Question Paper

Digital Electronics (EE-204-F)

MDU Examination May 2015

1. (a) represent (32)10 in
   - (i) BCD 8421 code
   - (ii) Excess-3 code
   - (iii) ASCII code

   (b) Design half adder using only NAND gates.

   (c) Give logic symbol, Truth table and circuit diagram for a clocked SR flip-flop.

   (d) A combinational circuit is defined by the function

   $F1=\sum m(1,5,7)$

   $F2=\sum m(5,6,7)$

   Implement the circuit with a PLA

   SECTION-A

2. (a) What is a hamming code? A seven bit even parity hamming code is received as 1110101. What is the correct code?

   (b) Convert each of decimal number to BCD and add.
   - (i) (65)10 + (58)10
   - (ii) (113)10 + (101)10

   (c) What do you mean by self complementing code? Write two self complementing codes.

   (d) Obtain simplified expression in POS using K-maps for:
   - (i) $F(w,x,y,z)=\pi M(1,3,5,7,13,15)$
   - (ii) $F(w,x,y,z)=x'z' + z' + y'z' + yz' + xyz$

3 (a) What I Quine mcClusky method? Use QM method to reduce each expression to a minimum SOP form

   (i) $Y= A'B'C'D' + A'B'C'D + ABCD + ABCD'$

   (ii) $Y= A'B(C'D' + C'D) + AB(C'D' + C'D) + AB'C'D$

   (b) Reduce by K-Map and implement using NOR-NOR gate:

   $F=\sum m(1,2,3,4,6,7,10,11,13,14)$

SECTION-B

4. (a) Design a full subtractor using only NOR gates.
   (b) Design a 3-bit comparator using three one-bit comparator and logic gates

5 (a) Implement a full adder circuit with 3:8 decoder and few logic gates.

   (b) Implement the following Boolean function using 8:1 multiplexer:
   $F(A,B,C,D)= A'BD' + ACD + B'CD + A'C'D$

SECTION – C

6 (a) Convert the following:

(i) JK flip-flop into D flip-flop

(ii) SR flip-flop into JK flip-flop

(b) List the basic types of shift registers in terms of data movement

(c) Compare ripple counter with synchronous counter.

7 (a) Design a MOD-10 synchronous counter with JK flip-flops

(b) Design MOD-8 ripple counter(up-counter) using JK flip-flops

UNIT-D

8 (a) List basic type of programmable logic devices.

(b) Implement the following Boolean expression using ROM

(i) $F1(A,B,C)= \sum m(0,2,4,7)$

(ii) $F2(A,B,C)= \sum m(1,3,5,7)$

(c) Explain AND-OR structure of PAL and PLA

9 (a) A combinational logic is defined by the function:

$F1(A,B,C) = \sum m(3,5,6,7)$

$F2(A,B,C) = \sum m(0,2,4,7)$

Implement the circuit with PLA having 3 inputs, 4 product terms and two outputs.

(b) Write short notes on:

(i) Hazards

(ii) ASMs

Digital Electronics

SOLUTION-May 2015

1. (a) represent $(32)_{10}$ in

(i) BCD 8421 code

$(32)_{10}$ = 00110010

(ii) Excess-3 code

$(32)_{10}$ in BCD is given as 00110010

We can get the Excess-3 code from BCD by adding 3 to each BCD digit

Thus $(32)_{10}$ in Excess-3 = 65 = 01100101

(iii) ASCII code

ASCII are 7 or 8-bit codes which are printable characters. These codes are obtained by oring 30h with the unpacked BCD digit. Thus the ASCII code for $(32)_{10}$ is given as '3' '2' which is 0x33 and 0x32 in hexadecimal

(b) Design half adder using only NAND gates.

A half adder is a combinational circuit that adds two bits and generates a sum and carry outputs.

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

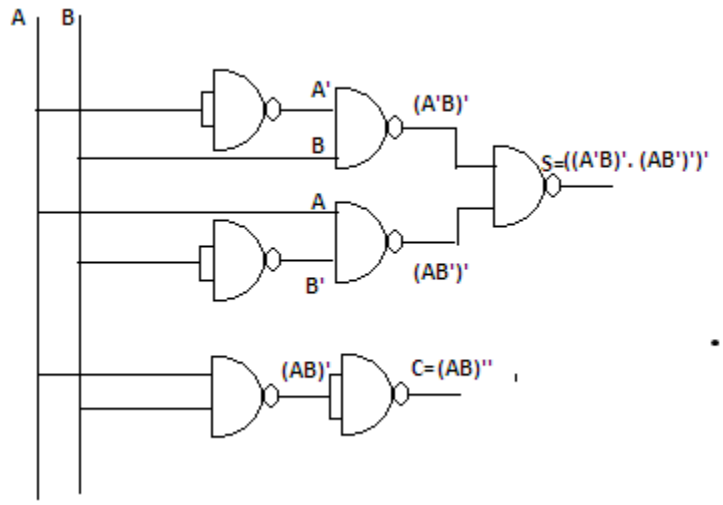
The SOP equation for S and C are:

S= A'B + AB' and C= AB

NAND equations are obtained by using De-Morgan's theorem as:

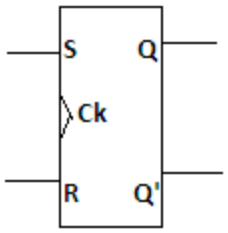
S=S''=(A'B + AB')'' = ((A'B)' . (AB')')'

C=C''= (AB)''



(c) Give logic symbol, Truth table and circuit diagram for a clocked SR flip-flop.

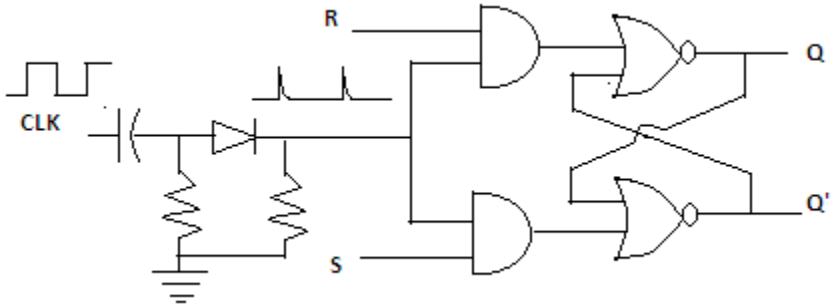
The logic symbol for a SR flip-flop is given below;



The truth table of SR flip-flop

CK	S	R	Qn
↓	x	X	No Change
↑	0	0	No change
↑	0	1	Reset
↑	1	0	Set
↑	1	1	? invalid or forbidden

The circuit diagram for SR Flip-flop is given below:



(d) A combinational circuit is defined by the function

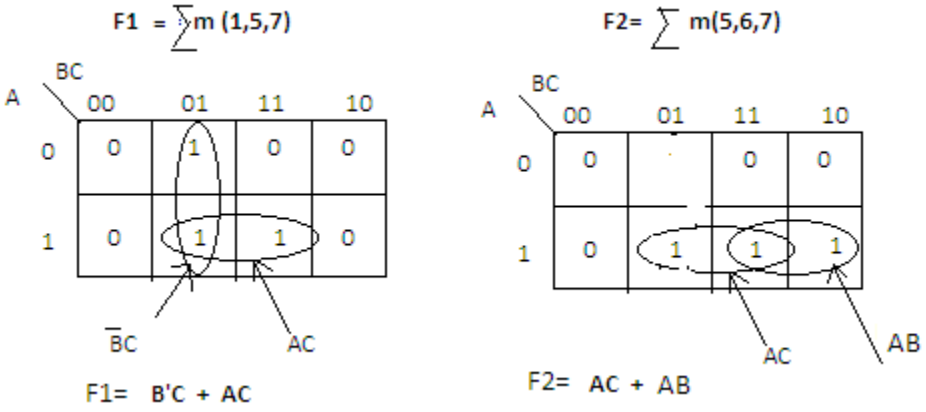
F1=∑m(1,5,7)

F2=∑m(5,6,7)

Implement the circuit with a PLA

SOLUTION:

Step-1: K- Simplification(K-Map or any other)



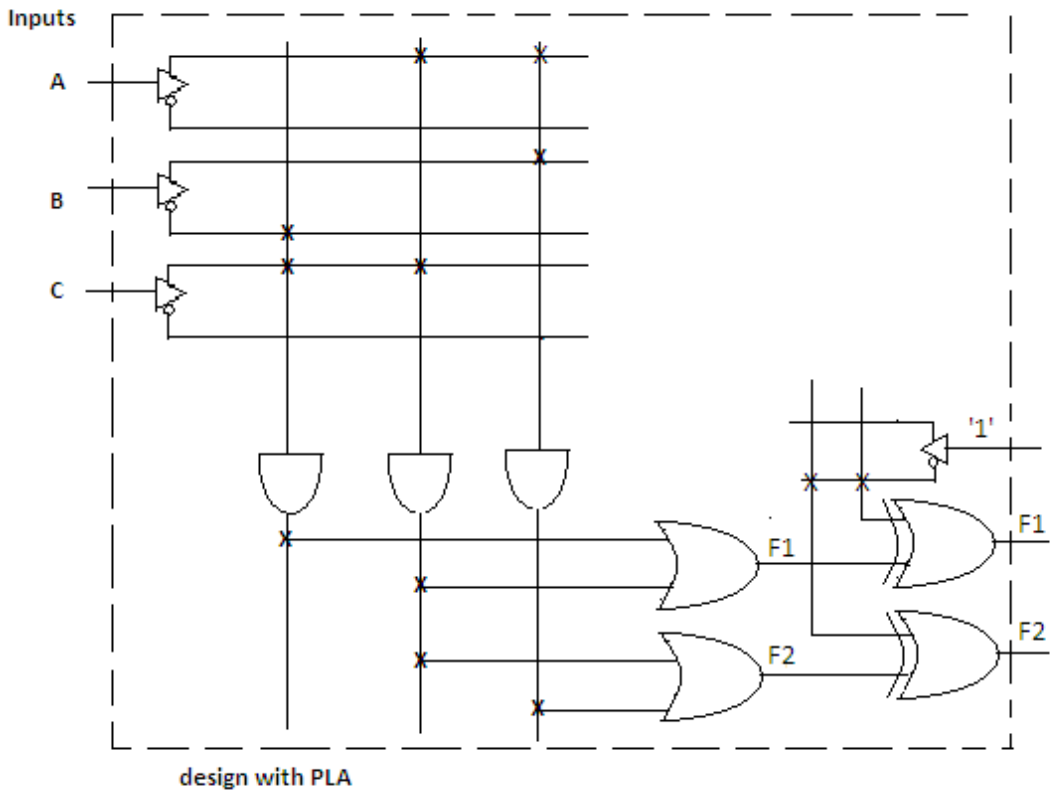
Step-2:

Preparing the PLA programming Table:

We write the minterms from the solution of both the expressions in the table but common terms are written only once:

Product Terms	Inputs			Outputs	
	A	B	C	F1	F2
B'C	-	0	1	1	-
AC	1	-	1	1	1
AB	-	1	0	-	1

Step-3 : Implement with PLA



design with PLA

SECTION-A

2. **(a) What is a hamming code? A seven bit even parity hamming code is received as 1110101. What is the correct code?**

SOLUTION:

Hamming code is a single bit error detection and correction code. We can detect the position the bit in error. We can then complement that bit to correct the error. The code uses multi parity bits at positions 1,2,4,8,16------. These parity bits are calculated as:

P1= checking bits 1,3,5,7,….. for odd or evenness

P2= bits starting from bit-2, then leaving two bits as: 2,3, 6,7,10,11…… for odd or evenness

P4= Checking bits in group of four starting from bit-4 as : 4,5,6,7, 12,13,14,15…. for odd or evenness

P8=checking bits in group of eight starting from bit-8 as : 8,9,10,11, 12,13,14,15 for odd or evenness.

7	6	5	4	3	2	1
			P4		P2	P1
1	1	1	0	1	0	1

Checking the parity bit for evenness

C1= checking alternate bits 7,5,3,1 = 1,1,1,1= even; correct

C2= Checkin twobits leaving to bits, so cheching bits 7,6,3,2 = 1, 1, 1, 0 = odd ; incorrect

C4= checking four bits 7,6,5,4= 1,1,1,0 = odd ; incorrect

So error is in bit

C4 c2 c1

1 1 0

That is bit-6 is in error; and so the correct code is obtained by complementing the bit-6 of the received code and hence the correct data sent was : **1010101**

(b) **Convert each of decimal number to BCD and add.**

Solution:

(i) (65)10 + (58)10

We represent +65 and +58 using 4 digit BCD numbers, one digit for sign and three digits to hold the sum

+65 = 0065 = 0000 0000 0110 0101

+58 = 0058 = 0000 0000 0101 1000

Adding ----------

0000 0001 1011 1101

Adding 6(0110) 0110 0110

0000 0001 0010 0011

The answer in BCD is 0123 MSd 0 means +ve, so answer is +123

(ii) $(113)_{10} + (101)_{10}$

We represent +65 and +58 using 4 digit BCD numbers, one digit for sign and three digits to hold the sum

+113 = 0113 = 0000 0001 0001 0011

+101 = 0101 = 0000 0001 0000 0001

Adding + ----------

0000 0010 0001 0100

Since the addition results in each nibble which are less than 9, so BCD adjust is not required.

So answer is 0214 which is +214

(c) What do you mean by self complementing code? Write two self complementing codes.

Solution:

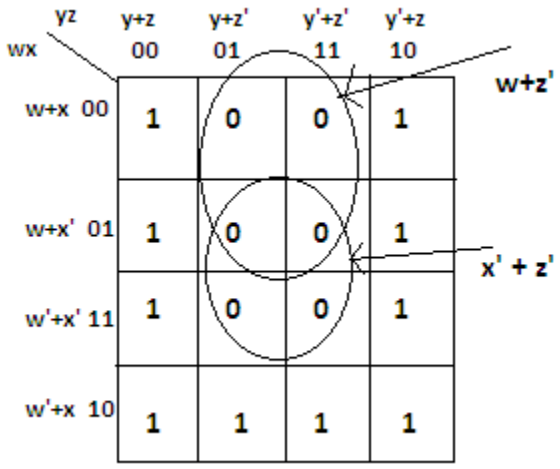
Self complementing codes are those codes which can be obtained by completing the bits. Examples of self complementing codes are XS-3. In XS-3 Complementing 1 gives 9, complementing 2 gives value for 8 and so on and vice versa.

Decimal digits	XS-3 Codes	Complement	Decimal Value
0	0011	1100	9
1	0100	1011	8
2	0101	1010	7
3	0110	1001	6
4	0111	1000	5
5	1000	0111	4
6	1001	0110	3
7	1010	0101	2
8	1011	0100	1
9	1100	0011	0

It is clearly seen in the table given above that compliment of 0011 is 1100 which are codes for 0 and 9 respectively, same is valid for all other digit as well. This indicates that XS-3 is self complementing code.

(d) Obtain simplified expression in POS using K-maps for:

(i) F(w,x,y,z)=πM(1,3,5,7,13,15)



The upper group of zeros formed by grouping Maxterm 1,3,5,7 give the Maxterm (w+z')

The lower group of zeros formed by grouping Maxterm 5,7,13,15 give the Maxterm (x'+z')

So, the complete simplification result the simplified POS equation as :

F'(w,x,y,z) = **(w+z')(x'+z')**

(iii) F(w,x,y,z) = X'Z' + Z' + Y'Z' + YZ' + XYZ

i. Convert to canonical sop form

= X'Z'(Y+Y') + Z'(X+X')(Y+Y') + Y'Z'(X+X') + YZ'(X+X') + XYZ

= X'YZ' + X'Y'Z' + Z'(XY+XY'+X'Y + X'Y') + XY'Z'+X'Y'Z' + XYZ'+X'YZ' + XYZ

= ~~X'YZ'~~ + ~~X'Y'Z'~~+ ~~XYZ'~~+~~XY'Z'~~+~~X'YZ'~~ + ~~X'Y'Z'~~+ XY'Z'+X'Y'Z' + XYZ'+X'YZ' + XYZ

= XY'Z'+X'Y'Z' + XYZ'+X'YZ' + XYZ

=$\sum m(0,2,4,6,7)$

ii. POS is obtained by considering the missing terms and appending Π symbol as:

= $\Pi_M(1,3,5)$

=(X' +Y'+Z)(X'+Y+Z)(X+Y'+Z)

3 (a) What is Quine McClusky method? Use QM method to reduce each expression to a minimum SOP form

(i) Y= A'B'C'D' + A'B'C'D + ABCD + ABCD'

(ii) Y= A'B(C'D' + C'D) + AB(C'D' + C'D) + AB'C'D

SOLUTION:

McClusky : McClusky is a tabular method of simplifying the Boolean Expressions.

i. Y= A'B'C'D' + A'B'C'D + ABCD + ABCD'

Minterms	A'B'C'D'	A'B'C'D	ABCD	ABCD'
Binary	0000	0001	1111	1110
No. of 1's	0	1	4	3

No. of 1		Minterm	Binary		1st pairing	Reduced Minterm
0	0	A'B'C'D'	0000	√	0,1	000-
1	1	A'B'C'D	0001	√		
3	14	ABCD'	1110	√	14, 15	111-
4	15	ABCD	1111	√		

The simplification is: Y= A'B'C' + ABC

PI Chart

	0	1	14	15
A'B'C' **(EPI)**	√	√		
ABC **(EPI)**			√	√
	√	√	√	√

Both the terms are essential prime implicants, and hence are part of solution.

Therefore the final solution is Y= A'B'C' + ABC

ii. Y = A'B(C'D' + C'D) + AB(C'D' + C'D) + AB'C'D

= A'BC'D' + A'BC'D + ABC'D' + ABC'D + AB'C'D

Minterms	A'BC'D'	A'BC'D	ABC'D'		ABC'D	AB'C'D
Binary	0100	0101	1100		1101	1001
No. of 1's	1	2	2		3	2

Writing the minterms by grouping terms as per number of 1's

No. of 1	Minterms	Minterms	Binary	Tick used terms	pairing	Reduced Minterm in Binary	Reduced Minterm
1	4	A'BC'D'	0100	√	4,5	010-	A'BC'
2	5	A'BC'D	0101	√	9,13	1-01	AC'D
	9	AB'C'D	1001	√			
3	13	ABC'D	1101	√			

The reduced expression is Y= A'BC' + AC'D

The PI Chart:

	4	5	9	13
A'BC' **(EPI)**	√	√		
AC'D **(EPI)**			√	√
	√	√	√	√

Both the PI minterms are the essential PIs and thus are part of the final solution :

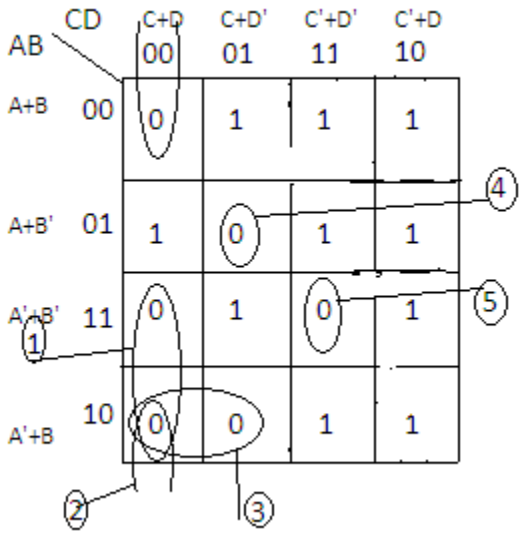
Y= A'BC' + AC'D

(b) Reduce by K-Map and implement using NOR-NOR gate:

$F=\sum m(1,2,3,4,6,7,10,11,13,14)$

Solution:

To obtain NOR-NOR solution, we will group the zeros instead of 1's. This will give solution for F'.



As seen in the K-Map, there are 5 groups, three quads, two pairs and one single, these groups are marked in circle 5 through 1. There solutions are:

Group-1: formed with maxterms 8,12 gives: A’ + C + D

Group-2: formed with maxterms 0,8 gives: B + C + D

Group-3 : formed with minterms 8,9 gives : A’ + B +C

Group-4 :formed with single maxterms 5 gives: A + B’ + C +D’

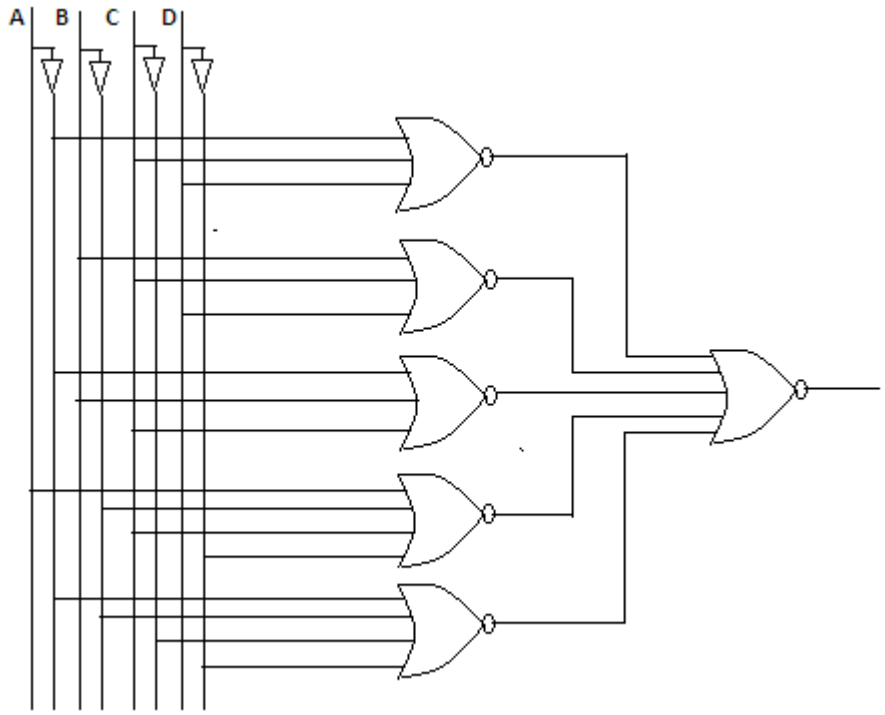
Group-5: formed with maxterms 15 gives: A’ + B’ + C’ + D’

So the solution for POS is: F’(A,B,C,D) = (A’+C+D).(B+C+D).(A’+B+C).(A+B’+C+D’).(A’B’C’D’)

Complementing we get F(A,B,C,D)’’ = ((A’+C+D).(B+C+D).(A’+B+C).(A+B’+C+D’).(A’B’C’D’))’’

= ((A’+C+D)’ + (B+C+D)’ + (A’+B+C)’ +(A+B’+C+D’)’ + (A’B’C’D’)’)’

This gives a NOR-NOR solution for the given SOP function and is implemented as:



SECTION-B

4. (a) Design a full subtractor using only NOR gates.

SOLUTION: A ful subtract is a combinational circuit that two one bit number using a borrow from the next stage. The truth table of a Full Subtractor is given below:

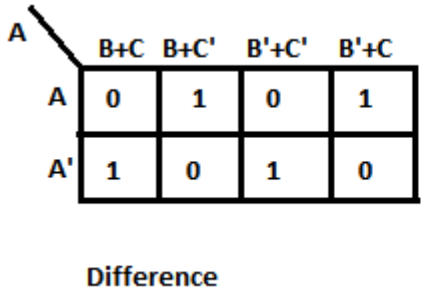
A	B	C	Diff	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0

1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

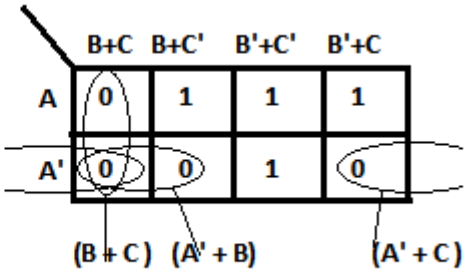
The POS equation for the Diff and Borrow are obtained by using the 0's and thus we get the expressions for (Diff)' and (Borrow)' and is given below:

Diff: (A+B+C) . (A+B'+C') . (A' + B + C') . (A' + B' + C)

Borrow= (A+B + C) . (A' + B + C) . (A' + B + C') . (A' + B' + C)



Difference



Borrow

Taking the double complement of the above equation

Diff'' = [(A+B+C) . (A+B'+C') . (A' + B + C') . (A' + B' + C)]''

= [(A+B+C)' +. (A+B'+C')' + (A' + B + C')' + (A' + B' + C)']'

Borrow'' = [(B+C). (A'+B).(A'+C)]''

= [(B+C) + (A'+B) + (A'+C)']'

Implementation using NOR gates

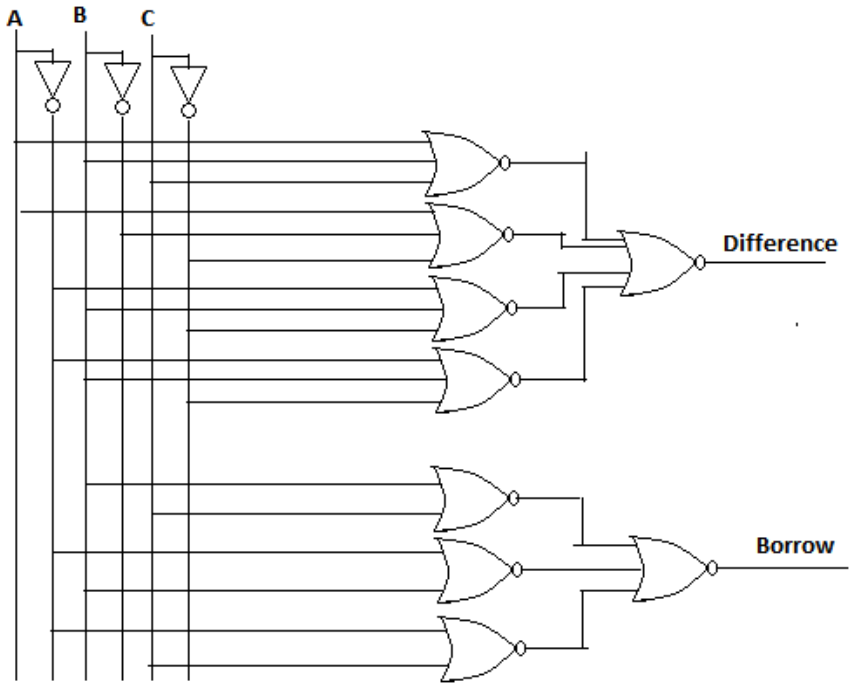
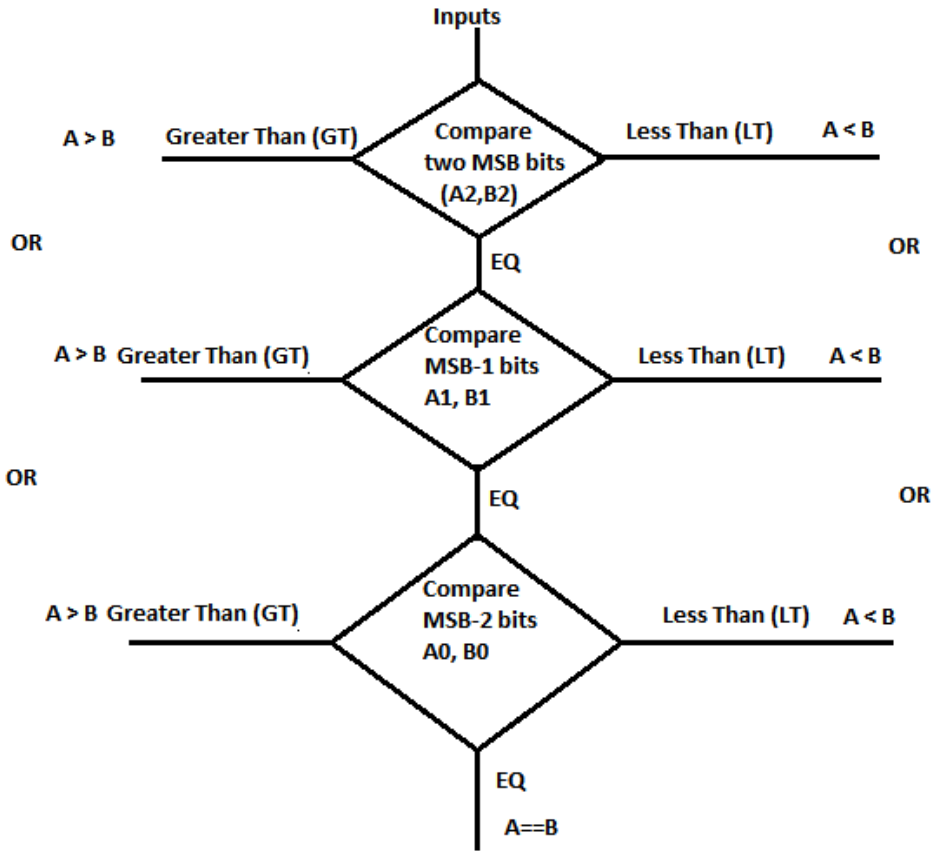


Fig 4-a: Full Subtractor

(b) Design a 3-bit comparator using three one-bit comparator and logic gates

SOLUTION:

As the number of bits for comparison becomes more than 2, the design of comparator becomes complex and tedious. So the algebraic or K-Map solution using truth table is not feasible. We utilize the algorithmic approach for such design as shown in the following flowchart:

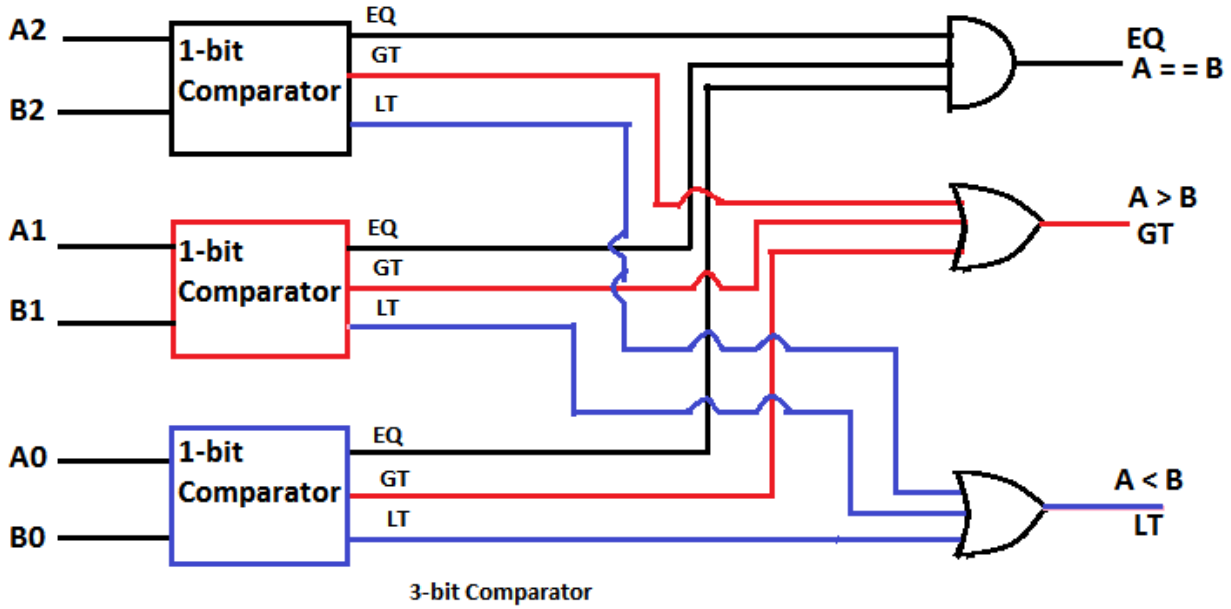


For equality each bit must be equal only then a decision for equality can be arrived.

For GT (>) case, we first check MSBs if A2 > B2 we arrive at decision, If not greater then we compare A1B1 for GT(>) or if false then we check A0B0. In this case the decision is at MSB or MSB-1, or MSB-2; see left side of flowchart.

Similarly, for LT(<), we first check A2,B2, or if false then A1B1, or if false then A0B0.

So for GT and LT, if any of the comparison, staring from MSB, is true, then we arrive at the final answer, therefore choce is to use OR gate to combine all GT together and use another OR gate to combine all LT. The complete design for the 3-bit comparator is shown below:



3-bit Comparator

5 (a) Implement a full adder circuit with 3:8 decoder and few logic gates.

SOLUTION:

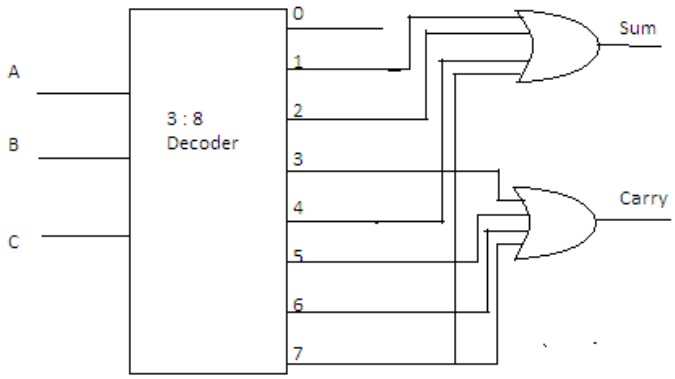
A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The SOP equation from the truth table generated by considering the minterms that generate a high for the Sum and carry output

Sum = ∑m(1,2,4,7)

Carry =∑m(3,5,6,7)

This can be implemented using a 3 x 8 decoder with two OR gates to OR the minterms of two equations as shown in figure:



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

F(A,B,C,D)= A'BD' + ACD + B'CD + A'C'D

SOLUTION:

F(A,B,C,D) = A'BD' + ACD + B'CD + A'C'D

= A'BD'(C+C') + ACD(B+B') + B'CD (A+A')+ A'C'D(B+B')

=A'BCD' + A'BC'D' + ABCD + AB'CD + AB'CD + A'B'CD + A'BC'D + A'B'C'D

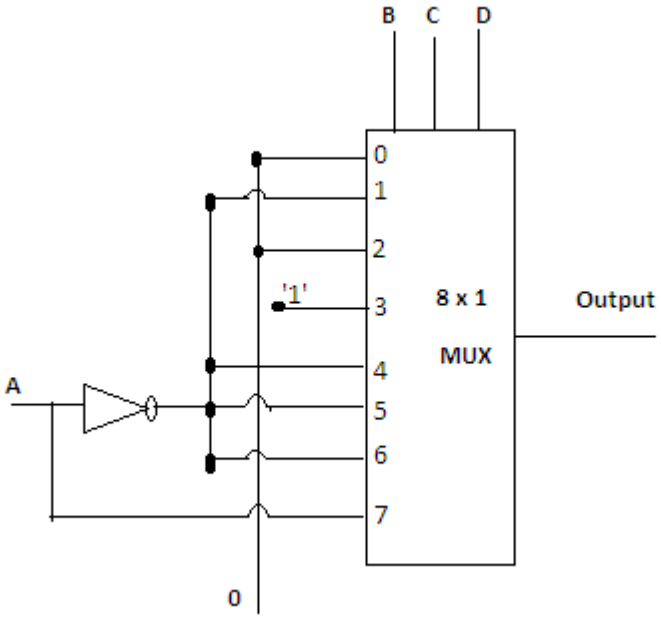
Writing the common terms only once:

=A'BCD' + A'BC'D' + ABCD + AB'CD + A'B'CD + A'BC'D + A'B'C'D

Now the standard SOP contains all the variable in each minterm. Now we can implement it using 8X1 MUX, but first we need to reduce it as shown below:

A	0	1	2	3	4	5	6	7
0		1		1	1	1	1	
1				1				1
Input	0	A'	0	1	A'	A'	A'	A

As seen from the table above that the inputs can be connected as per row number 3. And the control inputs to the MUX will be BCD and the implementation is shown in figure below:



SECTION – C

6 (a) Convert the following:

(i) JK flip-flop into D flip-flop

SOLUTION:

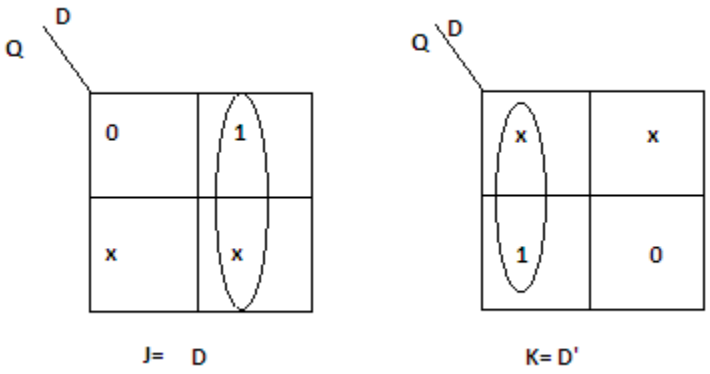
The conversion process is shown through the following steps:

Step-1:

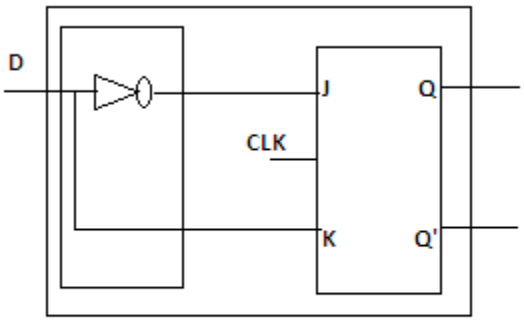
Write the table showing the characteristic for the desired flip-flop and the excitation table of the given flip-flop

Desired Flip-Flop Characteristic table			Excitation Table of Given Flip-flop	
Present State	Input	Next State	J	K
0	0	0	0	X
0	1	1	1	x
1	0	0	x	1
1	1	1	x	0

K-map Solution:



The implementation of JK flip-flop into a d Flip-Flop:



JK flip-Flop to D Flip-Flop

(ii) SR flip-flop into JK flip-flop

SOLUTION:

The conversion process is shown through the following steps:

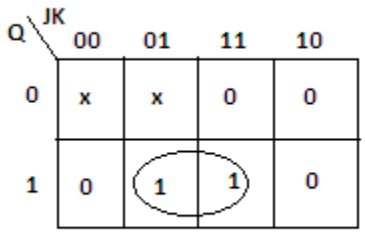
Step-1:

Write the table showing the characteristic for the desired flip-flop and the excitation table of the given flip-flop

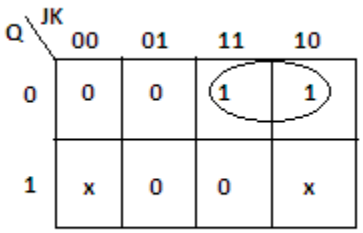
	Desired Flip-Flop Characteristic table	Excitation Table of Given Flip-flop

Present State	J	K	Next State	R	S
0	0	0	0	X	0
0	0	1	0	X	0
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	1	0	X
1	0	1	0	1	0
1	1	0	1	0	X
1	1	1	0	1	0

Solving using k- Map

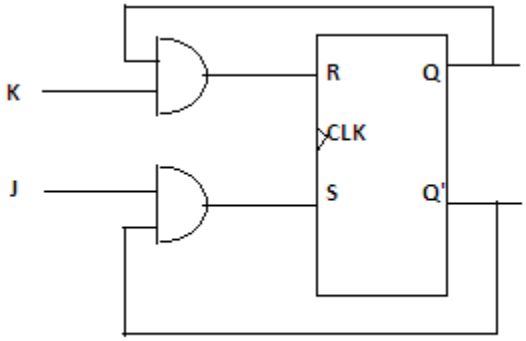


R = QK



S = Q'J

Implementation for SR to JK flip-flop:



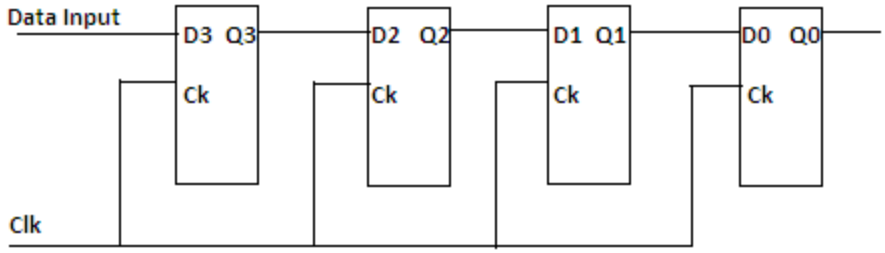
Conversion of SR to JK Flip-Flop

(b) List the basic types of shift registers in terms of data movement

SOLUTION:

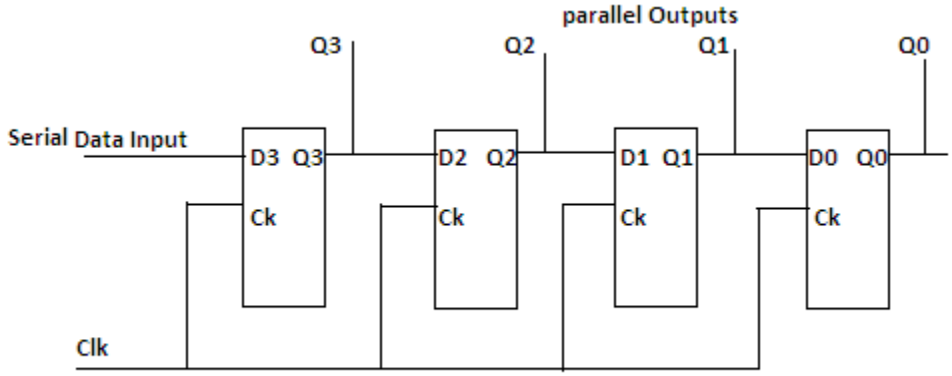
The different types of registers used for data movement are :

Serial in Serial Out(SISO) register:



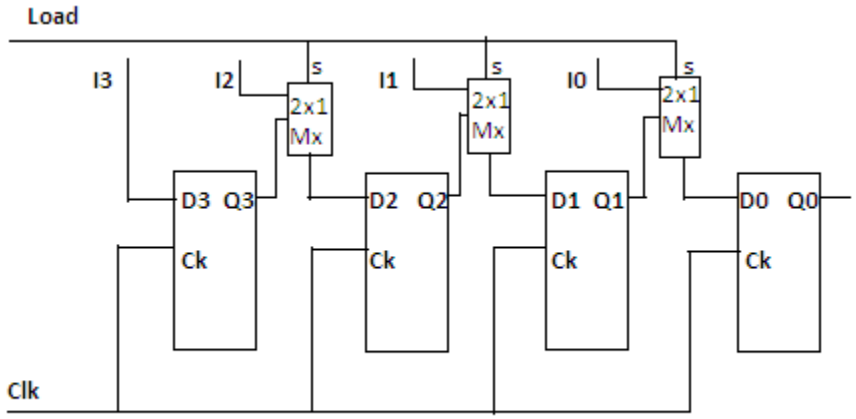
Serial In Serial Out register

Serial in Parallel Out(SIPO) register:



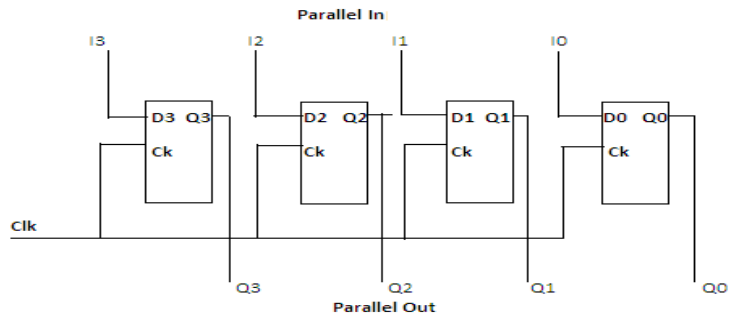
Serial In Parallel Out register

Parallel in Serial Out(PISO) register:



Parallel In Serial Out register

Parallel in Parallel Out(PIPO) register:



Parallel In Parallel Out register

(c) Compare ripple counter with synchronous counter.

SOLUTION:

Ripple Counter	Synchronous Counter
Logic is quite Simple	Logic become quite complicated with increase in FFs
Q or Q' Output of the previous stage FF is connected as clock input to the next stage FF	All FFs are connected to a common clock source
As the clock source to each FF is from different source, the FF are not clocked simultaneously	All FFS receive the clock simultaneously
Total propagation delay Pd=nx(td); where n is the number of FFs and td is propagation delay /FF	Total propagation delay Pd=(td)FF + (td)gate. Pd is quite shorter than in asynchronous counters.
Because of long propagation frequency of operation is Low	Because of long propagation frequency of operation is high

7 (a) Design a MOD-10 synchronous counter with JK flip-flops

SOLUTION:

To design a mode-10 synchronous counter, we follow the below

Step-1: We need to remember the excitation table of the JK flip flop as it will be used in the following table:

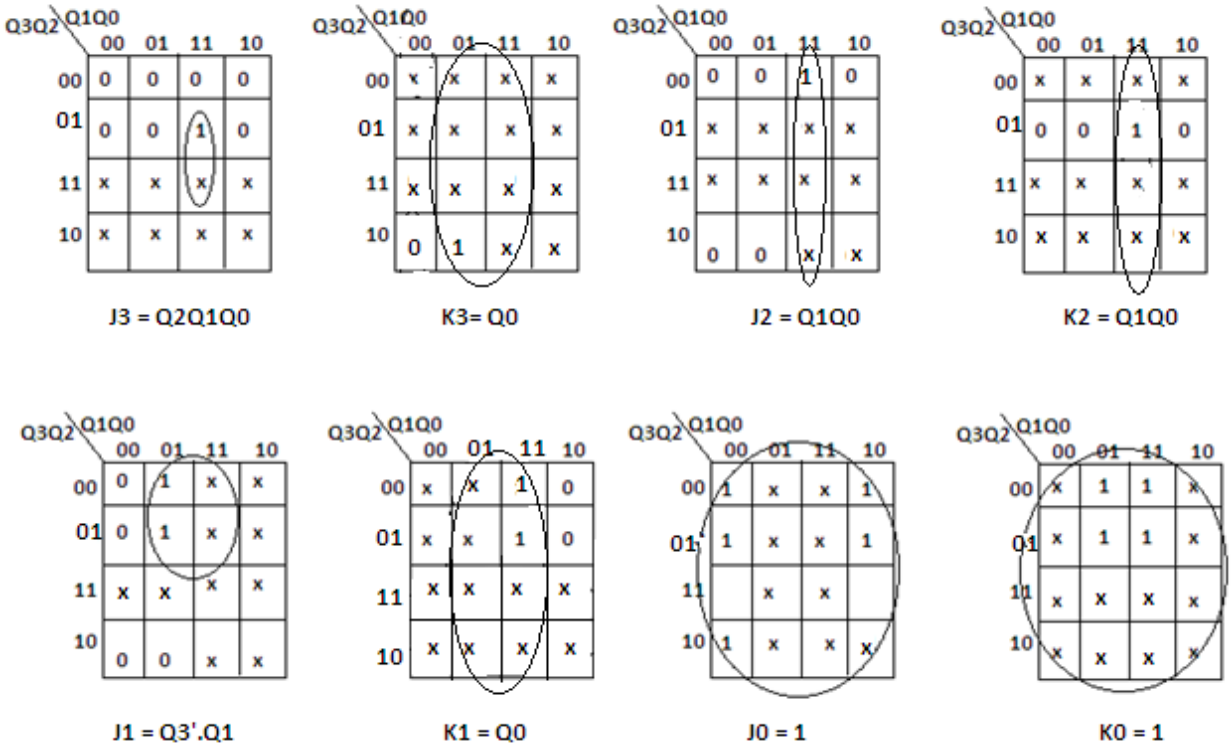
Excitation Table JK flip-Flop

Present State	Next State	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

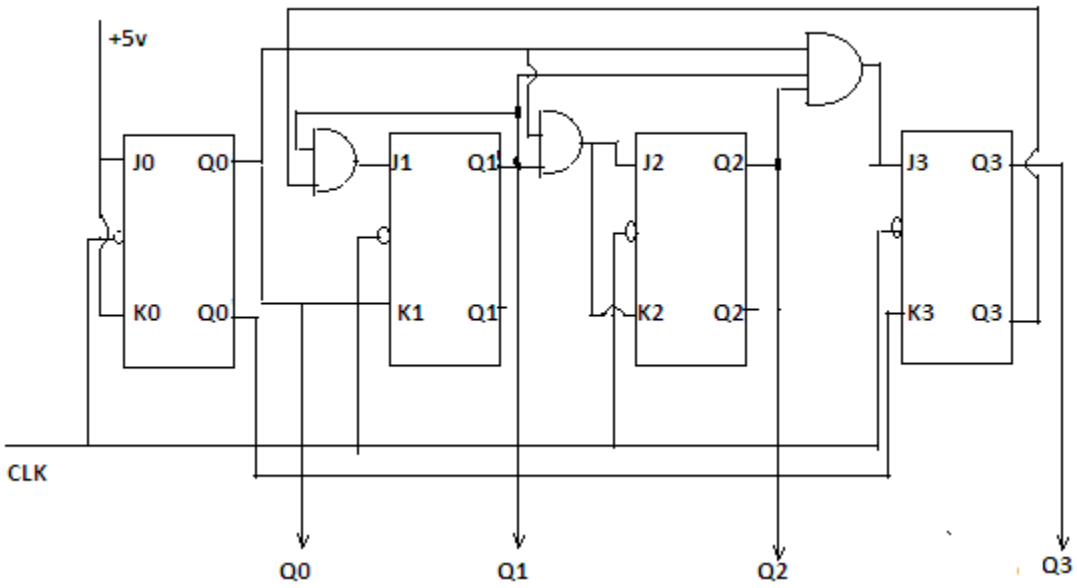
Present State				Next State				INPUTS Excitation Table for JK flip-flops							
Q3	Q2	Q1	Q0	Q3+t	Q2+t	Q1+t	Q0+t	J3	K3	J2	K2	J1	K1	J0	K0
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	x	1
0	0	1	0	0	0	1	1	0	X	0	X	x	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	x	1	x	1
0	1	0	0	0	1	0	1	0	X	x	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	x	0	1	X	x	1
0	1	1	0	0	1	1	1	0	X	x	0	x	0	1	X
0	1	1	1	1	0	0	0	1	X	x	1	x	1	x	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	0	0	0	0	X	1	0	X	0	X	x	1
1	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X

1	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X

The K-Map Solution for J3,K3,J2,K2,J1,K1,J0,K0 is given below:



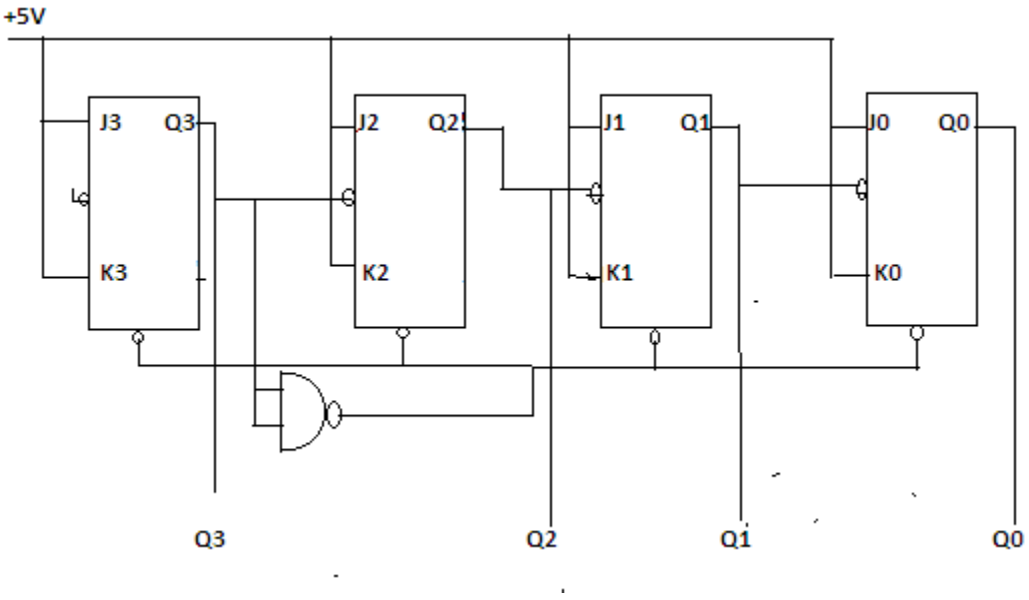
The complete circuit diagram is given below;



(b) Design MOD-8 ripple counter(up-counter) using JK flip-flops

Solution:

- Find the no. of FF required, In the given question of Mod-8 counter design $8=2^3$ i.e. 3 FF, also the count will be from $2^3 - 1 = 7$.
- Then connect all FF as ripple counter
- Find binary no. for N
- Connect all FF output , for which Q=1 when the count is 'N' , as inputs to NAND gate
- Connect the NAND output to clear input of FFs.



UNIT-D

8 (a) List basic type of programmable logic devices.

SOLUTION:

The following are the programmable logic devices:

- PROM: Programmable Read Only Memory
- PAL: Programmable Array Logic
- PLA: Programmable Logic Array
- GAL: Generic Array Logic
- CPLD: Complex Programmable logic Device
- FPGA: Field Programmable Gate Array

(b) Implement the following Boolean expression using ROM

(i) F1(A,B,C)= ∑m(0,2,4,7)

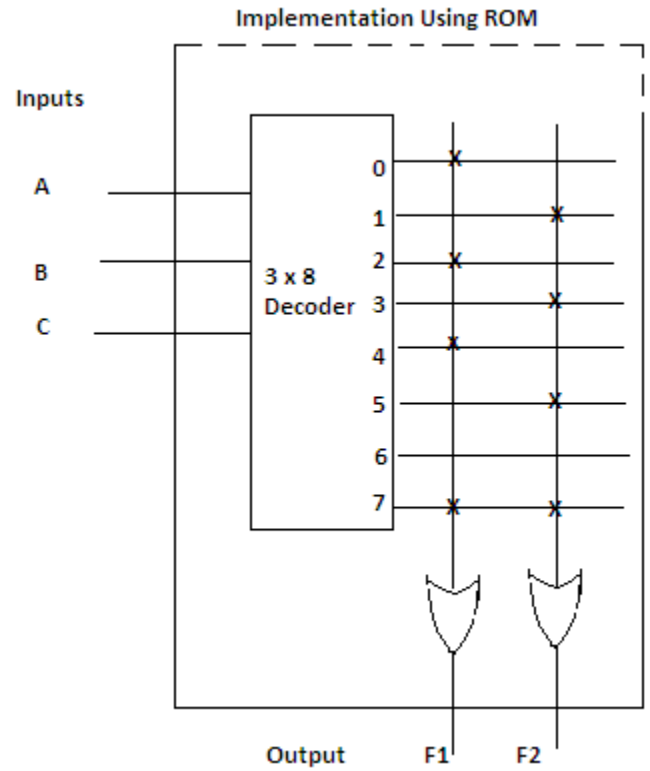
(ii) F2(A,B,C)= ∑m(1,3,5,7)

SOLUTION: The above functions can be implemented using a ROM. As we are aware that a ROM stores the information at certain address. The information may be in single or multibits at a location. The given question can be implemented using a ROM of size 8 x 2-bit ROM means a ROM having 8 locations and at each location 2-bit of information. Let us see the table showing the above functions.

C	B	A	F1(A,B,C)	F2(A,B,C)
0	0	0	1	0
0	0	1	0	1

0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

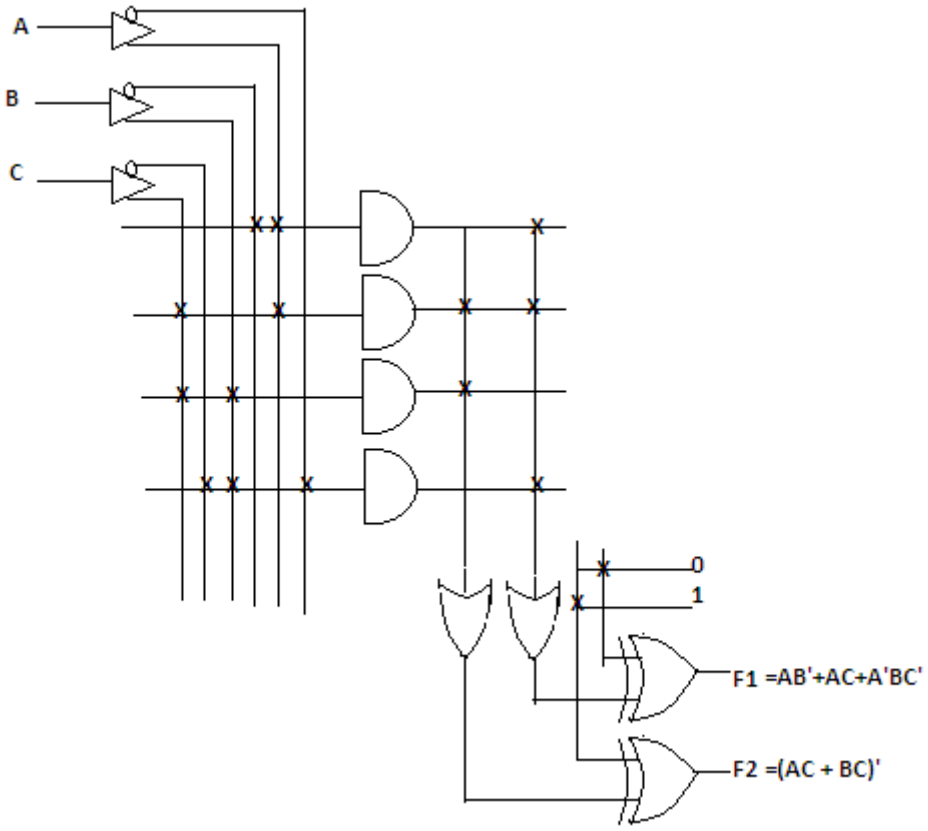
We show this implementation by showing the blocks of black and white cells representing logic '1' and logic '0'



(c) Explain AND-OR structure of PAL and PLA

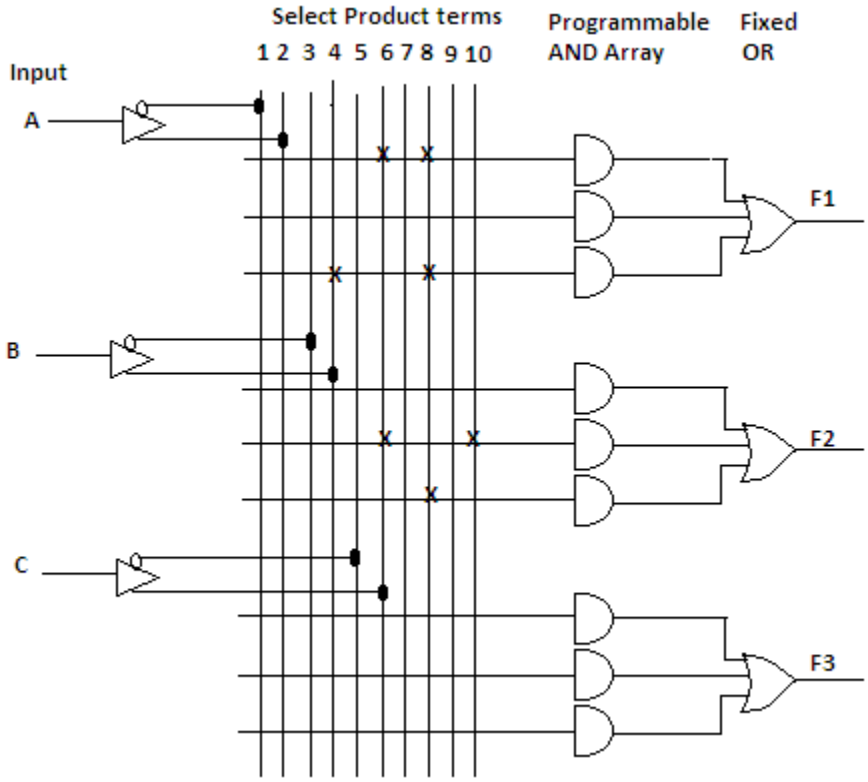
SOLUTION:

PLA structure is similar in concept to the PROM, except that the PLA does not provide full decoding of the variable and does not generate all the minterms. The decoding is replaced by an array of AND gates that can be programmed to generate any product terms of the input variables. The product terms are then connected to OR gates to provide SOP for the required Boolean expression. Figure below shows a PLA



with three inputs, four product terms and two outputs.

PAL: The PAL is is a programmable logic device with a fixed OR array and a programmable AND array. Because only the AND array is programmable, the PAL is easier to program, however it is not as flexible as the PLA. The programmable AND array are arranged in group e.g 3 AND gates and one OR gate etc. as shown in figure below:



9 (a) A combinational logic is defined by the function:

F1(A,B,C) = ∑m(3,5,6,7)

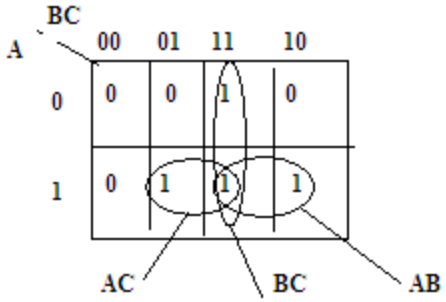
F2(A,B,C) = ∑m(0,2,4,7)

Implement the circuit with PLA having 3 inputs, 4 product terms and two outputs.

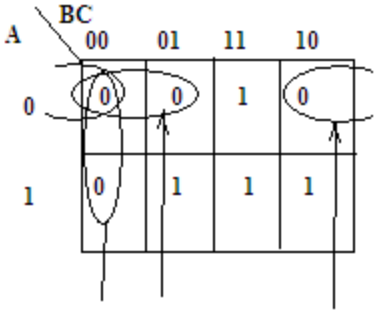
SOLUTION:

In implementing a combinational circuit with a PLA, careful investigation must be undertaken to reduce the number of distinct product terms, since the PLA has finite number of AND gates. This can be done by simplifying each Boolean function to a minimum number of terms. Both the true and complement of each function should be simplified to see which one can be expressed with fewer product terms and which one provides products terms that are common to other functions.

Solving for F1 and F1' :-

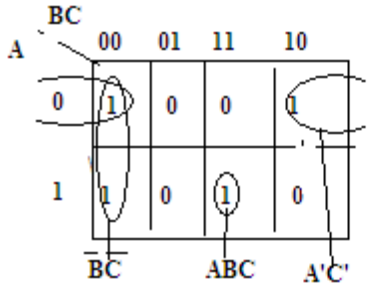


F1 = AC + BC + AB

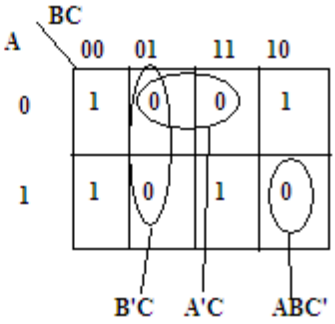


F1' = B'C' + A'B' + A'C'

Simplifying for F2 and F2' : -



F2 = B'C' + ABC + A'C'



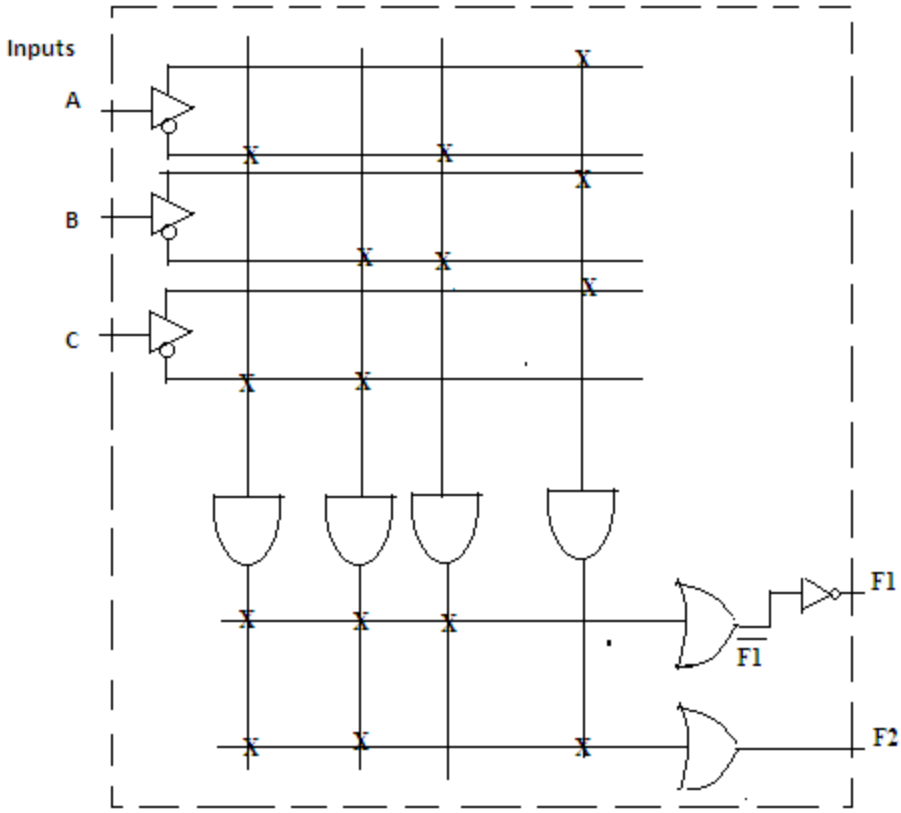
F2' = B'C + A'C + ABC'

Now we compare F1 with F2 and see that there are 6 terms, but the function is to be implemented with four terms.

Now we compare F1' and F2 and see that the terms B'C' and A'C' are common and can be written only once, rest two terms are independent, thus giving a total of four minterms. So we use F1' and F2 for the solution

	A	B	C	F1	F2
A'C'	0	-	0	1	-
B'C'	-	0	0	1	1
A'B'	0	0	-	1	-
ABC	1	1	1	-	1

IMPLEMENTATION:



(b) Write short notes on:

(i) Hazards

SOLUTION:

Hazard in digital systems causes the circuit to malfunction. They are the unwanted switching transients that may appear at the output of a circuit because of different path delays. These path delays could be due to different propagation delays, wiring delays. Therefore, in case of combinational circuits, if the outputs are observed before the signal travel to the final stage through different gates, this delay may cause momentary false output value. In sequential circuit, especially asynchronous circuits, the behavior will be different; hazards in these circuits will cause transition to a wrong stable state. So hazards must be carefully checked and determine whether they will cause improper operation of the circuit. If so, then proper care must be taken to eliminate such hazards in the digital circuits. Accordingly hazards can be categorized as static, dynamic or essential hazards, where static and dynamic hazard are caused by delays in combinational circuits, essential hazards are caused in sequential circuits.

Static hazard: Static hazard is a condition which results in a momentary incorrect output due to change in single input variable. If the output goes to '0' state when it was expected to remain in state '1' as per steady state analysis, hazard is called static-1 hazard. Similarly, If the output goes to '1' state when it was expected to remain in state '0' as per steady state analysis, hazard is called static-0 hazard. This type of hazard can be removed by including the redundant group that covers both (or more) group of 1's.

Dynamic Hazard: When the output is supposed to change from '0' to '1' (or from '1' to '0'), the circuit may through three or more transients and produce one or more glitch. Such multiple glitch situations is called dynamic hazard.

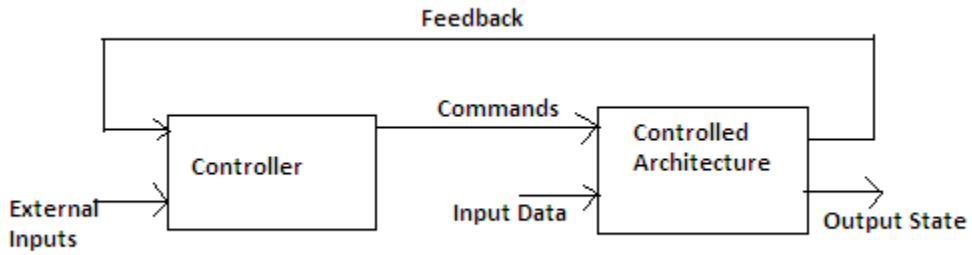
Essential Hazard: it is caused in asynchronous sequential circuits. It is generally an operational error caused by an excessive delay to a feedback variable in response to an input change, leading to a transient to an improper state.

(ii) ASMs

SOLUTION:

A digital system generally handles two types of binary information. This information is classified as data or the control. Data is discrete information that is manipulated by performing arithmetic operation performed by combinational units like adders, multiplier, decoder, muxes etc. the control information is provides command signals that coordinates and executes various operation in data section of machine in order to accomplish the data operation task.

Thus there are basically two types of block in a digital system e.g. controller and the controlled architecture with controller generating the commands and timings and controlled architecture actually performing the task. This is shown in the figure below:



Arrangements of various states in a sequence control the data processor. Hence, the controller can be regarded as a hardware algorithm operating following a finite number of prescribed states. It is referred to as Algorithmic State machine or simply ASM.