Solution to digital logic 2067

1.)What is the magnitude comparator? Design a logic circuit for 4 bit magnitude comparator and explain it,

A Magnitude comparator is a combinational circuit that compares two numbers, A and B, and determines their relative magnitudes. The outcome of the comparison is specified by three binary variables that indicate whether A > B, A = B, or A < B.

Consider two numbers, A and B, with four digits each. Write the coefficients of the numbers with descending significance as follows:

$A = A_3A_2A_1A_0$

$B = B_3B_2B_1B_0$

Where each subscripted letter represents one of the digits in the number, the two numbers are equal if all pairs of significant digits are equal, i.e., if $A_3=B_3$ and $A_2=B_2$and $A_1= B_1$ and $A_0=B_0$.

When the numbers are binary, the digits are either 1 or 0 and the equality relation of each pair of bits can be expressed logically with an equivalence function:

$Xi = A_iB_i + A_i'B_i'$, i = 0, 1, 2, 3

Where Xi = 1 only if the pair of bits in position i are equal, i.e., if both are 1's or both are 0's.

Algorithm

(A=B)

For the equality condition to exist, all X_i variable must be equal to 1. This dictates and AND operation of allvariable.

(A=B)=$X_3X_2X_1X_0$

The binary variable **(A=B)** is equal to 1 only if all pairs of digits of the two numbers are equal.

(A<B) or (A>B)

To determine if A is greater than or less than B, we check the relative magnitudes of pairs of significant digits starting from the most significant position. If the two digits are equal, we compare the next lower significant pair of digits. This comparision continues until a pair of uneual digits is reached.

A > B: If the corresponding digit of A is 1 and that of B is 0.

A < B: If the corresponding digit of A is 0 and that of B is 1.

The sequential comparison can be expressed logically by the following two Boolean functions:

(A>B)=$A_3B_3' + X_3X_2A_1B_1' + X_3X_2X_1A_0B_0'$

$A < B = A_3'B_3 + X_3X_2A_1'B_1 + X_3X_2X_1A_0'B_0$

The symbols (A > B) and (A < B) are binary output variables that are equal to 1 when A > B or A < B respectively.

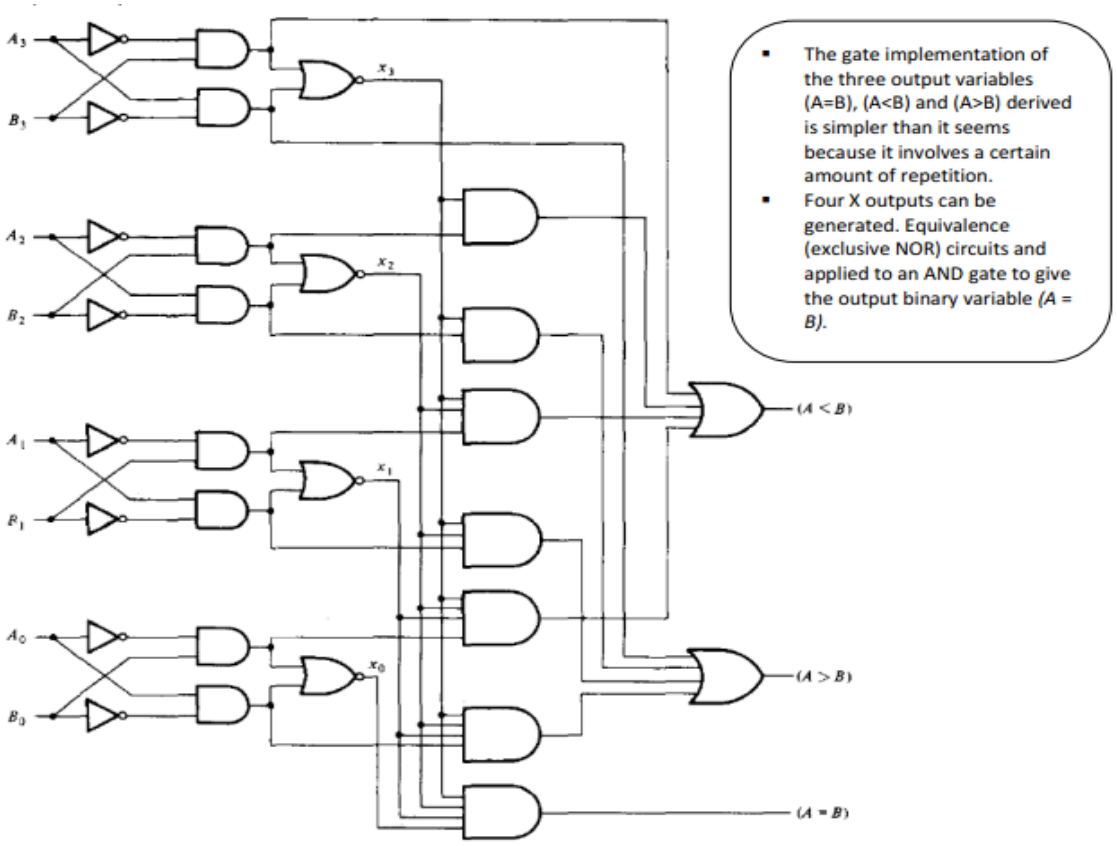


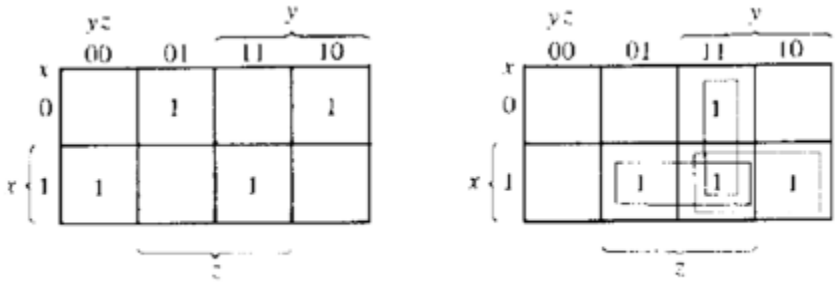
Fig: 4-bit magnitude comparator

2.)What do you mean by full adder and full subtractor? Design a 3 to 8 line decoder using two 2 to 4 line decoder and explain it.

A full adder is a combination circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables, denoted by x and y, represent the two significant bits to be added. The third input z, represents the carry from the previous lower significant position.

Truth table formulation				
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The S output is equal to 1 when only one input is equal to 1 or when all three inputs are equal to 1. The C output has a carry of 1 if two or three inputs are equal to 1.



S = x'y'z + x'yz' + xy'z' + xyz

C = xy + xz + yz

- **Implementation:**

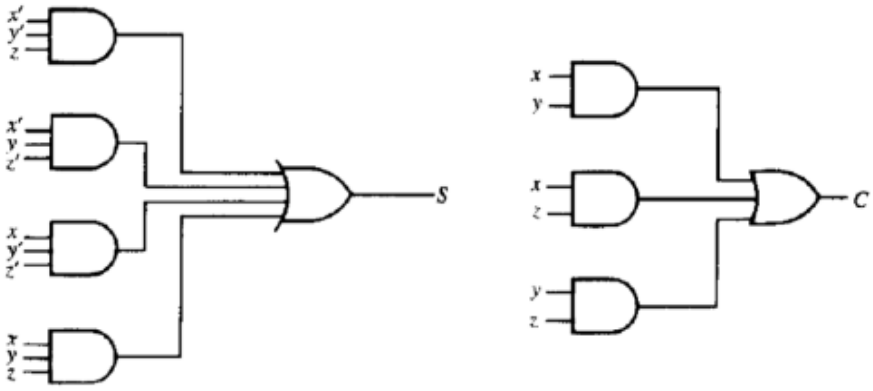


Fig: Implementation of a full-adder in sum of products

Full substractor

A full-subtractor is a combinational circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage. This circuit has three inputs and

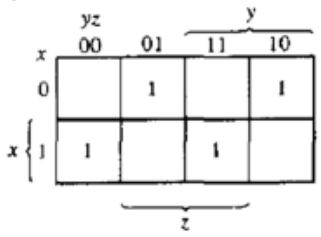
two outputs. The three inputs, x, y, and z, denote the minuend, subtrahend, and previous borrow, respectively. The two outputs, D and B, represent the

difference and output-borrow, respectively.

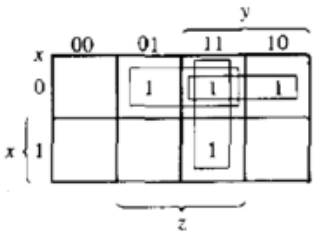
Truth table and output function formulation				
x	**y**	**z**	**B**	**D**
0	**0**	**0**	**0**	**0**
0	**0**	**1**	**1**	**1**
0	**1**	**0**	**1**	**1**
0	**1**	**1**	**1**	**0**
1	**0**	**0**	**0**	**1**
1	**0**	**1**	**0**	**0**
1	**1**	**0**	**0**	**0**
1	**1**	**1**	**1**	**1**

The 1's and 0's for the output variables are determined from the subtraction of x - y - z. The combinations having input borrow z = 0 reduce to the same four conditions of the half-adder. For x = 0, y = 0, and z = 1, we have to borrow a 1 from the next stage, which makes B = 1 and adds 2 to x. Since 2 - 0 - 1 = 1, D = 1. For x = 0 and yz = 11, we need to borrow again, making B = 1 and x = 2. Since 2 - 1 - 1 = 0, D = 0. For x = 1 and yz = 01, we have x - y - z = 0, which makes B = 0 and D = 0.Finally, for x = 1, y = 1, z =1I, we have to borrow 1, making B = 1 and x = 3, and 3 - 1 - 1 = 1, making D=1.

The simplified Boolean functions for the two outputs of the full-subtractor are derived in the maps:



D = x'y'z + x'yz' + xy'z' + xyz



B = x'y + x'z + yz

- Circuit implementations are same as Full-adder except B output (analogous to C) is little different. (Don't worry! ladies and gentlemen, we will discuss it in class…)

3.)What is JK master slave flip-flop? Design its logic circuit, truth table and explain the working principle.

Master-slave JK flip-flop constructed with NAND gates is shown in Fig. below. It consists of two flip-flops; gates 1 through 4 form the master flip-flop, and gates 5 through 8 form the slave flip-flop.

The information present at the J and K inputs is transmitted to the master flip-flop on the positive

edge of a clock pulse and is held there until the negative edge of the clock pulse occurs, after which

it is allowed to pass through to the slave flip-flop.

Operation:

o The clock input is normally 0, which prevents the J and K inputs from affecting the master flip-flop.

o The slave flip-flop is a clocked RS type, with the master flip-flop supplying the inputs and the clock input being inverted by gate 9.

o When the clock is 0, Q = Y, and Q' = Y'.

o When the positive edge of a clock pulse occurs, the master flip-flop is affected and may switch states.

o The slave flip-flop is isolated as long as the clock is at the 1 level

o When the clock input returns to 0, the master flip-flop is isolated from the J and K inputs and the slave flip-flop goes to the same state as the master flip-flop.

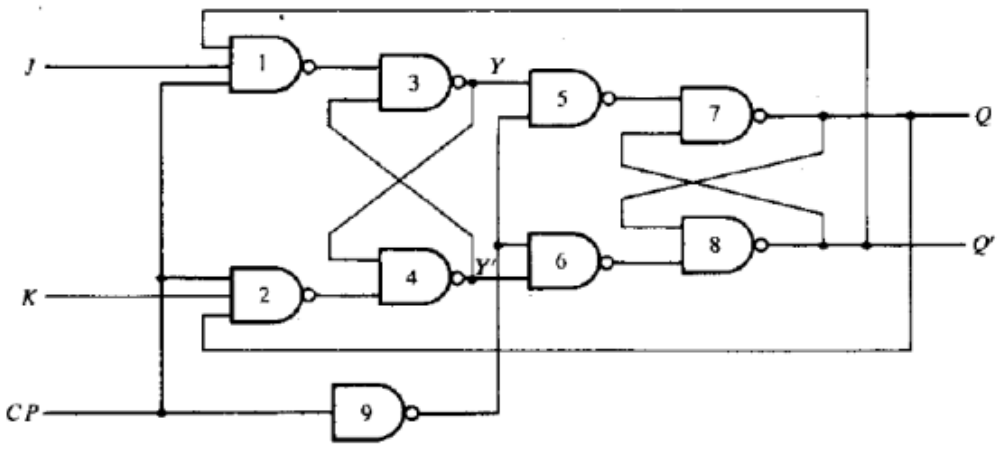


Fig: Clocked master-slave JK flip-flop

4.)Short answer question

Convert the following hexadecimal number to decimal and octal numbers

a) 0FFF **b) 3FFF**

Converting 0FFF to binary
i.e 0000111111111111
it's decimal is $1 \times 2^{11} + 1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$=4095
Now for octal we have

7777

decimal of 3FFF is 16383

Octal of 3FFF is 37777

5.)Design a half adder logic circuit using NOR gates only.

6.)Proof the 1st and 2nd law of De Morgan's theorem.

Theorem1: Idempotence (a) x + x = x (b) x.x = x

Theorem2: Existence: 0&1 (a) x + 1 = 1 (b) x.0 = 0

Proofs:

(a) The proofs of the theorems with one variable are presented below:

THEOREM 1(a): x + x = x

x + x = (x + x) . 1 (P4: Identity element)

= (x + x)(x + x') (P5: Existence of inverse)

= x + xx' (P3: Distribution)

=x + 0 (P5: Existence of inverse)

=x (P4: Identity element)

THEOREM 1(b): x·x = x

x.x = xx + 0 (P4: Identity element)

=xx + xx' (P5: Existence of inverse)

=x(x + x') (P3: Distribution)

= x.1 (P5: Existence of inverse)

=x (P4: Identity element)

Hey! Each step in theorem 1(b) and 1(a) are dual of each other.

THEOREM 2(a): x + 1 = 1

x + 1 = 1· (x + 1) (P4: Identity element)

= (x + x')(x + 1) (P5: Existence of inverse)

=x + x'·1 (P3: Distribution)

= x + x' (P4: Identity element)

= 1 (P5: Existence of inverse)

THEOREM 2(b): x.0 = 0 by duality.

Draw logic gate and truth table urself.

7.)What do you mean by Universal gate? Realize the following logic gates using NOR gates.

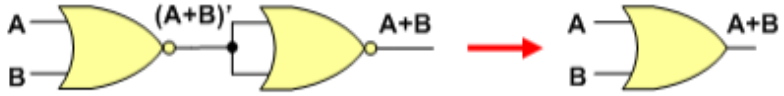
a) OR gate

b) AND gate

A universal gate is a gate which can implement any Boolean function without need to use any other gate type. The NAND and NOR gates are universal gates. In practice, this is advantageous since NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all IC digital logic families.

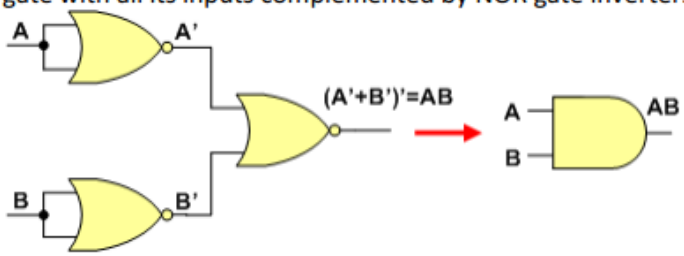
OR gate

- **Implementing OR Using only NOR Gates:** The OR is replaced by a NOR gate with its output complemented by a NOR gate inverter.



AND gate

- **Implementing AND Using only NOR Gates:** The AND gate is replaced by a NOR gate with all its inputs complemented by NOR gate inverters.



Thus, the **NOR gate** is a universal gate since it can implement the AND, OR and NOT functions.

8.)Draw a logic circuit of 4×1 multiplexer.

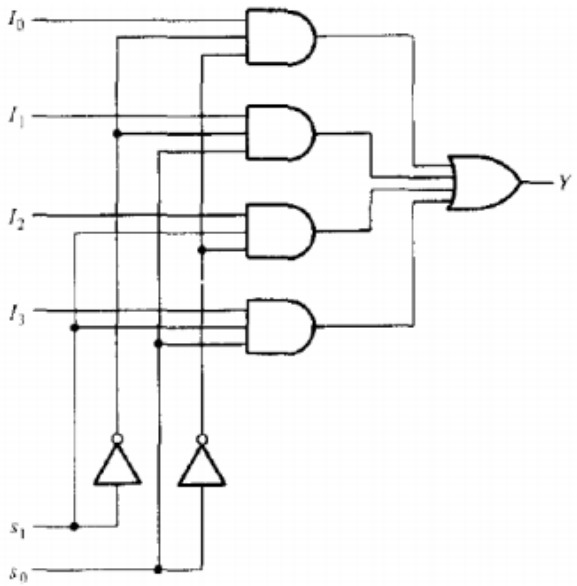


Fig: Logic Diagram: 4-to-1 line Multiplexer

s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Table: Function table

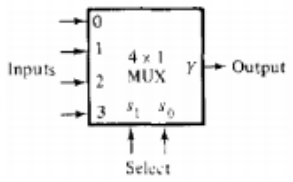


Fig: Block Diagram of Multiplexer

9.)What is flip-flop? Mention the application of flip-flop.

The memory elements used in clocked sequential circuits are called flip-flops. These circuits are binary

cells capable of storing one bit of information. A flip-flop circuit has two outputs, one for the normal

value and one for the complement value of the bit stored in it. Binary information can enter a flip-flop in a variety of ways, a fact that gives rise to different types of flip-flops.

- A flip-flop circuit can maintain a binary state indefinitely (as long as power is delivered to the circuit) until directed by an input signal to switch states.
- The major differences among various types of flip-flops are in the number of inputs they possess and in the manner in which the inputs affect the binary state.

10.)Explain the ripple counter.

In a ripple counter (Asynchronous Counter); flip-flop output transition serves as a source for triggering other flip-flops. In other words, the CP inputs of all flip-flops (except the first) are triggered not by the incoming pulses, but rather by the transition that occurs in other flip-flops.

Binary ripple counter

A binary ripple counter consists of a series connection of complementing flip-flops (T or JK type), with the output of each flip-flop connected to the CP input of the next higher-order flip-flop. The flip-flop holding the least significant bit receives the incoming count pulses. The diagram of a 4-bit binary ripple counter is shown in Fig. below. All J and K inputs are equal to 1.

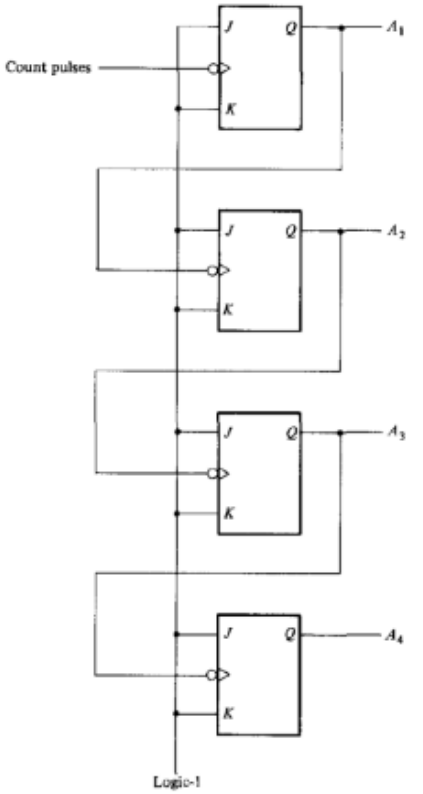


Fig: 4-bit binary ripple counter

All *J* and *K* inputs are equal to 1. The small circle in the *CP* input indicates that the flip-flop complements during a negative-going transition or when the output to which it is connected goes from 1 to 0.

To understand the operation of the binary counter, refer to its count sequence given in Table.

- It is obvious that the lowest-order bit A_1 must be complemented with each count pulse. Every time A_1 goes from 1 to 0, it complements A_2. Every time A_2 goes from 1 to 0, it complements A_3, and so on.
- For example: take the transition from count 0111 to 1000. The arrows in the table emphasize the transitions in this case. A_1 is complemented with the count pulse. Since A_1 goes from 1 to 0, it triggers A_2 and complements it. As a result, A_2 goes from 1 to 0, which in turn complements A_3. A_3 now goes from 1 to 0, which complements A_4. The output transition of A_4, if connected to a next stage, will not trigger the next flip-flop since it goes from 0 to 1. The flip-flops change one at a time in rapid succession, and the signal propagates through the counter in a *ripple* fashion.

Count Sequence				Conditions for Complementing Flip-Flops	
A_4	A_3	A_2	A_1		
0	0	0	0	Complement A_1	
0	0	0	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2
0	0	1	0	Complement A_1	
0	0	1	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2; A_2 will go from 1 to 0 and complement A_3
0	1	0	0	Complement A_1	
0	1	0	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2
0	1	1	0	Complement A_1	
0	1	1	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2; A_2 will go from 1 to 0 and complement A_3; A_3 will go from 1 to 0 and complement A_4
1	0	0	0	and so on . . .	

11.)Design the decimal adder.

Computers or calculators that perform arithmetic operations directly in the decimal number system represent decimal numbers in binary-coded form.

 Decimal adder is a combinational circuit that sums up two decimal numbers adopting particular encoding technique.

 A decimal adder requires a minimum of nine inputs and five outputs, since four bits are required to code each decimal digit and the circuit must have an input carry and output carry.

 Of course, there is a wide variety of possible decimal adder circuits, dependent upon the code used to represent the decimal digits.

12.)What do you mean by shift registers? explain.

 A register capable of shifting its binary information either to the right or to the left is called a shift register. The logical configuration of a shift register consists of a chain of flip-flops connected in cascade, with the output of one flip-flop connected to the input of the next flip-flop. All flip-flops receive a common clock pulse that causes the shift from one stage to the next.

 The Shift Register is used for data storage or data movement and are used in calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format. The individual data latches that make up a single shift register are all driven by a common clock (Clk) signal making them synchronous devices. Shift register IC's are generally provided with a clear or reset connection so that they can be "SET" or "RESET" as required.

Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being:

 Serial-in to Parallel-out (SIPO) - the register is loaded with serial data, one bit at a time, with the stored data being available in parallel form.

 Serial-in to Serial-out (SISO) - the data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.

 Parallel-in to Serial-out (PISO) - the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.

 Parallel-in to parallel-out (PIPO) - the parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.

13.)Write short notes on

a) ***Decoder***
b) ***Integrated Circuit***
c) ***PLA***

Decoder

Decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2n unique output lines.

If the n-bit decoded information has unused or don't-care combinations, the decoder output will have fewer than 2n

outputs.

n-to-m-line decoders have m <= 2n

.

Example: **3-to-8 line decoder**

The 3 inputs are decoded into 8 outputs, each output representing one of the minterms of the 3-input variables.

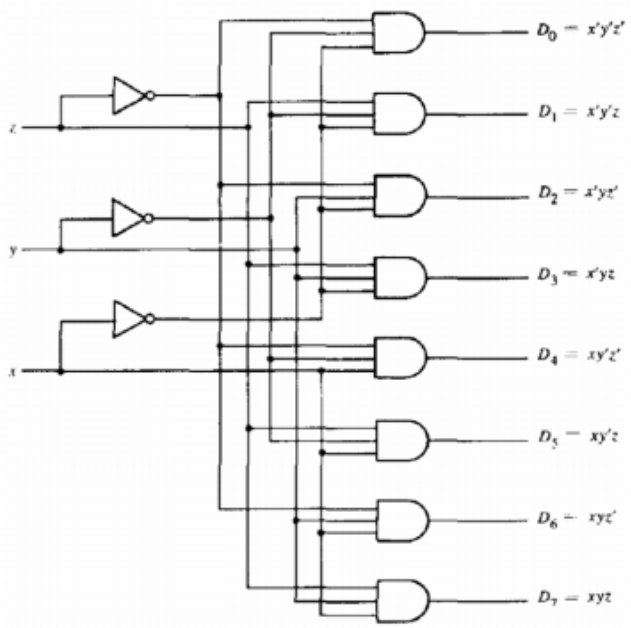


Fig: 3-to-8 line decoder

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

Output variables are mutually exclusive because only one output can be equal to 1 at anyone time. The output line whose value is equal to 1 represents the minterm equivalent of the binary number presently available in the input lines.

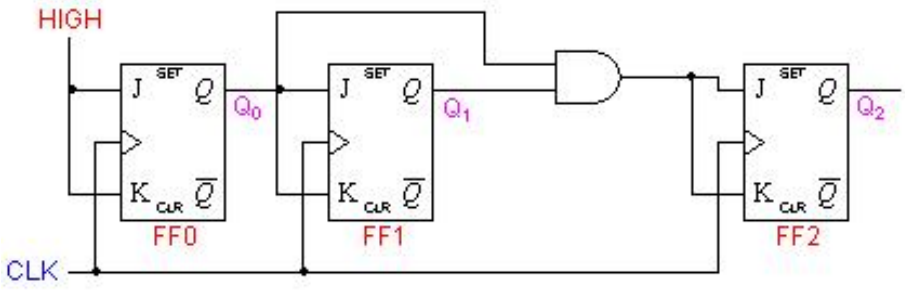
Table: Truth-table for 3-to-8 line Decoder

Solution to digital logic 2068

Q.N.1)Draw a block diagram truth table and logic circuit of 1×16 demultiplexer and explain its working principle?

Q.N.2) Design a 3 bit synchronous counter and explain it.

In *synchronous counters*, the clock inputs of all the flip-flops are connected together and are triggered by the input pulses. Thus, all the flip-flops change state simultaneously (in parallel). The circuit below is a 3-bit synchronous counter. The J and K inputs of FF0 are connected to HIGH. FF1 has its J and K inputs connected to the output of FF0, and the J and K inputs of FF2 are connected to the output of an AND gate that is fed by the outputs of FF0 and FF1.



Please refer to note for explaination and truth table

Q.N.3) What is magnitude comparator. Design a logic circuit for 4 bit comparator and explain it.

A Magnitude comparator is a combinational circuit that compares two numbers, A and B, and determines their relative magnitudes. The outcome of the comparison is specified by three binary variables that indicate whether A > B, A = B, or A < B.

Consider two numbers, A and B, with four digits each. Write the coefficients of the numbers with descending significance as follows:

$A = A_3A_2A_1A_0$

$B = B_3B_2B_1B_0$

Where each subscripted letter represents one of the digits in the number, the two numbers are equal if

all pairs of significant digits are equal, i.e., if $A_3=B_3$ and $A_2=B_2$and $A_1= B_1$ and $A_0=B_0$.

When the numbers are binary, the digits are either 1 or 0 and the equality relation of each pair of bits can be expressed logically with an equivalence function:

$Xi = A_iB_i + A_i'B_i'$, i = 0, 1, 2, 3

Where Xi = 1 only if the pair of bits in position i are equal, i.e., if both are 1's or both are 0's.

Algorithm

(A=B)

For the equality condition to exist, all X_i variable must be equal to 1. This dictates and AND operation of allvariable.

(A=B)=$X_3X_2X_1X_0$

The binary variable **(A=B)** is equal to 1 only if all pairs of digits of the two numbers are equal.

(A<B) or (A>B)

To determine if A is greater than or less than B, we check the relative magnitudes of pairs of significant digits starting from the most significant position. If the two digits are equal, we compare the next lower significant pair of digits. This comparision continues until a pair of uneual digits is reached.

A > B: If the corresponding digit of A is 1 and that of B is 0.

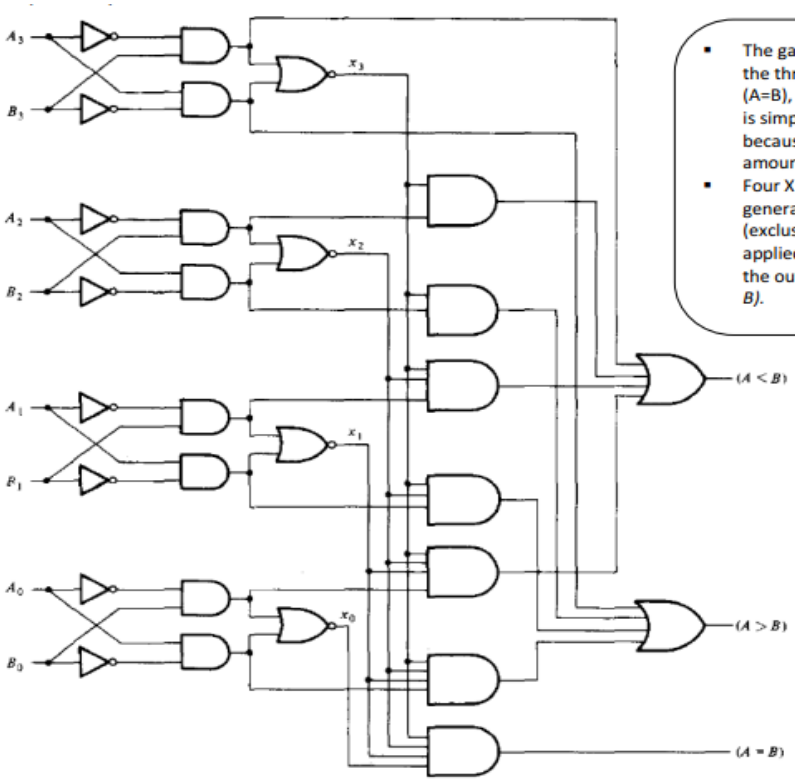
A < B: If the corresponding digit of A is 0 and that of B is 1.

The sequential comparison can be expressed logically by the following two Boolean functions:

(A>B)=$A_3B_3' + X_3X_2A_1B_1' + X_3X_2X_1A_0B_0'$

$A < B = A_3'B_3 + X_3X_2A_1'B_1 + X_3X_2X_1A_0'B_0$

The symbols (A > B) and (A < B) are binary output variables that are equal to 1 when A > B or A < B respectively.

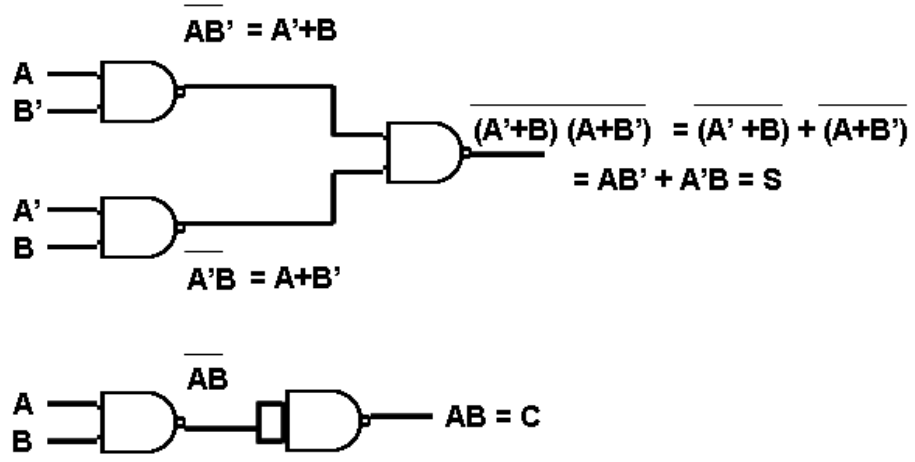


- The gate implementation of the three output variables (A=B), (A<B) and (A>B) derived is simpler than it seems because it involves a certain amount of repetition.
- Four X outputs can be generated. Equivalence (exclusive NOR) circuits and applied to an AND gate to give the output binary variable *(A = B)*.

Fig: 4-bit magnitude comparator

Short answer question

Q.N.4) Design a half subtractor only using NAND gate.



Q.N.5) Convert the following decimal numbers into Hexadecimal and octal number

a) 504

converting to binary we ge

111 111 000

now its octal is

770

b)250

Converting to binary we get

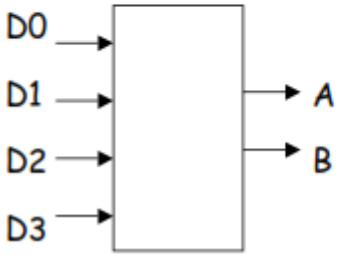
011 111 010

it's octal is

372

Q.N.6) Design an encoder using universal gate.

A typical encoder has 2^n inputs and n outputs.



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

A 4-to-2 encoder and its truth table

$$A = D1 + D3$$

$$B = D2 + D3$$

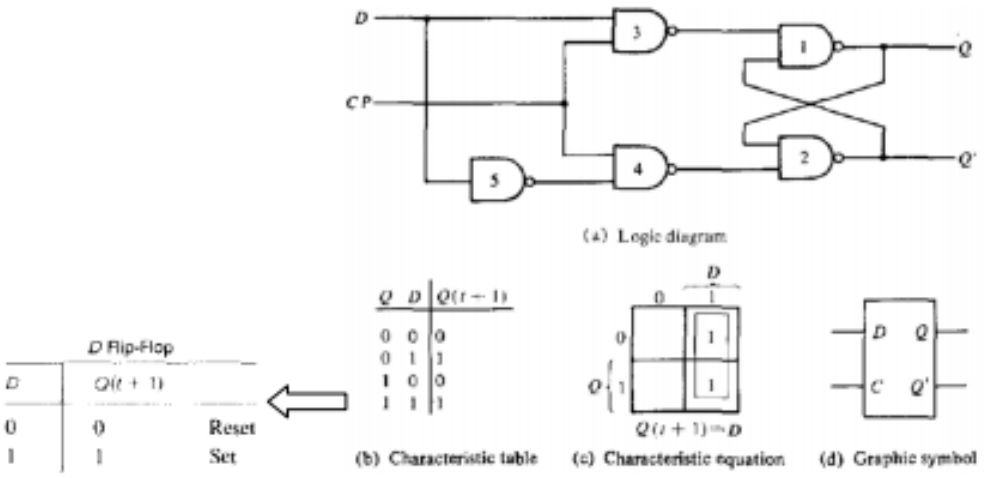
Q.N.7) What do you mean by D-flip-flop?

One way to eliminate the undesirable condition of the indeterminate state in the RS flip-flop is to ensure that inputs S and R are never equal to 1 at the same time. This is done in the D flip-flop shown in Fig. below. The D flip-flop has only two inputs: D and CP. The D input goes directly to the S input and its complement is applied to the R input.

As long as CP is 0, the outputs of gates 3 and 4 are at the 1 level and the circuit cannot change state regardless of the value of D.

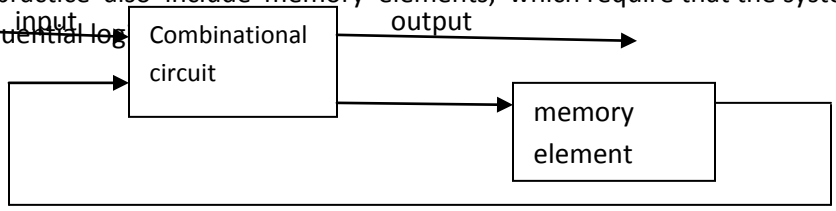
The D input is sampled when CP = 1.

- If D is 1, the Q output goes to 1, placing the circuit in the set state.
- If D is 0, output Q goes to 0 and the circuit switches to the clear state.



Q.N.8) What is sequential logic? What are the important feature?

Although every digital system is likely to have combinational circuits, most systems encountered in practice also include memory elements, which require that the system be described in terms of sequential log



Memory elements are devices capable of storing binary information within them. The binary information stored in the memory elements at any given time defines the state of the sequential circuit.

Block diagram shows external outputs in a sequential circuit are a function not only of external inputs, but also of the present state of the memory elements. Thus, a sequential circuit is specified by a time sequence of inputs, outputs, and internal states.

There are two main types of sequential circuits. Their classification depends on the timing of their signals.

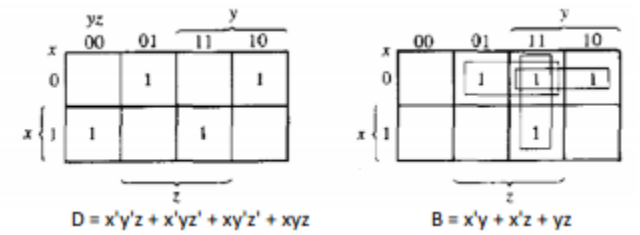
Synchronous sequential circuit: whose behavior can be defined from the knowledge of its signals at discrete instants of time

- A synchronous sequential logic system, by definition, must employ signals that affect the memory elements only at discrete instants of time. One way of achieving this goal is to use pulses of limited duration throughout the system so that one pulse-amplitude represents logic-1 and pulse amplitude (or the absence of a pulse) represents logic-0.
- The difficulty with a system of pulses is that any two pulses arriving from separate independent sources to the inputs of the same gate will exhibit unpredictable delays, will separate the pulses slightly, and will result in unreliable operation.
- Practical synchronous sequential logic systems use fixed amplitudes such as voltage levels for the binary signals. Synchronization is achieved by a timing device called a master-clock generator, which generates a periodic train of clock pulses. The clock pulses are distributed throughout the system in such a way that memory elements are affected only with the arrival of the synchronization pulse. Synchronous sequential circuits that use clock pulses in the inputs of memory elements are called clocked sequential circuits. Clocked sequential circuits are the type encountered most frequently. They do not manifest instability problems and their timing is easily divided into independent discrete steps, each of which is considered separately. The sequential circuits discussed in this chapter are exclusively of the clocked type.

Asynchronous sequential circuit: Behavior depends upon the order in which its input signals change and can be affected at any instant of time. The memory elements commonly used in asynchronous sequential circuits are time-delay devices.

Q.N.9)Simplify the Boolean function using K-map?

$$F = X'y'z + X'yz' + Xy'z' + Xyz$$



Q.N.10) Draw a parallel-in paralle- out shift register and explain it.

The final mode of operation is the Parallel-in to Parallel-out Shift Register. This type of register also acts as a temporary storage device or as a time delay device similar to the SISO configuration above. The data is presented in a parallel format to the parallel input pins P_A to P_D and then transferred together directly to their respective output pins Q_A to Q_D by the same clock pulse. Then one clock pulse loads and unloads the register. This arrangement for parallel loading and unloading is shown below.

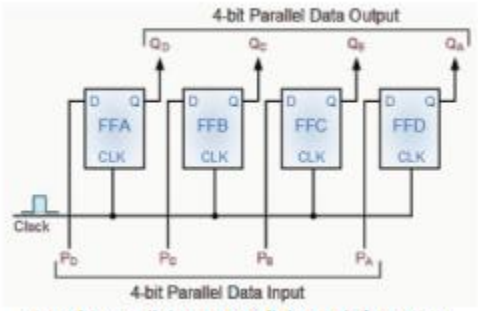


Fig: 4-bit Parallel-in to Parallel-out Shift Register

The PIPO shift register is the simplest of the four configurations as it has only three connections, the parallel input (PI) which determines what enters the flip-flop, the parallel output (PO) and the sequencing clock signal (Clk). Similar to the Serial-in to Serial-out shift register, this type of register also acts as a temporary storage device or as a time delay device, with the amount of time delay being varied by the frequency of the clock pulses. Also, in this type of register there are no interconnections between the individual flip-flops since no serial shifting of the data is required.

Q.N.11) Explain the 4-bit ripple counter?

Binary Ripple Counter

A binary ripple counter consists of a series connection of complementing flip-flops (T or JK type), with the output of each flip-flop connected to the CP input of the next higher-order flip-flop. The flip-flop holding the least significant bit receives the incoming count pulses. The diagram of a 4-bit binary ripple counter is shown in Fig. below. All J and K inputs are equal to 1.

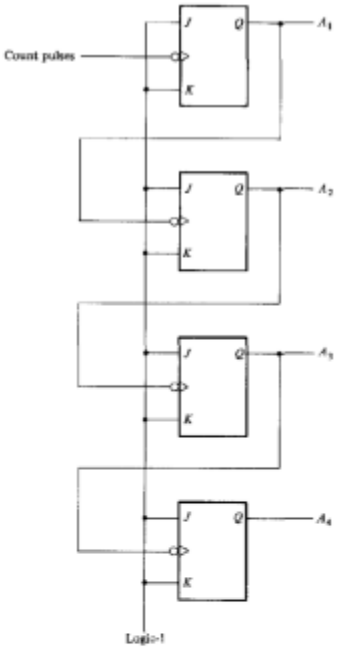


Fig: 4-bit binary ripple counter

All *J* and *K* inputs are equal to 1. The small circle in the *CP* input indicates that the flip-flop complements during a negative-going transition or when the output to which it is connected goes from 1 to 0.

To understand the operation of the binary counter, refer to its count sequence given in Table.

- It is obvious that the lowest-order bit A_1 must be complemented with each count pulse. Every time A_1 goes from 1 to 0, it complements A_2. Every time A_2 goes from 1 to 0, it complements A_3, and so on.
- For example: take the transition from count 0111 to 1000. The arrows in the table emphasize the transitions in this case. A_1 is complemented with the count pulse. Since A_1 goes from 1 to 0, it triggers A_2 and complements it. As a result, A_2 goes from 1 to 0, which in turn complements A_3. A_3 now goes from 1 to 0, which complements A_4. The output transition of A_4, if connected to a next stage, will not trigger the next flip-flop since it goes from 0 to 1. The flip-flops change one at a time in rapid succession, and the signal propagates through the counter in a *ripple* fashion.

Count Sequence				Conditions for Complementing Flip-Flops	
A_4	A_3	A_2	A_1		
0	0	0	0	Complement A_1	
0	0	0	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2
0	0	1	0	Complement A_1	
0	0	1	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2; A_2 will go from 1 to 0 and complement A_3
0	1	0	0	Complement A_1	
0	1	0	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2
0	1	1	0	Complement A_1	
0	1	1	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2; A_2 will go from 1 to 0 and complement A_3; A_3 will go from 1 to 0 and complement A_4
1	0	0	0	and so on . . .	

Q.N.12) Explain the programmable logic array?

A combinational circuit may occasionally have don't-care conditions. When implemented with a ROM, a don't care condition becomes an address input that will never occur. The words at the don't-care addresses need not be programmed and may be left in their original state (all 0's or all 1's). The result is that not all the bit patterns available in the ROM are used, which may be considered a waste of available equipment.

Defn: Programmable Logic Array or PLA is LSI component that can be used in economically as an alternative to ROM where number of don't-care conditions is excessive.

Block Diagram of PLA

A block diagram of the PLA is shown in Fig. below. It consists of n inputs, m outputs, k product terms, and m sum terms. The product terms constitute a group of k AND gates and the sum terms constitute a group of m OR gates. Fuses are inserted between all n inputs and their complement values to each of the AND gates. Fuses are also provided between the outputs of the AND gates and the inputs of the OR gates.

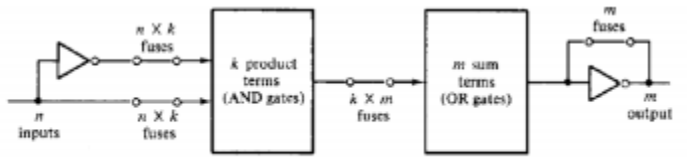


Fig: PLA block diagram

PLA program table and Boolean function Implementation

The use of a PLA must be considered for combinational circuits that have a large number of inputs and outputs. It is superior to a ROM for circuits that have a large number of don't-care conditions. Let me explain the example to demonstrate how PLA is programmed.

Consider a truth table of the combinational circuit:

Consider a truth table of the combinational circuit:

A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

PLA implements the functions in their sum of products form (standard form, not necessarily canonical as with ROM). Each product term in the expression requires an AND gate.

Q.N.13) Write short notes on

a) Asynchronous counter

In a ripple counter (Asynchronous Counter); flip-flop output transition serves as a source for triggering other flip-flops. In other words, the CP inputs of all flip-flops (except the first) are triggered not by the incoming pulses, but rather by the transition that occurs in other flip-flops.

b)Multiplexer

A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.

The selection of a particular input line is controlled by a set of selection lines.

Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.

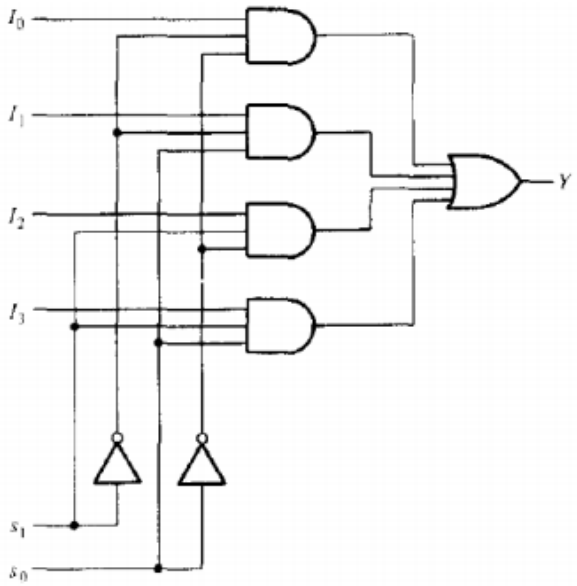


Fig: Logic Diagram: 4-to-1 line Multiplexer

s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Table: Function table

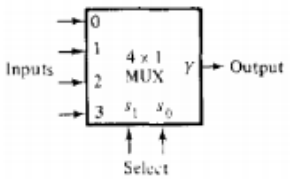


Fig: Block Diagram of Multiplexer

c) Static reduction table

Solution to digital logic 2066

Long answer question

Q.N.1) Design the 4 bit synchronous up/down counter with timing diagram, logic diagram and truth table.

Binary Up-Down Counter

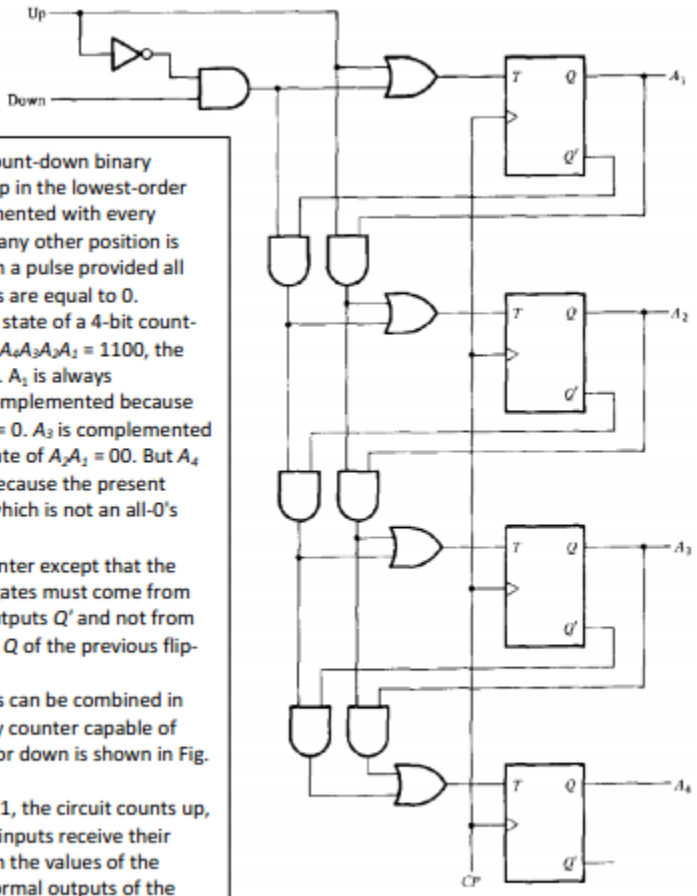


Fig: 4-bit up-down counter

- In a synchronous count-down binary counter, the flip-flop in the lowest-order position is complemented with every pulse. A flip-flop in any other position is complemented with a pulse provided all the lower-order bits are equal to 0.

Example: if the present state of a 4-bit count-down binary counter is $A_4A_3A_2A_1$ = 1100, the next count will be 1011. A_1 is always complemented. A_2 is complemented because the present state of A_1 = 0. A_3 is complemented because the present state of A_2A_1 = 00. But A_4 is not complemented because the present state of $A_3A_2A_1$ = 100, which is not an all-0's condition.

- Same as Binary counter except that the inputs to the AND gates must come from the complement outputs Q' and not from the normal outputs Q of the previous flip-flops.
- The two operations can be combined in one circuit. A binary counter capable of counting either up or down is shown in Fig. by side.
  - When **up** = 1, the circuit counts up, since the T inputs receive their signals from the values of the previous normal outputs of the flip-flops.
  - When **down** = 1 and **up** = 0, the circuit counts down

Q.N.2) Design a Full subtractor with truth table and logic gates.

Full-Subtractor

A full-subtractor is a combinational circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage.

This circuit has three inputs and two outputs. The three inputs, x, y, and z, denote the minuend,

subtrahend, and previous borrow, respectively. The two outputs, D and B, represent the difference and output-borrow, respectively.

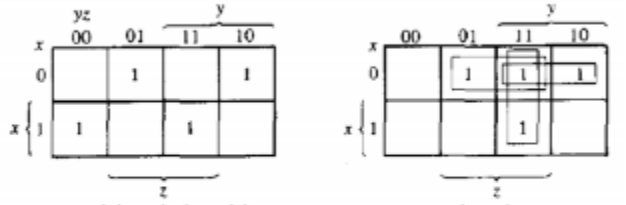
Truth-table and output-function formulation:

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

The 1's and 0's for the output variables are determined from the subtraction of x - y - z.

- The combinations having input borrow z = 0 reduce to the same four conditions of the half-adder.
- For x = 0, y = 0, and z = 1, we have to borrow a 1 from the next stage, which makes B = 1 and adds 2 to x. Since 2 - 0 - 1 = 1, D = 1.
- For x = 0 and yz = 11, we need to borrow again, making B = 1 and x = 2. Since 2 - 1 - 1 = 0, D = 0.
- For x = 1 and yz = 01, we have x - y - z = 0, which makes B = 0 and D = 0.
- Finally, for x = 1, y = 1, z =1l, we have to borrow 1, making B = 1 and x = 3, and 3 - 1 - 1 = 1, making D=1.

The simplified Boolean functions for the two outputs of the full-subtractor are derived in the maps:



D = x'y'z + x'yz' + xy'z' + xyz B = x'y + x'z + yz

- Circuit implementations are same as Full-adder except B output (analogous to C) is little different. (Don't worry! ladies and gentlemen, we will discuss it in class...)

Q.N.3) Design a decimal adder with logic diagram and truth table.

Computers or calculators that perform arithmetic operations directly in the decimal number system represent decimal numbers in binary-coded form.

Decimal adder is a combinational circuit that sums up two decimal numbers adopting particular encoding technique.

A decimal adder requires a minimum of nine inputs and five outputs, since four bits are required to code each decimal digit and the circuit must have an input carry and output carry.

Of course, there is a wide variety of possible decimal adder circuits, dependent upon the code used to represent the decimal digits.

Short answer question

Q.N.4) Differentiate between Analog and Digital System

Analog System

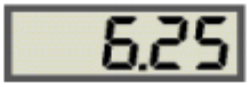
Analog systems process analog signals (continuous time signals) which can take any value within a range, for example the output from a speaker or a microphone.

An analog meter can display any value within the range available on its scale. However, the precision of readings is limited by our ability to read them. E.g. meter on the right shows 1.25V because the pointer is estimated to be half way between 1.2 and 1.3. The analogue meter can show any value between 1.2 and 1.3 but we are unable to read the scale more precisely than about half a division.

Digital System

- Digital systems process digital signals which can take only a limited number of values (discrete steps), usually just two values are used: the positive supply voltage (+Vs) and zero volts (0V).
- Digital systems contain devices such as logic gates, flip-flops, shift registers and counters.

A digital meter can display many values, but not every value within its range. For example the display on the right can show 6.25 and 6.26 but not a value between them. This is not a problem because digital meters normally have sufficient digits to show values more precisely than it is possible to read an analogue display.



The general purpose digital computer is a best known example of **digital system**.

Q.N.5) Convert the following octal number to hexadecimal.

a) 1760.46

converting to decimal

$=1\times 8^3+7\times 8^2+6\times 8^1+0\times 8^0+4\times 8^{-1}+6\times 8^{-2}$

=1008.594 [Note: the number behind point is multiplied by 16 and the whole number is written as a hexadecimal and the remaining point is multiplied again until u get satisfied. other method is shown in below]

Now converting to hexadecimal we have

=3F0.98

b)6055.263

converting to binary

=1 0111 1010 0111. 1000 0011 1

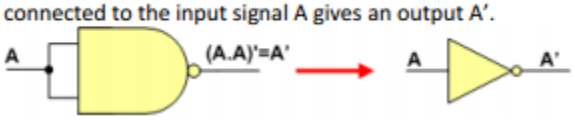
Binary to hexadecimal conversion

17A7.831

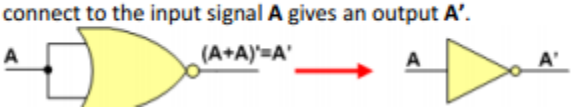
Q.N.6) Which gates can be used as inverters in addition to the NOT gate and how?

NAND and NOR gate are universal gate so they can be used instead of inverter

- **Implementing an Inverter Using only NAND Gate:** All NAND input connected to the input signal A gives an output A'.



Implementing neither an Inverter Using only NOR Gate: All NOR input pins connect to the input signal **A** gives an output **A'**.



Q.N.7) Draw a logic gates that implements the following

$$a)\ A = (Y_1 \oplus Y_2)(Y_3 \odot Y_4) + (Y_5 \oplus Y_6 \oplus Y_7)$$
$$b) A = (X_1 \odot X_2) + (X_3 \odot X_4) + (X_4 \odot X_5) \oplus (X_6 \odot X_7)$$

Q.N.8)State and prove De-Morgan's theorem 1st and 2nd with logic gates and truth table.

Theorem1: Idempotence (a) x + x = x (b) x.x = x

Theorem2: Existence: 0&1 (a) x + 1 = 1 (b) x.0 = 0

Proofs:

(a) The proofs of the theorems with one variable are presented below:

THEOREM 1(a): x + x = x

x + x = (x + x) . 1 (P4: Identity element)

= (x + x)(x + x') (P5: Existence of inverse)

= x + xx' (P3: Distribution)

=x + 0 (P5: Existence of inverse)

=x (P4: Identity element)

THEOREM 1(b): x·x = x

x.x = xx + 0 (P4: Identity element)

=xx + xx' (P5: Existence of inverse)

=x(x + x') (P3: Distribution)

= x.1 (P5: Existence of inverse)

=x (P4: Identity element)

Hey! Each step in theorem 1(b) and 1(a) are dual of each other.

THEOREM 2(a): x + 1 = 1

x + 1 = 1· (x + 1) (P4: Identity element)

= (x + x')(x + 1) (P5: Existence of inverse)

=x + x'·1 (P3: Distribution)

= x + x' (P4: Identity element)

= 1 (P5: Existence of inverse)

THEOREM 2(b): x.0 = 0 by duality.

Q.N.9) Reduce the following expression using K map

a. $\overline{A} + B(A + \overline{B} + D)(\overline{B}$***+C) (B+C+D)***

Q.N.10) Differentiate between a MUX and DEMUX

A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.

The selection of a particular input line is controlled by a set of selection lines.

Normally, there are 2n input lines and n selection lines whose bit combinations determine which input is selected.

A demultiplexer is a circuit that receives information on a single line and transmits this information on one of 2n possible output lines. The selection of a specific output line is controlled by the bit values of n selection lines.

A Decoder with an enable input can function as a demultiplexer.

Here, enable input and input variables for decoder is taken as data input line and selection lines for the demultiplexer respectively.

Q.N.11) Explain the operation of Decoder

Decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2n unique output lines.

- If the n-bit decoded information has unused or don't-care combinations, the decoder output will have fewer than 2noutputs.
- n-to-m-line decoders have m <= 2n

Example: **3-to-8 line decoder**

The 3 inputs are decoded into 8 outputs, each output representing one of the minterms of the 3-input variables.

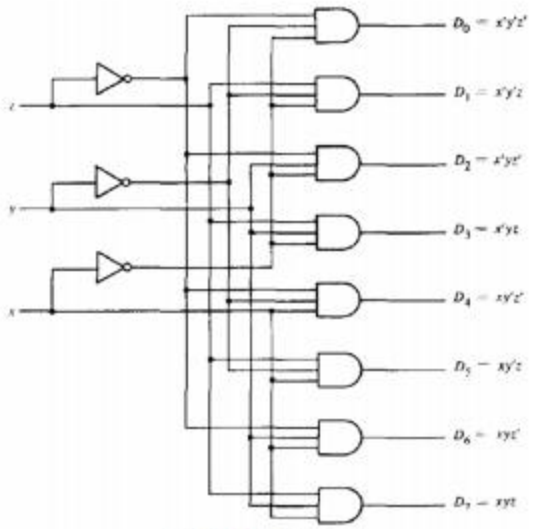


Fig: 3-to-8 line decoder

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

Output variables are mutually exclusive because only one output can be equal to 1 at anyone time. The output line whose value is equal to 1 represents the minterm equivalent of the binary number presently available in the input lines.

Table: Truth-table for 3-to-8 line Decoder

.

Q.N.12) What are the various types of shift registers?

Shift Registers

A register capable of shifting its binary information either to the right or to the left is called a shift register. The logical configuration of a shift register consists of a chain of flip-flops connected in cascade, with the output of one flip-flop connected to the input of the next flip-flop. All flip-flops receive a common clock pulse that causes the shift from one stage to the next.

The Shift Register is used for data storage or data movement and are used in calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format. The individual data latches that make up a single shift register are all driven by a common clock (Clk) signal making them synchronous devices. Shift register IC's are generally provided with a clear or reset connection so that they can be "SET" or "RESET" as required.

Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being:

- Serial-in to Parallel-out (SIPO) - the register is loaded with serial data, one bit at a time, with the stored data being available in parallel form.
- Serial-in to Serial-out (SISO) - the data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.
- Parallel-in to Serial-out (PISO) - the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.
- Parallel-in to parallel-out (PIPO) - the parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.

Q.N.13) What do you mean by synchronous counter?

Synchronous counter, the input pulses are applied to all CP inputs of all flip-flops. The change of state of a particular flip-flop is dependent on the present state of other flip-flops.

Synchronous counters are distinguished from ripple counters in that clock pulses are applied to the CP inputs of all flip-flops. The common pulse triggers all the flip-flops simultaneously, rather than one at a time in succession as in a ripple counter. The decision whether a flip-flop is to be complemented or not is determined from the values of the J and K inputs at the time of the pulse. If J = K = 0, the flip-flop remains unchanged. If J = K = 1, the flip-flop complements.

For eg of a synchronous counter we can look after binary counter

Binary Counter

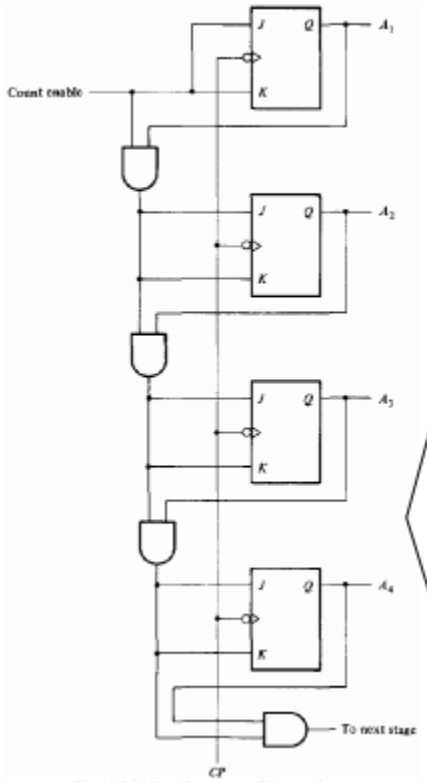


Fig: 4-bit Synchronous Binary Counter

- The design of synchronous binary counters is so simple that there is no need to go through a rigorous sequential-logic design process. In a synchronous binary counter, the flip-flop in the lowest-order position is complemented with every pulse. This means that it's *J* and *K* inputs must be maintained at logic-1. A flip-flop in any other position is complemented with a pulse provided all the bits in the lower-order positions are equal to 1, because the lower-order bits (when all 1's) will change to 0's on the next count pulse.
- Synchronous binary counters have a regular pattern and can easily be constructed with complementing flip-flops and gates. The regular pattern can be clearly seen from the 4-bit counter depicted in Fig by side.
- The *CP* terminals of all flip-flops are connected to a common clock-pulse source. The first stage A_1 has its *J* and *K* equal to 1 if the counter is enabled. The other *J* and *K* inputs are equal to 1 if all previous low-order bits are equal to 1 and the count is enabled. The chain of AND gates generates the required logic for the *J* and *K* inputs in each stage. The counter can be extended to any number of stages, with each stage having an additional flip-flop and an AND gate that gives an output of 1 if all previous flip-flop outputs are 1's.