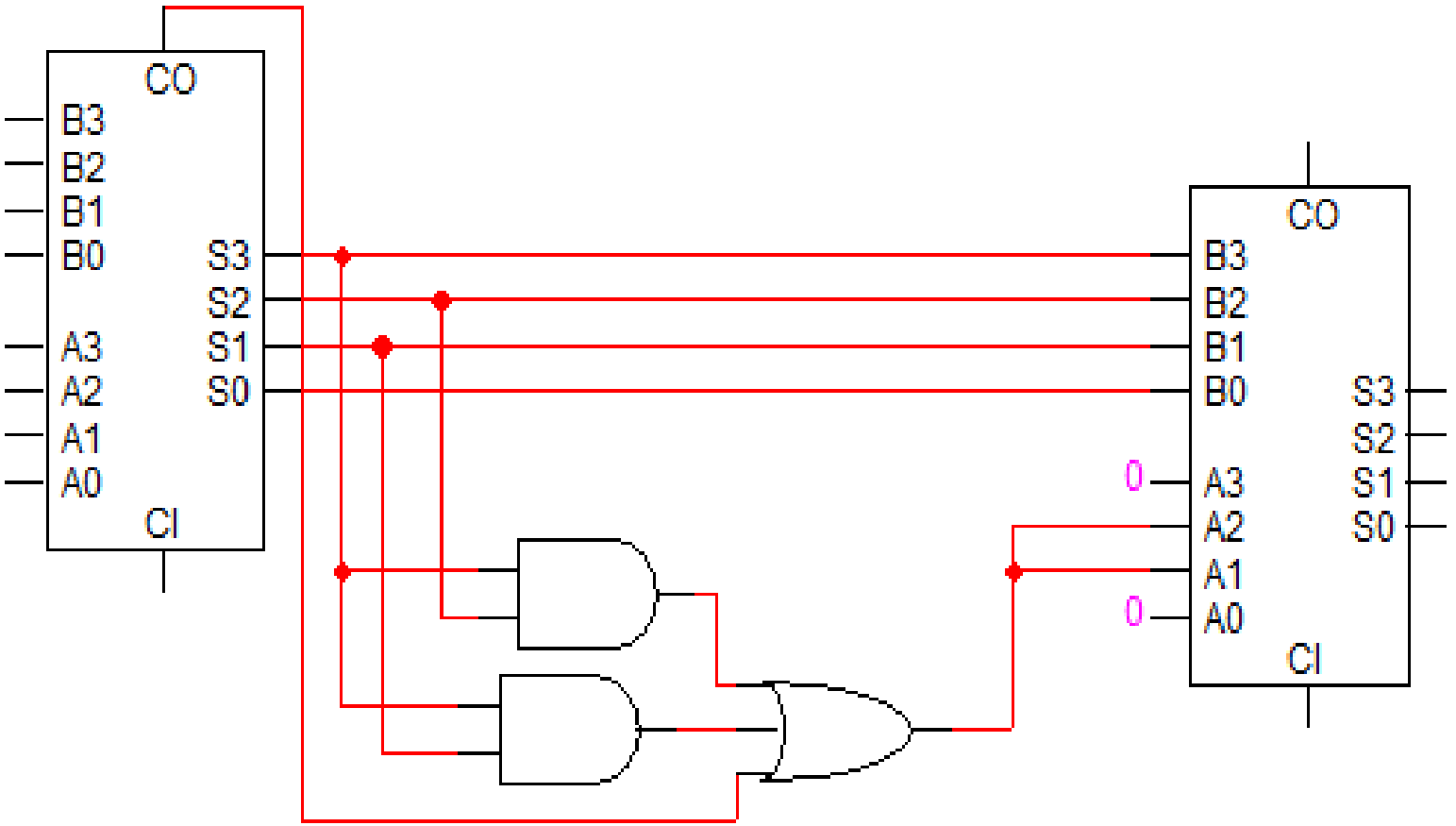


$$Y = S_3S_2 + S_3S_1$$

To include the second correction condition it is necessary to add to this function Cout.

$$Fcor = S_3S_2 + S_3S_1 + Cout$$

Block diagram



Comparators

- **A circuit that compares two binary words and shows whether they are equal is called a comparator.**
- Some comparators also indicate an arithmetic relationship (greater or less than) between the words. These devices are often called **magnitude comparators**.

Truth Table for 1-bit comparator

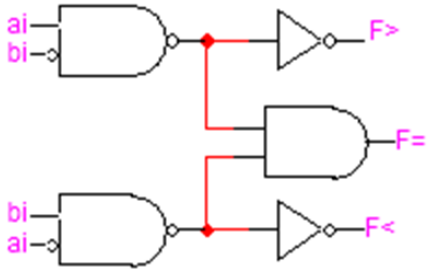
a_i	b_i	$F_=$	$F_<$	$F_>$
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	1	0	0

$$F_= = \overline{a_i \oplus b_i} = \overline{\overline{a_i} b_i + a_i \overline{b_i}} = \overline{\overline{a_i} b_i} \cdot \overline{a_i \overline{b_i}}$$

$$F_< = \overline{a_i} b_i$$

$$F_> = a_i \overline{b_i}$$

The 1-bit magnitude comparator.

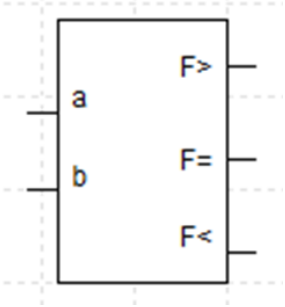


Exclusive OR and Exclusive NOR gates may be viewed as 1-bit comparators

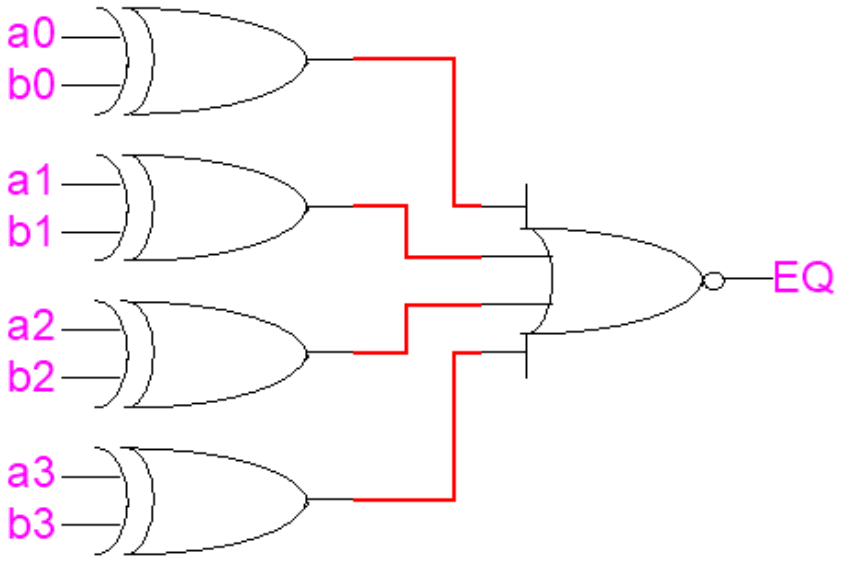
The active-high output Eq is asserted if the inputs are equal.

The active-high output diff is asserted if the inputs are different

Logic symbol



Ex of 4-bit parallel comparator



4-bit magnitude comparator

- the equality relation A = B $F_{A=B} = f_{=3} f_{=2} f_{=1} f_{=0}$,

because the relation A = B assumes that:

$a_3 = b_3$ and
$a_2 = b_2$ and
$a_1 = b_1$ and
$a_0 = b_0$;

- Function larger $A > B$ assumes that: $F_{A>B} = f_{>3} + f_{=3} f_{>2} + f_{=3} f_{=2} f_{>1} + f_{=3} f_{=2} f_{=1} f_{>0}$

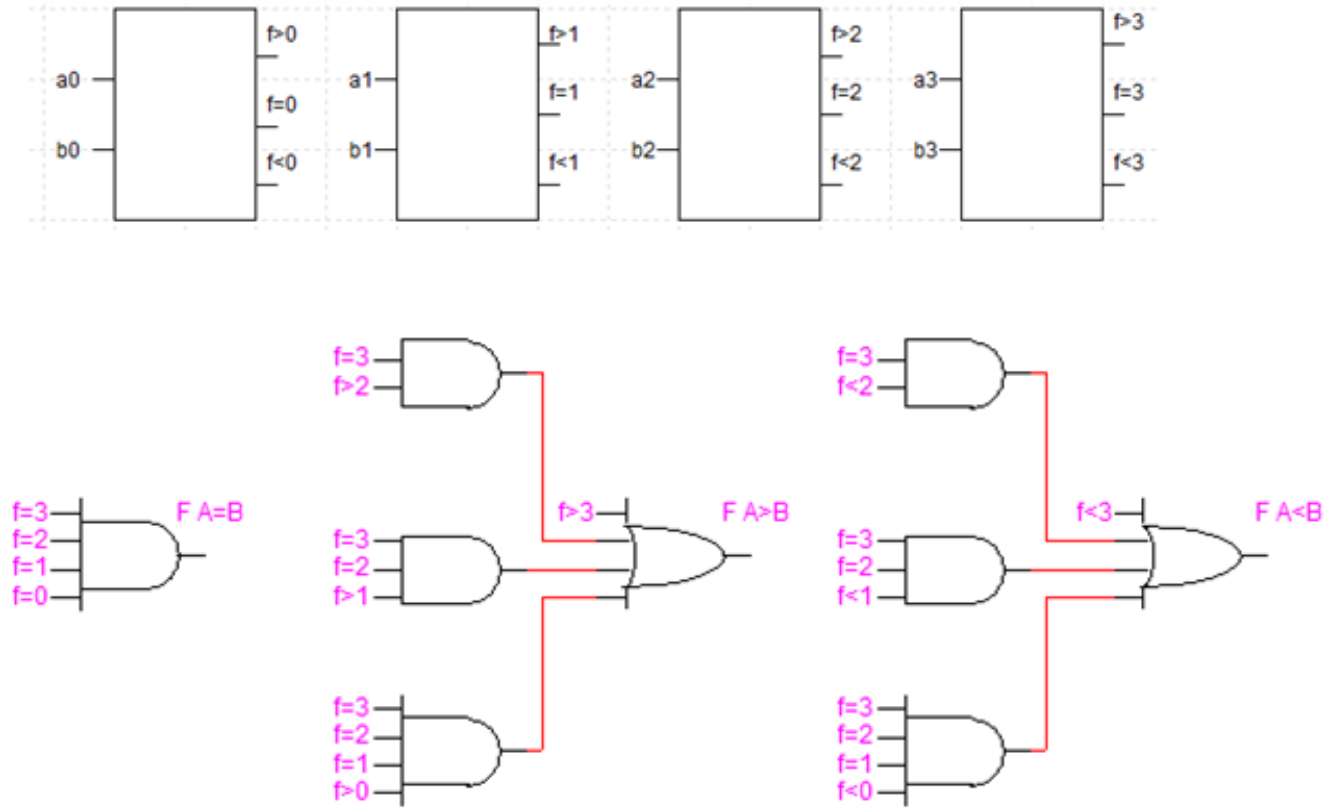
$a_3 > b_3$; or
$a_3 = b_3$ and $a_2 > b_2$; or
$a_3 = b_3$ and $a_2 = b_2$ and $a_1 > b_1$; or
$a_3 = b_3$ and $a_2 = b_2$ and $a_1 = b_1$ and $a_0 > b_0$

- Fuction smaler $A < B$ assumes that: $F_{A<B} = f_{<3} + f_{=3} f_{<2} + f_{=3} f_{=2} f_{<1} + f_{=3} f_{=2} f_{=1} f_{<0}$

$a_3 < b_3$; or
$a_3 = b_3$ and $a_2 < b_2$; or
$a_3 = b_3$ and $a_2 = b_2$ and $a_1 < b_1$; or
sau $a_3 = b_3$ and $a_2 = b_2$ and $a_1 = b_1$ and $a_0 < b_0$

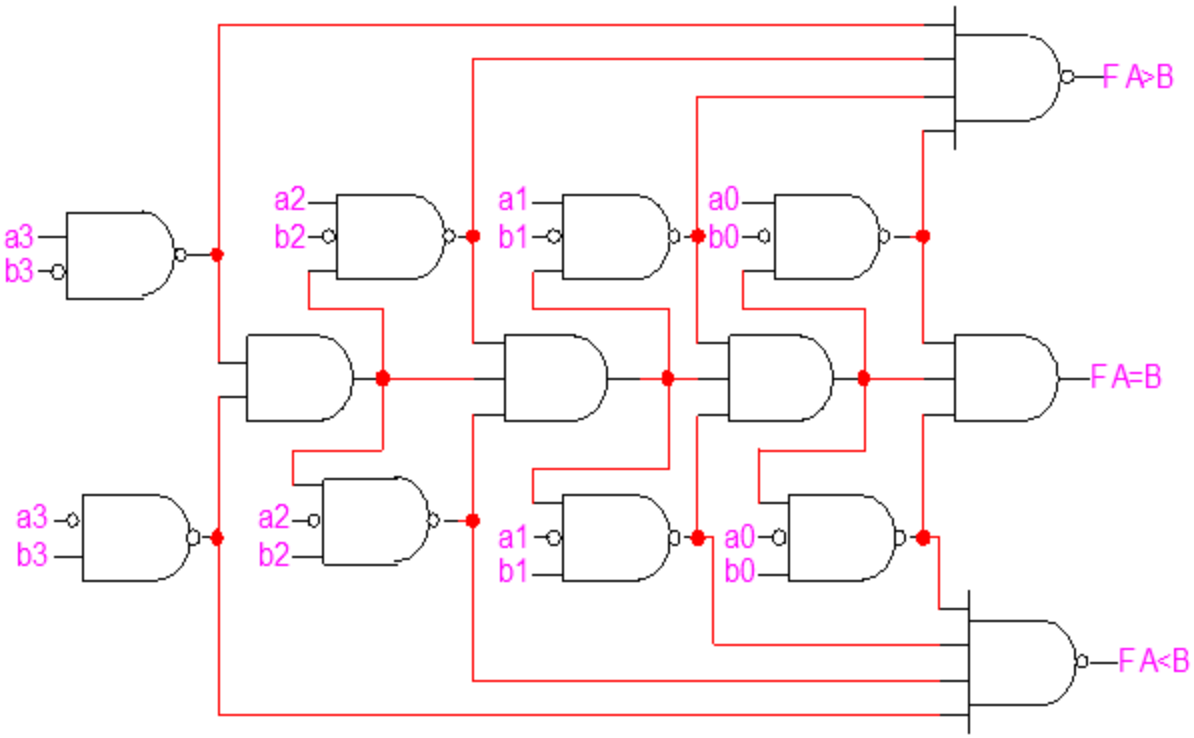
4-bit magnitude comparator

Logic circuit of the parallel comparator that performs these functions for 4 bits:

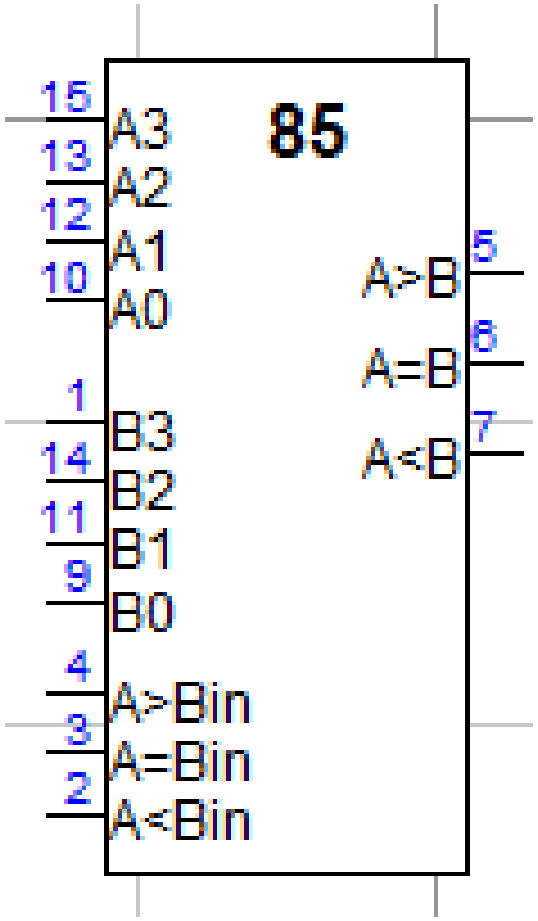


4-bit magnitude comparator

The same circuit in the sequential version:



74_85 magnitude comparator



Decoders

A decoder is a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different.

The most commonly used input code is an n-bit binary code, where an n-bit word represents one of 2^n different coded values, normally the integers from 0 troughs 2^n-1.

Sometimes an n-bit binary code is truncated to represent fewer than 2^n values. For example, in the BCD code the 4-bit combinations 0000 through 1001 represent the decimal digits 0-9, and combinations 1010 through 1111 are not used.

Binary Decoder

The most common decoder circuit is an n-to-2^n decoder or binary decoder. Such a decoder has an n-bit binary input code and a 1-out-of-2^n output code that means that only 1 output bit is asserted at any time.

The truth table of a 2-to-4decoder

Inputs			Outputs			
En	I1	I0	Y3	Y2	Y1	Y0
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

If EN=0 then all of the outputs are 0

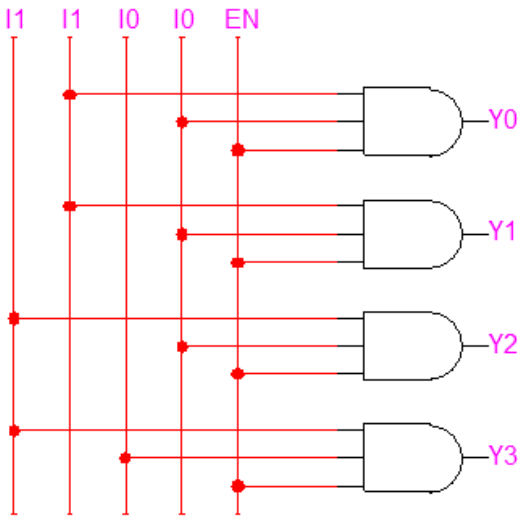
If EN=0 one of Yi are equal to 1

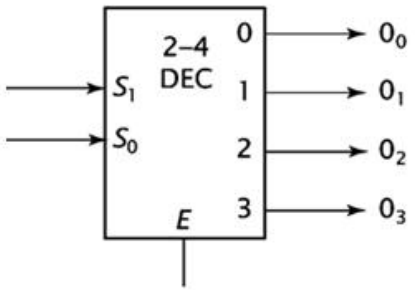
$Y_0 = \overline{I}_1 \overline{I}_0$

$Y_1 = \overline{I}_1 I_0$

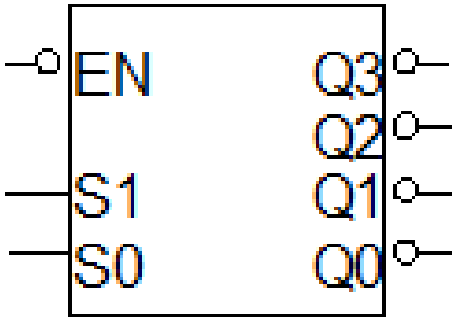
$Y_2 = I_1 \overline{I}_0$

$Y_3 = I_1 I_0$





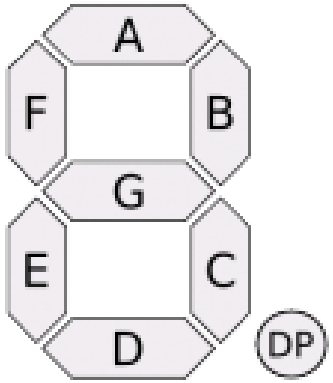
Most decoders were originally designed with active-low outputs, since TTL inverting gate are generally faster that non inverting ones.



Seven-Segment Decoder

A seven-segment display uses LED or LCD elements and is used in watches, calculators and instruments to display decimal data.

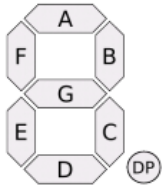
A digit is displayed by illuminating a subset of the seven line segments



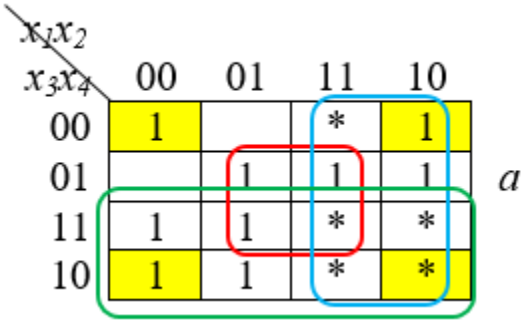
Individual segments 7 Segment Display marked

Truth Table

Decimal number	Code				Outputs						
	8	4	2	1							
	x_1	x_2	x_3	x_4	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	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	*	*	*	*	*	*	*



Individual segments 7 Segment Display marked



$$F_a = x_1 + x_3 + \bar{x}_2\bar{x}_4 + x_2x_4$$

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.

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.

Address Decoders

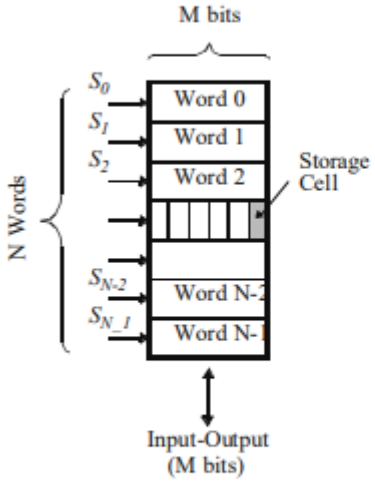
Amongst its many uses, a decoder is widely used to decode the particular memory location in the computer memory system. Decoders accept the address code generated by the CPU which is a combination of address bits for a specific location in the memory.

In a memory system, there are several memory ICs are combined and each one has their unique address to distinguish from other memory locations.

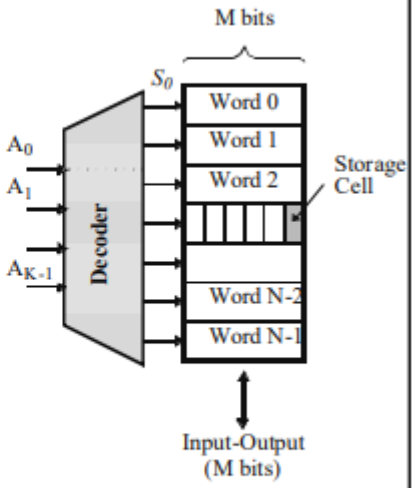
Instruction Decoder

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.

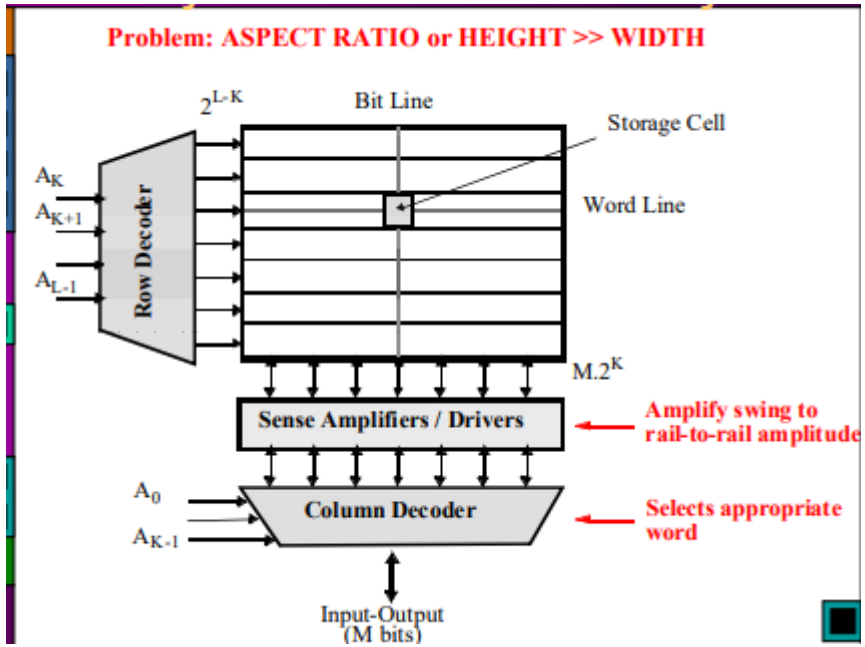
Memory Decoders



N words => N select signals
Too many select signals



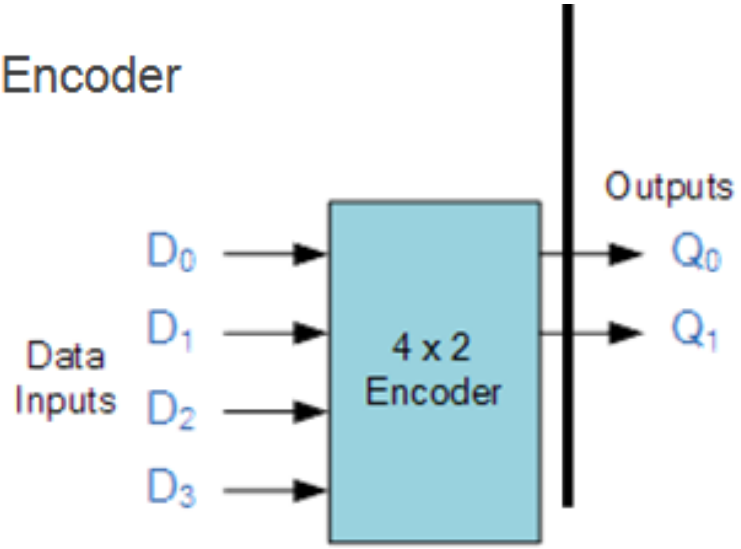
Decoder reduces # of select signals
$K = log_2 N$



Encoder

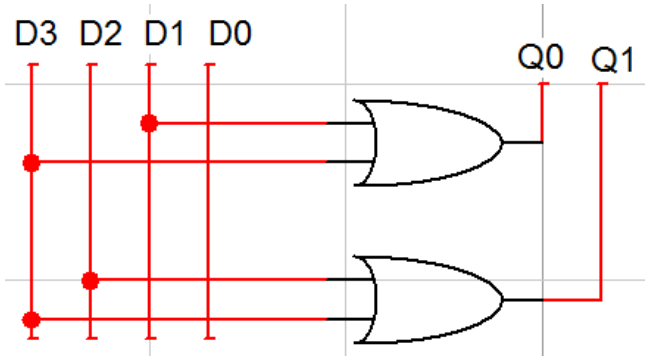
An encoder has just the opposite function as a decoder. A 2^n-to-n binary encoders has an 1-out-of-2^n inputs code (only one input is 1) and an n-bit binary output code.

4-to-2 Bit Binary 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

$$Q_0 = D_1 + D_3$$
$$Q_1 = D_2 + D_3$$



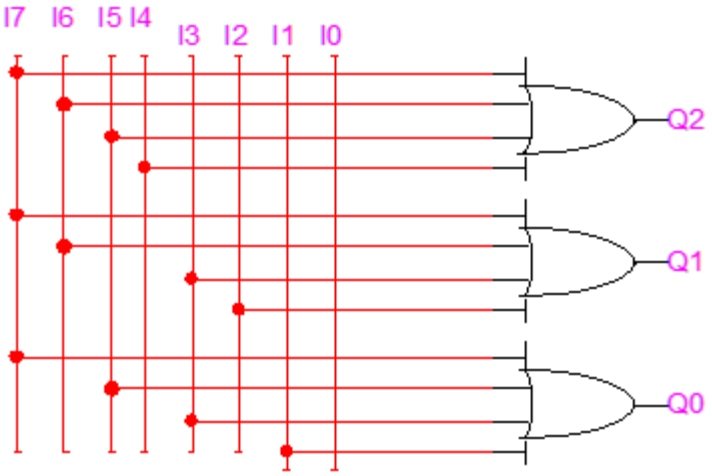
8-to-3 encoder

I7	I6	I5	I4	I3	I2	I1	I0	Q2	Q1	Q0
1	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0

$Q2=I7+I6+I5+I4$

$Q1=I7+I6+I3+I2$

$Q0=I7+I5+I3+I1$

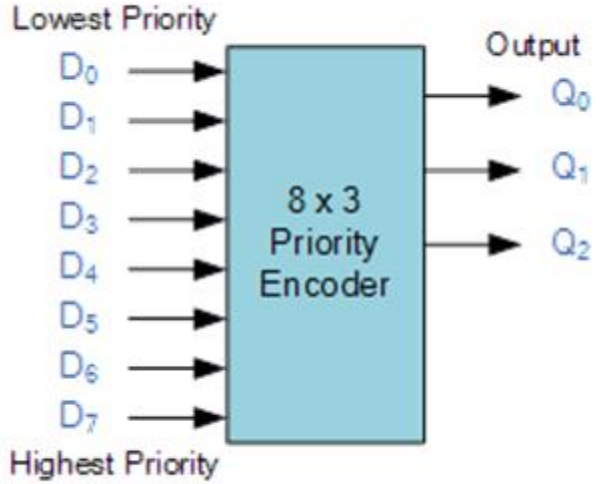


Priority Encoder

One of the main disadvantages of standard digital encoders is that they can generate the wrong output code when there is more than one input present at logic level “1”.

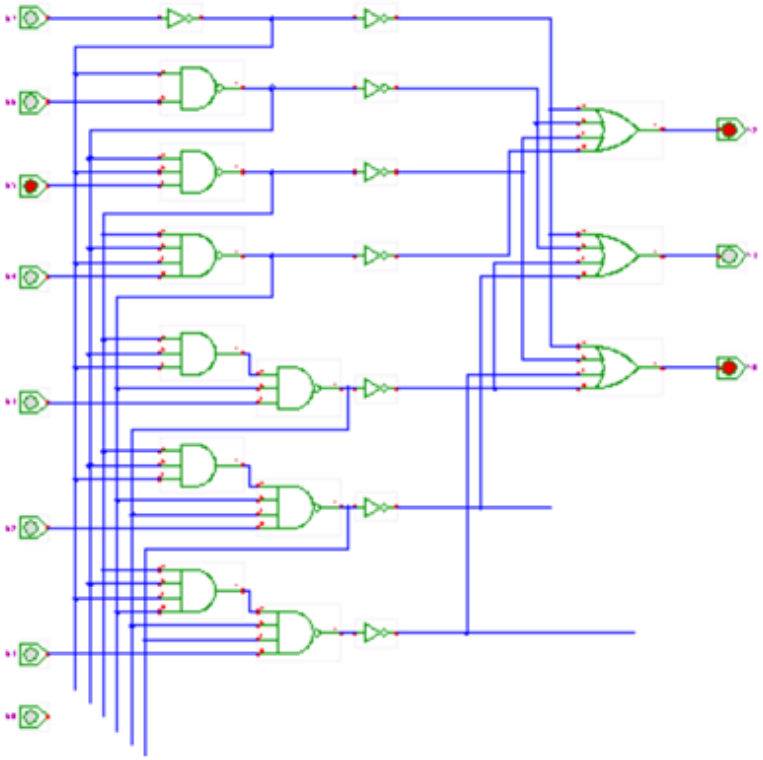
One simple way to overcome this problem is to “Prioritise” the level of each input pin and if there was more than one input at logic level “1” the actual output code would only correspond to the input with the highest designated priority. Then this type of digital encoder is known commonly as a **Priority Encoder** or **P-encoder** for short.

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 = don't care



$$Q_0 = \sum\left(\overline{D}_6\left(\overline{D}_4\overline{D}_2D_1 + \overline{D}_4D_3 + D_5\right) + D_7\right)$$

$$Q_1 = \sum\left(\overline{D}_5\overline{D}_4\left(D_2 + D_3\right) + D_6 + D_7\right)$$

$$Q_2 = \sum\left(D_4 + D_5 + D_6 + D_7\right)$$

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.

Positional Encoders

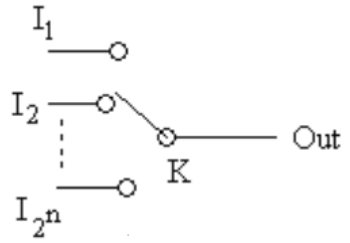
- Another more common application is in magnetic positional control as used on ships navigation or for robotic arm positioning etc. Here for example, the angular or rotary position of a compass is converted into a digital code by a 74LS148 8-to-3 line priority encoder and input to the systems computer to provide navigational data.

Interrupt Requests

- Other applications especially 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 so needs some way of knowing when a particular peripheral device wants to communicate with it.

  The processor does this by using “Interrupt Requests” or “IRQ” signals to assign priority to all the peripheral 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.

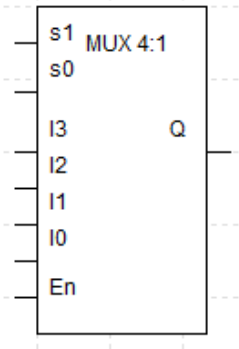
Multiplexer



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

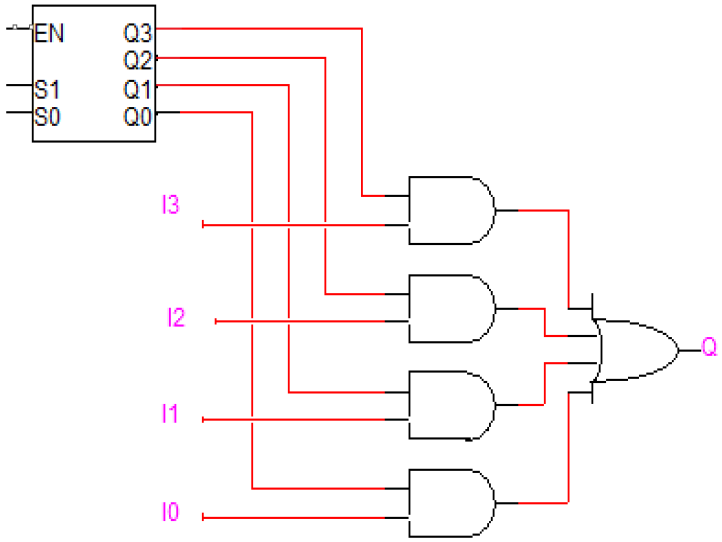
4x1 Multiplexer

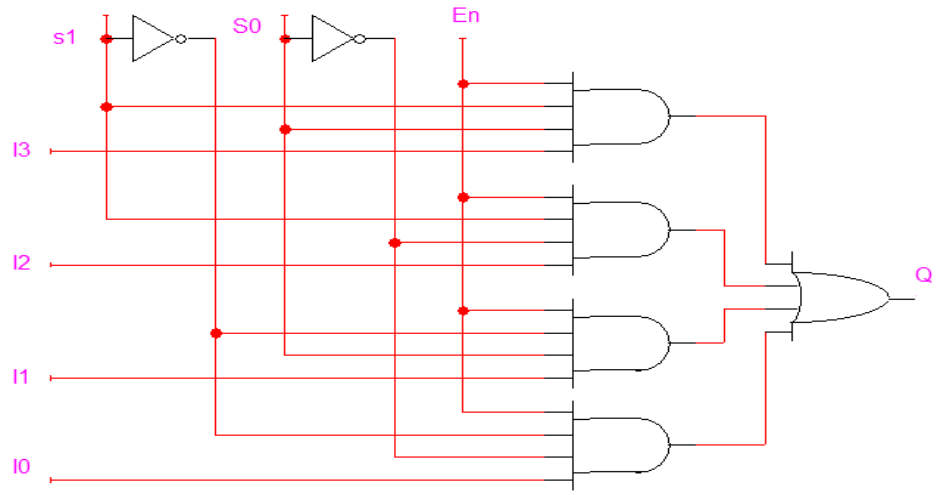
- 4x1 Multiplexer has four data inputs I_3, I_2, I_1 & I_0, two selection lines s_1 & $s_0$and one output Y.



S1	S0	I3	I2	I1	I0	En	Q
*	*	*	*	*	*	0	0
0	0	*	*	*	I0	1	I0
0	1	*	*	I1	*	1	I1
1	0	*	I2	*	*	1	I2
1	1	I3	*	*	*	1	I3

$$Q = I_3 s_1 s_0 En + I_2 s_1 \bar{s}_0 En + I_1 \bar{s}_1 s_0 En + I_0 \bar{s}_1 \bar{s}_0 En$$

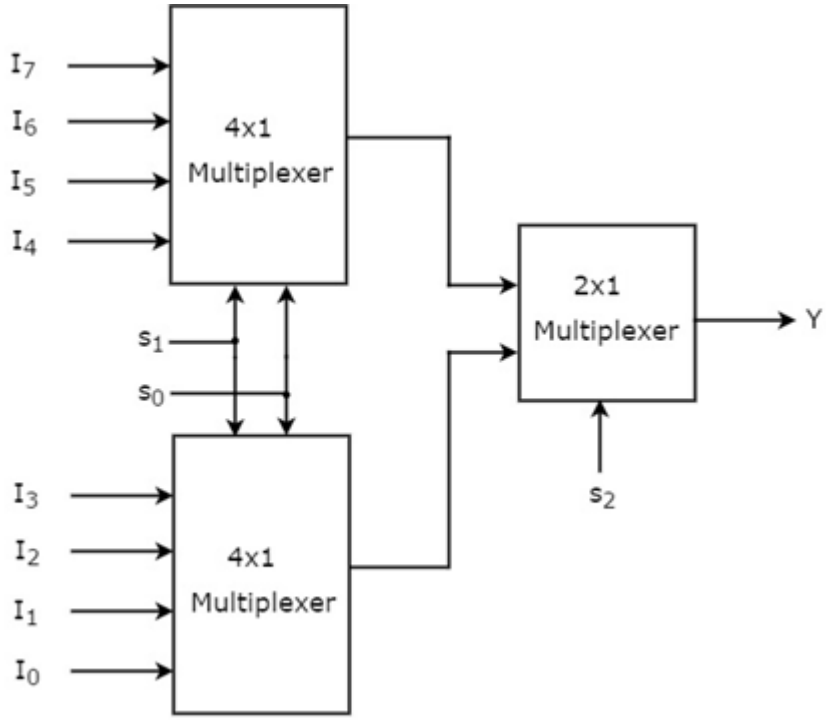




Implementation of Higher-order Multiplexers

Let us implement 8x1 Multiplexer using 4x1 Multiplexers and 2x1 Multiplexer.

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



Applications of Multiplexers

Multiplexers are used in various applications wherein multiple-data need to be transmitted by using single line.

- **Communication System**

  A communication system has both a communication network and a transmission system. By using a multiplexer, the efficiency of the communication system can be increased by allowing the transmission of data, such as audio and video data from different channels through single lines or cables.

- **Computer Memory**

  Multiplexers are used in computer memory to maintain a huge amount of memory in the computers, and also to reduce the number of copper lines required to connect the memory to other parts of the computer.

- **Telephone Network**

  In telephone networks, multiple audio signals are integrated on a single line of transmission with the help of a multiplexer.

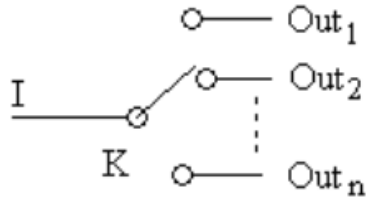
- **Transmission from the Computer System of a Satellite**

  Multiplexer is used to transmit the data signals from the computer system of a spacecraft or a satellite to the ground system by using a GSM satellite.

Demultiplexer

The process of getting information from one input and transmitting the same over one of many outputs is called demultiplexing.

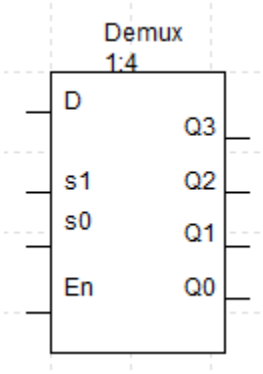
A demultiplexer is a combinational logic circuit that receives the information on a single input and transmits the same information over one of 2n possible output lines.



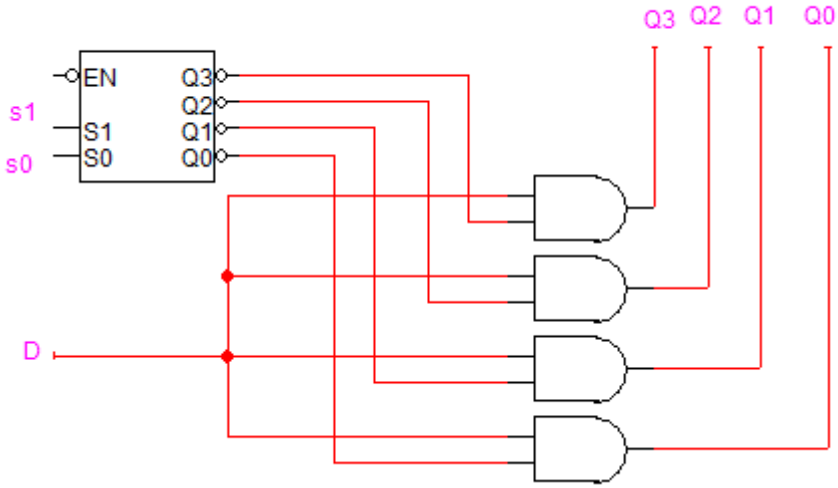
Demultiplexers are also called as data distributors, since they transmit the same data which is received at the input to different destinations.

Thus, a demultiplexer is a 1-to-N device where as the multiplexer is an N-to-1 device.

1-to-4 Demultiplexer



En	D	S1	S2	Q3	Q2	Q1	Q0
0	*	*	*	0	0	0	0
1	D	0	0	0	0	0	**D**
1	D	0	1	0	0	**D**	0
1	D	1	0	0	**D**	0	0
1	D	1	1	**D**	0	0	0



$$Q_3 = D \cdot s_1 s_0 En$$

$$Q_2 = D \cdot s_1 \bar{s}_0 En$$

$$Q_1 = D \cdot \bar{s}_1 s_0 En$$

$$Q_0 = D \cdot \bar{s}_1 \bar{s}_0 En$$