C.P.PATEL & F.H.SHAH COMMERCE COLLEGE

(MANAGED BY SARDAR PATEL EDUCATION TRUST)

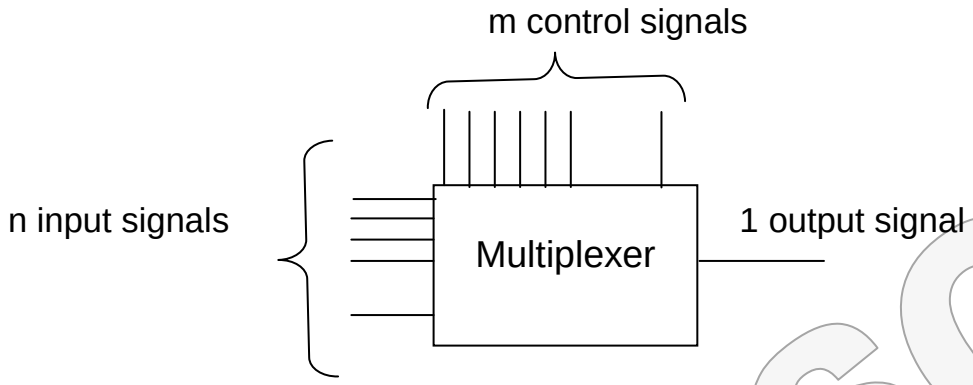
BCA, BBA(ITM) & PGDCA PROGRAMME

PGDCA-UNIT-4

❖ MULTIPLEXER :

Multiplexer means many into one.
A multiplexer is a circuit with many inputs but only one output. By applying control signal we can generate any input to the output. Multiplexer is also known as **Data Selector** because the output bit depends on the input data bit that is selected.

The following figure shows general idea of multiplexer.



◆ 8 TO 1 Multiplexer :

Figure from notebook.

In the figure we have 8 input data bits from D_0 to D_7 . From these bits only one of them can be transmitted as output that depends on the value of ABC(control word)

For **e.g.** If ABC=000 then only upper AND gate is enabled and remaining are disabled (i.e. the upper AND gate has output as D_0 and remaining AND gates have output 0) and as a result OR gate generates the output as D_0 . i.e. $Y=D_0$.

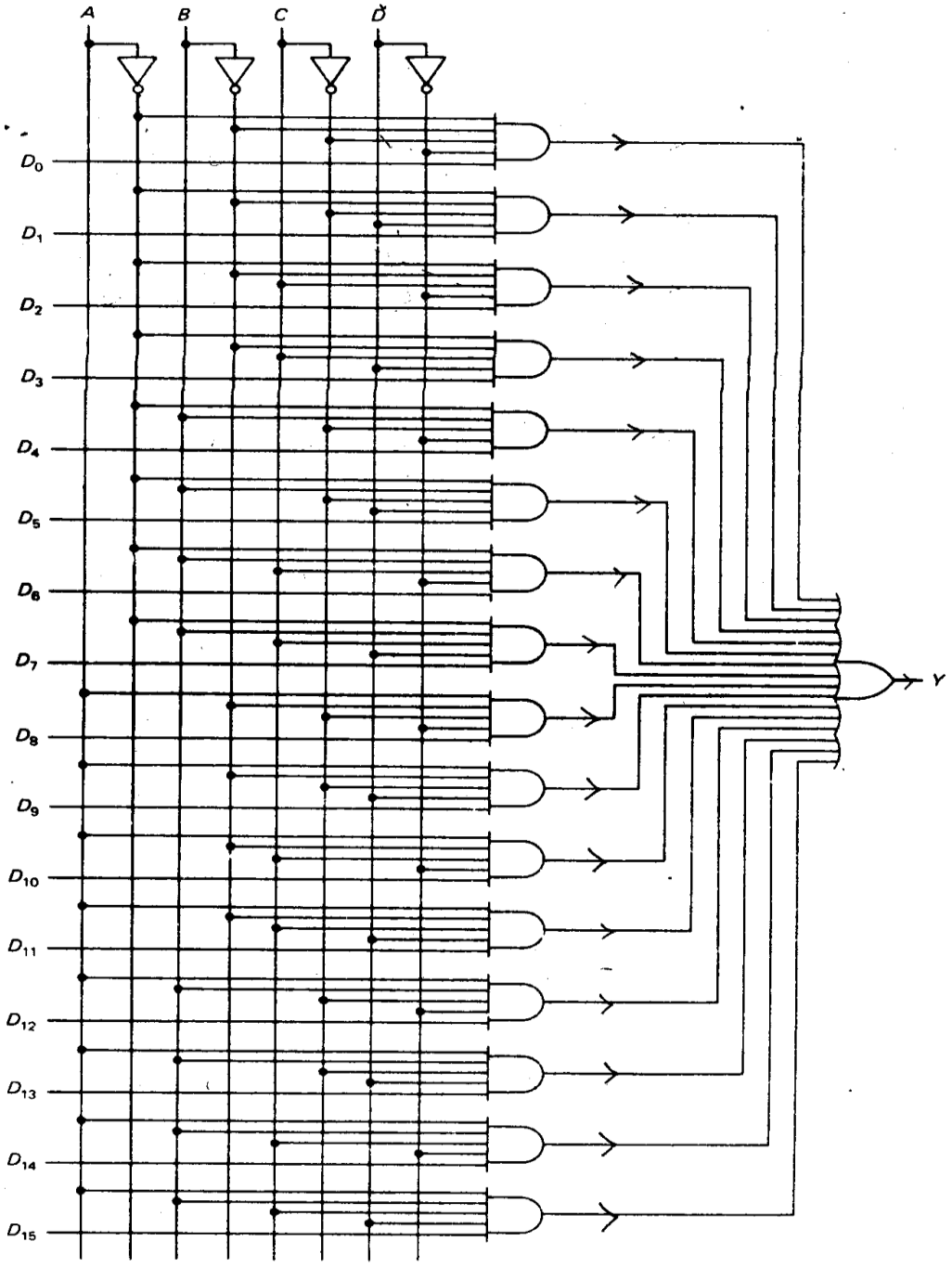
Thus from the eight input signals D_0 to D_7 one of the signal is transmitted as output depending upon the **control signal ABC.**

◆ 16 TO 1 Multiplexer :

In the figure we have 16 input data bits from D_0 to D_{15}. From these bits only one of them can be transmitted as output that depends on the value of ABCD(control word)
For **e.g.** If ABCD=0000 then only upper AND gate has output as D_0 and remaining AND gates have output 0 and as a result OR gate generates the output as D_0 .

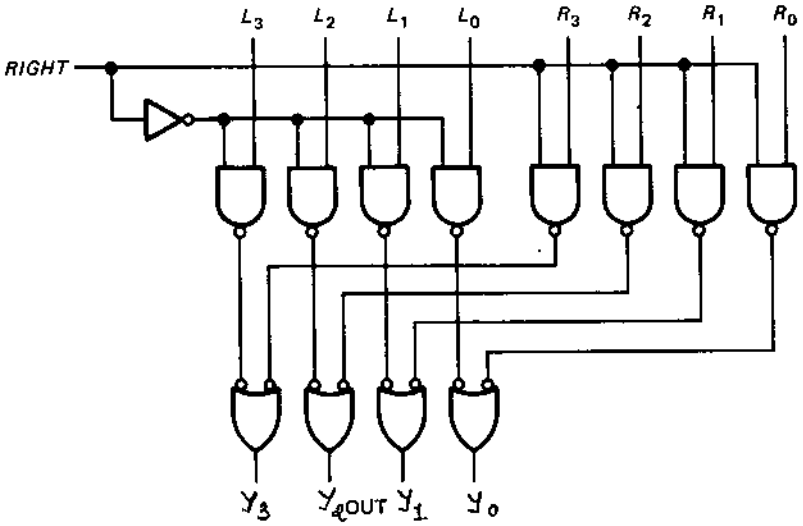
i.e. $Y=D_0$.

Thus from the sixteen input signals D_0 to D_{15} one of the signal is transmitted as output depending upon the **control signal ABCD.**



16 to 1 multiplexer

❖ Nibble Multiplexer (Word Multiplexer):



The figure shows a Nibble Multiplexer that has two input nibble(words) i.e. left nibble and right nibble and one output word.

The input nibble on the left is : $L_3\ L_2\ L_1\ L_0$ and on the right is $R_3\ R_2\ R_1\ R_0$.
When the control signal **RIGHT** is low then the four AND gates on the left are acti activated. Therefore the output $Y_3\ Y_2\ Y_1\ Y_0 = L_3\ L_2\ L_1\ L_0$.

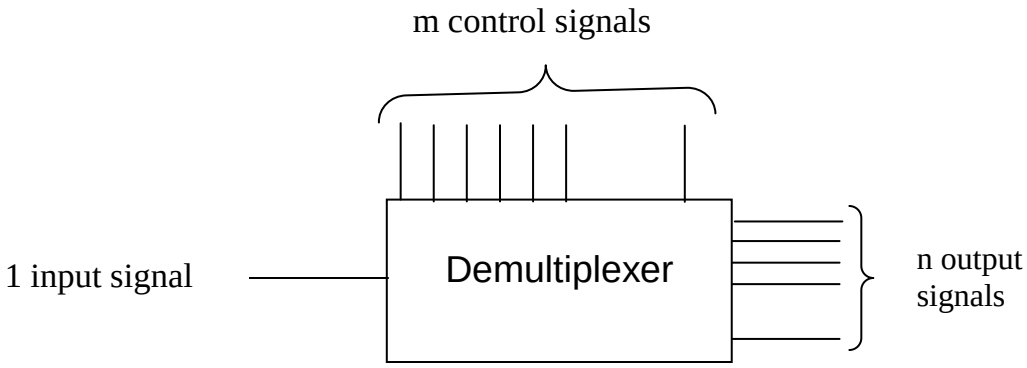
Thus When RIGHT=0 then output= left nibble

when the control signal **RIGHT** is high then the four AND gate on the right are activated. Therefore the output $Y_3\ Y_2\ Y_1\ Y_0 = R_3\ R_2\ R_1\ R_0$

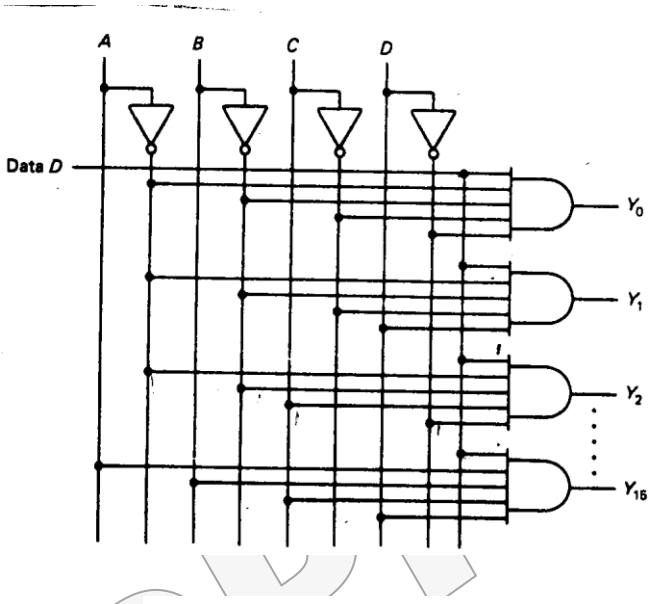
Thus When RIGHT=1 then output= right nibble

❖ Demultiplexer

Demultiplexer means one into many. A demultiplexer is a logic circuit with one input and many outputs. By applying control signals, we can steer the input signal to one of the output lines.



◆ 1 to 16 Demultiplexer :



1 to 16 demultiplexer

The input bit is labeled as D and is transmitted to the data bit of the output lines. It depends on the value of ABCD, the control input.

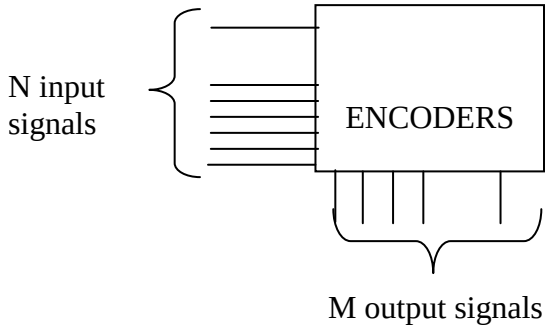
When ABCD=0000 the upper AND gate is enabled and all other AND gates are disabled. So data bit D is transmitted to the output Y_0 thus Y_0=D.
If D is low Y_0 is low and if D is high Y_0 is high.

When ABCD=1111 the lower AND gate is enabled and all other AND gates are disabled. So data bit D is transmitted to the output Y_{15} thus Y_{15}=D.
If D is low Y_{15} is low and if D is high Y_{15} is high.

Ass : Explain 1 to 8 demultiplexer in detail

ENCODER :

It converts an active input signal into a coded output signal.



Out of n input lines only one of input line is active. Internal logic with the encoder converts this active input to a coded binary output signal.

An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2^n input lines and n output lines. The output lines generate the binary code corresponding to the input value. 8-to-3 encoder has eight inputs, one for each of the octal digits, and three outputs that generate the corresponding binary number. It is assumed that only one input has a value of 1 at any given time; otherwise the circuit has no meaning.

The encoder can be implemented with OR gates whose inputs are determined directly from the truth table. Output z is equal to 1 when the input octal digit is 1 or 3 or 5 or 7. Output y 1 for octal digits 2,3,6 or 7, and output x is 1 for digits 4,5,6 or 7. These conditions can be expressed by the following output Boolean functions:

$$z = D_1 + D_3 + D_5 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$x = D_4 + D_5 + D_6 + D_7$$

Truth Table of Octal-to-Binary Encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

The encoder is implemented with three OR gates. The encoder defined has limitation that only one input can be active at any given time. If two inputs are active simultaneously, the output produces an undefined combination.

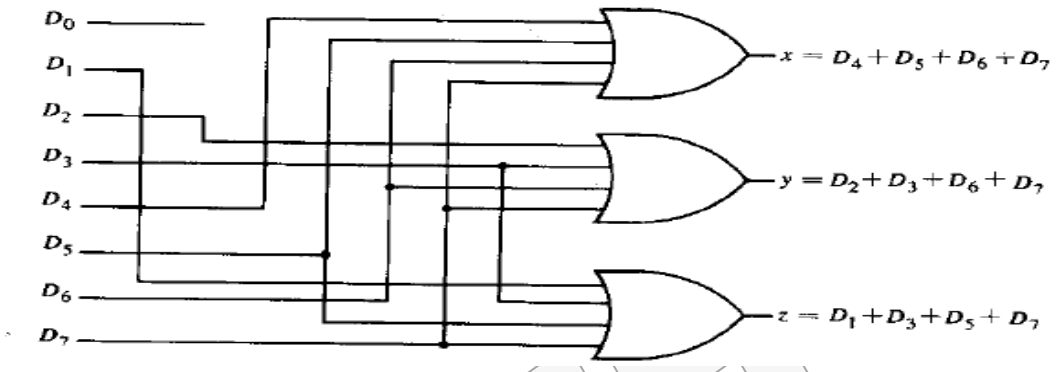


Figure – 8 to 3 encoder (octal to binary encoder)

DECODER

A decoder is a digital circuit that performs the inverse operation of a encoder. A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. The decoders presented here are called n-to-m line decoders, where $m \leq 2n$. their purpose is to generate the 2^n minterms of n input variables.

In 3-to-8 line decoder circuit, three inputs are decoded into eight outputs, each output representing one of the minterms of the 3-input variables. The three inverters provide the complement of the inputs, and each of this decoder would be a binary-to-octal conversion. The input variables may represent a binary number, and the outputs will the represent the eight digits in the octal number system. However, a 3-to-8 line decoder can be used for decoding any 3-bit code to provide eight output , one for each element of the code.

Truth Table of a 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

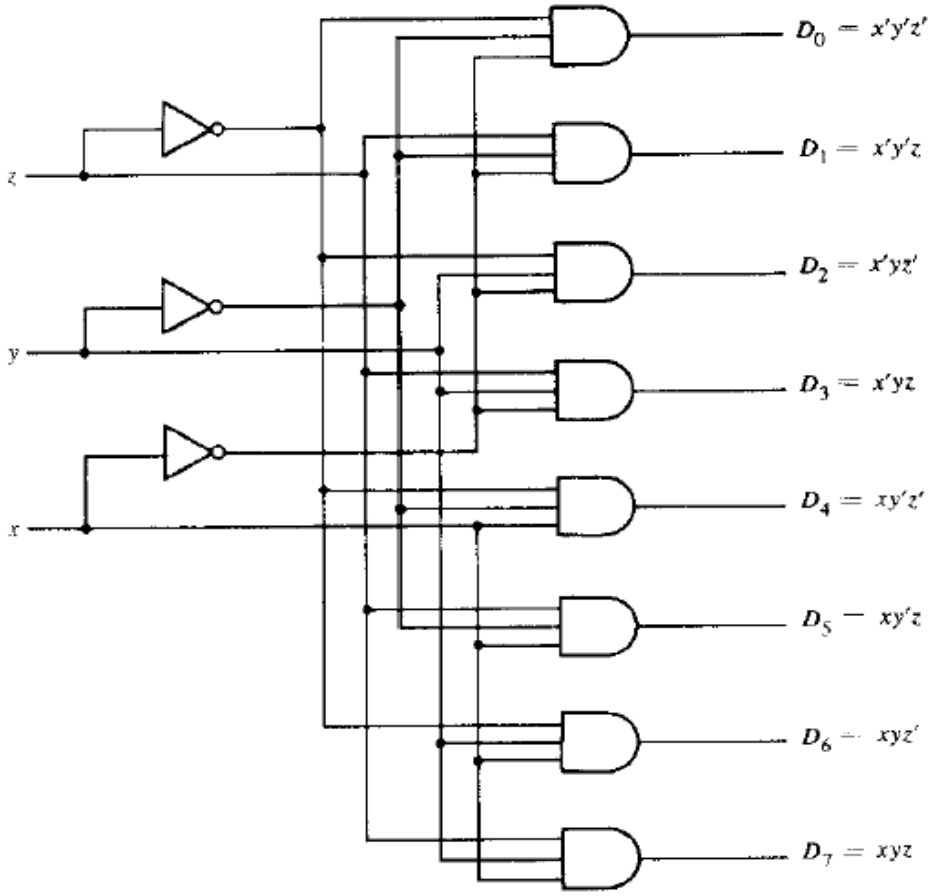


Figure – 3 to 8 line decoder

Comparator

The circuit is a word comparator; it recognizes two identical words. The first XOR gate compares A0 and B0; if they are the same, output of XOR is a 0. The second XOR gate compares A1 and B1; if they are the same output is a 0. In turn, the remaining XOR gates compare the bits that are left, producing a 0 output for equal bits and 1 output for unequal bits.

If the words A and B are identical, all XOR gates have law outputs and the NOR gate has a high EQUAL. If word A and B differ in one or more bit positions, the NOR gate has a low EQUAL.

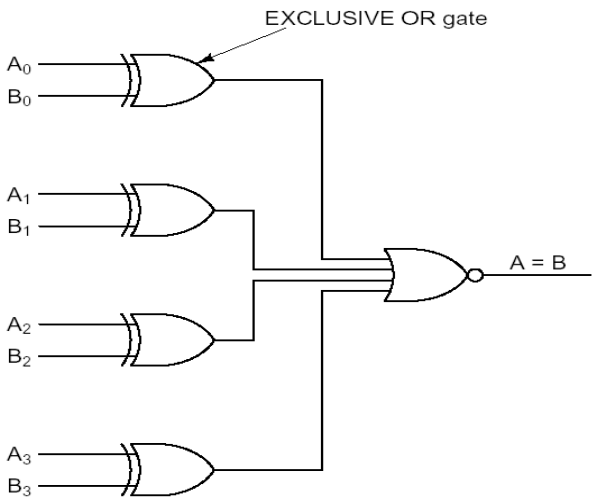


Fig- A 4-bit Comparator

EQUATIONS TO REMEMBER

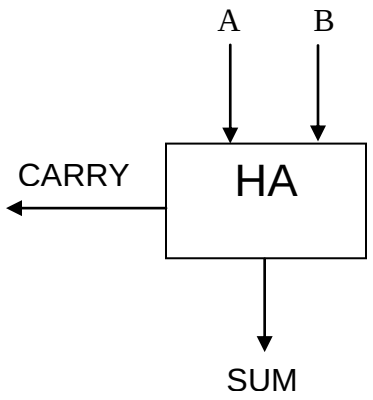
1	A + B = B + A	Commutative
2	AB = BA	
3	A + (B + C) = (A + B) + C	Associative Law
4	A (BC) = (AB) C	
5	A (B + C) = AB + AC	Distributive Law
6	A + 0 = A	Boolean Relation about OR Operation
7	A + A = A	
8	A + 1 = 1	
9	A + A = 1	
10	A . 0 = 0	Boolean Relation about AND Operation
11	A . A = A	
12	A . 1 = A	
13	A + A = 0	
14	A = A	Double Inversion
15	A+B = A B	De' Morgans Theorem
16	A B = A + B	
17	A + AB = A	
18	A + AB = A + B	
19	A (A + B) = A	
20	A (A + B) = AB	

HALF ADDER :

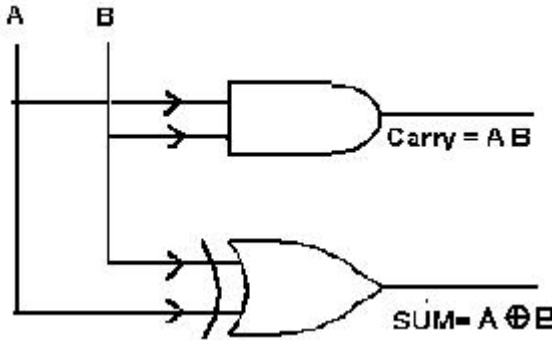
It is a logic circuit that adds 2-bits and generates the output as **SUM** and **CARRY.**
Two gates used to construct half adder are AND and XOR gate.
XOR gate produce the SUM and AND gate produce the CARRY.

Thus output of Half Adder are : SUM = A B
CARRY = A B
Therefore SUM is 1 when high inputs(A and B) are odd.
CARRY is 1 when both inputs(A and B) are high .

Abbreviated symbol of Half Adder :



↳**Logic circuit :**



↳**Truth Table :**

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

From truth table we can see that **CARRY** is generated by AND gate and **SUM** is generated by XOR gate. i.e. CARRY = A B , SUM =

↳ **Explanation for Truth Table :**

For 2 inputs XOR gate output is 1 when high inputs are odd and in any other cases output is 0.
In AND gate output is high when all inputs are high.

Case 1: If A=0, B=0 then sum=0 and carry =0 , according to XOR and AND gate condition.

Case 2: If A=0 , B=1 then sum=1 and carry=0 according to XOR and AND gate condition.

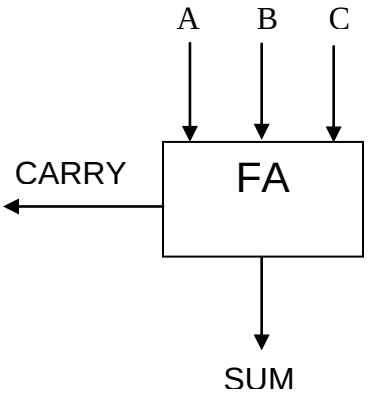
Case 3: If A=1, B=0 then sum=1 and carry=0 according to XOR and AND gate condition.

Case 4: If A=1 , B=1 then sum=0 and carry=1 according to XOR and AND gate condition.

❖ FULL ADDER :

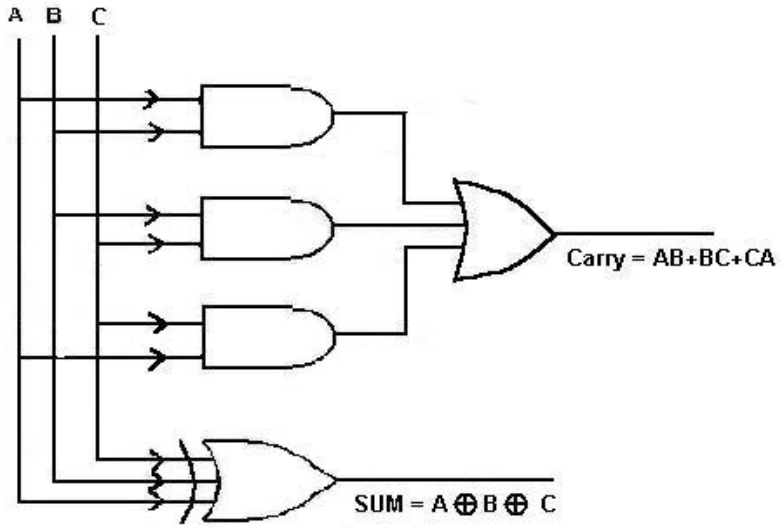
It is a logic circuit that adds 3-bits and generates the output as **SUM** and **CARRY.**
Three gates used to construct full adder are AND, OR and XOR gate.
The output of Full Adder are : SUM = A B C
CARRY = A B + B C + C A
Therefore SUM is 1 when the number of high inputs are odd.
CARRY is 1 when two or more inputs are 1's.

↳ Abbreviated symbol for Full Adder :



Logic circuit :

A	B	C	SUM	CAR
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



From truth table we can see that **SUM** is generated by XOR gate.
i.e. SUM =A B C

CARRY is generated by OR gate, which combines two inputted three AND gate.
i.e. CARRY =A B + B C + CA

↳ Explanation for Truth Table :

Case 1: If A=0, B=0, C=0 then sum=0 and carry =0

Case 2: If A=0, B=0,C=1 then sum=1 and carry=0

Case 3: If A=0, B=1, C=0 then sum=1 and carry =0

.

.
.

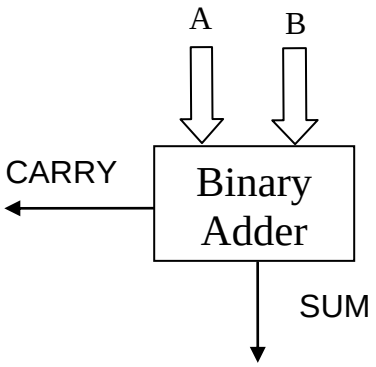
Case 8: If A=1, B=1,C=1 then sum=0 and carry=1

Similarly other cases can be explained.

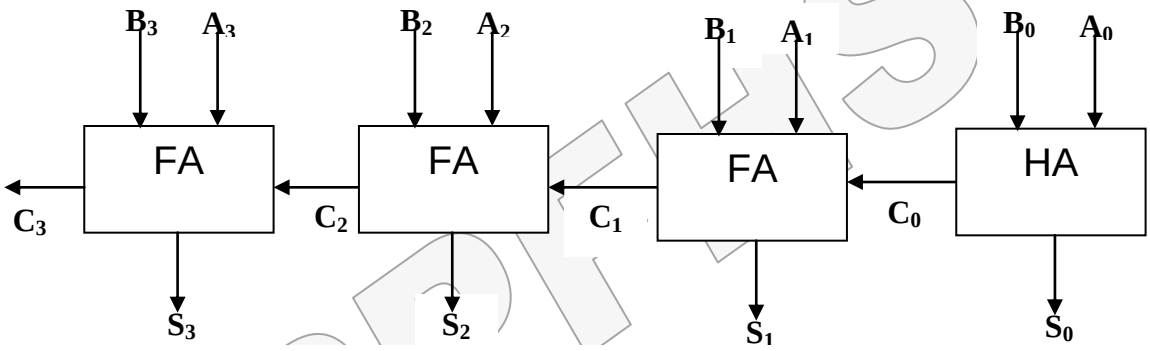
❖ BINARY ADDER :

It is a logic circuit that can adds two binary numbers.

Abbreviated symbol of Binary Adder :



↳**Logic circuit :**



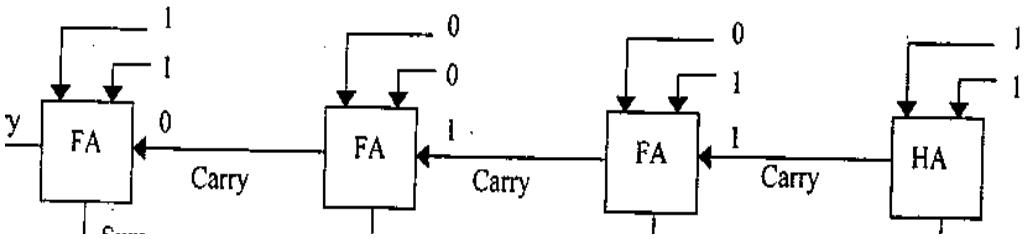
The block on the right labeled HA represents a Half Adder. Their inputs are A_0 and B_0. The outputs are S_0(sum) and C_0(carry). All the others are Full Adder (FA). Each of these has three inputs An, Bn, Cn-1 and two outputs are Cn and Sn.

This circuit performs the following addition :

$$\begin{array}{llllll} & C_2 & C_1 & C_0 & & \\ & A_3 & A_2 & A_1 & A_0 & \\ + & B_3 & B_2 & B_1 & B_0 & \\ \hline C_3 & S_3 & S_2 & S_1 & S_0 & \end{array}$$

↳**Let us see how Binary Adder will add two binary numbers :**

Let two binary numbers be $A_3A_2A_1A_0 = 1001$ and $B_3B_2B_1B_0 = 1011$ which are binary equivalent of decimal number 9 and 11 respectively.



As shown in above figure, the Half Adder adds 1+1 to give a sum of 0 and a carry of 1. The carry goes into the first Full Adder, which adds 0+1+1 to get a sum of 0 and a carry of 1. This carry goes into the next Full Adder which adds 0+0+1 to get a sum of 1 and a carry of 0. The last Full Adder adds 1+1+0 to get a sum of 0 and a carry of 1.

The final output of the system is 10100(decimal – 20) which is the correct decimal sum of 9 and 11.

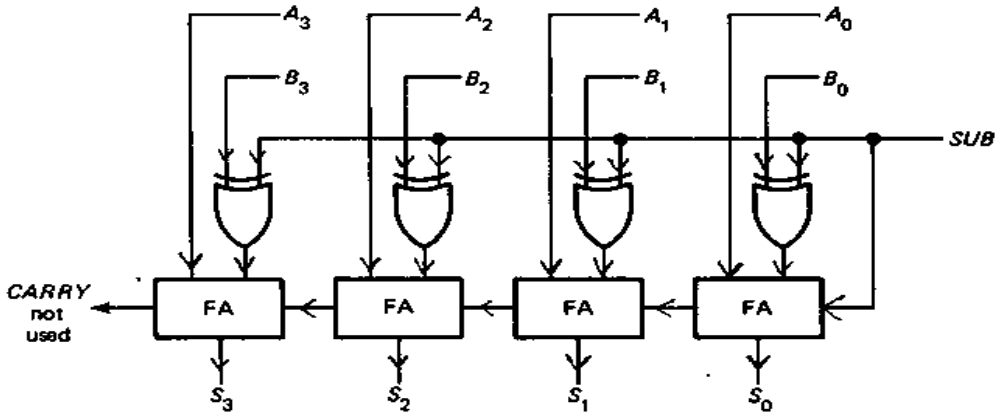
The binary adder of figure has limited capacity. The largest binary numbers that can be added are 1111 and 1111. Hence its maximum capacity is

```
  15        1 1 1 1
+ 15        1 1 1 1
  30      1 1 1 1 0
```

In order to increase the capacity more full adders can be connected to the left end of the system. For e.g. to add six bit numbers, two more full adders must be connected to the left end of full adder.

2'S Complement Adder/Subtractor :

It is a logic circuit that can add or subtract 4 bits binary numbers.



2'S Complement Adder/Subtractor

When the **SUB** is **low** the word **B = $B_3B_2B_1B_0$** passes through the controlled inverter without inversion(any change). As a result full adder produce the **SUM=A+B**.

When the **SUB** is **high**, the controlled inverter produce the one's complement of word B. Also high SUB adds 1 to the first full adder. This addition of 1 to the 1's complement B forms **two's complement** of word B.
As a result full adder produce the **SUM = A + B i.e. SUM=A-B**

e.g. Let $A_3A_2A_1A_0$ =0100 which is binary of +4
$B_3B_2B_1B_0$ =0010 which is binary of +2

Let SUB=0 then all controlled inverter pass the B bits as it is.
Hence we have $C_3C_2C_1C_0$ =0000 and $S_3S_2S_1S_0$=0110 and sum=0110 which is nothing but a binary of +6. Here it gives addition of two numbers A and B.

Let SUB=1 then controlled inverter produce 1's complement of B. As SUB is high it gives 2's complement of B. Hence we have $C_3C_2C_1C_0$ =1101 and $S_3S_2S_1S_0$=0010 which is binary of +2. ie. Subtraction of two numbers A and B.

FLIP FLOP:

The outputs of the digital circuits are dependent entirely on the inputs. i.e. If the input changes , the output is also changed.
However there are requirements for a digital circuits, whose output will remain unchanged, once set, even if there is a change in the input. A Flip flop is one such circuit. **Such a device could be used to store a binary number**.

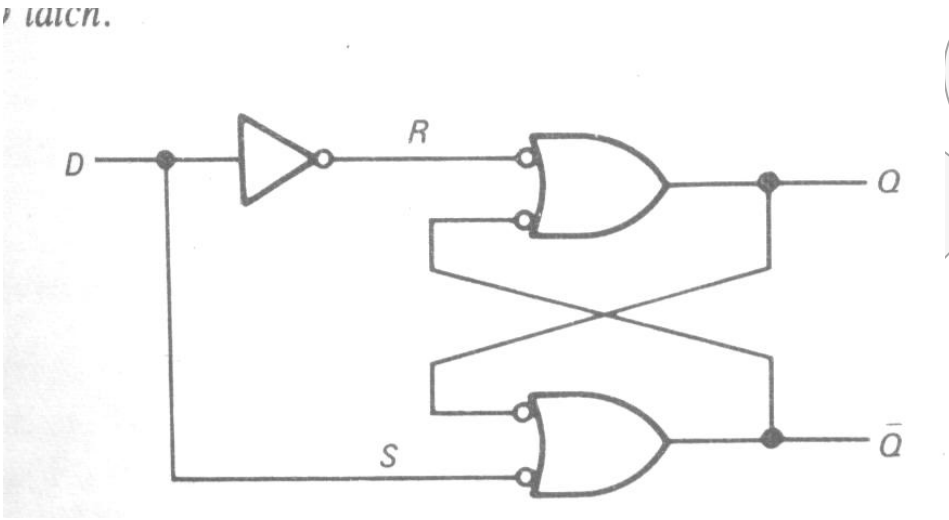
A flip flop is a device (circuit) with two stable states , it remains in one of these states until triggered into the other.

A flip flop can be constructed with two NOR or two NAND gates.

D Flip flop:

To eliminate the possibility of a race condition , D flip flop is used.

- **Unclocked D Flip flop :**



In D latch , because of inverter R and S will always be in opposite state.
i.e. If D is 1 then R is 0 and if D is 0 then R is 1

Therefore there is no possibility of race condition in D latch.

↳ **Truth Table** :

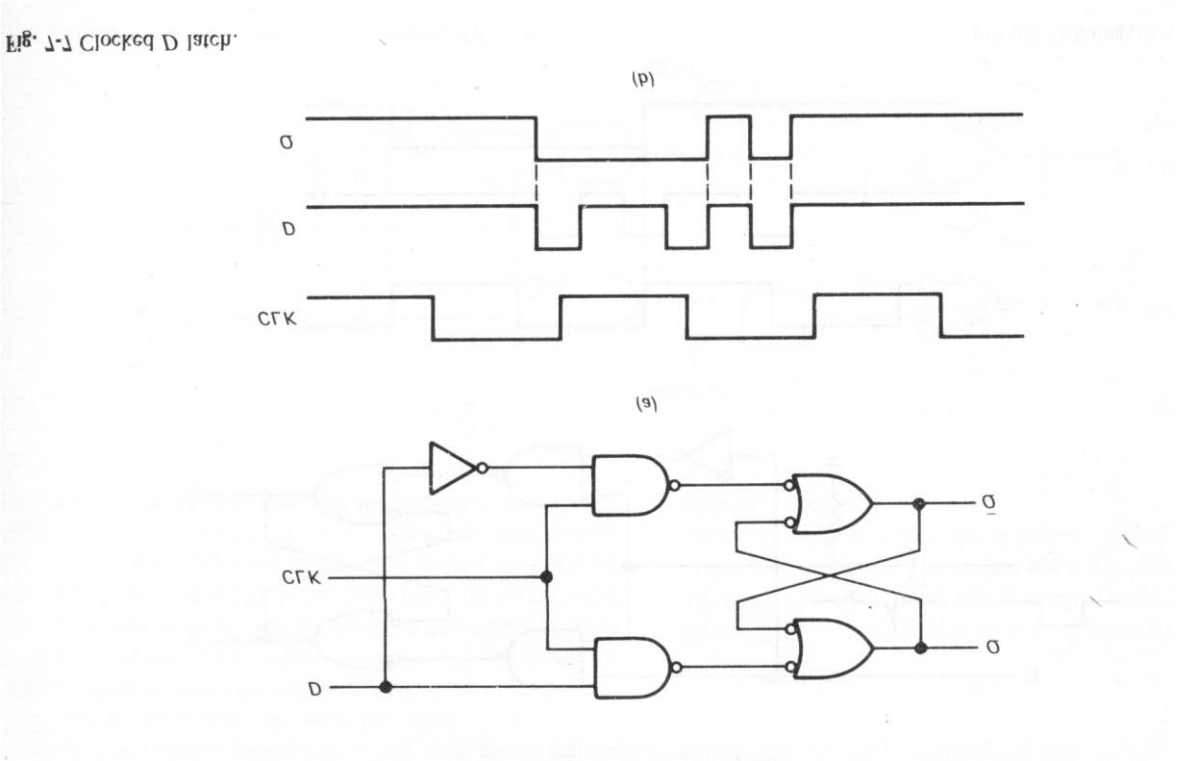
D	Q
0	0
1	1

↳ **Explanation:**

A high D sets the output Q as high

A low D sets the output Q as low

◆ **Clocked D Flip flop :**



When **CLK is low**, it does not matter what the value of D is. The output cannot change.
i.e. When CLK is low, it does not matter what the value of D is. The output cannot change.
When **CLK is high**, D controls the output.
i.e. When CLK is high then
Q is high, if D is high
Q is low, if D is low
i.e. If CLK is high , the output Q = D

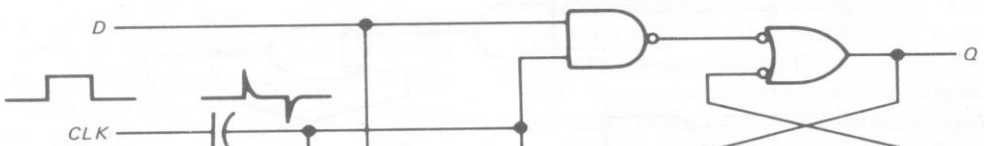
↳**Truth Table :**

CLK	D	Q
0	X	NC
1	0	0
1	1	1

↳ **Timing Diagram** :

If CLK is low, output Q can not changed.
If CLK is high, Q equals D
i.e. When D goes high, Q goes high
When D goes low, Q goes low

◆ **Edge Triggered D Flip flop :**



Edge triggered D flip flop is most common type of D flip flop.

The RC time constant is much smaller than the clock 's pulse width. Because of this , the capacitor can charge fully when CLK goes high, this exponential charging produce a positive spike across the resistor. The trailing edge of the clock pulse results in a negative spike. **The positive spike enable the input gates, the negative spike does nothing.**

This kind of operation is called edge triggered because the flip flop response only when the CLK is changing the states.

↳**Truth Table** :

CLK	D	Q
0	X	NC
1	X	NC
↓	X	NC
↑	0	0
↑	1	1

The up and down arrow represent the positive and negative edges of the clock.
The first three entries indicates that there is no change in output Q when CLK is low, high or on its negative edge.

The last two entries indicate that an output change on the positive edge of the clock.

Thus input data D is stored only on the positive edge of the clock.

↳ **Timing Diagram** :

The output changes only on the positive edge of the clock.i.e. data is stored only on positive edge.

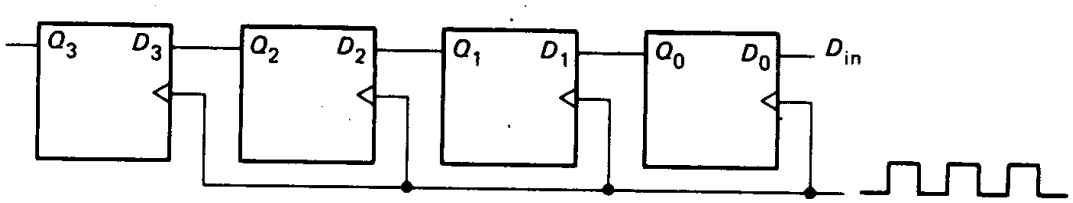
REGISTERS:

- Register is a group of flip flops that can be used to store binary numbers.
- It can also be defined as a group of memory elements that work together as a unit.

⇨ Shift Register :

A shift registers moves the stored bit left or right.
This bit shifting is generally used for arithmetic and logic operations.

◆ Shift Left Register :



Above figure is of shift left register.

The clock input is given to each flip-flop separately.

D_{in} sets up the right most flip-flop .Q_0 sets up the second flip flop. Q_1 sets up the third flip flop. Q_2 sets up the fourth flip flop and so on.

i.e.
When first positive clock edge arrives, D_{in} sets up the right most flip flop.
At the second positive clock edge, Q_0 sets up the second flip flop.
At the third positive clock edge, Q_1 sets up the third flip flop.
At the fourth positive clock edge, Q_2 sets up the fourth flip flop.

For e.g.

- Suppose $\mathbf{D_{in}=1}$ and **Q = 0000**

  when first positive clock edge arrives , D_{in} shifts to first flip flop
  So **Q=0001**

  when second positive clock edge arrives , Q_0 shifts to second flip flop
  So **Q=0011**

  At third positive clock edge
  Q=0111

  At fourth positive clock edge
  Q=1111

This output will remain as it is till $D_{in} = 1$

- Now Suppose **D_{in} =0**

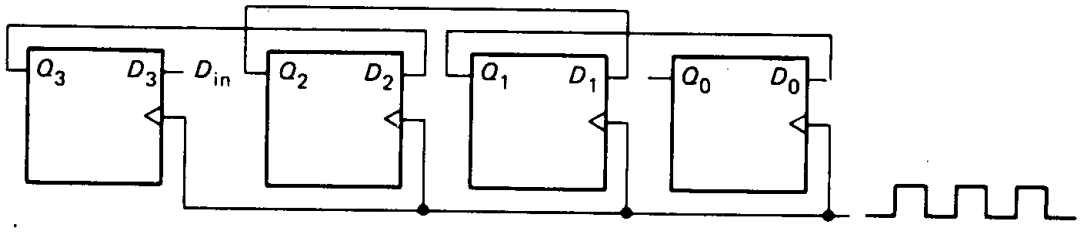
when first positive clock edge arrives , D_{in} shifts to first flip flop
So **Q=1110**

when second positive clock edge arrives , Q_0 shifts to second flip flop
So **Q=1100**

At third positive clock edge
Q=1000

At the fourth positive clock edge
Q=0000

- **Shift Right Register :**



Above figure is of Shift Right register .

The clock input is given to each flip-flop separately.

D_{in} sets up the left most flip-flop .Q_3 sets up the second flip flop. Q_2 sets up the third flip flop. Q_1 sets up the fourth flip flop and so on.

i.e.
When first positive clock edge arrives, D_{in} sets up the left most flip flop.
At the second positive clock edge , Q_3 sets up the second flip flop.
At the third positive clock edge, Q_2 sets up the third flip flop.
At the fourth positive clock edge, Q_1 sets up the fourth flip flop.

For e.g.

- Suppose **D_{in} =1** and **Q = 0000**

when first positive clock edge arrives , D_{in} shifts to left most flip flop
So **Q=1000**

when second positive clock edge arrives , Q_3 shifts to second flip flop
So **Q=1100**

At third positive clock edge
Q=1110

At fourth positive clock edge
Q=1111

This output will remain as it is till $D_{in} = 1$

- Now Suppose **D_{in} =0**

when first positive clock edge arrives , D_{in} shifts to first flip flop.
So **Q=0111**

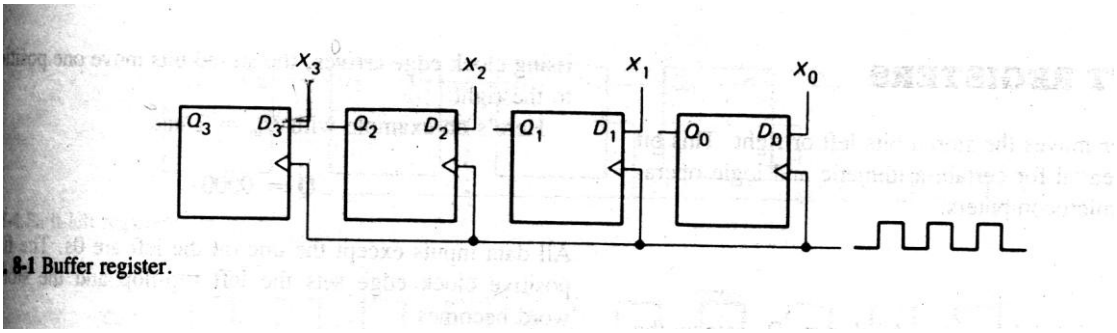
when second positive clock edge arrives , Q_3 shifts to second flip flop.
So **Q=0011**

At the third positive clock edge
Q=0001

At the fourth positive clock edge
Q=0000

Buffer Register :

Buffer register is use to temporary store binary numbers.



8-1 Buffer register.

In above figure, there are four flip-flops.
X_0, X_1, X_2, X_3 are inputs and Q_3,Q_2,Q_1,Q_0 are output. Clock input is separately given to each flip- flop. A simple arrow at the clock input of every flip-flop indicates that the flip-flop will trigger(activate) on the positive edge of the clock.

When positive clock edge arrives the output **$Q_3Q_2Q_1Q_0 = X_3X_2X_1X_0$**
i.e. output **Q=X**

COUNTER:

Counter is a special kind of register used to count the number of clock pulses.
There are three types of counters:

(1) Ripple or Asynchronous or Serial counter
(2) Parallel or synchronous counter
(3) Ring counter

Ring counter:

Ring counter have only a single high bit.

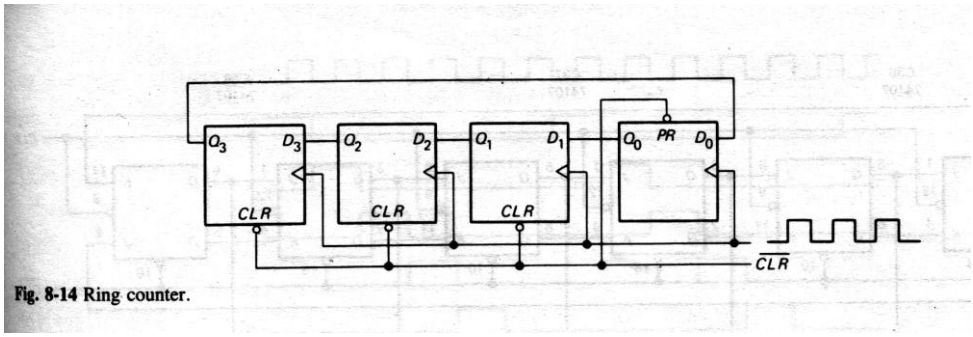


Fig. 8-14 Ring counter.

Ring counter is constructed using D flip-flop. The Q_0 output sets up the D_1 input , the Q_1 output sets up the D_2 input , Q_2 output sets up the D_3 input , and Q_2 output sets up the D_3 input. Therefore a ring counter act as a shift-left register but the final output is again fed back to the D_0 input . This kind of action is known as **rotate left** because bits are shifted left and fed back to the input.

For e.g..

Suppose initial output Q=0001

On first positive clock edge output Q=0010

On second positive clock edge Q=0100

On third positive clock edge final output Q=1000

Application of Ring counters:

Ring counter is normally used to control a sequence of operations. Because each ring word has only one high bit, so you can activate one of the several devices.

During a computer run , many digital circuits participates. Computer has to activate these circuits at the right time and in the right sequence. So in such situation ring counter is most widely used.