QUESTION BANK

DEPARTMENT: CSE **SEMESTER – II**
SUBJECT NAME: DIGITAL PRINCIPLES AND SYSTEM DESIGN
SUBJECT CODE: CS6201

UNIT – I: Boolean Algebra and Logic Gates

PART -A (2 Marks)

1. What are the limitations of Karnaugh map? (AUC MAY 2013)

1. For more than 4 input variables, K-map is complex to solve.
2. Redundant terms cannot be avoided in K-Map.

2. What are don‟t care condition ? (AUC MAY 2013)

In some logic circuits certain conditions never occurs ,therefore the corresponding output never appears. These output levels are indicated by “X” or “d” in the truth tables and are called don‟t care conditions or incompletely specified functions.

3. State Demorgan‟s theorem. (AUC MAY 2012 ,APR 2010)

Theorem 1 : The compliment of a product is equal to the sum of the compliments.

$$\overline{AB} = A' + B'$$

Theorem 2 : The compliment of a sum is equal to the product of the compliments.

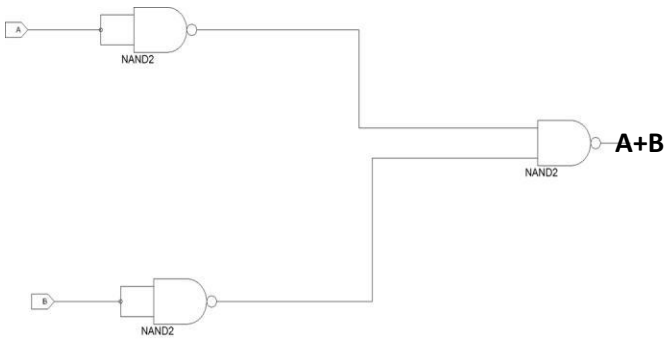
$$\overline{A+B} = A'.B'$$

4. Map the standard SOP expression on a karnaugh map. (AUC NOV 2011) ABC + A‟ B C +ABC‟ +A‟B‟C

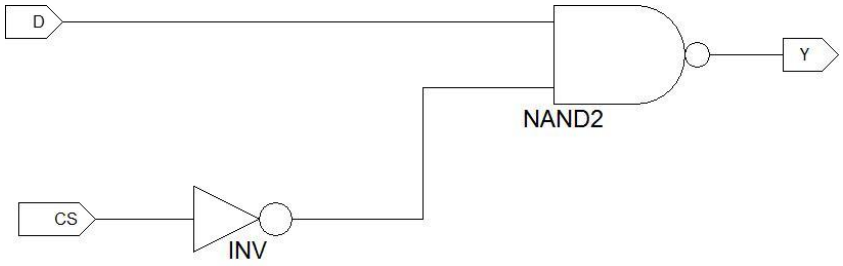
A				
		1	1	
			1	1
A	B‟C‟	B‟C	BC	BC‟

5. Draw the logic diagram of OR gate using universal gates. (AUC NOV 2011)

A



6. Draw an active-high tri-state buffer and write its truth table. (AUC APR 2010)



PUT	CHIP SELECT INPUT	UTPUT
0	0	1
0	0	1
0	0	1
1	0	0
X	1	Z

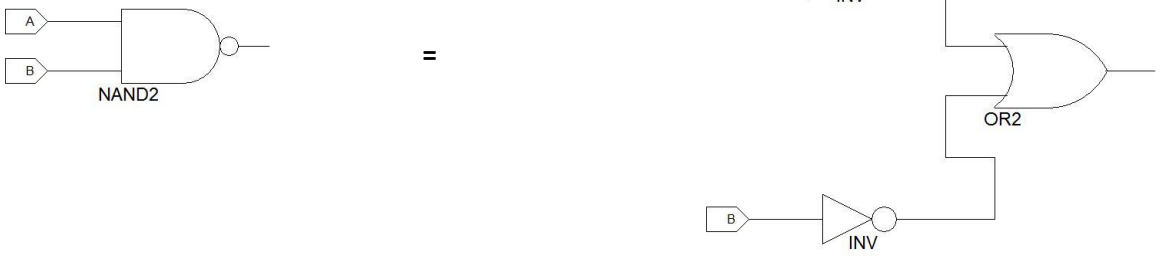
7. **Prove that the logical sum of all minterms of a Boolean function of 2 variables is 1. (AUC NOV 2009)**

Minterms formed by 2 variables = 2^n combinations (2^2 =4)

Let the 2 input variables are x and y and the 4 possible combinations are x‟ y‟ ,x‟y, xy , xy‟.

Sum of all minterms are x‟y‟ +x‟y+ xy + xy‟ = x(y+y‟)+x‟(y+y‟) =x+x‟ =1

8. **Show that a positive logic NAND gate is a negative logic NOR gate. (AUC NOV 2009)**



9. **Simplify the given function : F=A‟BC + AB‟C‟+ABC‟+ABC. (AUC NOV 2008)**

F=BC(A+A‟)+AC‟(B+B‟)

F=BC+AC‟

10. **Using Boolean algebra prove x+x‟y + xy‟ =x + y (AUC NOV 2007)**

x(1+y‟)+x‟y = x+x‟y = x+y

11. **Minimize the function using Boolean algebra f=x(y+w‟z)+wxz (AUC JUNE 2007**)

f=xy+w‟xz+wxz = xy +xz(w+w‟) = xy+xz =x(y+z)

12. **What is tristate logic? What are its demerits? (AUC MAY 2012)**

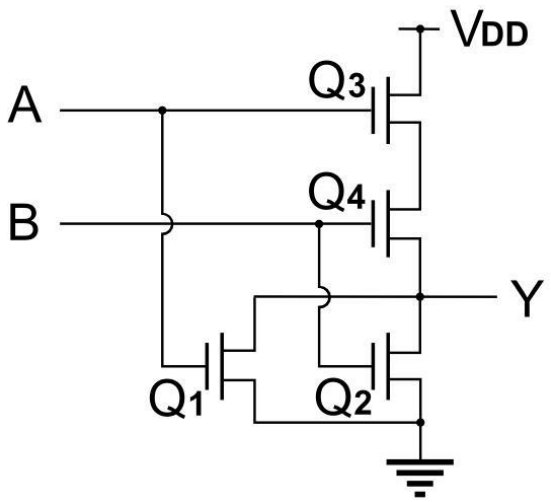
In the tristate logic, in addition to low impedance outputs 0 and 1 there is a third state known as high impedance state. When the gate is disabled it is in the third state.

When the output changes from low to high,a large current is drawn from the supply during the transition. This generates noise spike in the supply system.

13. State the features of bipolar logic families. (AUC MAY 2012)

- Speed of operation
- Power dissipation
- Fan-in
- Fan-out
- Noise margin
- Operating temperature

14. Draw a 2 input CMOS NOR gate. (AUC NOV 2007)



15. Define fanout of a digital IC. (AUC NOV 2007)

It is the maximum number of similar logic gates that a gate can drive without any degradation in its voltage level is called Fan-out.

16. What is the advantage of using schottky TTL gate. (AUC JUNE 2007)

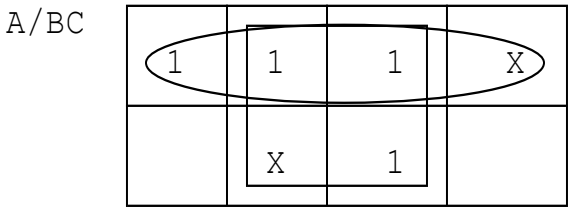
- Low power dissipation
- Higher speed of operation
- Driving capability.

PART –B (16 Marks)

1. **Reduce the following functions using Karnaugh map technique i)f(A,B,C)= ∑m(0,1,3,7)+ ∑d(2,5) (AUC MAY 2013) ii)F(w,x,y,z)= ∑m(0,7,8,9,10,12)+ ∑d(2,5,13)**

i)

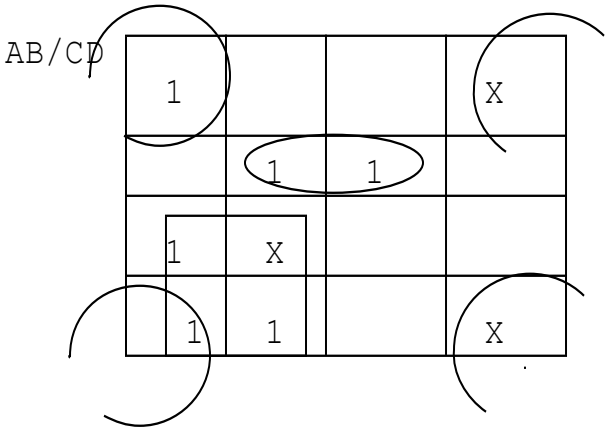
K MAP FOR F



F1=A'C

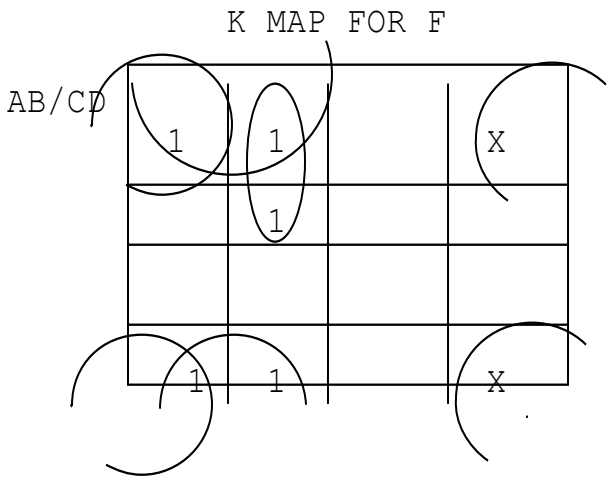
ii)

K MAP FOR F



F=AB'+B'D'+AB'D

2. Simplify the given Boolean function into
i)Sum of products form ii) product of sum form
f(A,B,C,D)=∑m(0,1,2,5,8,9,10)(AUC MAY 2013)



F=B'C'+B'D'+A'C'D

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

B'D'=B'D'(A+A')(C+C') =AB'CD+A'B'CD+AB'C'D+A'B'C'D

AC'D(B+B')=A'BC'D+A'BC'D

SOP : A'B'C'D'+A'B'C'D+A'B'CD+A'BC'D+AB'C'D'+AB'C'D+AB'CD

POS :(A+B+C'+D)(A+B'+C+D)(A+B'+C'+D)
(A+B'+C'+D')(A'+B+C'+D)(A'+B'+C+D)
(A'+B'+C'+D)(A'+B'+C'+D')

3. Simplify the Boolean function using Quine Mc Clusky method f(A,B,C,D)=∑m(0,5,7,8,9,10,11,14,15)(AUCMAY 2013,2012, 2007)

LIST OF MINTERMS

0√	0000	0
8√	1000	1
5√	0101	2
9√	1001	
10√	1010	
11√	1011	3
7√	0111	
14√	1110	
15√	1111	4

0	0000	0
5	0101	2
7	0111	3
8	1000	1
9	1001	2
10	1010	2
11	1011	3
14	1110	3
15	1111	4

0,8*	-000
5,7√	01-1
8,9*	100-
8,10√	10-0
9,11√	10-1
11,15*	1-11
14,15*	111-

8,10,9,11	10--
5,7,9,11	1--1

PRIME IMPLICANTS

	0	5	7	8	9	10	11	14	15
8,10,9,11*				√	√	√	√		
5,7,9,11*		√	√		√		√		
11,15							√		√
14,15*								√	√
0,8*	√			√					
8,9				√	√				
	*	*				*		*	

PRIME IMPLICANTS are 8,10,9,11* , 5,7,9,11* , 14,15* , 0,8* **F= ab'+ad+abc+b'c'd'**

4. Elaborate the basic laws of Boolean alzebra with sample. (AUC MAY 2012) Write the steps for multiplying a logic expression using a karnaugh map.

Description of the Laws and Theorems

Annulment Law - A term AND´ed with a "0" equals 0 or OR´ed with a "1" will equal 1.

A . 0 = 0, A variable AND'ed with 0 is always equal to 0.

A + 1 = 1, A variable OR'ed with 1 is always equal to 1.

Identity Law - A term OR´ed with a "0" or AND´ed with a "1" will always equal that term.

A + 0 = A, A variable OR'ed with 0 is always equal to the variable.

A . 1 = A, A variable AND'ed with 1 is always equal to the variable

Indempotent Law - An input AND´ed with itself or OR´ed with itself is equal to that input.

A + A = A, A variable OR'ed with itself is always equal to the variable.

A . A = A, A variable AND'ed with itself is always equal to the variable.

Complement Law - A term AND´ed with its complement equals "0" and a term OR´ed with its complement equals "1".

A . A = 0, A variable AND'ed with its complement is always equal to 0.

A + A = 1, A variable OR'ed with its complement is always equal to 1.

Commutative Law - The order of application of two separate terms is not important.

A . B = B . A, The order in which two variables are AND'ed makes no difference.

A + B = B + A, The order in which two variables are OR'ed makes no difference.

Double Negation Law - A term that is inverted twice is equal to the original term.

A = A, A double complement of a variable is always equal to the variable.

de Morgan´s Theorem - There are two "de Morgan´s" rules or theorems,

Two separate terms NOR´ed together is the same as the two terms inverted (Complement) and AND´ed for example, A+B = A. B.

Two separate terms NAND´ed together is the same as the two terms inverted (Complement) and OR´ed for example, A.B = A +B.

Distributive Law - This law permits the multiplying or factoring out of an expression.

Absorptive Law - This law enables a reduction in a complicated expression to a simpler one by absorbing like terms.

Associative Law - This law allows the removal of brackets from an expression and regrouping of the variables.

Procedure to solve k –map

Consider the karnaugh map

1 **0**	1 **1**	1 **3**	x **2**
 4	x **5**	1 **7**	 **6**

Consider no of one's adjacent to each other
We can form combinations as octet ,quad, pair and unique
Combining 8 no of 1's is octet
Combining 4 no of 1's is quad
Combining 2 no of 1's is pair
Uncombined form is termed as unique term
X are don't care condition
X will be treated as 1 when it is adjacent to atleast one minterm (1)
Combining two groups with atleast one minterm seems to be free is called overlapping.
Combining the minterms in the corner is called rolling.

5. **Simplify the logic function using Quine –McCluskey method and realize using NAND gates. (AUC NOV 2011,2007)**

   f(A,B,C,D)=∑m(1,3,5,9,10,11)+∑d(6,8)

1	0001	1
3	0011	2
5	0101	2
D6	0110	2
D8	1000	1
9	1001	2
10	1010	2
11	1011	3

1√	0001	1
D8√	1000	
3√	0011	2
5√	0101	
D6	0110	
9√	1001	
10√	1010	
11√	1011	3

1,3√	00-1
1,5*	0-01
1,9*	-110
3,11*	-011
9,11√	10-1
10,11*	101-

1,3,9,11 -0-1

P

RIME IMPLICANTS

	1	3	5	9	10	11
1,3,9,11*	√	√		√		√
1,5*	√		√			
1,9	√			√		
3,11*		√				
10,11*					√	√
			*		*	*

PRIME IMPLICANTS are 1,3,9,11*,1,5*,3,11*,10,11*

F=b'd+ a'c'd+ b'cd+ ab'c

6. Express the Boolean function as POS form

(1)SOP form *D* = (*A'* + *B*) (*B'* + *C*) (4)

D=(A'+B)(B'+C)
A'+B = A'+B+CC =(A'+B+C)(A'+B+C')

B'+C=B+C+AA' =(B'+C+A)(B'+C+A')

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

Minimize the given terms
***πM* (0, 1, 4, 11, 13, 15) + π *d* (5, 7, 8) using Quine-McClusky methods and verify the results using K-map methods. (12) (AUC APR 2010,2007)**

0	0000	0
1	0001	1
4	0100	1
D5	0101	2
D7	0111	3
D8	1000	1
11	1011	3
13	1101	3
15	1111	4

0 √	0000	0
1√	0001	1
4√	0100	
D8√	1000	
D5√	0101	2
D7√	0111	3
D11√	1011	
D13√	1101	
15√	1111	4

0,1√	000-
0,4√	0-00
0,8*	-000
1,5√	0-01
4,5√	010-
5,7√	01-1
5,13√	-101
7,15√	-111
11,15*	1-11
13,15√	11-1

PRIME IMPLICANTS :

(0,8) ,(11,15)

0,1,4,5 *	0-0-
5,7,13,15 *	-1-1
0,4,1,5	0-0-
5,13,7,15	-1-1

PRIME

IMPLICANTS : (0,1,4,5) (5,7,13,15)

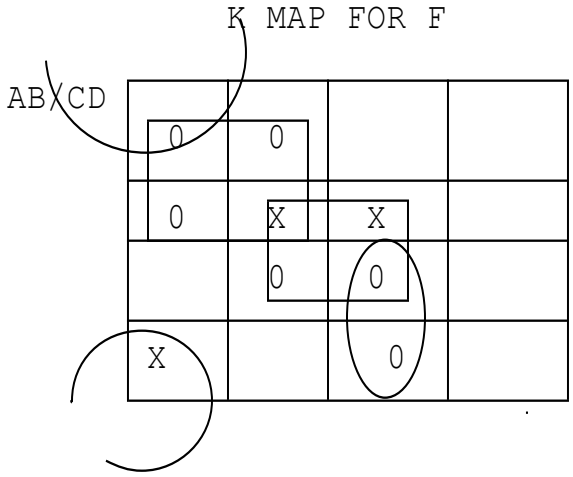
PRIME IMPLICANTS

	0	1	4	5	7	8	11	13	15
(0,8)*	√					√			
(11,15)*							√		√
(0,1,4,5)*	√	√	√	√					
(5,7,13,15)*				√	√			√	√
		*	*		*	*	*	*	

PRIME IMPLICANTS are (0,8)* (11,15)* (0,1,4,5)* (5,7,13,15)*

F=-000+1-11+0-0-+-1-1=B'C'D'+ACD+A'C'+BD
=(B+C+D)(A'+C'+D')(A+C)(B'+D')

K-MAP VERIFICATION



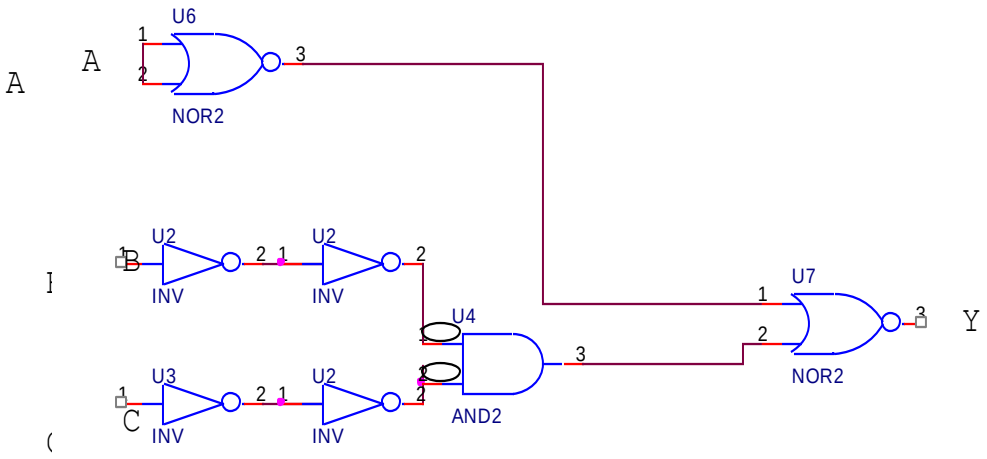
F=(B+C+D)(A'+C'+D')(A+C)(B'+D')

7. i)Implement the following function using NOR gates. (8) (AUC APR 2010)
Output = 1 when the inputs are Σ *m*(0,1,2,3,4) = 0 when the inputs are Σ *m*(5,6,7) .

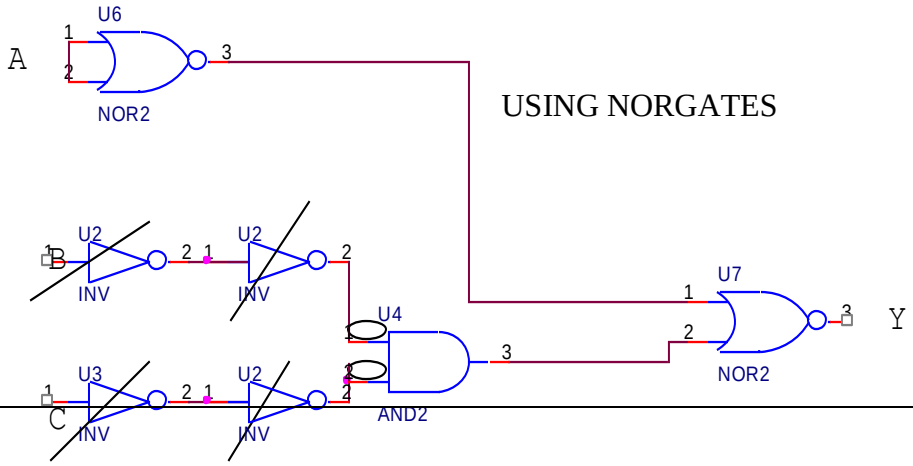
A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

A/B(

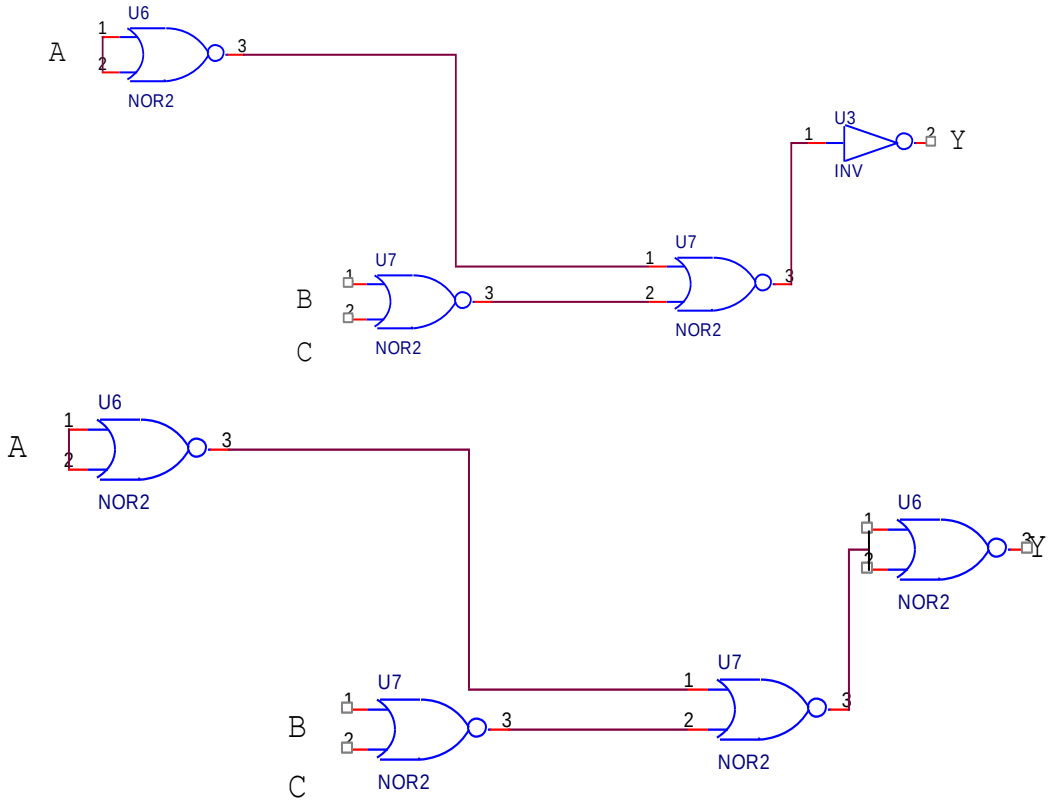
DIAGRAM USING BASIC GATES



DIAGRAM



USING NORGATES



8. Express the Boolean function *F = XY + XZ* in product of Maxterm.(6)

XY+XZ

XY(Z+Z')=XYZ+XYZ'

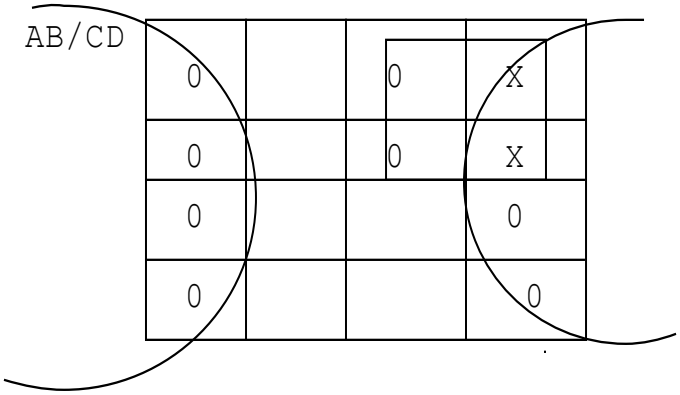
XZ(Y+Y')=XYZ+XY'Z

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

Reduce the following function using K-map technique (AUC NOV 2009)

***f* (*A*, *B*, *C*, *D*) = π (0, 3, 4, 7, 8, 10, 12, 14) + *d* (2, 6) . (10)) (AUC NOV 2009)**

K MAP FOR F



F=D(A+C')

9. **Simplify the following Boolean function by using Quine –Mcclusky method F(A,B,C,D)= Σ(0, 2, 3, 6, 7, 8, 10, 12, 13)(16) (AUC NOV 2009)**

0	0000	0
2	0010	1
3	0011	2
6	0110	2
7	0111	3
8	1000	1
10	1010	2
12	1100	2
13	1101	3

0 √	0000	0
2√	0010	1
8√	1000	
3√	0011	2
6√	0110	
10√	1010	
12√	1100	
7√	0111	3
13√	1101	

0,2√	00-0
0,8√	-000
2,3√	001-
2,6√	0-10
2,10√	-010
8,10√	10-0
8,12*	1-00
3,7√	0-11
6,7√	011-
12,13*	110-

PRIME IMPLICANTS :(8,10),(12,13)

2,3,6,7*	0-1-
0,2,8,10*	-0-0
2,6,3,7	0-1-
0,8,2,10	-0-0

PRIME IMPLICANTS : (2,3,6,7*) (0,2,8,10*)
PRIME IMPLICANTS

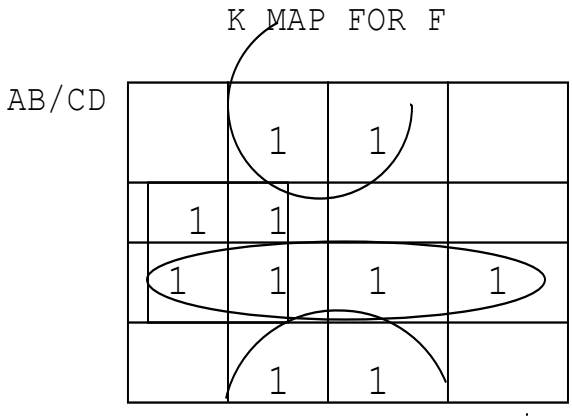
	0	2	3	6	7	8	10	12	13
(2,3,6,7)*		√	√	√	√				
(0,2,8,10)*	√	√				√	√		
(8,10)						√	√		
(12,13)*								√	√
	*		*	*	*			*	*

PRIME IMPLICANTS are
(2,3,6,7)*
(0,2,8,10)*
(12,13)*

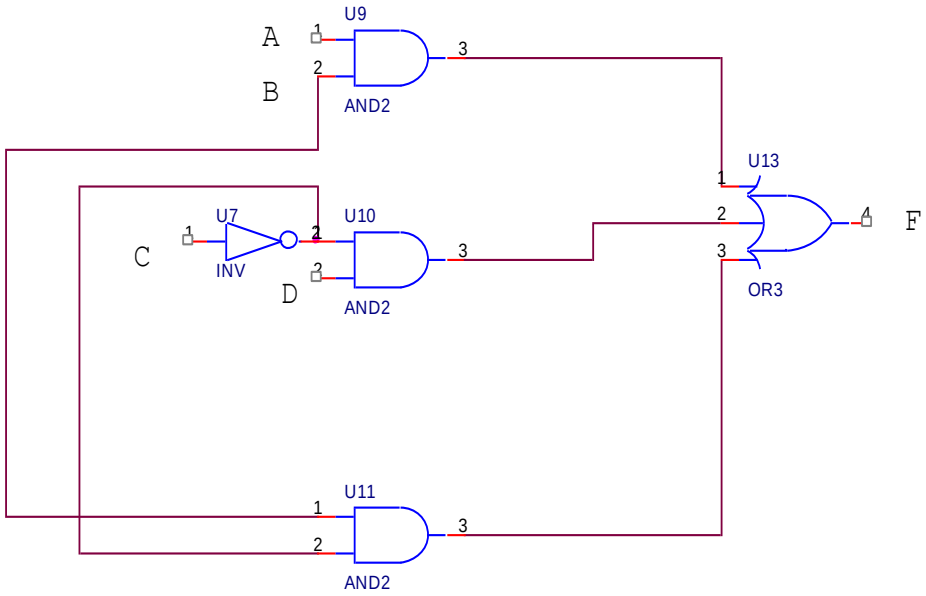
F=0-1-+110-+-0-0 =A'C+ABC'+B'D'

10. Simplify the given function using K map F=∑m(1,3,4,5,9,11,12,13,14,15) (AUC NOV 2008,2007)

List all the prime implicants and draw the logic diagram for the minimized expression using gates (4) (AUC NOV 2008)



F=AB+C'D+BC'



QUESTION BANK

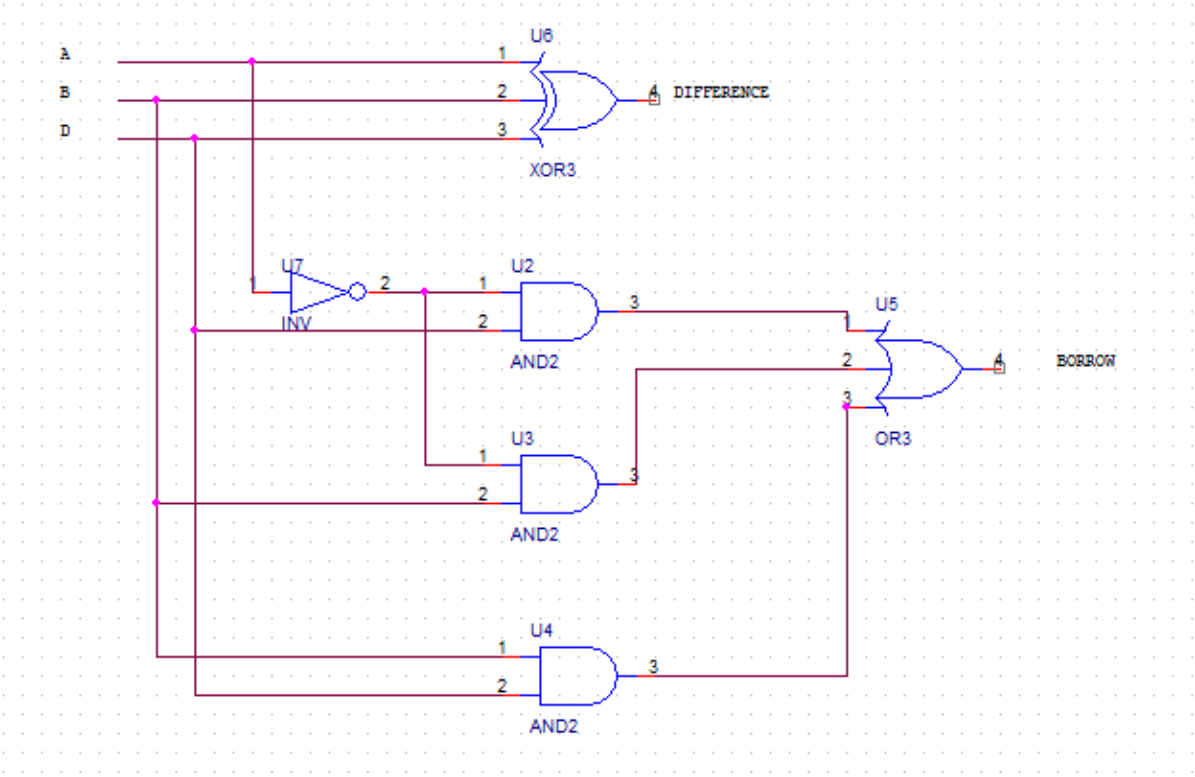
DEPARTMENT: CSE **SEMESTER – III**

SUBJECT NAME: DIGITAL PRINCIPLES AND SYSTEM DESIGN
SUBJECT CODE: CS6201

UNIT 2 : Design of Combinational Circuits

PART -A (2 Marks)

1. Design a half subtractor. **(AUC MAY 2013)**



2. **Write down the truth table of full subtractor**. **(AUC MAY 2012 , 2013)**

A	B	D	DIFFERENCE	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

3. **What is meant by look ahead carry** ?

(AUC NOV 2011,NOV 2008)

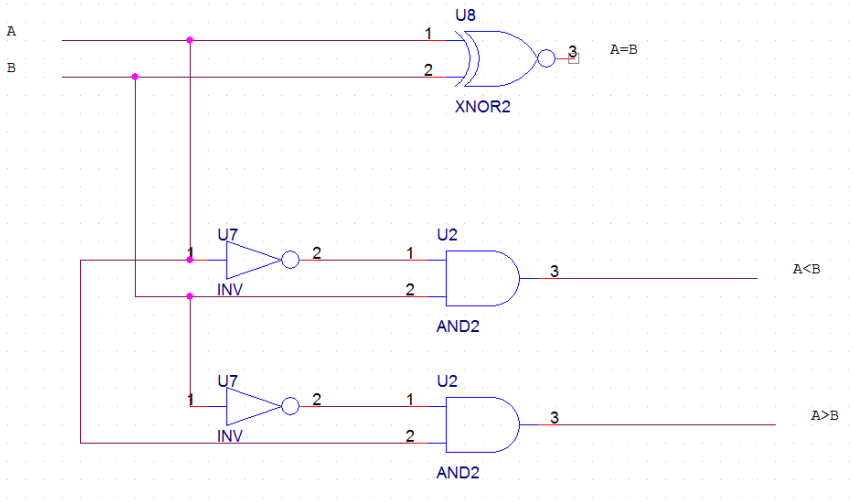
In parallel addition time delay occurs,the process of overcoming this time delay is called look ahead carry.

4. **Give the logical expression for sum output and carry output of a full adder**.

(AUC NOV 2011)

Sum = A ⊕ B ⊕ C Carry =AB+AC+BC

5. **Design a single bit magnitude comparator to compare two words A and B.** (**AUC APR 2011**)



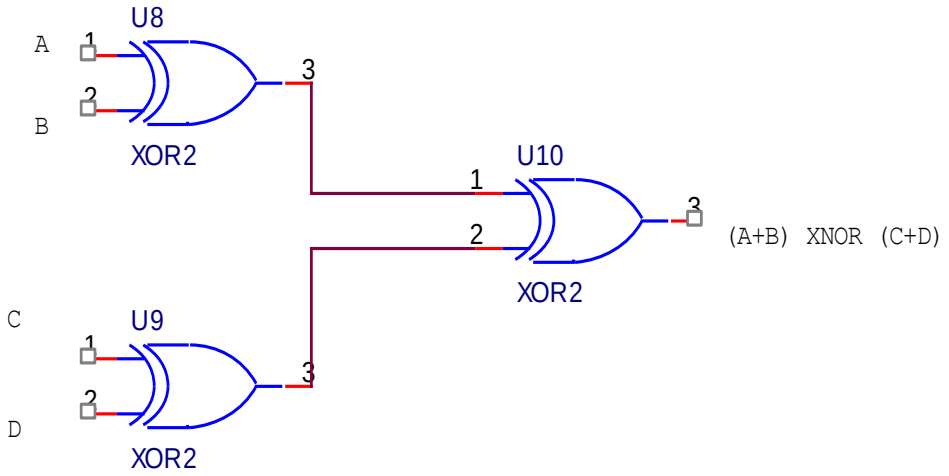
A	B	A=B	A>B	A<B
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

6. **Write an expression for borrow and difference in a full subtractor circuit**.

(AUC APR 2010)

Difference = A'B'D+A'BD'+AB'D'+ABD
Borrow = A ⊕ B ⊕ D

7. **Draw the circuits diagram for 4 bit Odd parity generator**. **(AUC APR 2010, APR 2007)**



8. **Suggest a solution to overcome the limitation on the speed of an adder**.

(AUC NOV 2009)

By eliminating the interstage carry delay we can increase the speed of addition.

9. **Differentiate a decoder from a demultiplexer**. **(AUC NOV 2009)**

DECODER	DEMULTIPLEXER
A decoder is a multiple –input ,multiple-output logic circuit which converts coded input into coded outputs, where the input and output codes are different.	A demultiplexer is a circuit that receives information on a single line and transmit this information on one of the 2^n possible output lines.

10. **Define combinational logic** .

When logic gates are connected together to produce a specified output for certain specified combinations of input variables, with no storage involved, the resulting circuit is called combinational logic.

11. **Define Decoder?**

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

12. **What is binary decoder?**

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2n out puts lines.

13. **Define Encoder?**

An encoder has 2n input lines and n output lines. In encoder the output lines generate the
binary code corresponding to the input value.

14. **What is priority Encoder?**
A priority encoder is an encoder circuit that includes the priority function. In priority encoder, if 2 or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.

15. **Define multiplexer?**
Multiplexer is a digital switch. If allows digital information from several sources to be routed onto a single output line.

16. **What do you mean by comparator?**
A comparator is a special combinational circuit designed primarily to compare the relative magnitude of two binary numbers.

17. **What is propagation delay?**
Propagation delay is the average transition delay time for the signal to propogate form input to output when the signals changes its value. It is expressed in ms.

Part –B (16 Marks)

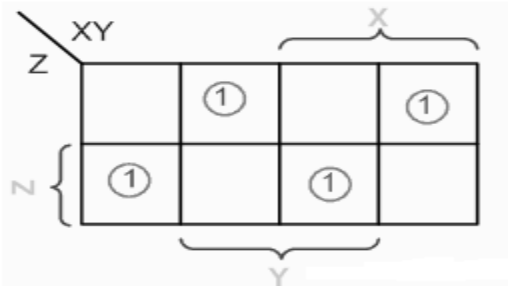
1. **Design a full adder using two half adders**. **(AUC MAY 2013)**
Full adder takes a three-bits input. Adding two single-bit binary values X, Y with a carry input bit C-in produces a sum bit S and a carry out C-out bit.

Truth Table

A	B	CIN	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

SUM (X, Y, Z) = Σ(1, 2, 4, 7) CARRY (X, Y, Z) = Σ(3, 5, 6, 7)

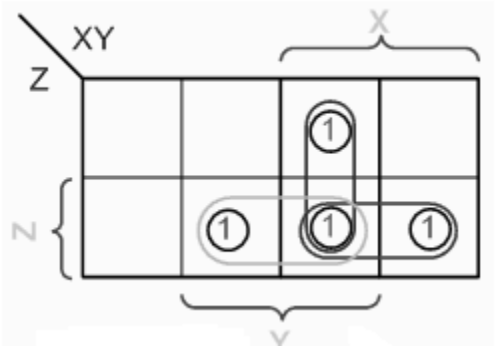
Kmap-SUM



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

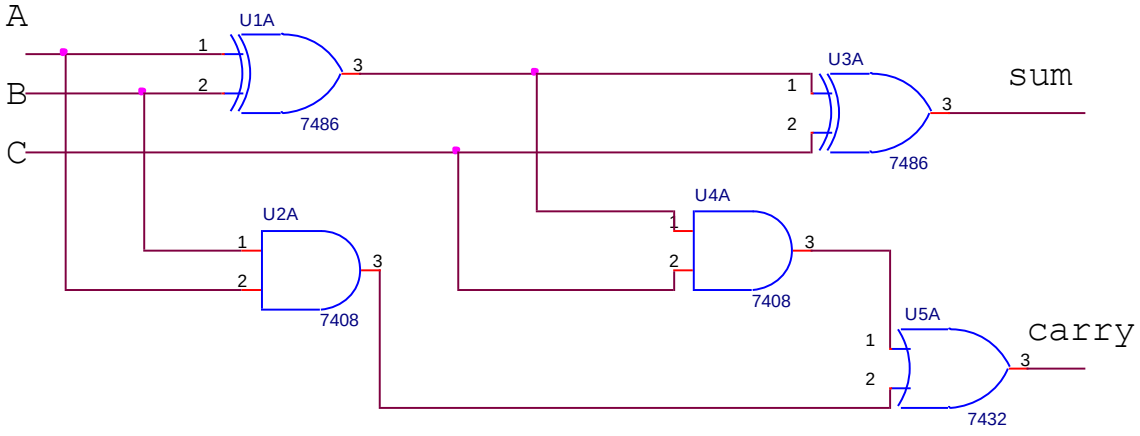
SUM = X ⊕Y ⊕Z

Kmap -CARRY



CARRY = XY + XZ + YZ

Logic diagram:
Full adder circuit using half adders:



SUM= A xor B xor CIN
CARRY = AB+BC+CA

2. Design a 4 bit magnitude comparator and draw the circuit. **(AUC MAY 2013)**

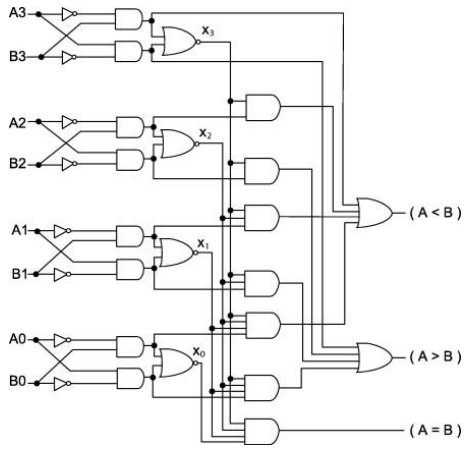
A magnitude comparator is a combinational circuit that compares the magnitude of two numbers (A and B) and generates one of the following outputs: A=B, A>B, A<B.

GROUP	A3	A2	A1	A0	B3	B2	B1	Bo	Condition
I	1	x	x	x	0	x	x	x	A>B
	0	x	x	x	1	x	x	x	A<B
II	1	1	x	x	1	0	x	X	A>B
	1	0	x	x	1	1	x	X	A<B
	0	1	x	x	0	0	x	X	A>B
	0	0	x	x	0	1	x	x	A<B
III	1	1	1	x	1	1	0	x	A>B
	1	1	0	x	1	1	1	X	A<B
	0	0	1	x	0	0	0	X	A>B
	0	0	0	X	0	0	1	X	A<B
IV	1	1	1	1	1	1	1	0	A>B
	1	1	1	0	1	1	1	1	A<B
	0	0	0	1	0	0	0	0	A>B
	0	0	0	0	0	0	0	1	A<B
V	1	1	1	1	0	0	0	0	A=B
	0	0	0	0	1	1	1	1	A=B

Let

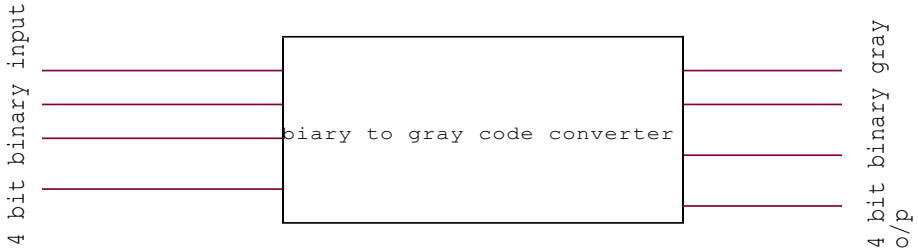
A3 EXNOR B3 =x3
A2 EXNOR B2 =x2
A1 EXNOR B1 =x1
A0 EXNOR B0 =x0
From the table
A>B = A3B3'+X3 A2B2'+X3X2A1B1'+X3X2X1A0B0'
A<B =A3'B3+X3A2'B2+X3X2A1'B1+X3X2X1A0'B0
A=B = X3X2X1X0

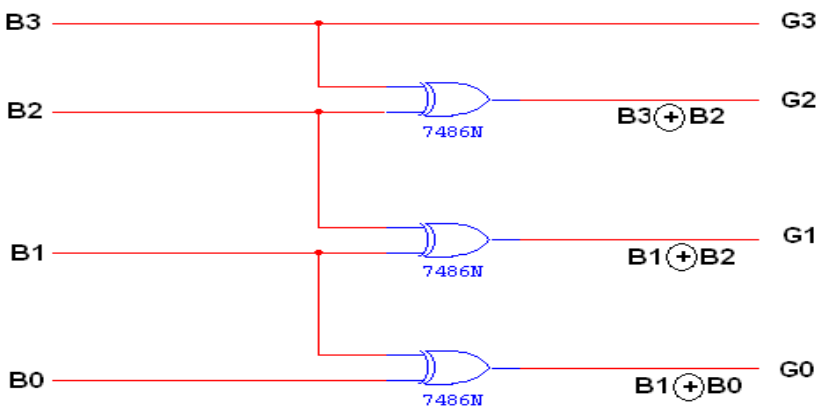
LOGIC DIAGRAM



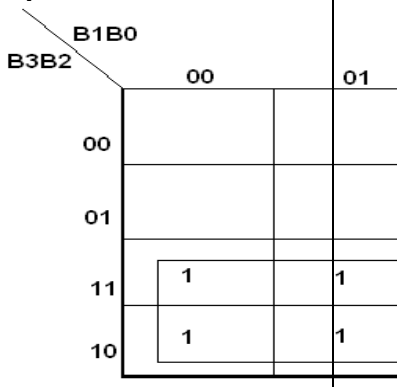
3. **Design a combinational circuit to convert binary to gray code. (AUC MAY 2013,2009)**

BLOCK DIAGRAM



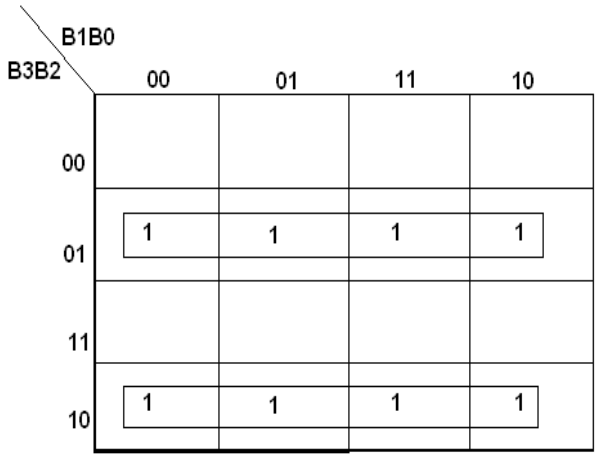


K-Map for G_3:



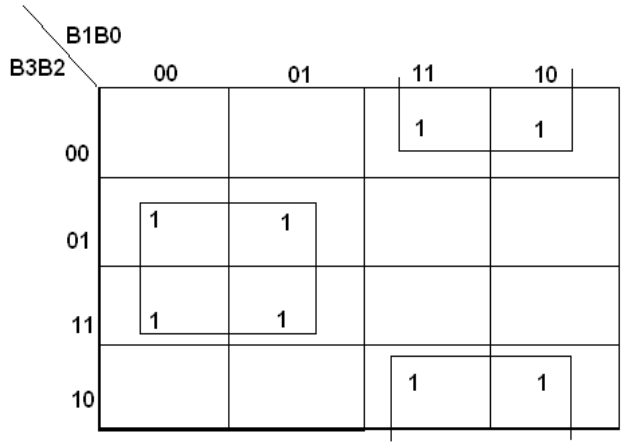
G_3 = B_3

K-Map for G_2:



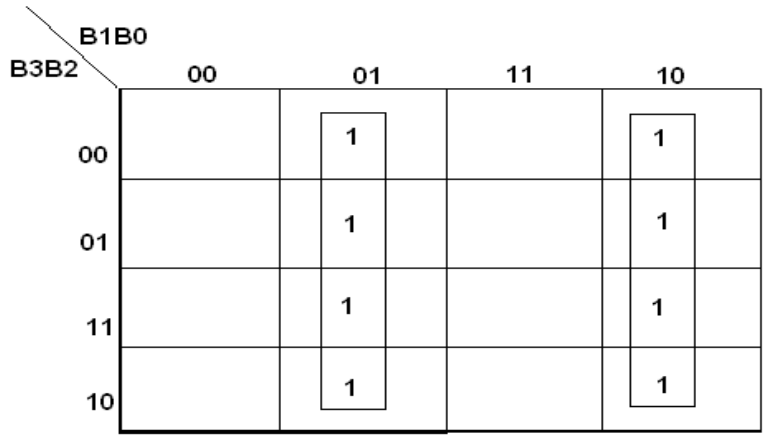
$G2 = B3 \oplus B2$

K-Map for G_1:



$G1 = B1 \oplus B2$

K-Map for G_0:



$G0 = B1 \oplus B0$

TRUTH TABLE:

| Binary input | Gray code output

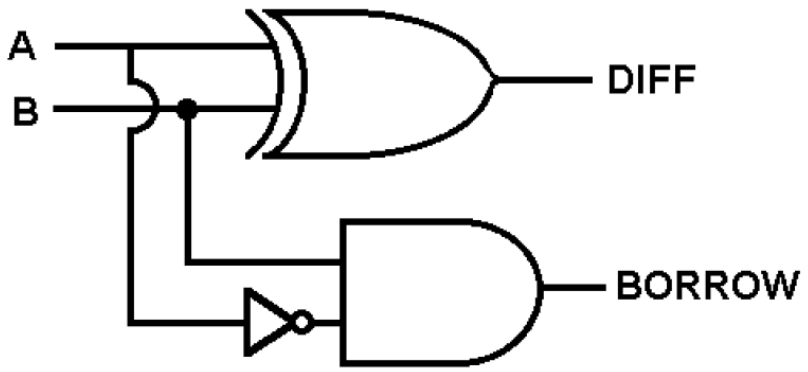
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

4. **Design Half and Full subtractor circuits (AUC MAY 2012)**

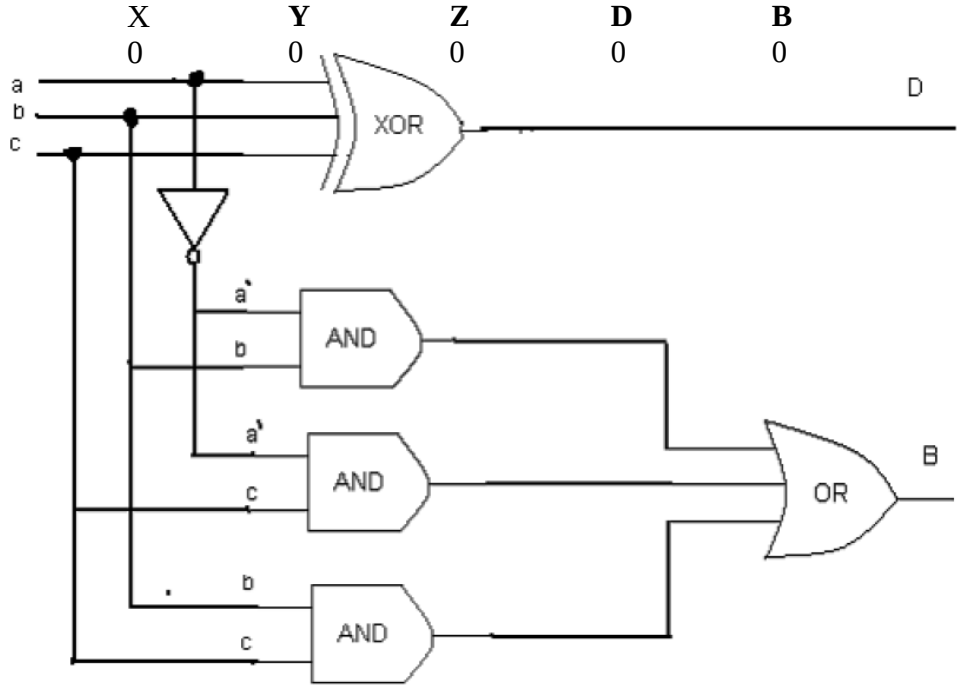
SUBTRACTOR

Half subtractor The half-subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, X (minuend) and Y (subtrahend) and two outputs D (difference) and B (borrow).

X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



The Full_subtractor is a combinational circuit which is used to perform subtraction of three bits. It has three inputs, X (minuend) and Y (subtrahend) and Z (subtrahend) and two outputs D (difference) and B (borrow).



X	Y	Z	D	B
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

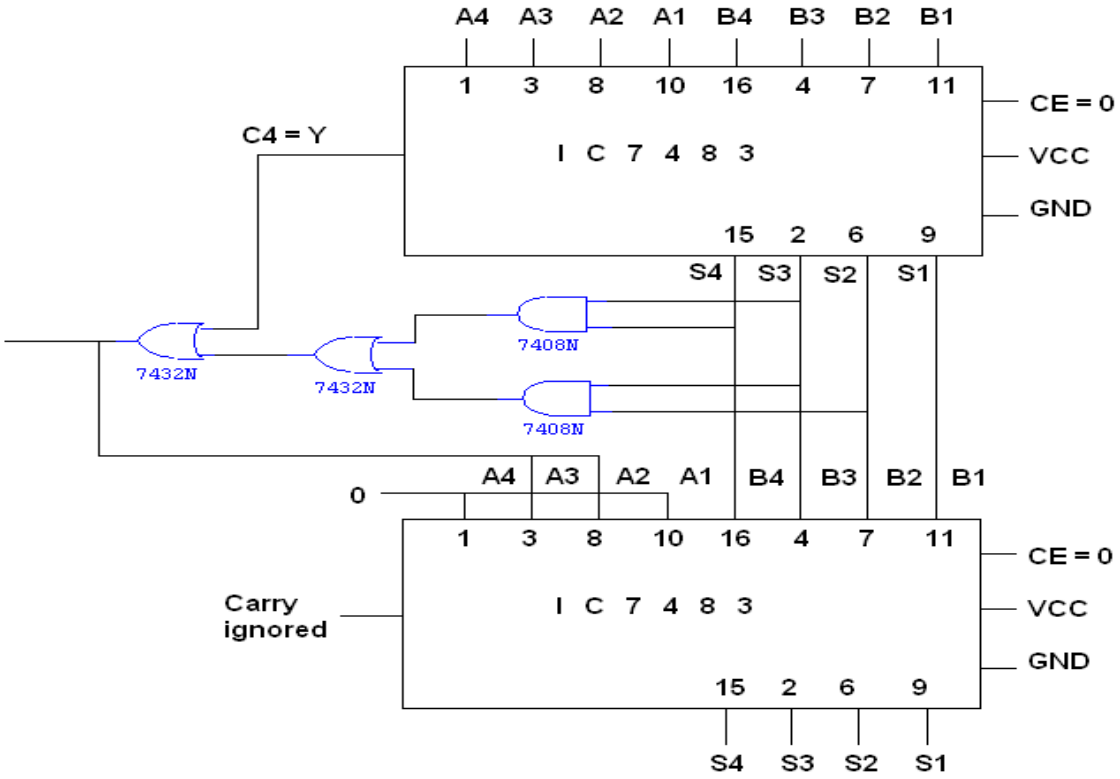
B=A‟ B + A‟ C + BC
C=A xor B xor C

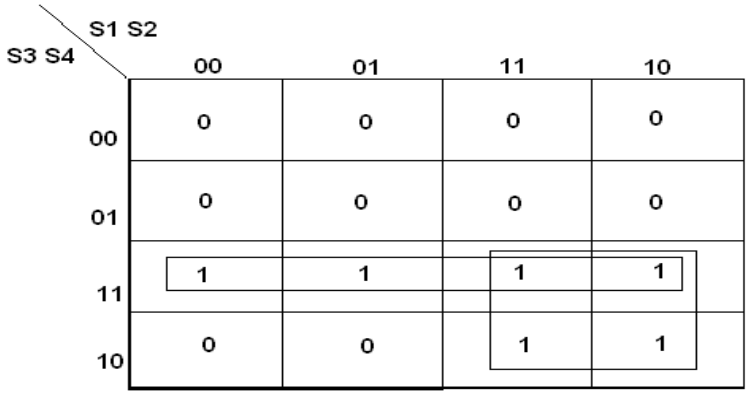
5. **a)Draw the circuit of BCD adder and explain**. **(AUC NOV 2011,2013)**

BCD addition is the same as binary addition with a bit of variation: whenever a sum is greater than 1001, it is not a valid BCD number, so we add 0110 to it, to do the correction. This will produce a carry, which is added to the next BCD position. Add the two 4-bit BCD code inputs.

if the sum of this addition is greater than 1001; if yes, then add 0110 to this sum and generate a carry to the next decimal position.

LOGIC DIAGRAM:





K MAP

Y = S4 (S3 + S2)

TRUTH TABLE:

BCD SUM				CARRY
S4	**S3**	**S2**	**S1**	**C**
0	**0**	**0**	**0**	**0**
0	**0**	**0**	**1**	**0**
0	**0**	**1**	**0**	**0**

0	**0**	**1**	**1**	**0**
0	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**
0	**1**	**1**	**0**	**0**
0	**1**	**1**	**1**	**0**
1	**0**	**0**	**0**	**0**
1	**0**	**0**	**1**	**0**
1	**0**	**1**	**0**	**1**
1	**0**	**1**	**1**	**1**
1	**1**	**0**	**0**	**1**
1	**1**	**0**	**1**	**1**
1	**1**	**1**	**0**	**1**
1	**1**	**1**	**1**	**1**

b) **What is priority encoder? How is it different from encoder? Draw the circuit of 4 bit priority encoder and explain. (AUC NOV 2011)**

If more than two inputs are active simultaneously, the output is unpredictable or rather it is not what we expect it to be. This ambiguity is resolved if priority is established so that only one input is encoded, no matter how many inputs are active at a given point of time. The priority encoder includes a priority function. The operation of the priority encoder is such that if two or more inputs are active at the same time, the input having the highest priority will take precedence.

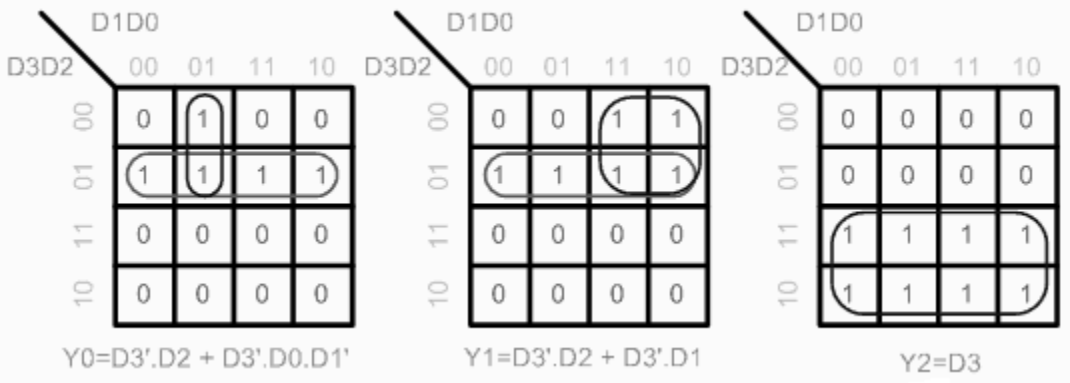
Example - 4to3 Priority Encoder

The truth table of a 4-input priority encoder is as shown below. The input D3 has the highest priority, D2 has next highest priority, and D0 has the lowest priority. This means output Y2 and Y1 are 0 only when none of the inputs D1, D2, D3 are high and only D0 is high. A 4 to 3 encoder consists of four inputs and three outputs, truth table and symbols of which is shown below.

Truth Table

D3	**D2**	**D1**	**D0**	**Y2**	**Y1**	**Y0**
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	x	0	1	0
0	1	x	x	0	1	1
1	x	x	x	1	0	0

Kmaps



6. (i) **Implement full subtractor using demultiplexer**. (10) **(AUC NOV 2009)**

Full Subtracter

A full subtracter is a combinational circuit that performs subtraction involving three bits, namely minuend, subtrahend, and borrow-in. The logic symbol and truth table are shown below.

Truth Table

A	B	Bin	difference	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

Difference = ∑m(1,2,4,7)
Borrow = ∑m(1,2,3,7)

	I0	I1	I2	I3
A’	0	**1**	**2**	3
A	**4**	5	6	**7**
	A	A’	A’	A

I0 ,I3 ------- A I1 ,I2 ----------A’

(ii) Implement the given Boolean function using 8 : 1 multiplexer

F(*A*, *B*, *C*) = Σ(1, 3, 5, 6) . (6) **(AUC NOV 2009)**

	I0	I1	I2	I3
A’	0	**1**	2	**3**
A	4	**5**	**6**	7
	0	1	A	A’

I0 ----------0 I1 ----------1 I2 ---------- A I3----------A’

QUESTION BANK

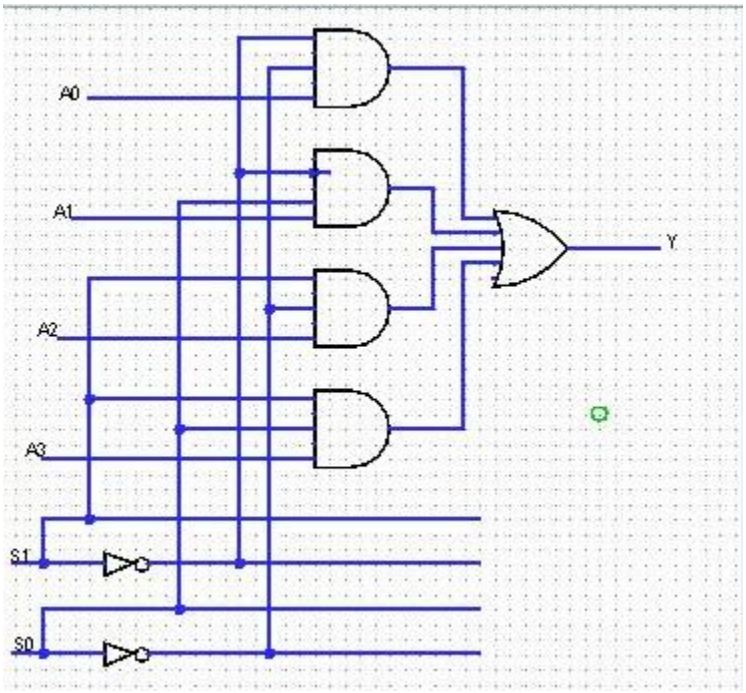
DEPARTMENT: CSE **SEMESTER – III**

SUBJECT NAME: DIGITAL PRINCIPLES AND SYSTEM DESIGN
SUBJECT CODE: CS6201

UNIT 3 : Design of Synchronous Sequential Circuits

PART –A (2 Marks)

1. **Draw the logic diagram 4:1 Multiplexer.(AUC MAY 2013)**



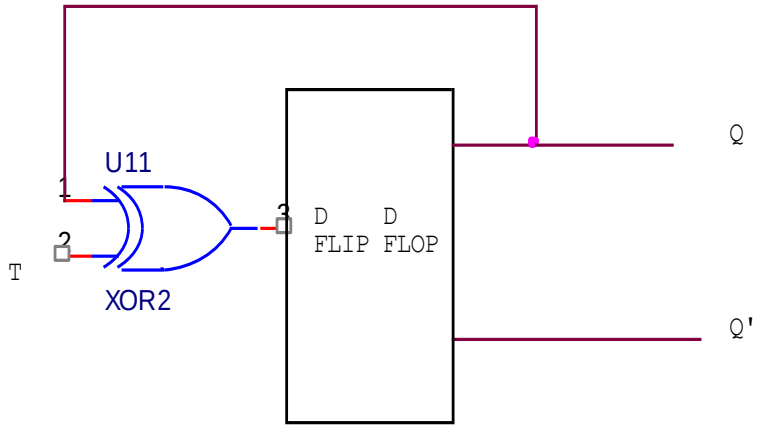
2. **Convert D flip flop to T flip flop. (AUC MAY 2013)**

Required flip flop (T)			Given flip flop (D)
Q (n)	T	Q(n+1)	Input (D)
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

Truth table

Q(n)	0	1	T
	1	0	

D= T ⊕ Q(n)



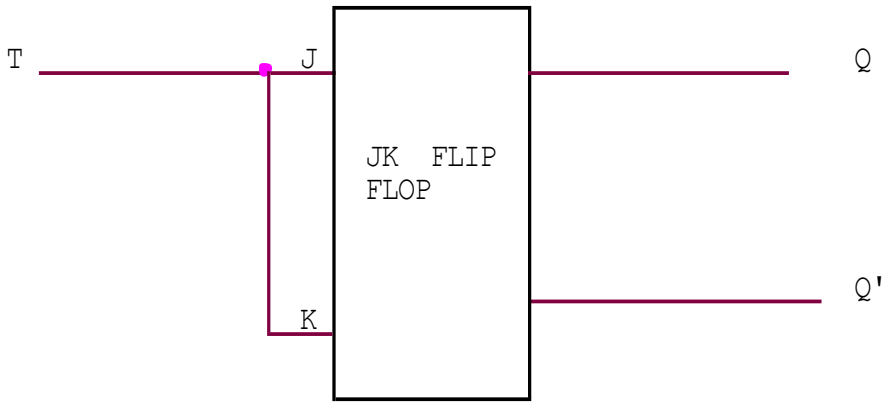
3. **How many flip flops are required to build a binary counter that counts from 0 to 1023.(AUC MAY 2013)**
   $2^n >= N$ $2^n >= 1023$ Number of flip flops required =10

4. **Distinguish between Decoder and demultiplexer.(AUC MAY 2013)**

DECODER	DEMULTIPLEXER
A decoder is a multiple –input ,multiple-output logic circuit which converts coded input into coded outputs, where the input and output codes are different.	A demultiplexer is a circuit that receives information on a single line and transmita this information on one of the 2^n possible output lines.

5. **State the principle of parity checker. (AUC MAY 2012)**
   A parity bit is an extra bit included with binary message to make number of 1's either odd or even. The parity checker produces an error if the transmitted bit does not correspond with the received one.

6. **Draw the logic diagram of T flip flop using JK flip flop. (AUC NOV 2011)**



7. **How can a SIPO register used as a SISO register? (AUC NOV 2011)**
   By using universal shift registers we can realize SIPO register as SISO register

8. **Mention any two differences between the edge triggering and level triggering. (AUCAPR 2010)**

Level Triggering	Edge triggering
Flip flop is enabled either by high or low level.	Flip flop is enabled by positive or negative edge of clock pulse.
Flip flop changes its state either at positive level or negative level.	Flip flop changes its state either at positive or negative edge.

9. **What is meant by programmable counter? Mention its application.(AUC APR 2010)**
   Synchronous counters are preloaded with binary numbers in parallel in prior to count operation. The preloading capability makes it possible to count sequence from 0 to any other number. These counters are called as programmable counters.

10. **Write down the characteristic equation for JK flipflop. (AUC NOV 2009,2013)**
    Q(t+1)=JQ'+K'Q

11. **Implement the function f=∑m(0,1,4,5,7) using 8:1 multiplexer. (AUC NOV 2007)**

	D0	D1	D2	D3
X	**0**	**1**	2	3
X'	**4**	**5**	6	**7**
	1	1	0	X'

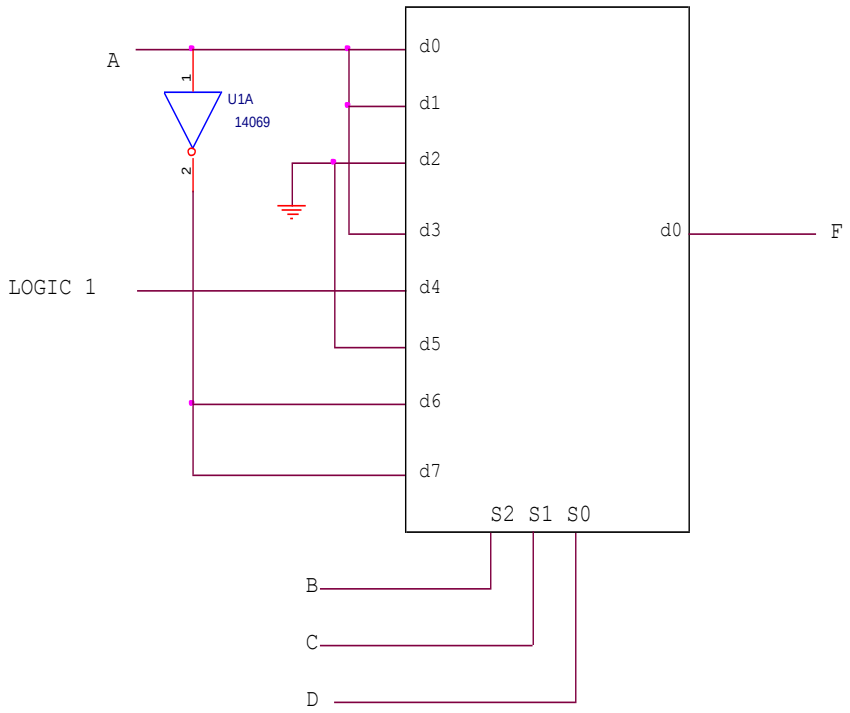
12. **What is VHDL**?
    VHDL is a hardware description language that can be used to model a digital system at many level of abstraction, ranging from the algorithmic level to the gate level. The VHDL language as a combination of the following language. a. Sequential language b. Concurrent language c. Net-list language d. Timing specification e. Waveform generation language. 93. What are the features of VHDL? The features of VHDL are a. VHDL has powerful constructs. b. VHDL supports design library. c. The language is not case sensitive.
13. **Define entity?**
    Entity gives the specification of input/output signals to external circuitry. An entity is modeled using an entity declaration and at least one architecture body. Entity gives interfacing between device and others peripherals.
14. **List out the different elements of entity declaration**?
    The different elements of entity declaration are: 1. entity_name 2. signal_name 3. mode 4. in: 5. out: 6. input 7. buffer 8. signal_type
15. **Give the syntax of entity declaration**?
    ENTITY entity_name is PORT (signal_name: mode signal_type;signal_names: mode signal_type; : : signal_names: mode signal_type;
    END entity_name;
16. **What do you meant by concurrent statement?**
    Architecture contains only concurrent statements. It specifies behavior, functionality, interconnections or relationship between inputs and outputs.
17. **What are operates used in VDHL language?**
    There are different types of operators used in VHDL language Logical operators : AND, OR, NOT, XOR, etc., Relational operator : equal to, <less than etc., Shift operators : SLL- Shift Left Logical,
    ROR- Rotate Right Logical etc., Arithmetic operators: Addition, subtraction etc., Miscellaneous operators: <= assign to etc.,
18. **Define VHDL package?**
    A VHDL, package is a file containing definitions of objects which can be used in other programs. A package may include objects such as signals, type, constant, function, procedure and component declarations.

PART –B (16 Marks)

1. **Implement the switching function F= ∑m(0,1,3,4,12,14,15) using an 8 input MUX.(AUC MAY 2013)**

	D0	D1	D2	D3	D4	D5	D6	D7
A	**0**	**1**	2	**3**	**4**	5	6	7
A'	8	9	10	11	**12**	13	**14**	**15**
	A	A	0	A	**1**	0	**A'**	**A'**

Number of variables **N=4** ; Control lines M =N-1=4-1=3
Type of multiplexer =2^M :1 =2^3 :1 =8:1 multiplexer



2. **Using D flip flops, design a synchronous counter which counts in the sequence 000,001,010,011,100,101,110,111.(AUC MAY 2013)**

PRESENT STATE	NEXT STATE
000	**001**
001	**010**
010	**011**
011	**100**
100	**101**
101	**110**
110	**111**
111	**XXX**

Q2	Q1	Q0	Q2+	Q1+	Q0+	D2	D1	D0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	X	X	X	X	X	X

D2

Q2/Q1Q0

		1	
1	1	X	1

D2=Q2+Q1Q0

D1

Q2/Q1Q0

	1		1
	1	X	1

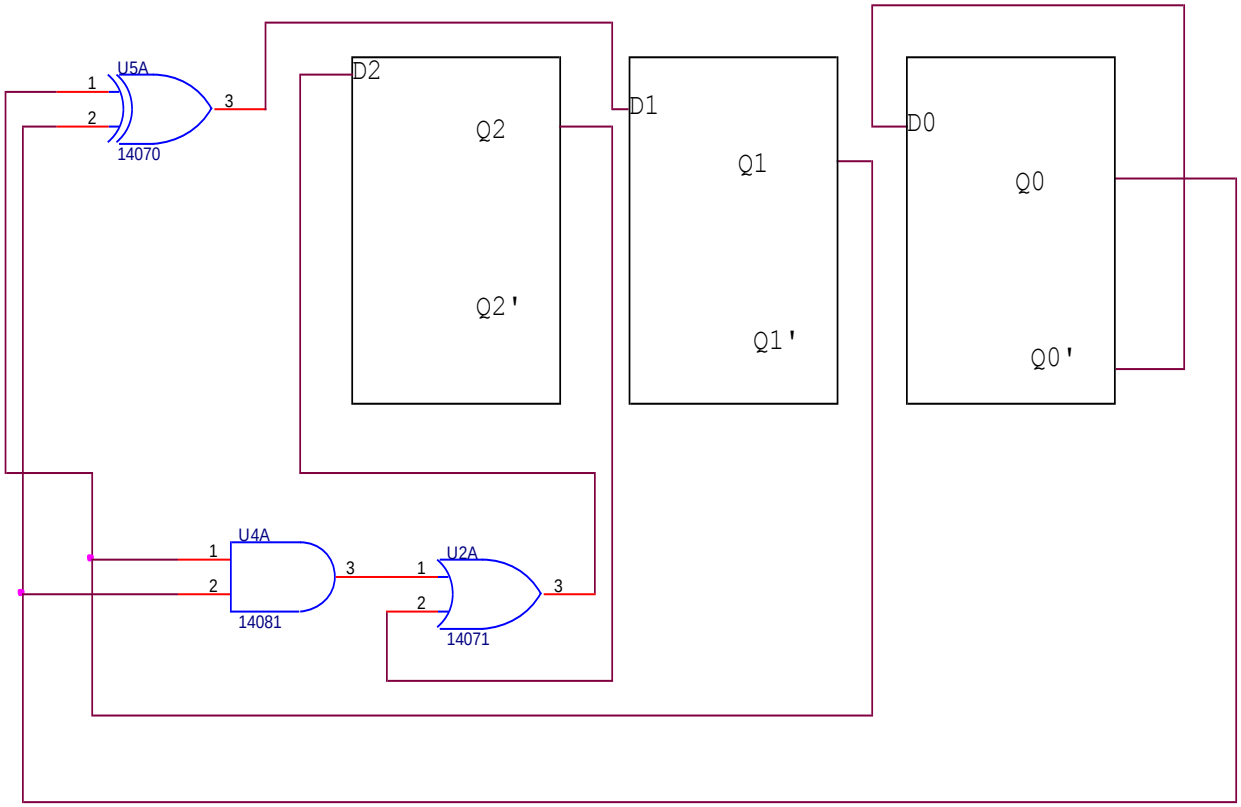
D2=Q1'Q0+Q1Q0'=Q1 ⊕ Q0

D0

Q2/Q1Q0

1			1
1		X	1

D2=Q0'



3. **Design a counter to count the sequence 0,1,2,4,5,6 using SR FFs (AUC MAY 2013, 2008)**

PRESENT STATE	NEXT STATE
000	001
001	010
010	100
100	101
101	110
110	000

Q2	Q1	Q0	Q2+	Q1+	Q0+	S2 R2	S1 R1	S0 R0
0	0	0	0	0	1	0 x	0 x	1 0
0	0	1	0	1	0	0 x	1 0	0 1
0	1	0	1	0	0	1 0	0 1	0 x
1	0	0	1	0	1	X 0	0 x	1 0
1	0	1	1	1	0	X 0	1 0	0 1
1	1	0	0	0	0	0 1	0 1	0 x

S2

Q2/Q1Q0

		X	1
X	X	X	

D2=Q2'Q1

R2

Q2/Q1Q0

X	X	X	
		X	1

D2=Q2Q1

S1

Q2/Q1Q0

	1	X	
	1		

S1=Q1Q0'

R1

Q2/Q1Q0

X		X	1
X	1	X	1

R1=Q2+Q1

S0

Q2/Q1Q0

1		X	
1		X	

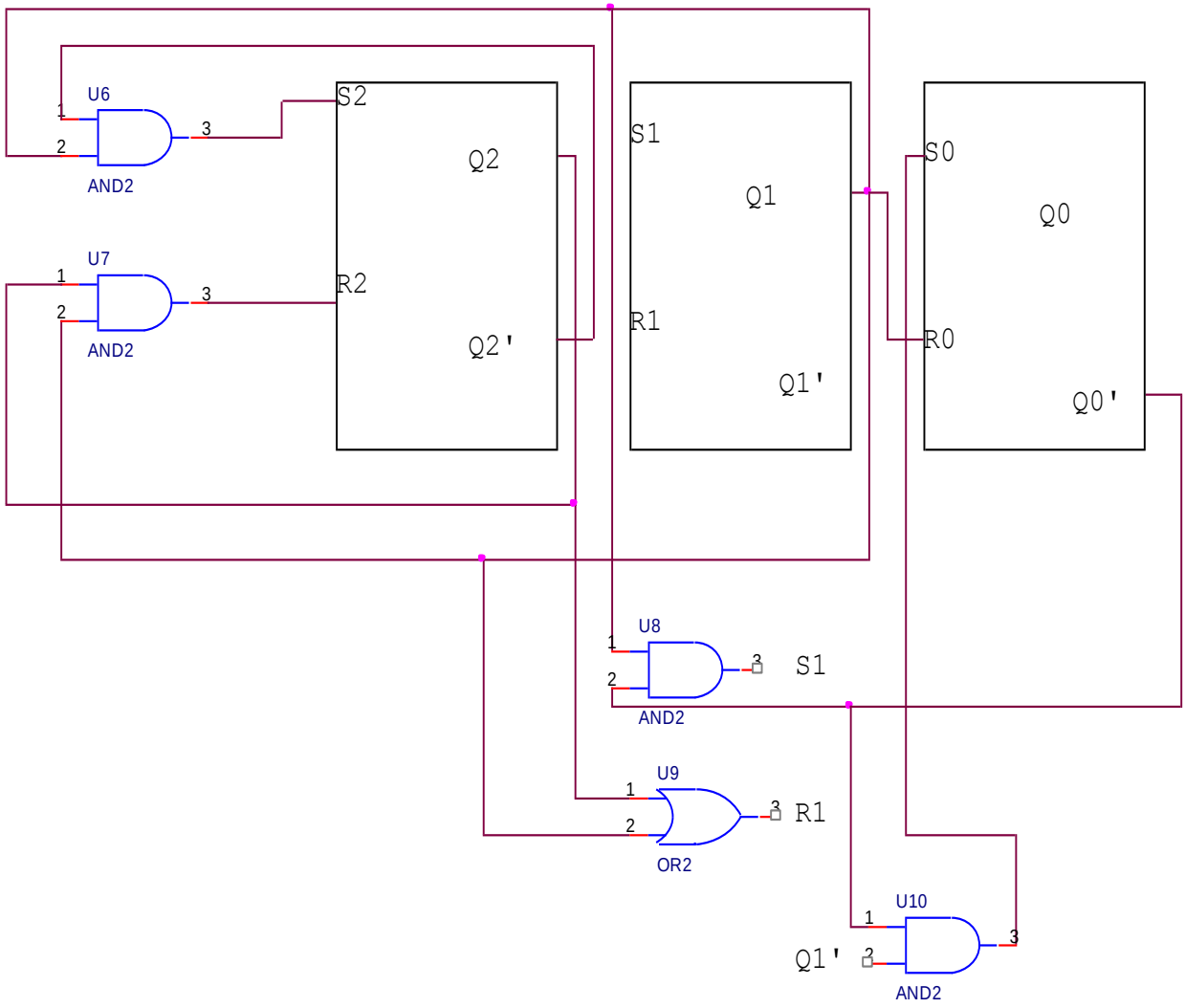
S0=Q1'Q0'

R0

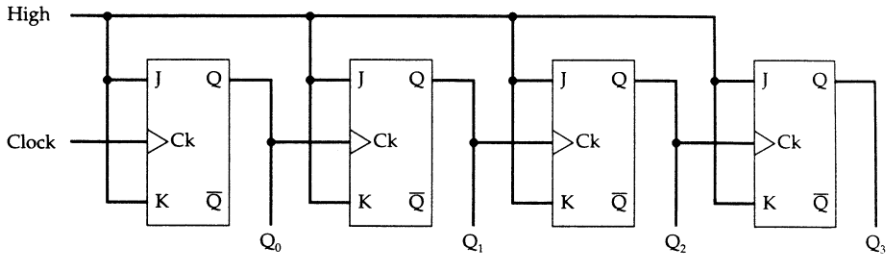
Q2/Q1Q0

	1	X	X
	1	X	

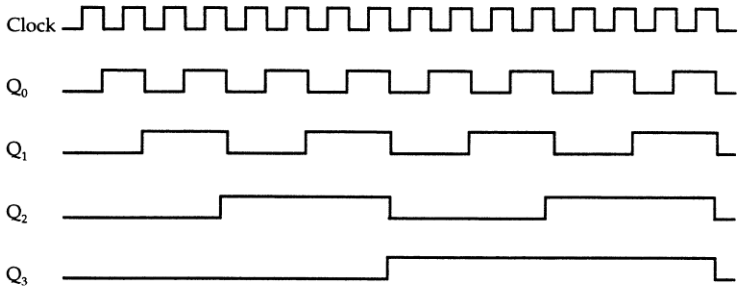
R0=Q1


U6
AND2
S2
Q2
U7
AND2
R2
Q2'
S1
Q1
R1
Q1'
S0
Q0
R0
Q0'
U8
AND2
S1
U9
OR2
R1
U10
AND2
Q1'

4. **Design a 4 bit asynchronous ripple counter and explain the operation with timing diagram.(AUC MAY 2013)**

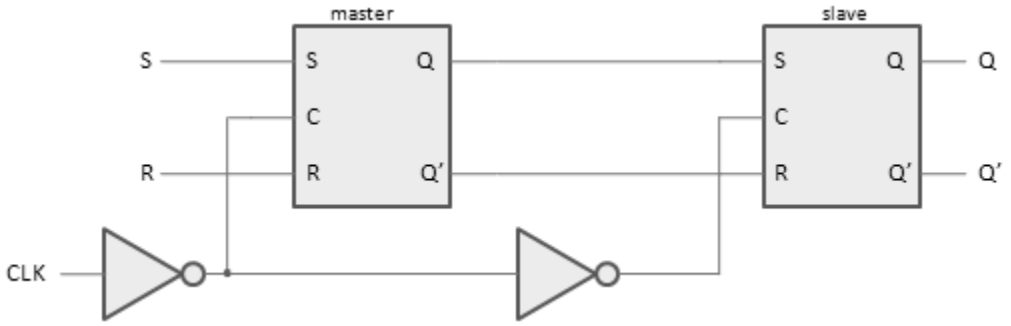


(a) Sequential Circuit



(b) Timing Diagram

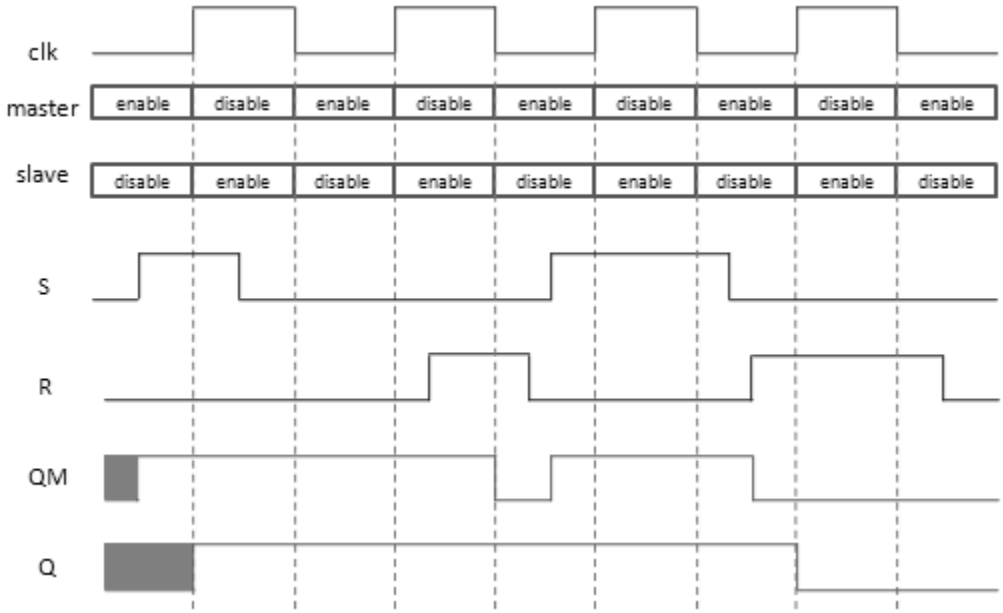
5. a i) **Draw the logic diagram of master slave SR flip flop and explain its working with truth table.(**10) **(AUC NOV 2011)**



Working

- When clk='0', master SR latch C is '1' and S&R excitation inputs can change the master output. But C input of slave is '0'. So changes in S-R of slave (which are connected to outputs of master latch which are changing) latch can't be reflected at output.
- When clk='1', C input of master latch becomes '0', hence it can't respond to S-R inputs. Hence master latch output can't change. C input of slave is '1', so it can change state here.

- Thus the total circuit changes state only at positive edge of clock. Thus it behaves as flip-flop.
- When S=0 AND R=0 the output qn+1 remains in its present state qn.
- When S=0 and R=1 the flip flop resets to 0.
- When S=1 and R=0 the flip flop sets to 1.
- When S=0 and R=0 the output of both gates will produce 0.



Truth table

P.S	N.S	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Characteristic equation:
Q*=S+R'Q

ii) **Design a D flip flop using JK flip flop and explain with its truth table**.(6)
(AUC NOV2011,2009)

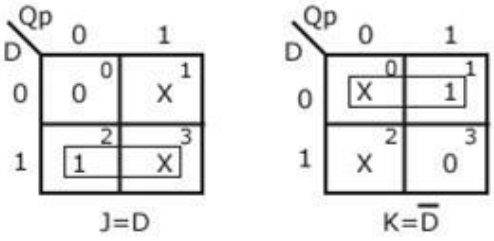
Procedure

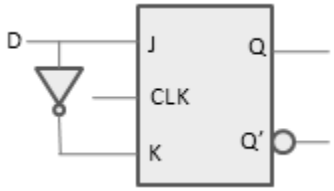
- Write the characteristic table for D flip flop.
- Write the excitation table for JK flip flop.
- Determine the expression of inputs J and k using K map.
- Draw the flip flop conversion logic diagram

Conversion Table

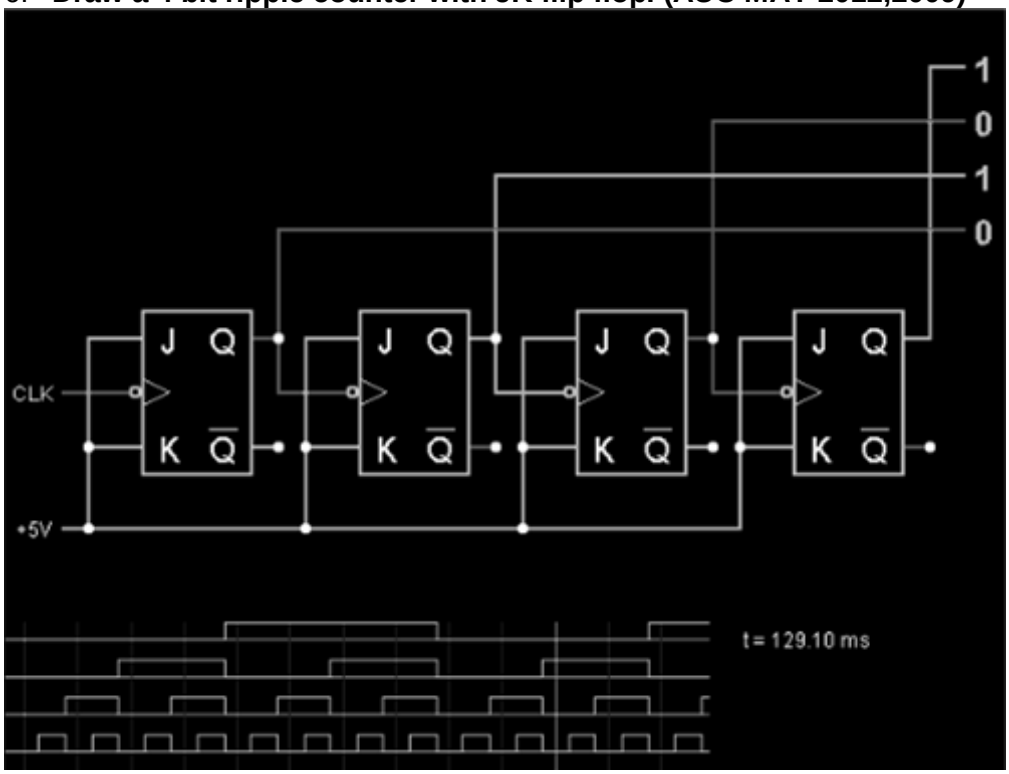
D Input	Outputs Qp	Outputs Qp+1	J-K Inputs J	J-K Inputs K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	0	X	0

K-maps





6. **Draw a 4 bit ripple counter with JK flip flop. (AUC MAY 2012,2009)**



7. Draw the logic diagram of 4 bit binary up /down synchronous counter and explain with truth table .Also draw the timing diagram.(AUC NOV 2011)

The up-down counter has the option for both the up and down counting. It is a combination of up counter and down counter. It can also be called as a multimode counter.

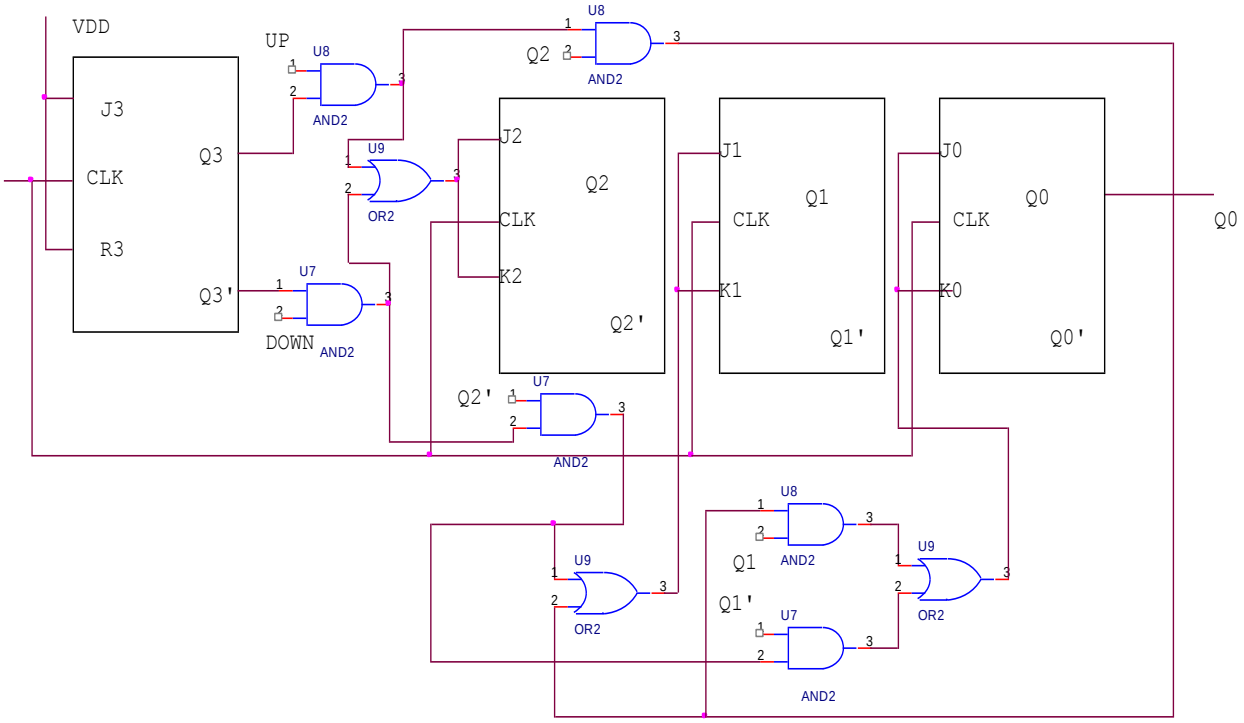
A 4 bit up-down counter is constructed using JK flip flop.

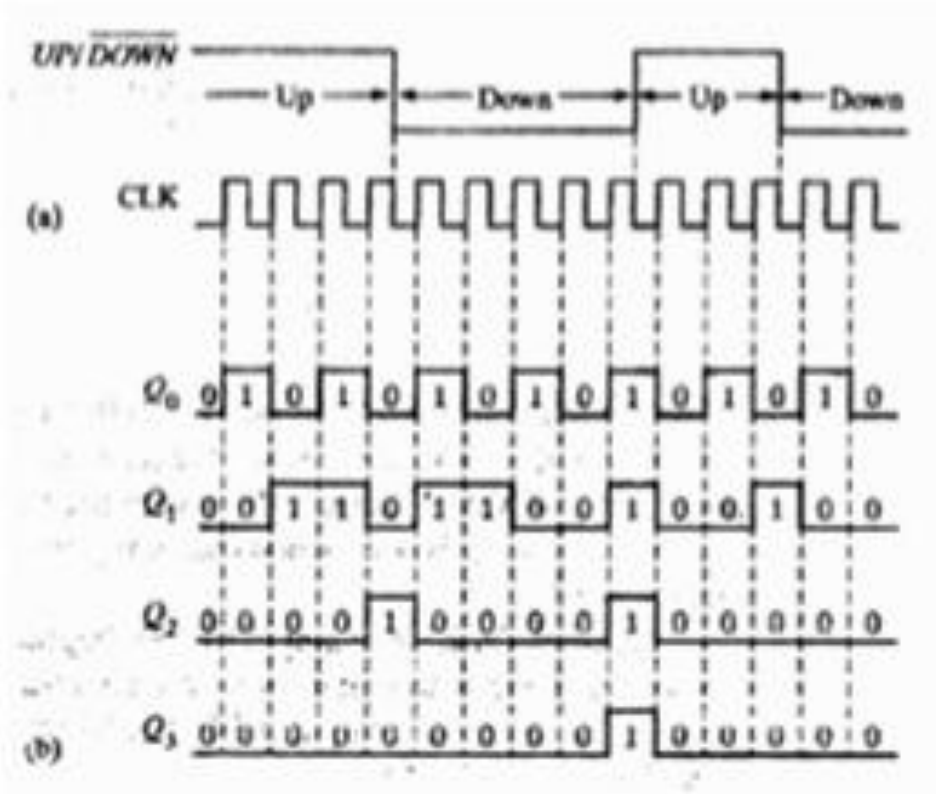
The up /down counter consist of an input called up / down '. If this input is made as 1,the counter will work as a up counter. Similarly ,if the input is made as 0,the counter will work as a down counter.

When up/down' is high, the upper AND gates give the value of Q to the clock pin of the next stage flip flop through an OR gate.

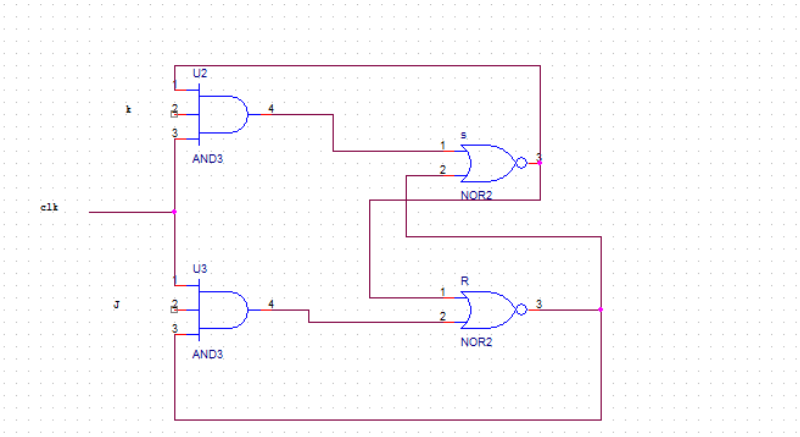
When up/down' is low, the lower AND gates give the value of Q to the clock pin of the next stage flip flop through an OR gate.

COUNT UP MODE					COUNT DOWN MODE				
States	QD	QC	QB	QA	States	QD	QC	QB	QA
0	1	1	1	1	0	0	0	0	0
1	1	1	1	0	1	0	0	0	1
2	1	1	0	1	2	0	0	1	0
3	1	1	0	0	3	0	0	1	1
4	1	0	1	1	4	0	1	0	0
5	1	0	1	0	5	0	1	0	1
6	1	0	0	1	6	0	1	1	0
7	1	0	0	0	7	0	1	1	1
8	0	1	1	1	8	1	0	0	0
9	0	1	1	0	9	1	0	0	1
10	0	1	0	1	10	1	0	1	0
11	0	1	0	0	11	1	0	1	1
12	0	0	1	1	12	1	1	0	0
13	0	0	1	0	13	1	1	0	1
14	0	0	0	1	14	1	1	1	0
15	0	0	0	0	15	1	1	1	1

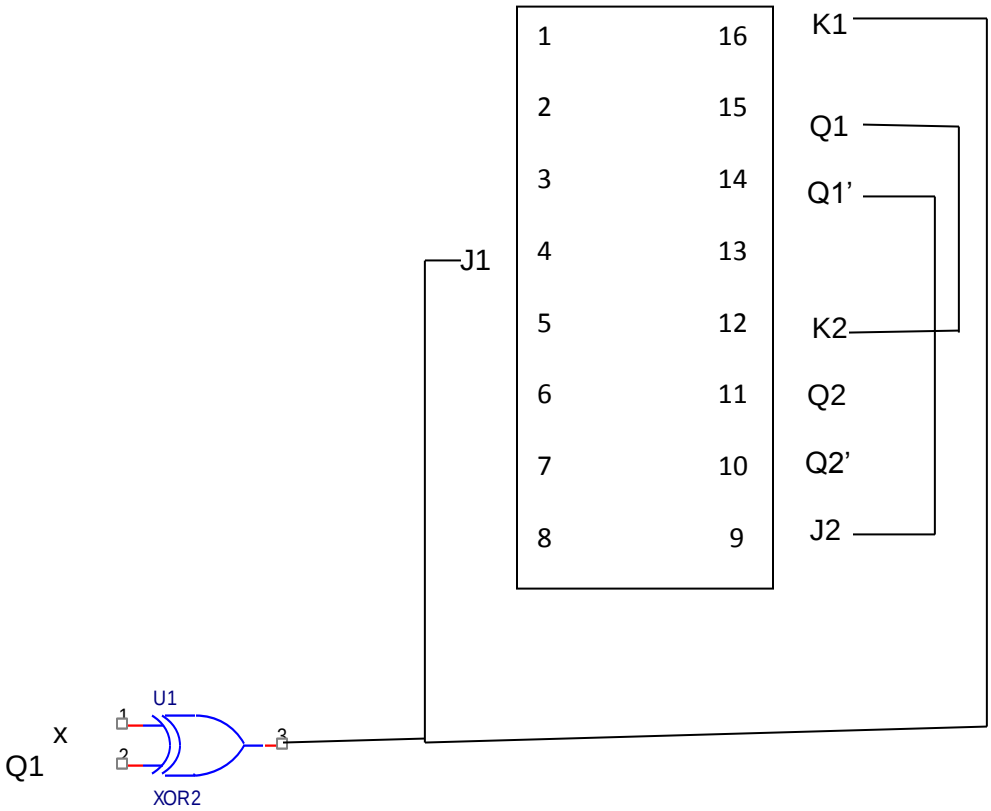

VDD
UP
J3
CLK
R3
Q3
Q3'
U8
AND2
U9
OR2
U7
DOWN
AND2
Q2
AND2
J2
Q2
CLK
K2
Q2'
J1
Q1
CLK
K1
Q1'
J0
Q0
CLK
K0
Q0'
Q0
Q2'
U7
AND2
U8
Q1
AND2
U9
OR2
Q1'
U7
AND2
U9
OR2


UP/DOWN
Up
Down
Up
Down
(a)
CLK
Q_0
Q_1
Q_2
Q_3
(b)

8. **(a) (i) Construct a clocked JK flip flop which is triggered at the positive edge of the clock pulse from a clocked SR flip flop consisting of NOR gates. (4)**



(ii) **Design a synchronous up/down counter that will count up from zero to one to two to three, and will repeat whenever an external input *x* is logic 0, and will count down from three to two to one to zero, and will repeat whenever the external input *x* is logic 1. Implement your circuit with one TTL SN74LS76 device and one TTL SN74LS00 device.** (12) **(AUC APR 2010)**



X	Q2	Q1	Q0	Q2+	Q1+	Q0+	J2	K2	J1	K1	J0	K0
0	0	0	0	0	0	1	0	x	0	x	1	x
0	0	0	1	0	1	0	0	x	1	x	X	1
0	0	1	0	0	1	1	0	x	x	0	1	x
0	0	1	1	0	0	0	0	x	x	1	X	1
1	0	0	0	0	1	1	0	x	1	x	1	x
1	0	0	1	0	0	0	0	x	0	x	X	1
1	0	1	0	0	0	1	0	x	x	1	1	x
1	0	1	1	0	1	0	0	x	X	0	X	1

J1

Q1Q0

XQ2				
		1	X	
	x	x	x	x
	x	x	x	x
	1		x	x

J1 = X ⊕Q0

K 1

Q1Q0

XQ2				
		x	1	
	x	x	x	x
	x	x	x	x
	x		x	1

J1 = X ⊕Q0

J0

Q1Q0

XQ2				
	1	x	x	1
	x	x	x	x
	x	x	x	x
	x	x	x	1

J0 = Q0’

k0

Q1Q0

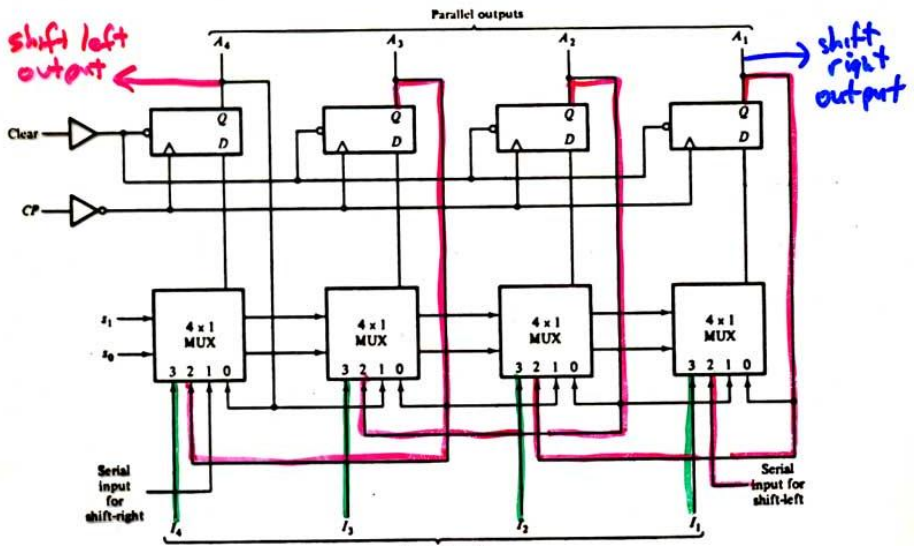
XQ2				
	x	1	1	x
	x	x	x	x
	x	x	x	x
	x	1	1	1

J0 = Q0

(b) (i)**Write down the Characteristic table for the JK flip flop with NOR gates.** (4)

CLOCK	J	K	$\overline{SET}$	$\overline{RESET}$	Q	$\overline{Q}$
-	-	-	0	1	1	0
-	-	-	1	0	0	1
↓	0	0	1	1	Q	$\overline{Q}$
↓	1	0	1	1	1	0
↓	0	1	1	1	0	1
↓	1	1	1	1	$\overline{Q}$	Q

(ii)What is meant by Universal Shift Register? Explain the principle of Operation of 4-bit Universal Shift Register. (12) **(AUC APR 2010)**



A register which can shift the data in either left or right is a unidirectional shift register. If the register has the capability to shift in both left and right direction and parallel load capability is

called a universal shift register.

A universal shift register has the following functions

Clear control : to clear the register to 0

Clock : for giving clock pulses

A shift left control to shift the data left.

A parallel load control to enable a parallel load

Parallel output lines

Control line used to hold the shift operation

The multiplexers are provided with selection lines s0 and s1 for selecting modes

When s1=0,s0=0 there is no shifting operation

When s1=0 ,s0=1 shift right operation takes place

When s1=0 ,s0=1 shift left operation takes place

When s1=1 ,s0=1 parallel loading of data in the mux takes place.

9. Design a synchronous counter which counts in the sequence 0,2,6,1,7,5 using D flip flop. Draw the logic and state diagram. **(AUC JUNE 2007)**

PRESENT STATE	NEXT STATE
000	010
010	110
110	001
001	111
111	101
101	000

Q2	Q1	Q0	Q2+	Q1+	Q0+	D2	D1	D0
0	0	0	0	1	0	0	1	0
0	1	0	1	1	0	1	1	0
1	1	0	0	0	1	0	0	1
0	0	1	1	1	1	1	1	1
1	1	1	1	0	1	1	0	1
1	0	1	0	0	0	0	0	0

D2

Q2/Q1Q0

	1	x	1
x		1	

D2=Q0Q2'+Q1Q0+Q2'Q1

D1

Q2/Q1Q0

1	1	X	1
X			

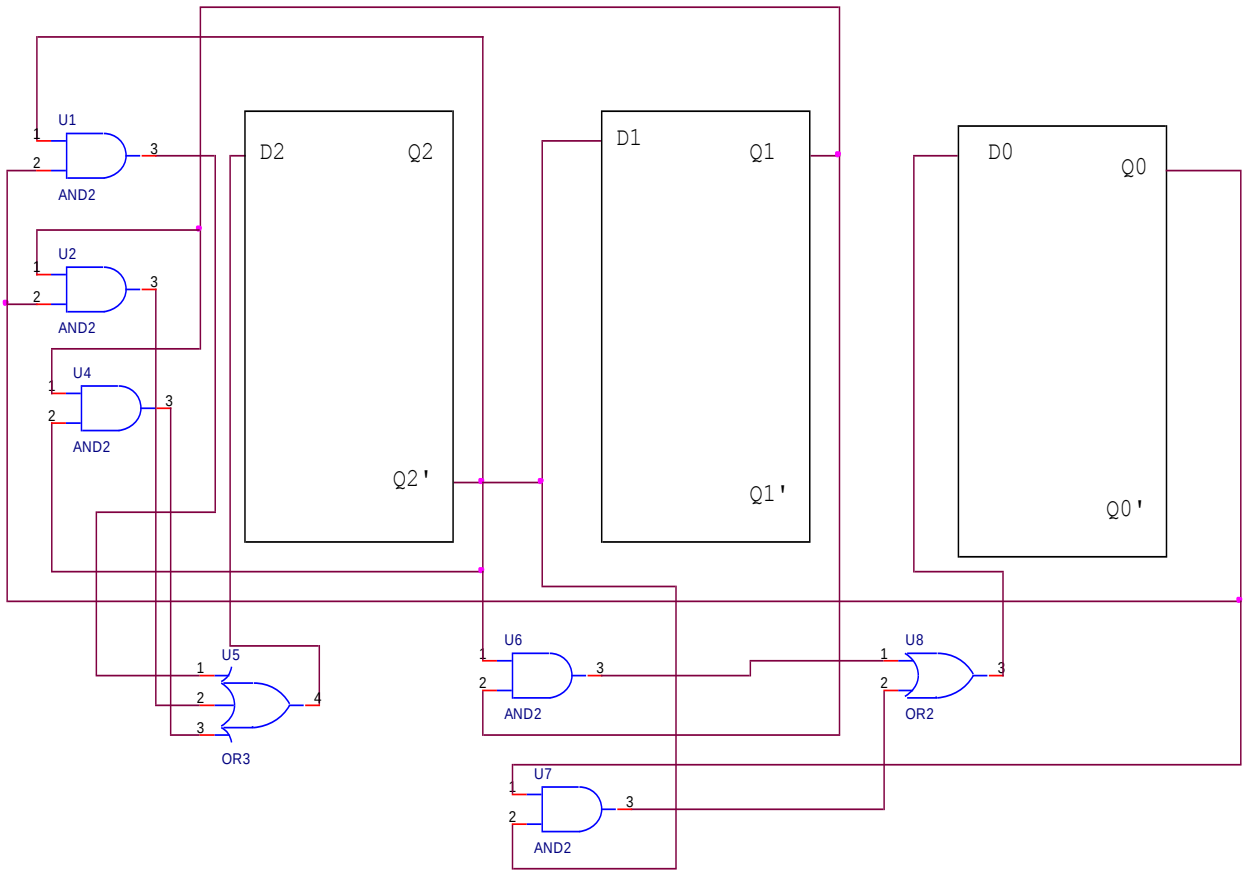
D2=Q2’

D0

Q2/Q1Q0

	1	X	
X		1	1

D2=Q2Q1+Q2’Q0



HDL FOR COMBINATIONAL AND SEQUENTIAL CIRCUITS

Basic Structure

module module_name (list_of_ports);

input/output declarations;

local net declarations;

parallel statements;

endmodule

Simple AND circuit for HDL

```
module simpleand (f, x, y);
 input x, y;
 output f;
assign f = x & y;
endmodule
```

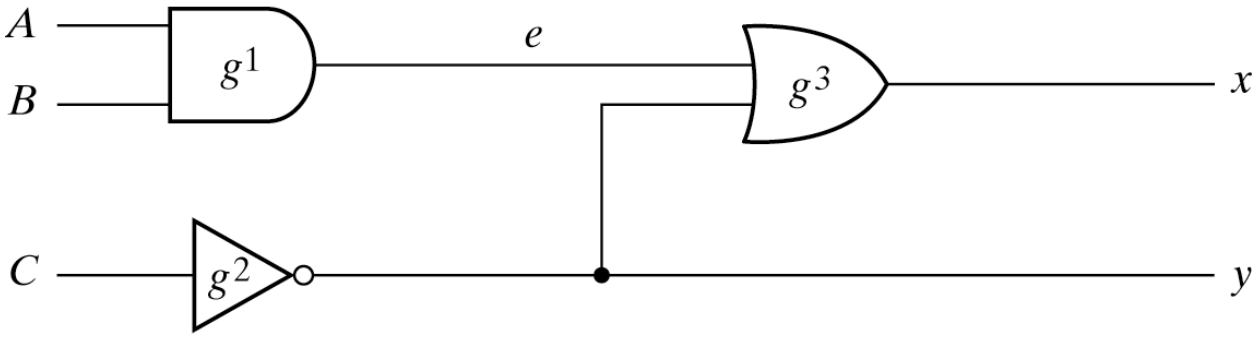


Fig. 3-37 Circuit to Demonstrate HDL

Simple HDL

```
module smpl_circuit(A,B,C,x,y);
input A,B,C;
output x,y;
wire e;
and g1(e,A,B);
    not g2(y, C);
    or g3(x,e,y);
    endmodule
```

Simple circuit with delay

```
    module circuit_with_delay (A,B,C,x,y);
    input A,B,C;
    output x,y;
    wire e;
    and #(30) g1(e,A,B);
    or #(20) g3(x,e,y);
    not #(10) g2(y,C);
    endmodule
```

Effect of Delay

Time (ns)	Input A B C	Output y e x
<0	0 0 0	1 0 1
0	1 1 1	1 0 1
10	1 1 1	0 0 1
20	1 1 1	0 0 1

Simple circuit for Boolean functions

$x = A.B + C^1$
$y = C^1$

```
//Circuit specified with Boolean equations
module circuit_bln (x,y,A,B,C);
  input A,B,C;
  output x,y;
  assign x = A | (B & ~C);
  assign y = ~C ;
endmodule
```

Dataflow description of 2-input Mux

```
module mux2x1_df (A,B,select,OUT);
  input A,B,select;
  output OUT;
  assign OUT = select ? A : B;
endmodule
```

Behavioral description of 2-input mux

```
module mux2x1_bh(A,B,select,OUT)
  input A,B,select;
  output OUT;
  reg OUT;
  always @ (select or A or B)
      if (select == 1) OUT = A;
      else OUT = B;
endmodule
```

```
//D flip–flop
module DFF(D, clock, Q, Qnot, Reset);
input D, clock, Reset;
output Q, Qnot;
reg Q, Qnot;
always @ (posedge clock or negedge Reset)
if (~Reset) {Q,Qnot} = 2'b01;
else {Q,Qnot} = {D, ~D};
endmodule
```

```
//D Latch
module DLatch(D, Control, Q, notQ);
        input D, Control;
        output Q, Qnot;
        reg Q, Qnot;
        always @ (Control or D)
        begin
                if (Control) Q = D;
                Qnot = ~Q;
        end
endmodule
```

QUESTION BANK

DEPARTMENT: CSE **SEMESTER – III**

SUBJECT NAME: DIGITAL PRINCIPLES AND SYSTEM DESIGN
SUBJECT CODE: CS6201

UNIT 4: Asynchronous Sequential Circuits

PART –A (2 Marks)

1. **What is primitive flow table. (AUC MAY 2013)**
   Flow chart one stable state per row is called primitive flow table.

2. **What are static '1' and static '0' hazards. (AUC MAY 2013)**
   The output goes momentarily 0 when it should remain at 1 is called static 1 hazard.
   The output goes momentarily 1 when it should remain at 1 is called static 0 hazard.

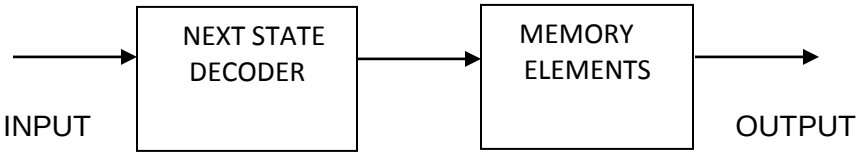
3. **Define Race condition. (AUC MAY 2012)**
   When two or more binary state variable changes their value in response to the change in input variable race occurs.

4. **Bring out the difference between fundamental mode and pulse mode sequential circuits. (AUC NOV 2011)**

Fundamental mode	Pulse mode
Design inputs are levels.	Design inputs are pulses.
Operation depends on the stability of the circuit.	Operation depends on the pulse width.

5. **What is meant by hazard and how it could be avoided? (AUC NOV 2011)**
   The unwanted switching transients are called hazards. By providing extra gates we hazards are avoided.

6. **Draw the block diagram for Moore model. (AUC APR 2010)**



7. **What are hazard free digital circuits? (AUC APR 2010)**
   The digital circuit with extra gates is used to prevent static and dynamic hazards. Such circuits are called hazard free digital circuits.

8. **Compare the ASM chart with a conventional flow chart. (AUC NOV 2009,2013)**
   An ASM chart describes the sequence of events as well as the timing relationship between states of sequential controller.
   A conventional flowchart describes sequence of Procedural steps without concern for their time relationship.

9. **What are the different types of races that occur in fundamental mode circuits. (AUC NOV 2007)**
   Critical race and Non critical race.

10. **Define cycle in asynchronous sequential circuits. (AUC NOV 2007)**
    The asynchronous circuit makes a transition through a series of unstable state. The series of unstable state are called cycle.

11. **What is a hazard in asynchronous sequential circuit. (AUC JUNE 2007)**
    The unwanted switching transients are called hazards.

12. **What are the different methods of operation in asynchronous sequential circuits. (AUC JUNE 2007)**
    Fundamental mode and Pulse mode

13. **What is an essential hazard? (AUC NOV 2008,2012)**
    The essential hazard is due to unequal delays along 2 or more path from the same input.

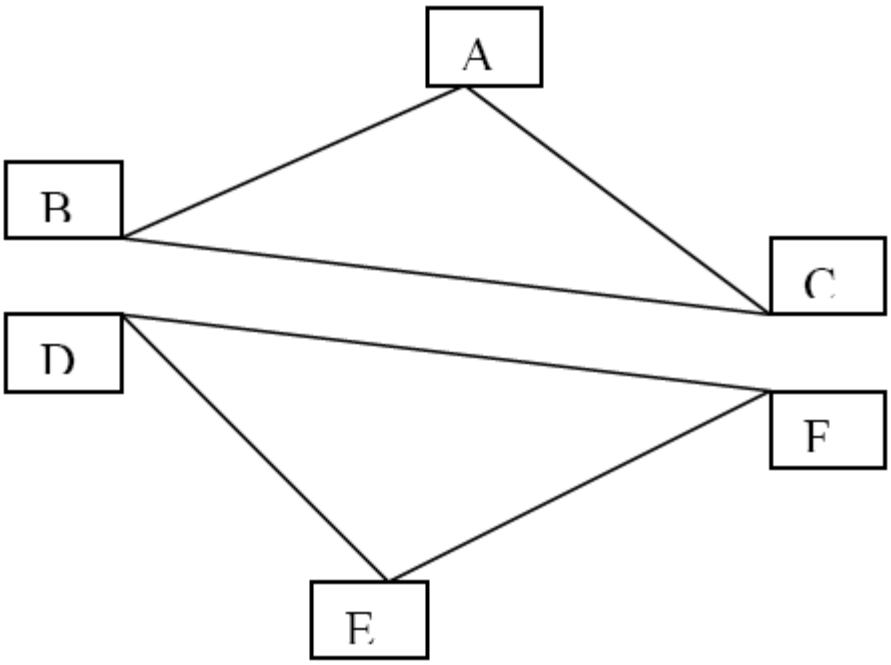
PART –B (16 Marks)

1. **Design a asynchronous sequential circuit with two inputs X and Y and with one output Z. Whenever Y is one, input X is transferred to Z. When Y is zero, the output does not change for any change in X. (AUC MAY 2012,2013)**

PRIMITIVE FLOW TABLE:

Present state	Next state, output Z For XY inputs			
	00	01	11	10
A	(A)	B,-	-,-	C,-
B	A,-	(B)	D,-	-,-
C	A,-	-,-	D,-	(C)
D	-,-	B,-	(D)	E,1
E	F,-	-,-	D,-	(E)
F	(F)	B,-	-,-	E,-

Merger graph for problem:



A, B, C → S0
D, E, F → S1

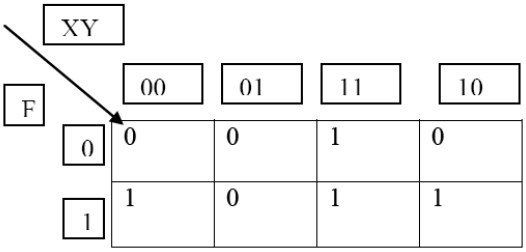
REDUCED FLOW TABLE:

Present state	Next state, output Z For XY inputs			
	00	01	11	10
S0	(S0)	(S0)	S1,-	(S0)
S1	(S1)	S0,-	(S1)	(S1)

Transition table:

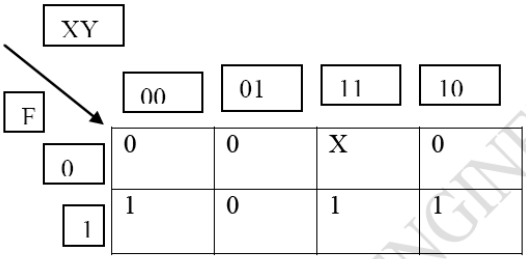
Present state	Next state, output Z For XY inputs			
	00	01	11	10
0	0	0	1,-	0
1	1	0,-	1	1

K-map simplification:



F \ XY	00	01	11	10
0	0	0	1	0
1	1	0	1	1

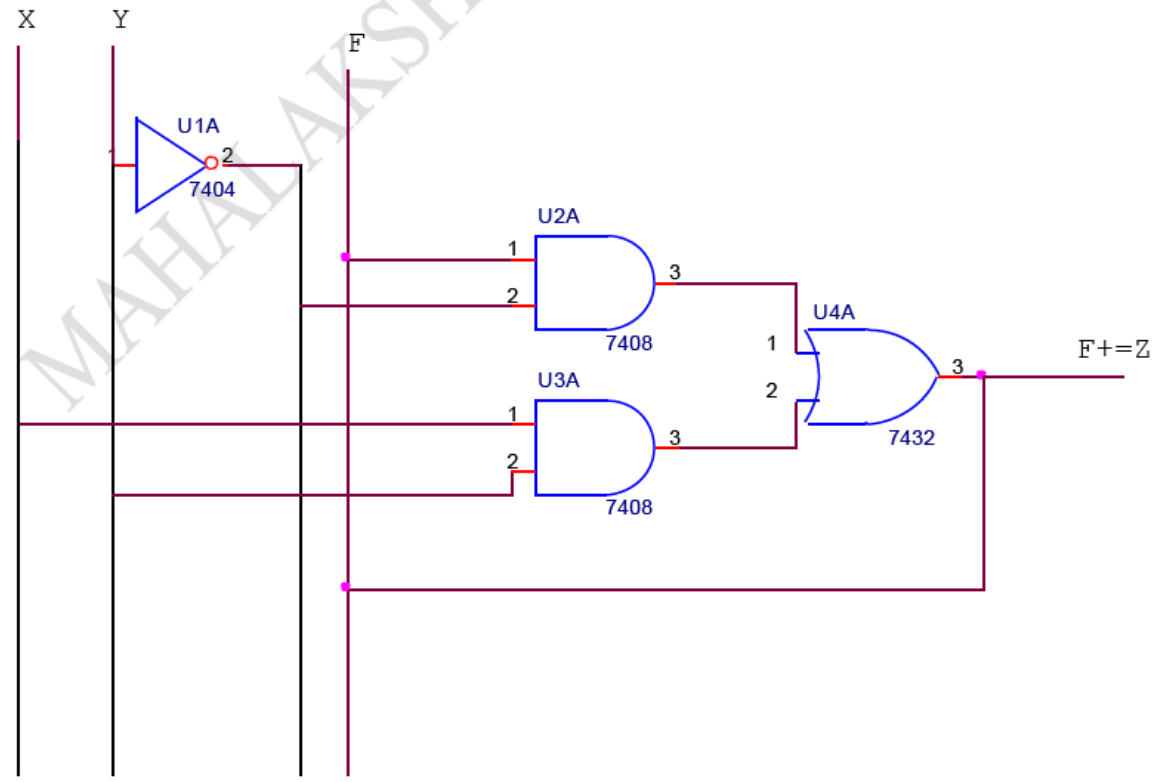
F+=FY'+XY



F \ XY	00	01	11	10
0	0	0	X	0
1	1	0	1	1

Z=FY'+XY

Logic diagram:

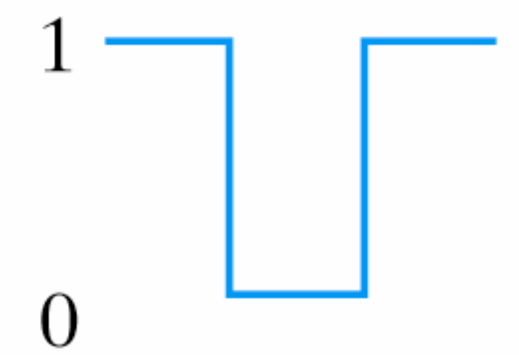


2. **Explain the types of hazards in digital circuits. (AUC MAY 2013)**

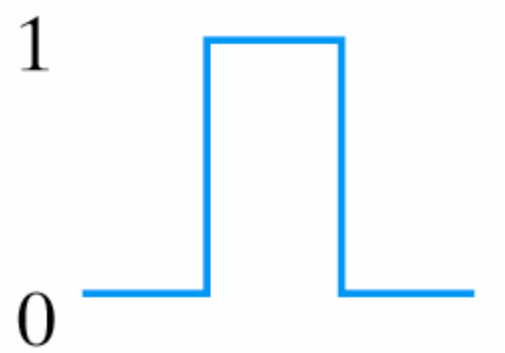
In digital logic, a **hazard** in a system is an undesirable effect caused by either a deficiency in the system or external influences. Logic hazards are manifestations of a problem in which changes in the input variables do not change the output correctly due to some form of delay caused by logic elements (NOT, AND, OR gates, etc.) This results in the logic not performing its function properly. The three different most common kinds of hazards are usually referred to as **static**, **dynamic** and **function hazards**. Hazards are a temporary problem, as the logic circuit will eventually settle to the desired function. However, despite the logic arriving at the correct output, it is imperative that hazards be eliminated as they can have an effect on other connected systems. A **static hazard** is

the situation where, when one input variable changes, the output changes momentarily before stabilizing to the correct value. There are two types of static hazards:

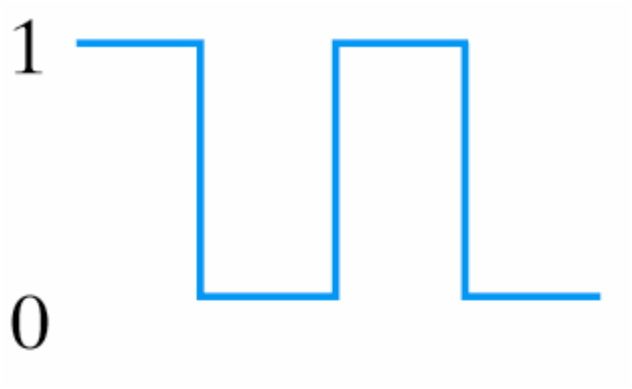
- Static-1 Hazard: the output is currently 1 and after the inputs change, the output momentarily changes to 0 before settling on 1



Static-0 Hazard: the output is currently 0 and after the inputs change, the output momentarily changes to 1 before settling on 0



A **dynamic hazard** is the possibility of an output changing more than once as a result of a single input change. Dynamic hazards often occur in larger logic circuits where there are different routes to the output (from the input). If each route has a different delay, then it quickly becomes clear that there is the potential for changing output values that differ from the required / expected output. e.g. A logic circuit is meant to change output state from **1** to **0**, but instead changes from **1** to **0** then **1** and finally rests at the correct value **0**. This is a dynamic hazard.



3. **Explain the steps for the design of asynchronous sequential circuits. (AUC MAY 2013)**

Design steps:

1. Construction of a primitive flow table from the statement. And intermediate step may include the development of a state diagram
2. Primitive flow table is reduced by eliminating redundant states by using state reduction techniques.
3. state assignment is made
4. The primitive flow table is realized using appropriate logic elements.

Design problems:

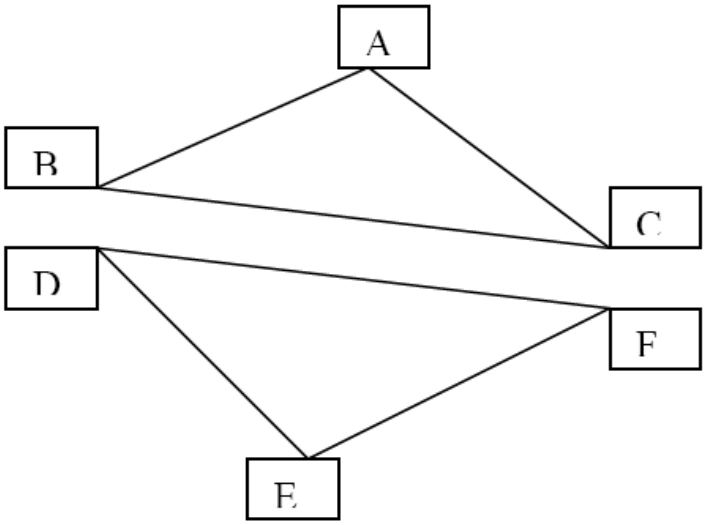
Example:

Design a asynchronous sequential circuit with two inputs X and Y and with one output Z. Whenever Y is one, input X is transferred to Z. When Y is zero, the output does not change for any change in X.

PRIMITIVE FLOW TABLE:

Present state	Next state, output Z For XY inputs			
	00	01	11	10
A	(A)	B,-	-,-	C,-
B	A,-	(B)	D,-	-,-
C	A,-	-,-	D,-	(C)
D	-,-	B,-	(D)	E,1
E	F,-	-,-	D,-	(E)
F	(F)	B,-	-,-	E,-

Merger graph for problem:



A, B, C → S0
D, E, F → S1

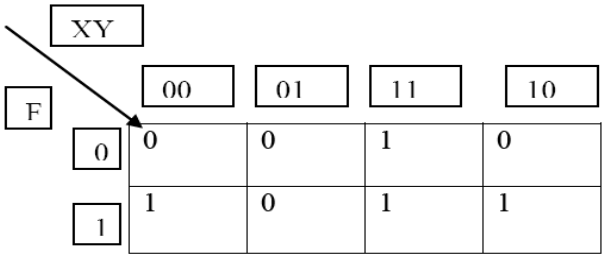
REDUCED FLOW TABLE:

Present state	Next state, output Z For XY inputs			
	00	01	11	10
S0	(S0)	(S0)	S1,-	(S0)
S1	(S1)	S0,-	(S1)	(S1)

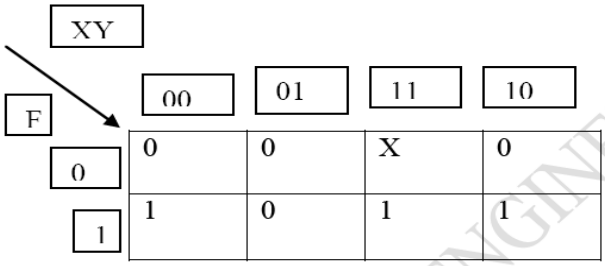
Transition table:

Present state	Next state, output Z For XY inputs			
	00	01	11	10
0	(0)	(0)	1,-	(0)
1	(1)	0,-	(1)	(1)

K-map simplification:

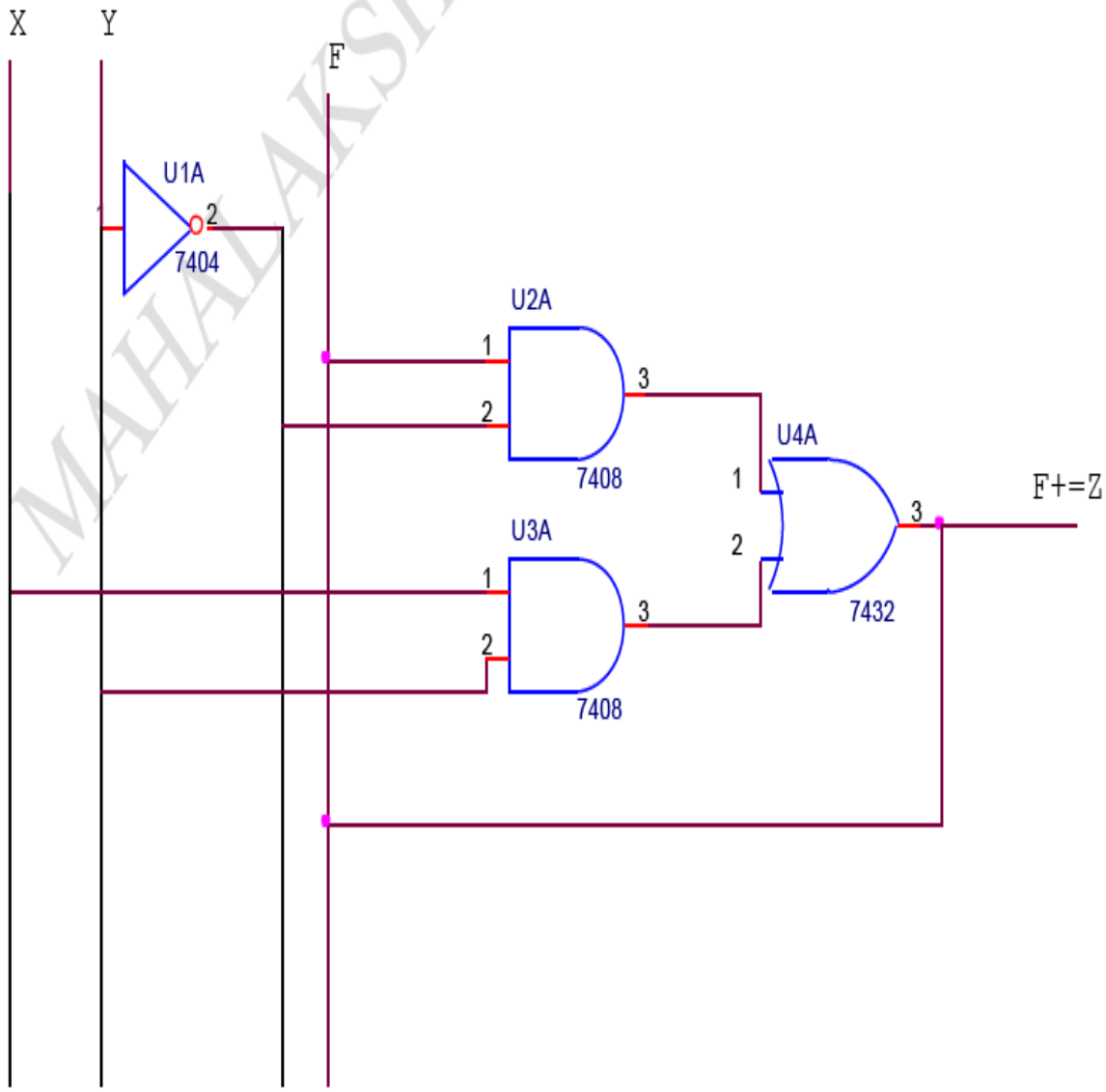


F+=FY'+XY

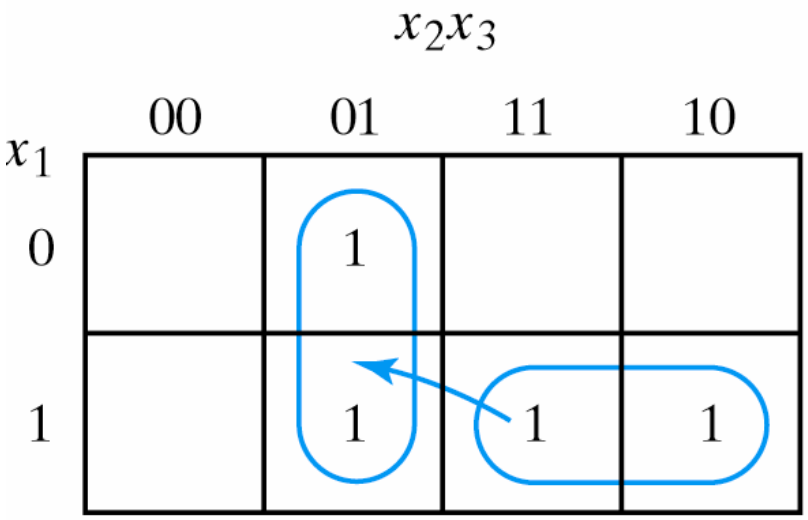


Z=FY'+XY

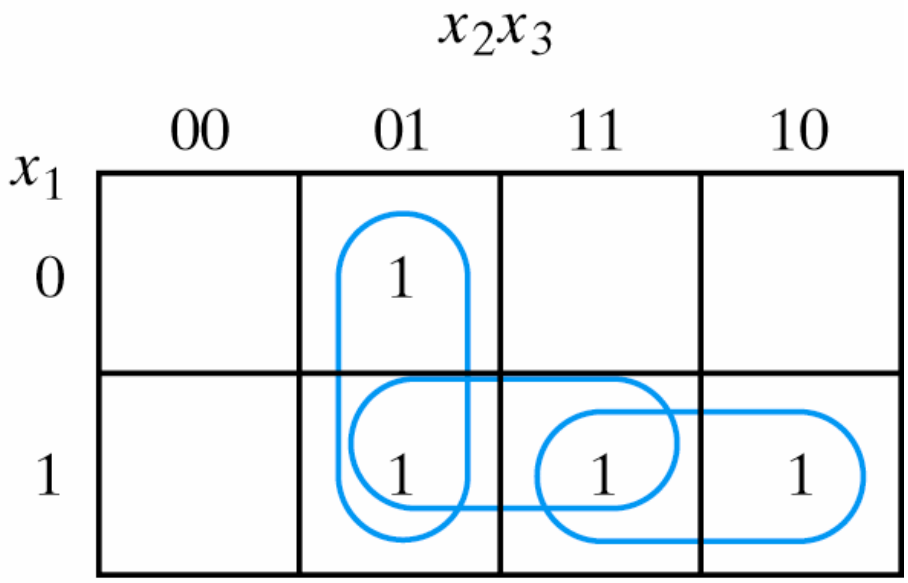
Logic diagram:



4. **Find the circuit that has no static hazards and implement the Boolean function F(A,B,C,D)=∑m(1,,5,6,7) (AUC MAY 2012,2013)**



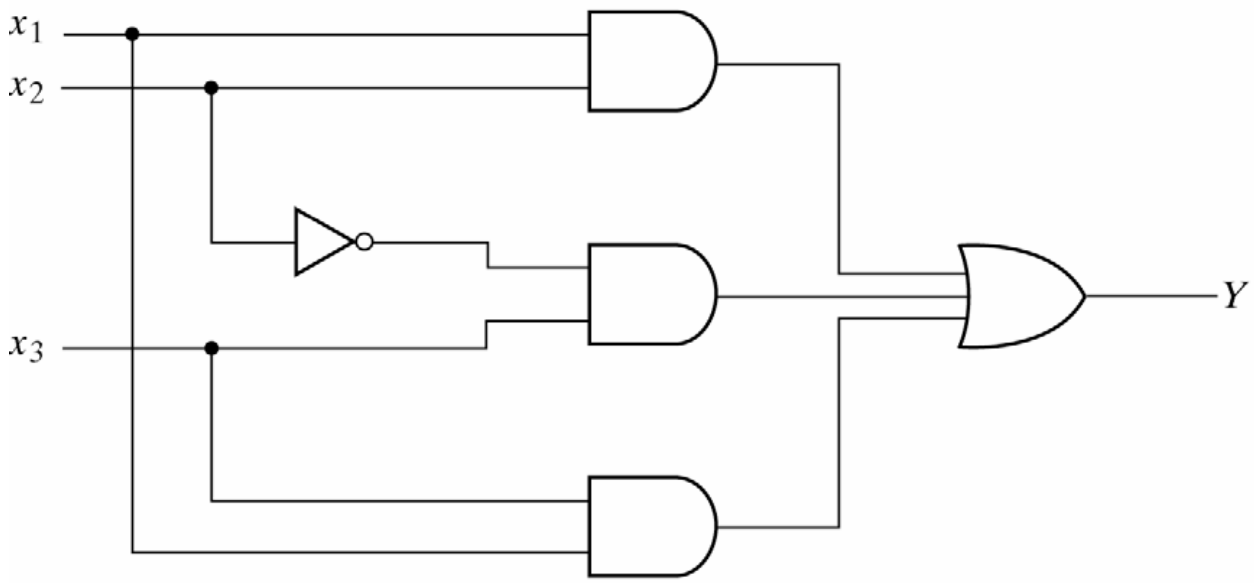
Normal K‟ Map answer is *Y* = *x*1*x*2 + *x*2′ *x*3



Hazard free map is

Y = *x*1*x*2 + *x*2′ *x*3 + *x*1*x*3

Hazard Free Logic diagram is



5. **Write short notes on shared row state assignment with an example.(8) (AUC NOV 2011)(8) (AUC NOV 2011)**

The one hot state assignment is a method to design race-free state assignment. One state variable is required for the each flow table. The additional row are introduced to provide single variable changes between internal state transition.

State variables F4	F3	F2	F1	State	Inputs x1 x2 00	01	10	11
0	0	0	1	A	A	B	C	D
0	0	1	0	B	A	B	C	D
0	1	0	0	C	A	B	C	D
1	0	0	0	D	D	B	C	D

Let us consider the above flow table. A state transition from states A to B requires two state variable changes, F2 should change from 0 to 1 and F1 should change from 1 to 0 .This creates a race in the circuit. To eliminate this race we are following a method called as one hot state assignment.

In between states A and B a new state E is introduced with both changing variables as 1 ie F2 & F1 =1

A= 0 0 0 1

E= 0 0 1 1

B= 0 0 1 0

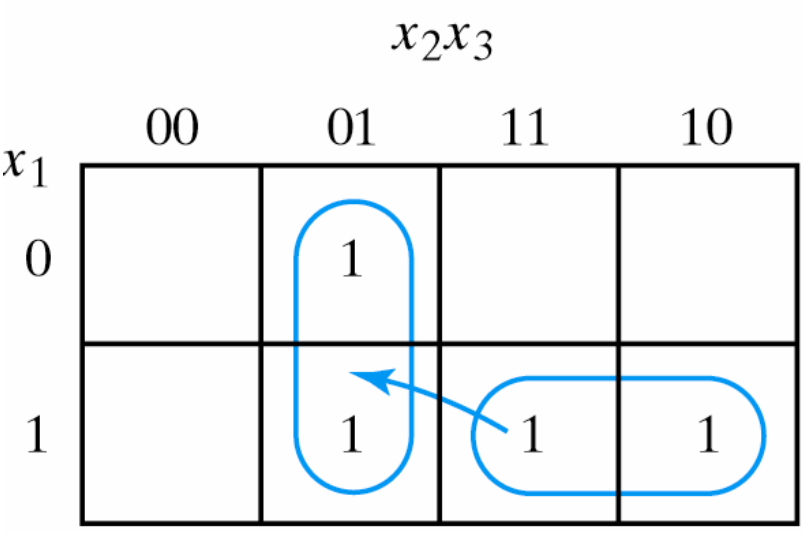
Similarly insert new state between states where two states changing at a time and a modified table with race free states is given as

State variables F4 F3 F2 F1	STATE	INPUTS X1 X2 00 01 11 10
0 0 0 1	A	A B [E] C [F] D[G]
0 0 1 0	B	A [E] B C [H] D[I]
0 1 0 0	C	A [F] B[H] C D[J]
1 0 0 0	D	A [G] B [I] C[J] D
0 0 1 1	E	A B - -
0 1 0 1	F	A - C -
1 0 0 1	G	A - - D
0 1 1 0	H	- B C -
1 0 1 0	I	- B - D
1 1 0 0	J	- - C D

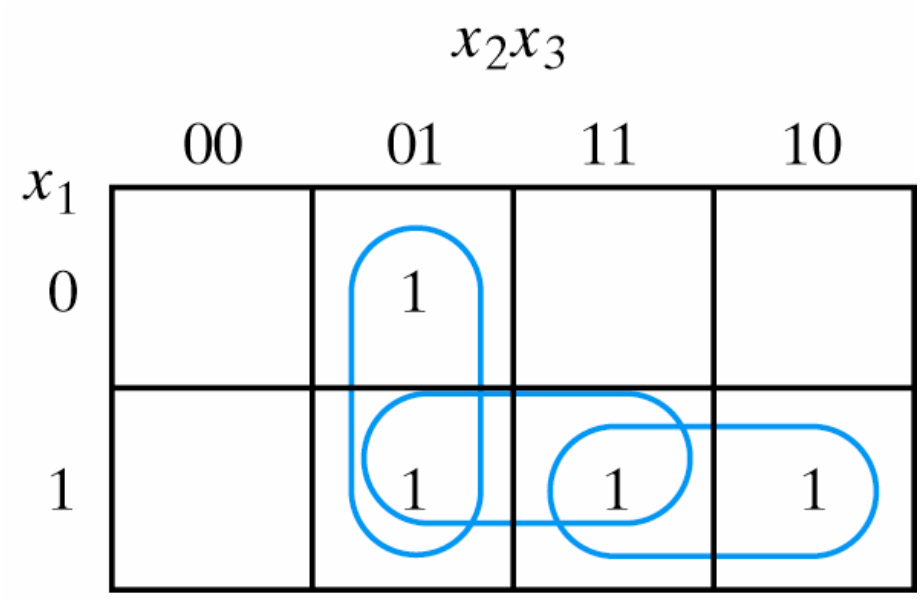
6. **i)Explain the method to eliminate static hazard in an asynchronous circuit with an example.(10) (AUC NOV 2011)**

To eliminate static hazard an additional minterm can be added which maintains the output as 1 when hazard occurs.

Example :



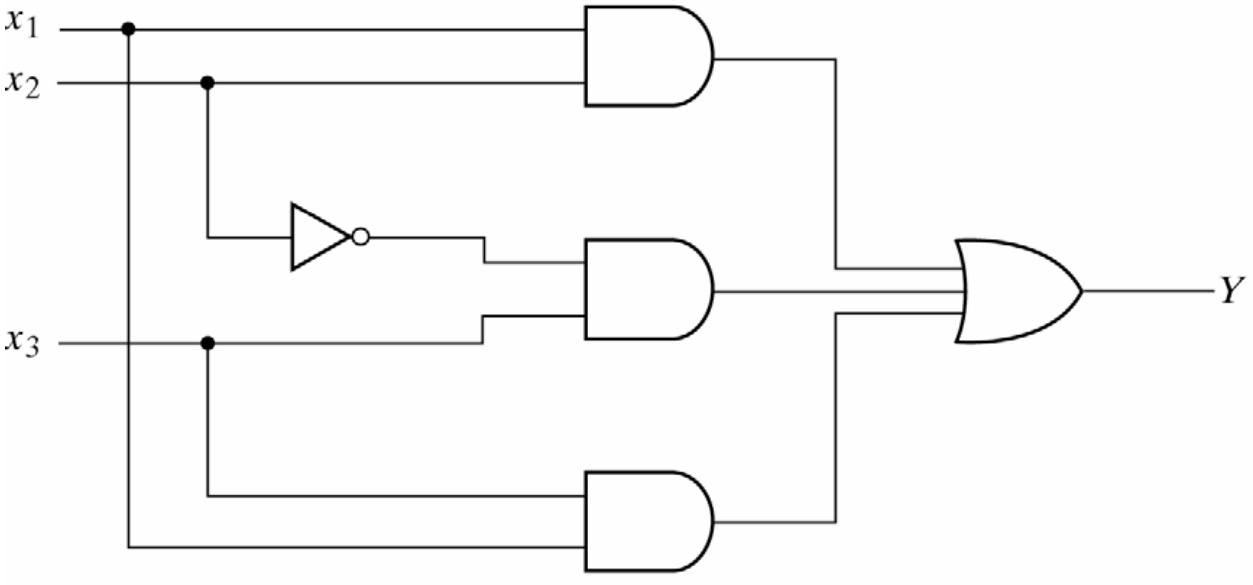
Normal K‟ Map answer is $Y = x1x2 + x2' x3$



Hazard free map is

$Y = x1x2 + x2' x3 + x1x3$

Hazard Free Logic diagram is



ii)Write short notes on verilog.(6) (AUC NOV 2011)

Verilog is a programming language to describe hardware

- The module starts with **module** keyword and finishes with **endmodule**.
- Internal signals are named with **wire**.
- Comments follow ***//***
- **input** and **output** are ports. These are placed at the start of the module definition.
- Each statement ends with a semicolon, except **endmodule**.
- Design entry using both structure and behaviour
- Simulation modelling
- Testing
- It is a standard language
- *Register* - Retains the last value assigned to it. Often used to represent storage elements.
- „wire" equivalent; when there are multiple drivers driving them, the outputs of the drivers are shorted together.
- Arithmetic operators - *, /, +, -, %
- Logical operators - ! logical negation && logical AND | | logical OR
- Relational operators >, <, >=, <=, ==, !=
- Bitwise operators ~, &, |, ^, ~^
- Reduction operators (operate on all the bits within a word) &, ~&, |, ~|, ^, ~^
- accepts a single word operand and produces a single bit as output
- Shift operators >>, <<
- Concatenation { }
- Replication { n { } }
- Conditional <condition> ? <expression1> : <expression2>

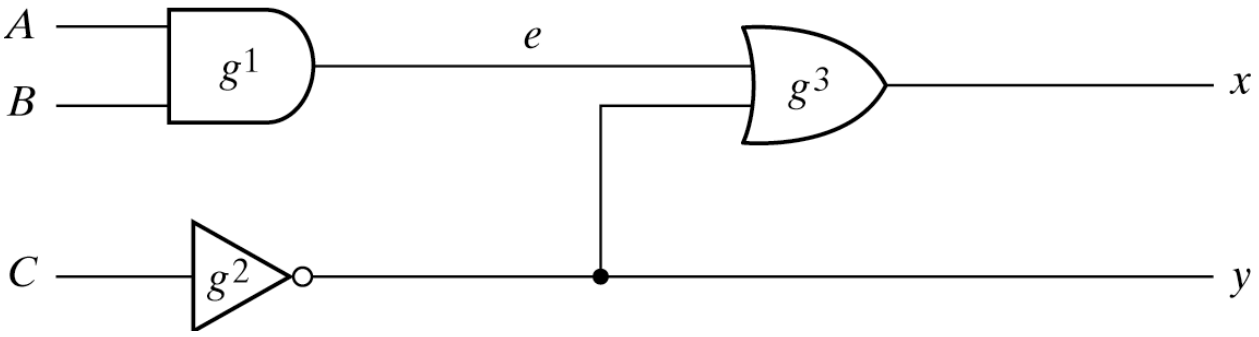
Basic Structure

```
module module_name (list_of_ports);
input/output declarations;
local net declarations;
parallel statements;
endmodule
```

Simple AND circuit for HDL

```
module simpleand (f, x, y);
input x, y;
output f;
assign f = x & y;
endmodule
```

simple circuit diagram

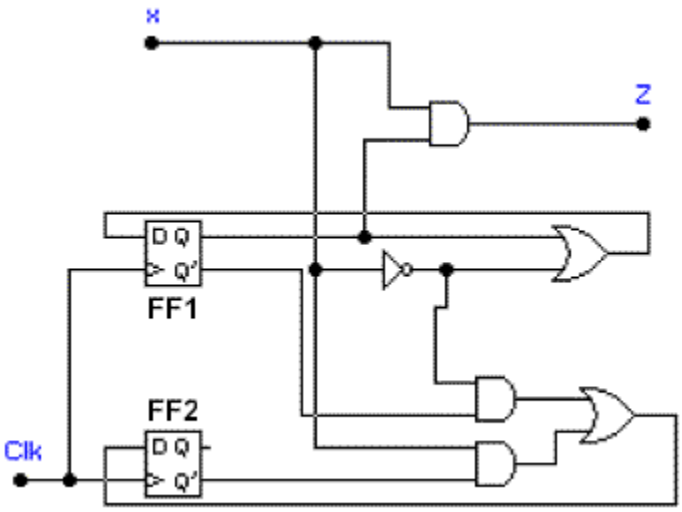


7. **(a) For the circuit shown in figure, write down the state table and draw the state diagram and analyze the operation. (16) (AUC APR 2010)**

   The state table representation of a sequential circuit consists of three sections labelled present state, next state and output. The present state designates the state of flip-flops before the occurrence of a clock pulse. The next state shows the states of flip-flops after the clock pulse, and the output section lists the value of the output variables during the present state.

   State Diagram

   In addition to graphical symbols, tables or equations, flip-flops can also be represented graphically by a state diagram. In this diagram, a state is represented by a circle, and the transition between states is indicated by directed lines (or arcs) connecting the circles. An example of a state diagram is shown in Figure below.Consider a sequential circuit .It has one input x, one output Z and two state variables Q1Q2 (thus having four possible present states 00, 01, 10, 11).



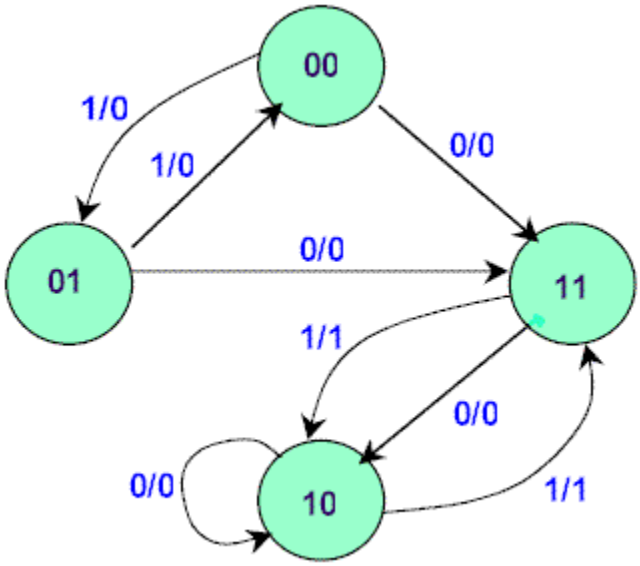
The behaviour of the circuit is determined by the following Boolean expressions:

Z = x*Q1
D1 = x' + Q1
D2 = x*Q2' + x'*Q1'

These equations can be used to form the state table. Suppose the present state (i.e. Q1Q2) = 00 and input x = 0. Under these conditions, we get Z = 0, D1 = 1, and D2 = 1. Thus the next state of the circuit D1D2 = 11, and this will be the present state after the clock pulse has been applied. The output of the circuit corresponding to the present state Q1Q2 = 00 and x = 1 is Z = 0. This data is entered into the state table as shown in Table 2.

Present State	Next State		Output	
Q1Q2	x = 0	x = 1	x = 0	x = 1
0 0	1 1	0 1	0	0
0 1	1 1	0 0	0	0
1 0	1 0	1 1	0	1
1 1	1 0	1 0	0	1

The state diagram for the sequential circuit



(b) What are called as essential hazards? How does the hazard occur in sequential circuits? How can the same be eliminated using SR latches? Give an example. (16)(AUC APR 2010)

Essential hazard is caused by unequal delays along two or more paths that originate from the same input. An excessive delay through an inverter circuit in comparison to the delay associated with the feedback path may cause such a hazard.

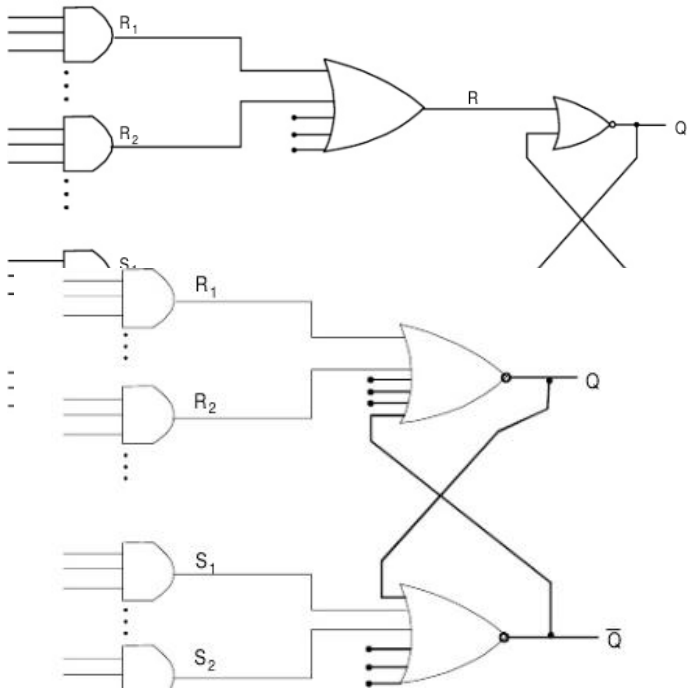
How does the hazard occur in sequential circuits?

In digital logic, a **hazard** in a system is an undesirable effect caused by either a deficiency in the system or external influences. Logic hazards are manifestations of a problem in which changes in the input variables do not change theoutput correctly due to some form of delay caused by logic elements (NOT, AND, OR gates, etc.) This results in the logic not performing its function properly. The three different most common kinds of hazards are usually referred to as **static**, **dynamic** and **function hazards**.

Elimination using SR latch

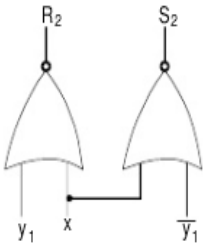
A typical network structure with the S-R flip-flop driven by 2-level AND-OR networks constructed from cross-coupled NOR gates is shown in the diagram. The equivalent network structure is provided with multiple input NOR gates. The two structure are equivalent since in both cases.

$$Q = (\overline{Q} + R_1 + R_2 + ...)'$$
$$\overline{Q} = (Q + S_1 + S_2 + ...)'$$

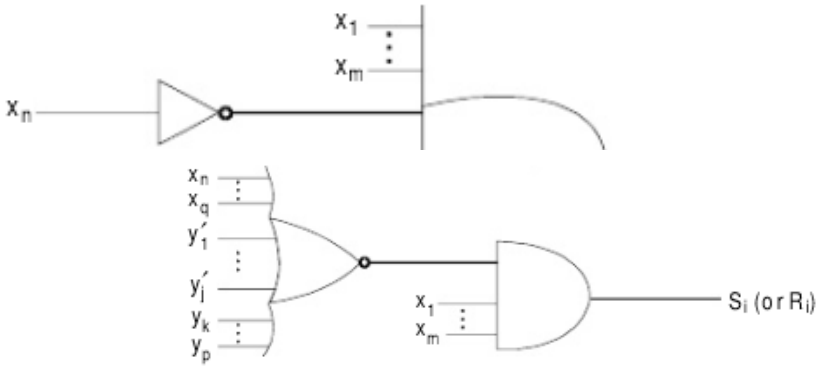


Even if an asynchronous network is realized using S-R flip-flops and S and R networks are free of 0-hazards, essential hazards may still be present. Such essential hazards may be eliminated as discussed previously by adding delays in the feedback paths for the state variables. Therefore, in an asynchronous network with S-R flip-flops, we can eliminate the essential hazards by arranging the gate structure so that the effect of any input change will propagate to all flip-flop inputs before any state variable changes can propagate back to the flip-flop inputs. For example, the essential hazard of Fig. 8.24 can be eliminated by replacing the R2 and S2 networks

Assuming that wiring delays are negligible that the gate delay is concentrated at the gate output any change in x will propagate to R2 and S2 before flip-flop 1 output y1 can change state and this change in y1 can propagate to R2 and S2



This eliminates the essential hazard. x's are external inputs to the circuit, and the y's are feedback from flip-flop outputs. If there are essential hazards in the flow table, then the circuit could malfunction due to the inverter delays. By replacing the AND gate with the NOR-AND network the inverters on the x variables are eliminated. Therefore by replacing all of the AND gate with the NOR-AND combinations as indicated, all of the estimate hazards will be eliminated.

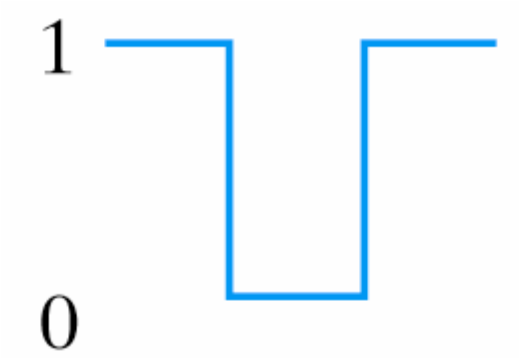


8. **Discuss on the different types of hazards that occur in asynchronous sequential circuits. (AUC NOV 2007)**

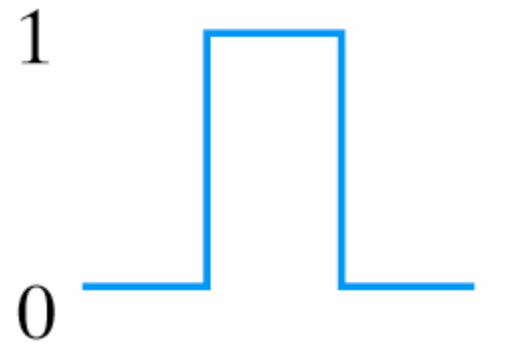
In digital logic, a **hazard** in a system is an undesirable effect caused by either a deficiency in the system or external influences. Logic hazards are manifestations of a problem in which changes in the input variables do not change the output correctly due to some form of delay caused by logic elements (NOT, AND, OR gates, etc.) This results in the logic not performing its function properly. The three different most common kinds of hazards are usually referred to as **static**, **dynamic** and **function hazards**. Hazards are a temporary problem, as the logic circuit will eventually settle to the desired function. However, despite the logic arriving at the correct output, it is imperative that hazards be

eliminated as they can have an effect on other connected systems. A **static hazard** is the situation where, when one input variable changes, the output changes momentarily before stabilizing to the correct value. There are two types of static hazards:

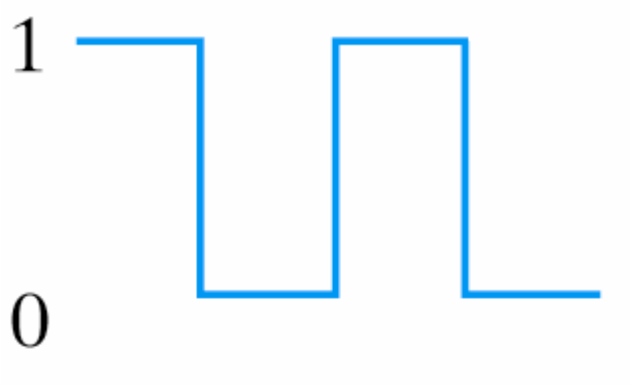
Static-1 Hazard: the output is currently 1 and after the inputs change, the output momentarily changes to 0 before settling on 1



Static-0 Hazard: the output is currently 0 and after the inputs change, the output momentarily changes to 1 before settling on 0



A **dynamic hazard** is the possibility of an output changing more than once as a result of a single input change. Dynamic hazards often occur in larger logic circuits where there are different routes to the output (from the input). If each route has a different delay, then it quickly becomes clear that there is the potential for changing output values that differ from the required / expected output. e.g. A logic circuit is meant to change output state from **1** to **0**, but instead changes from **1** to **0** then **1** and finally rests at the correct value **0**. This is a dynamic hazard.



9. **Write short notes on i) race free assignments ii) pulse mode circuits. (AUC NOV 2007)**

RACE-FREE STATE ASSIGNMENT:

- Choose a proper binary state assignment to prevent critical races.
- Only one variable can change at any given time when a state transition occurs.
- States between which transitions occur will be given adjacent assignments
- Two binary values are said to be adjacent if they differ in only one variable
- To ensure that a transition table has no critical races, every possible state transition should be checked

State assignments can be demonstrated by means of four row flow table example , multiple row flow table example, one hot state assignment technique

Multiple row assignment method

In this method ,each row in the original flow table is replaced by two or more equivalent rows. Thus there will be two or more assignments of state variables to each state.

A possible multiple row assignment is shown. Here there are two state variables for each state, and each is a logical compliment of the other. For instance state a is represented by a1 and a2 where a1 has an assignment of 000 while a2 is assigned 111.Also a1 is adjacent to b1 ,d1 and c2, while a2 is adjacent to b2,d2 and c1

	y_2y_3			
y_1	00	01	11	10
0	a_1	b_1	c_1	d_1
1	c_2	d_2	a_2	b_2

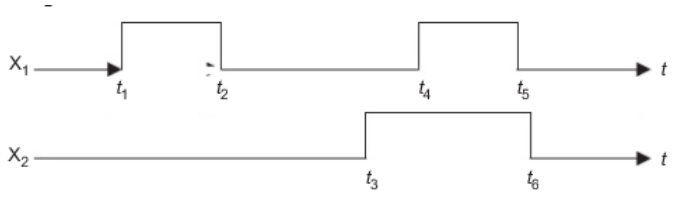
Expanded flow table

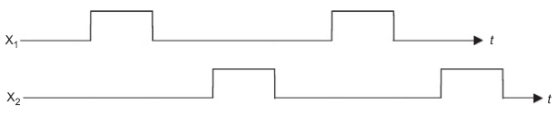
	x_1x_2 = 00	01	11	10
a_1	c_2	a_1	b_1	a_1
a_2	c_1	a_2	b_2	a_2
b_1	b_1	c_1	b_1	a_1
b_2	b_2	c_2	b_2	a_2
c_1	c_1	c_1	d_1	d_1
c_2	c_2	c_2	d_2	d_2
d_1	b_2	a_1	d_1	d_1
d_2	b_1	a_2	d_2	d_2

Here each row is replaced by two rows. For instance , row a is replaced by rows a1 and a2.In the original table ,state a makes transition to states b and c. In the expanded table a1 makes transition to b1 and c2 and a2 makes transition to b2 and c1.for state a1 with input 01 the sequence will be c2 ,d2,b1,c1,d1 01 : 00 ,10, 00,01,11

pulse mode circuits

In pulse mode, the inputs and outputs are represented by pulses. In this mode of operation the width of the input pulses is critical to the circuit operation. The input pulse must be long enough for the circuit to respond to the input but it must not be so long as to be present even after new state is reached. In such a situation the state of the circuit may make another transition. The minimum pulse width requirement is based on the propagation delay through the next state logic .The maximum pulse width is determined by the total propagation delay through the next state logic and the memory elements .In pulse-mode operation, only one input is allowed to have pulse present at any time. This means that when pulse occurs on any one input, while the circuit is in stable state, pulse must not arrive at any other input. X1 and X2 are the two inputs to a pulse mode circuit. In the diagram at time t3 pulse at input X2 arrives. While this pulse is still present, another pulse at X1 input arrives at t4. Therefore, this kind of the presence of pulse inputs is not allowed. Both fundamental and pulse mode asynchronous sequential circuits use unclocked S-R flip-flops or latches. In the design of both types of circuits, it is assumed that a change occurs in only one inputs and no changes occurs in any other inputs until the circuit enters a stable state.





10. **What is an essential hazard? Explain with example.(6) (AUC JUNE 2007)**

Refer Question No. 7b

11. **Explain how hazard free realization can be obtained for a boolean function.(8) (AUC JUNE 2007)**

Refer Question No. 6

12. **Discuss a method used for race free assignments with example.(8) (AUC JUNE 2007)**

Refer Question No. 5

QUESTION BANK

DEPARTMENT: CSE **SEMESTER – III**

SUBJECT NAME: DIGITAL PRINCIPLES AND SYSTEM DESIGN
SUBJECT CODE: CS6201

UNIT 5: Memory and Programmable Logic

PART –A (2 Marks)

1. **What are the different types of Programmable logic devices.(AUC MAY 2013)**
   PROM ,PAL,PLA,GAL,FPGA

2. **What is the need for output buffer in a PLA system?(AUC NOV 2011)**
   Without buffer the PLA provide AND – OR implementation.
   With buffer the PLA provide AND – OR – INVERT implementation.

3. **Give the difference between RAM and ROM.(AUC NOV 2011)**

RAM	ROM
Volatile memory	Non volatile memory
Read and write operations are possible	Read operation only takes place.

4. **What is meant by memory expansion? Mention its limit.(AUC APR 2010)**
   The memory expansion is achieved in two ways
   1. By expanding word size
   2. By expanding more memory capacity

   The limit is connecting more than one IC is not possible.

5. **What are the advantages of static RAM compared to Dynamic RAM?(AUC APR 2010,2013)**

STATIC RAM	DYNAMIC RAM
Less memory cells per unit area	more memory cells per unit area
Refreshing not required	Refreshing required
Cost is more	Cost is less

6. **Compare and contrast static RAM and dynamic RAM.(AUC NOV 2009)**

STATIC RAM	DYNAMIC RAM
Less memory cells per unit area	more memory cells per unit area
Refreshing not required	Refreshing required
Cost is more	Cost is less

7. **What is PAL? How does it differ from PLA? (AUC NOV 2009,2013)**

The PAL is a programmable logic device with a fixed OR array and a programmable AND array, whereas PLA contains both AND and OR arrays as programmable.

8. **How the semiconductor memories are classified?(AUC NOV 2008)**

RAM ,ROM ,PROM ,EPROM ,EEPROM

9. **Explain dynamic hazard** (**(AUC NOV 2008)**

When the output changes from 0 to 1 the circuit may go through three or more transients and produce glitch. These are called as dynamic hazards.

10. **What is race ?**(**AUC APR 2007**)

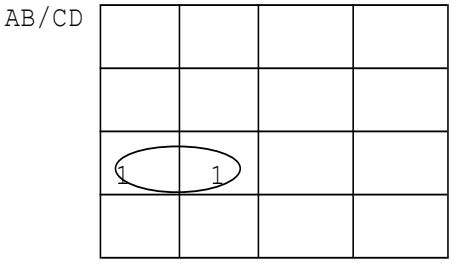
In JK flip flop when both inputs are high, the output toggles continuously. This condition is called as race.

PART –B(16 Marks)

1. **Design using PAL the following Boolean function. (AUC MAY 2013)**

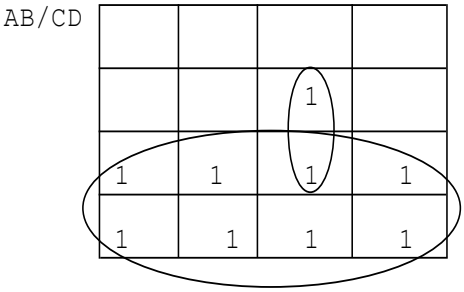
W(A,B,C,D)= Σ(2,12,13)
X(A,B,C,D)= Σ(7,8,9,10,11,12,13,14,15)
Y(A,B,C,D)= Σ(0,2,3,4,5,6,7,8,10,11,15)
Z(A,B,C,D)= Σ(1,2,8,12,13)

K MAP FOR W



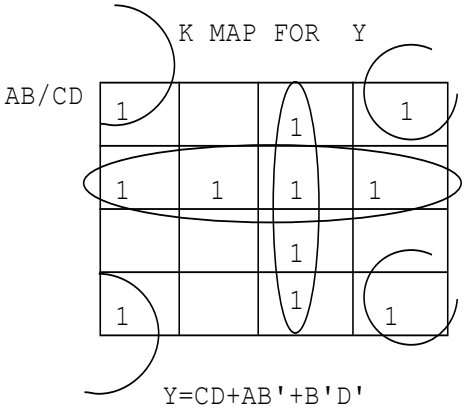
W=A'B'CD'+ABC'

K MAP FOR X



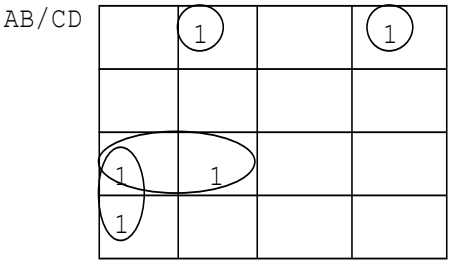
X=A+ACD

K MAP FOR Y



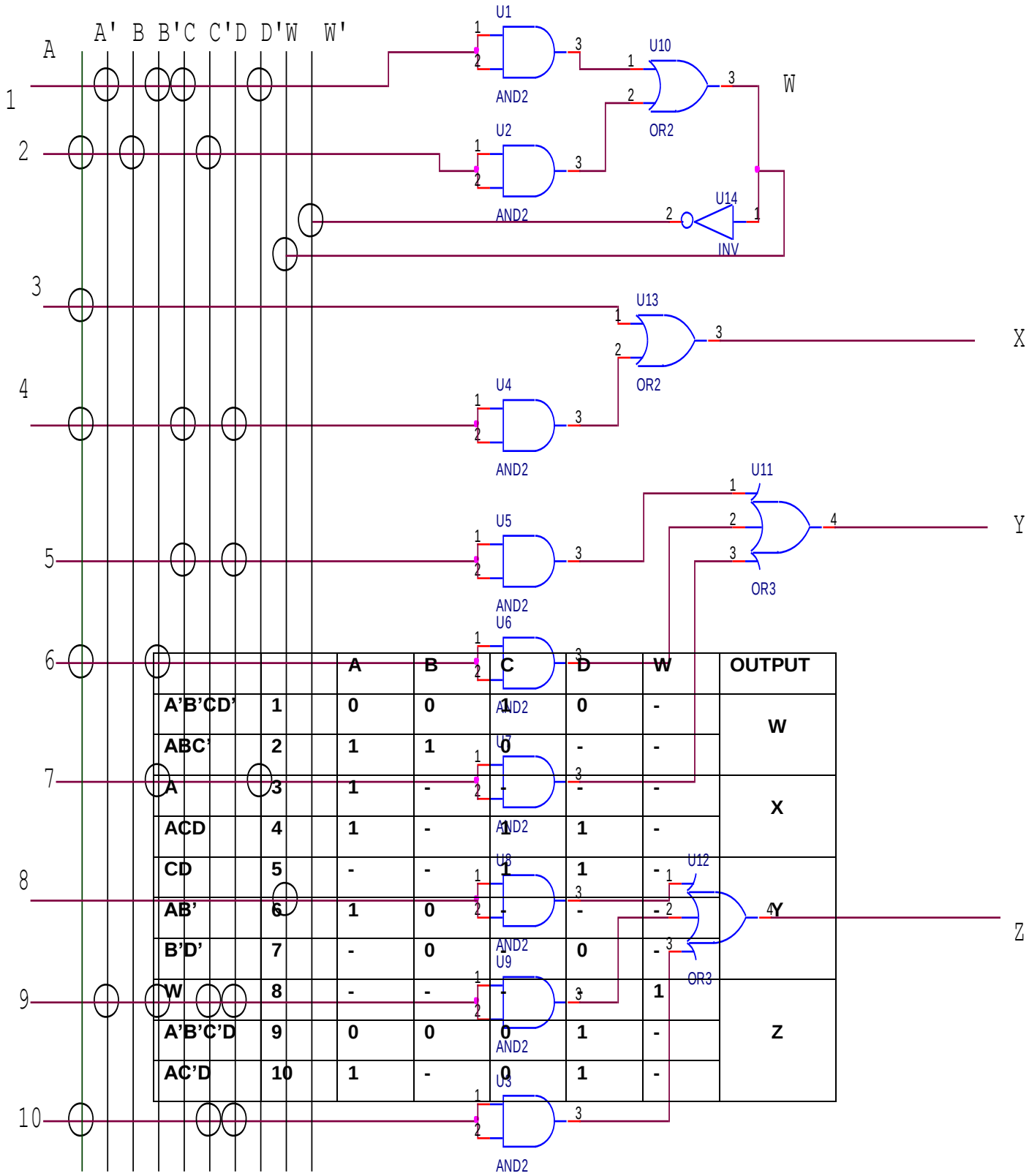
Y=CD+AB'+B'D'

K MAP FOR Z



Z=A'B'C'D+A'B'CD'+ABC'+AC'D'
Z=W+A'B'C'D+AC'D'

PROGRAMMING TABLE



		A	B	C	D	W	OUTPUT
A'B'CD'	1	0	0	1	0	-	W
ABC'	2	1	1	0	-	-	
A	3	1	-	-	-	-	X
ACD	4	1	-	1	1	-	
CD	5	-	-	1	1	-	Y
AB'	6	1	0	-	-	-	
B'D'	7	-	0	0	0	-	
W	8	-	-	-	-	1	Z
A'B'C'D	9	0	0	0	1	-	
AC'D	10	1	-	0	1	-	

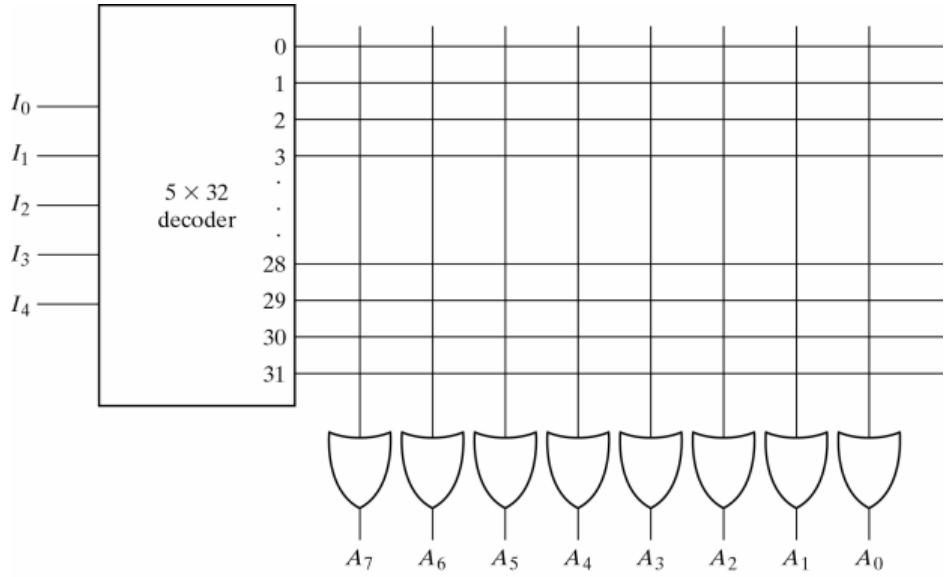
2. **Design and explain a 32x8 ROM. (AUC MAY 2013)**

A 32 x 8 ROM consists of 32 words of 8 bits each.

The five input lines are decodedby into 32 distinct outputs (memory addresses) using a 25 x 8 decoder.

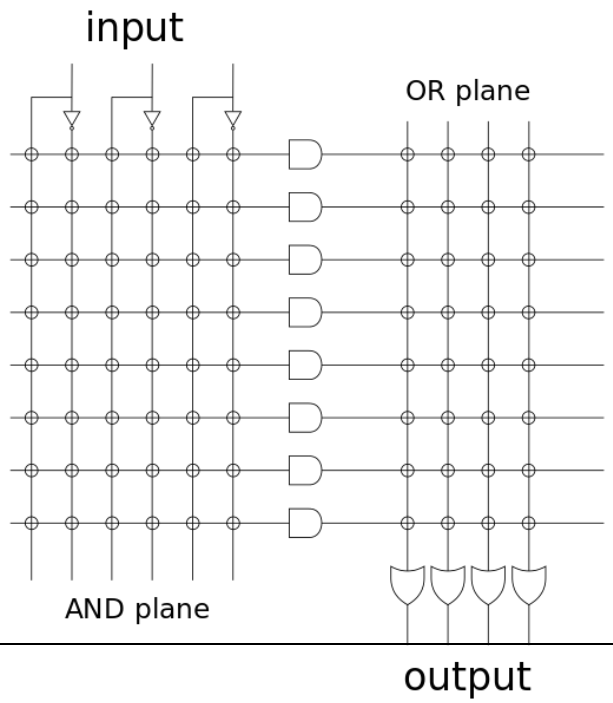
Each ORgate has 32 input connections à 32 x 8 ROM has internal connections 32 x 8.

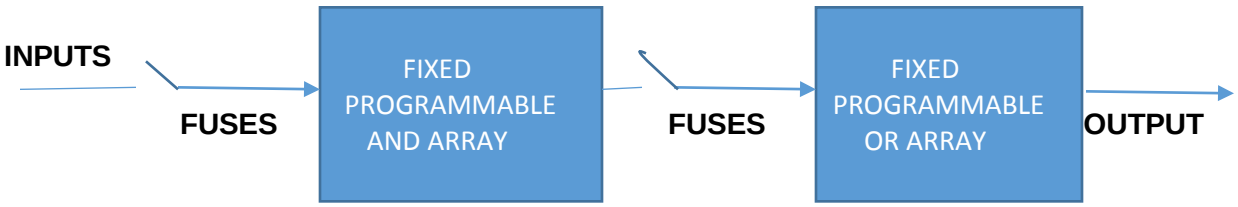
In general, a 2k x n ROM will have k x 2k decoder and n OR gates with 2k x ninternal connections.



3. **Draw the basic block diagram of PLA device and explain each block. List out its applications. Implement a combinational circuit using PLA by taking a suitable Boolean function. (AUC NOV 2011)**

A programmable logic array (PLA) is a kind of programmable logic device used to implement combinational logic circuits. The PLA has a set of programmable AND gate planes, which link to a set of programmable OR gate planes, which can then be conditionally complemented to produce an output. This layout allows for a large number of logic functions to be synthesized in the sum of products (and sometimes product of sums) canonical forms.





EXAMPLE

F1(A ,B ,C) = Σ(0, 1, 2, 4)
F2(A,B ,C) = Σ(0, 5, 6, 7)

A \ BC	00	01	11	10
0	1	1	0	1
1	1	0	0	0

$F_1 = A'B' + A'C' + B'C'$

$F_1 = (AB + AC + BC)'$

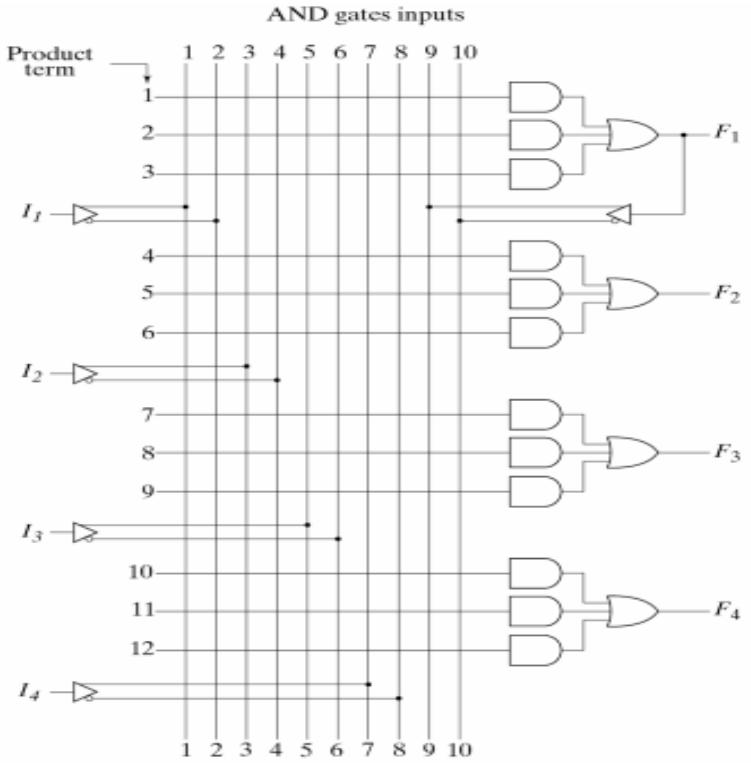
A \ BC	00	01	11	10
0	1	0	0	0
1	0	1	1	1

$F_2 = AB + AC + A'B'C'$

$F_2 = (A'C + A'B + AB'C')'$

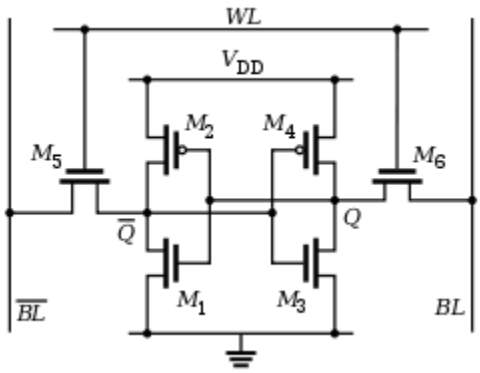
PLA programming table

	Product term	Inputs A	B	C	Outputs (C) F_1	(T) F_2
AB	1	1	1	–	1	1
AC	2	1	–	1	1	1
BC	3	–	1	1	1	–
$A'B'C'$	4	0	0	0	–	1



4. **a)Explain the operation of static and dynamic MOS RAM cell with necessary diagrams.(12) (AUC NOV 2011)**

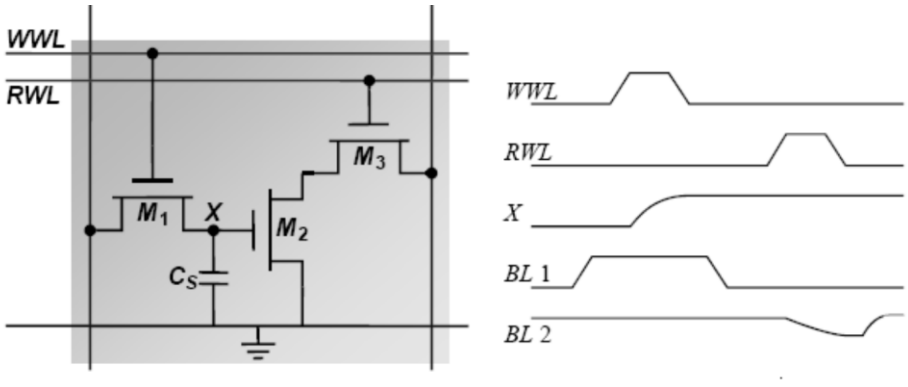
Static Random Access Memory (SRAM)



A single SRAM memory cell has two NMOS and two PMOS transistors (M_1 to M_4) forms the simple latch to store the data and two pass NMOS transistors (M_5 and M_6) are controlled by Word Line to pass Bit Line and $\overline{\text{Bit Line}}$ into the cell. A **Write** operation is performed by first charging the Bit Line and $\overline{\text{Bit Line}}$ with values that are desired to be stored in the memory cell. Setting the Word Line high performs the actual write operation, and the new data is latched into the circuit.

A **Read** operation is initiated by pre-charging both Bit Line and $\overline{\text{Bit Line}}$ to logic 1. Word Line is set high to close NMOS pass transistors to put the contents stored in the cell on the Bit Line and $\overline{\text{Bit Line}}$

Transistors M_1 to M_4 constitute the latch and are constantly toggling back and forth. During these switching the power consumption in CMOS circuits takes place and therefore, the sizes of these transistors are kept as small as possible. NMOS transistors are basically switches opening and closing access to the SRAM cell. To minimize the propagation delay caused by these transistors their sizes are kept relatively larger.



Dynamic Random Access Memory (DRAM)

DRAM stores each bit in a storage cell consisting of a capacitor and a transistor. Capacitors tend to lose their charge rather quickly; thus, the need for recharging. The presence or absence of charge in the capacitor determines whether the cell contains a '1' or a '0'. The **Read** operation begins by precharging the bit line to an intermediate value, $\frac{V_{DD}}{2}$. The word line is raised to a high potential and the charge stored on capacitor is shared with $\frac{V_{DD}}{2}$ that on the bit line. The change in the bit line voltage is given by the change on the bit line capacitor when the charge stored on capacitor C is shared with the bit line. Based on the access pattern, RWMs are classified as random access class and serial memories. FIFO (first-in-first-out) is an example for serial memories. Most memories belong to the random access class, which means memory locations can be read or written in random order. One would expect memories of this class to be called RAM (random access memory); nevertheless for historic reasons, RAM has been reserved for random access RWM memories. That means though most ROM units also provide random access, but the acronym RAM should not be used for them.

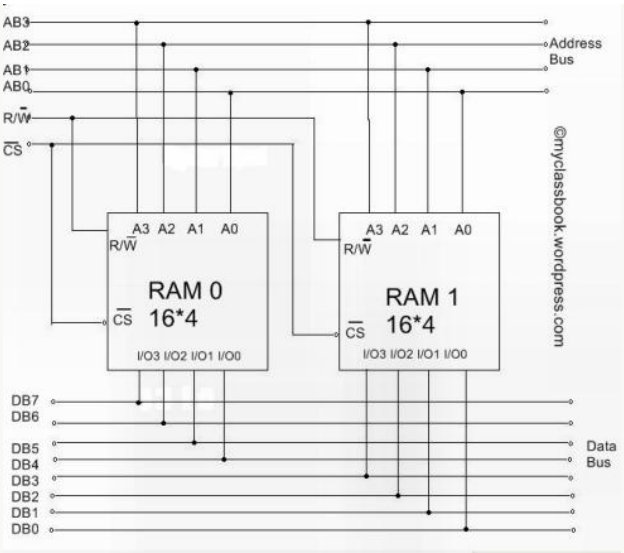
b)What are the advantages of FPGA. (AUC NOV 2011)

- FPGA can be used to implement logic circuits more than a million equivalent gates in size.
- FPGA package pins are very small ,hence more pins can be provided on a relatively small package.
- Speed of operation is high.
- Low power dissipation

5. **(a) (i) We can expand the word size of a RAM by combining two or more RAM chips. For instance, we can use two 32 × 8 memory chips where the number 32 represents the number of words and 8 represents the number of bits per word, to obtain a 32 × 16 RAM. In this case the number of words remains the same but the length of each word will two bytes long. Draw a block diagram to show how we can use two 16 × 4 memory chips to obtain a 16 × 8 RAM. (8) (AUC APR 2010)**

   Following figure shows how to produce 16 X 8 memory RAM using two 16 X 4 memory chips. Since each bit can store 16 4-bit words and since 16 8-bit words are to be stored, each chip is used to store half of each word. In other words, RAM 0 stores the four higher order bits of each of the 16 words, and RAM 1 stores the four lower order bits of each of the 16 words. A full –bit word is available at the RAM outputs connected to the data bus.Any one of the sixteen words is selected by applying the appropriate address code to the four line address bus (AB3, AB2, AB1 and AB0). The address lines typically originate at the CPU. Note that, each address bus line is connected to the corresponding address input of each chip. This means that once an address code is placed on the address bus, the same address code is applied to both chips such that

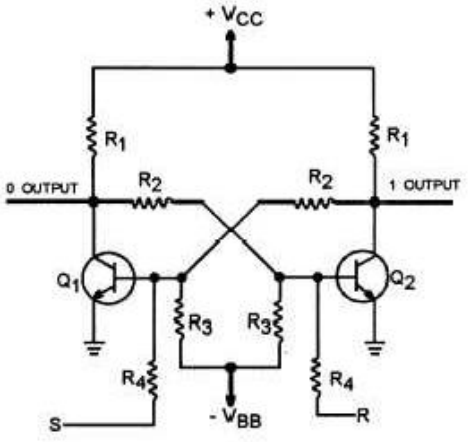
the same location on each chip is accessed at the same time.Once the addressed is selected, we can read or write at this address under the control of the common R/W' and CS' line. To read R/W' must be high and CS' must be low.



Combining two 16*4 RAMs for a 16*8 module

(ii) Explain the principle of operation of Bipolar SRAM cell. (8) (AUC APR 2010)

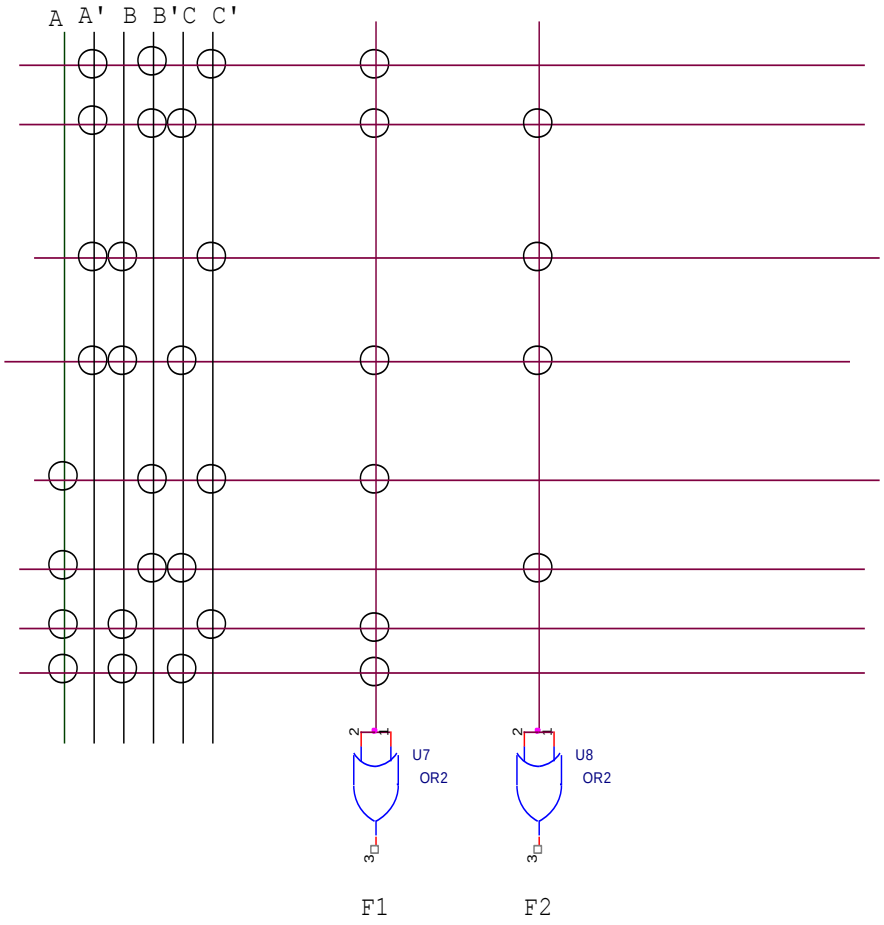
Static Random Access Memory (SRAM)



A single SRAM memory cell has two bipolar transistors (q1 &q2) forms the simple latch to store the data. Q1 and Q2 are controlled by Word Line to pass Bit Line and $\overline{\text{Bit Line}}$ into the cell. A **Write** operation is performed by first charging the Bit Line and $\overline{\text{Bit Line}}$ with values that are desired to be

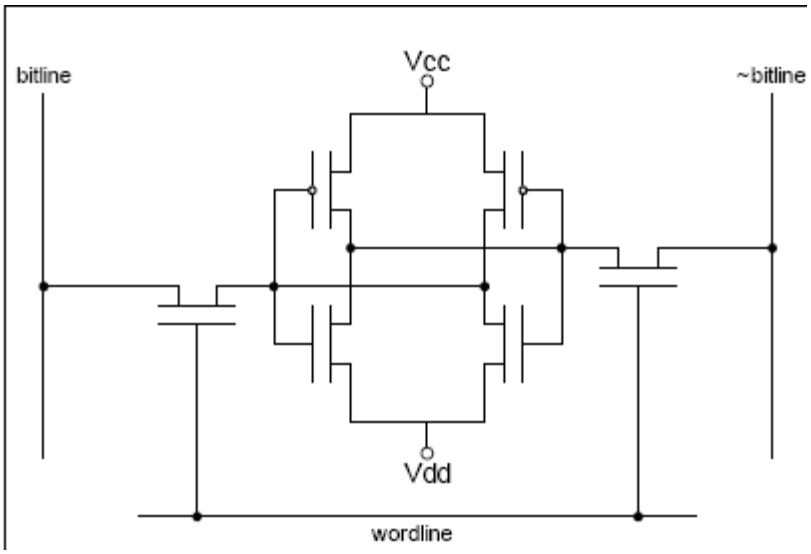
stored in the memory cell. Setting the Word Line high performs the actual write operation, and the new data is latched into the circuit. A **Read** operation is initiated by pre-charging both Bit Line and $\overline{\text{Bit Line}}$ to logic 1. Word Line is set high to close bipolar transistors to put the contents stored in the cell on the Bit Line and $\overline{\text{Bit Line}}$ Transistors q1 & q2 constitute the latch and are constantly toggling back and forth. During these switching the power consumption in CMOS circuits takes place and therefore, the sizes of these transistors are kept as small as possible. NMOS transistors are basically switches opening and closing access to the SRAM cell. To minimize the propagation delay caused by these transistors their sizes are kept relatively larger.

6. **(b) (i) A combinational circuit is defined as the functions(AUC APR 2010)**
   F1 = AB'C'+AB'C+ABC
   F2 = A'BC+AB'C+ABC Implement it using PROM



7. **i)Implement the digital circuit with a PLA having 3 inputs, 3 product terms, and 2 outputs. (8) (AUC APR 2010)**
   (ii) Write a note on SRAM based FPGA. (8) (AUC APR 2010)

FPGA (Field Programmable Gate Array) is an integrated circuitcontaining gate matrix which can be programmed by the user “in thefield” without using expensive equipment.
An FPGA contains a set of programmable logic gates and richinterconnect resources, making it possible to implement complexdigital circuits.FPGA devices are produced by a number of semiconductorcompanies: Xilinx, Altera, Actel, Lattice, QuickLogic and Atmel.



FPGA Implementation Technologies
Configuration bitstream can be stored in FPGA using various technologies. The majority of FPGAsis based on SRAM (Static RAM).
SRAM-based FPGAs
SRAM-based FPGA stores logic cells configuration data in the static memory (organized as an array of latches). Since SRAM is volatile and can't keep data without power source, such FPGAs must be programmed (configured) upon start. There are two basic modes of programming:
• *Master mode*, when FPGA reads configuration data from an external source, such as an external Flash memory chip.
• *Slave mode*, when FPGA is configured by an external master device, such as a processor. This can be usually done via a dedicated configuration interface or via a boundary-scan (JTAG)interface.
SRAM-based FPGAs with an internal flash memory
This type of FPGA is generally like the previous, except that these chips contain internal flash memory blocks, thus eliminating the need to have an external non-volatile memory.

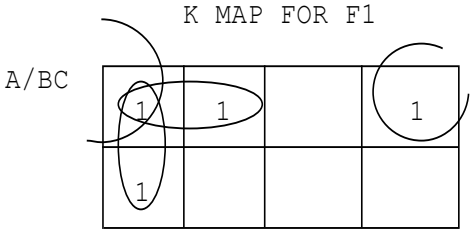
8. **Implement the following Boolean functions with a PLA**

F1(A ,B ,C) = Σ(0, 1, 2, 4)

F2(A,B ,C) = Σ(0, 5, 6, 7)

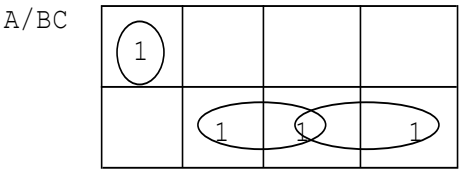
F3(A ,B , C) = Σ(0, 3, 5, 7) . (16) (AUC NOV 2009)

K MAP FOR F1



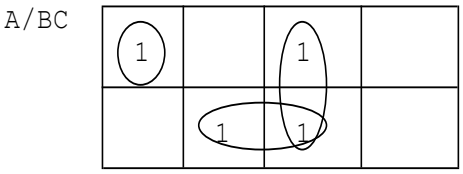
F1=B'C'+A'C'+A'B'

K MAP FOR F2

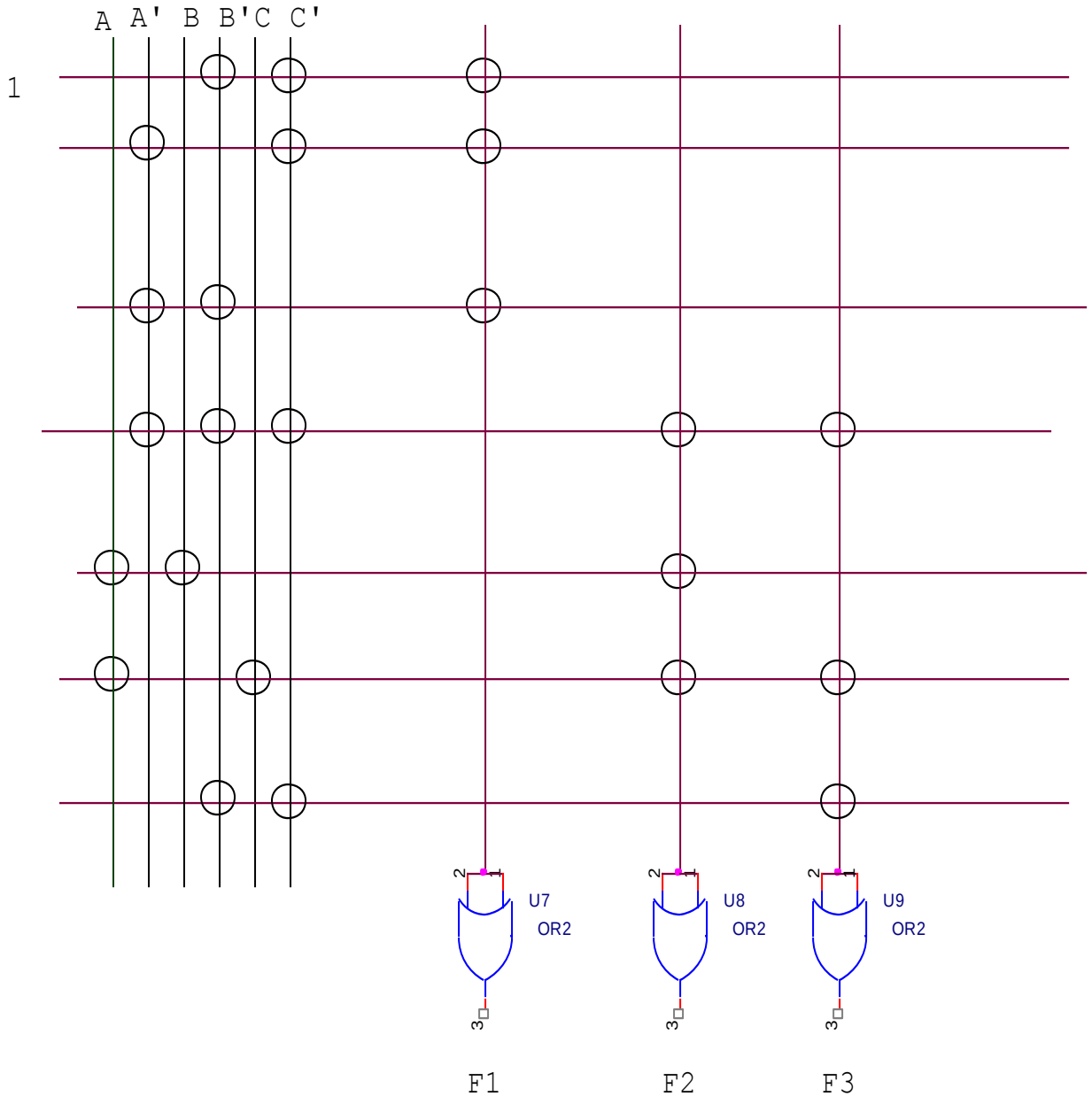


F2=A'B'C'+AC+AB

K MAP FOR F3



F3=A'B'C'+AC+BC


A A' B B'C C'
1
U7
OR2
U8
OR2
U9
OR2
F1
F2
F3

9. Design a combinational circuit using a ROM. The circuit accepts a three bit number and outputs a binary number equal to the square of the input number. (16) (AUC NOV 2009)

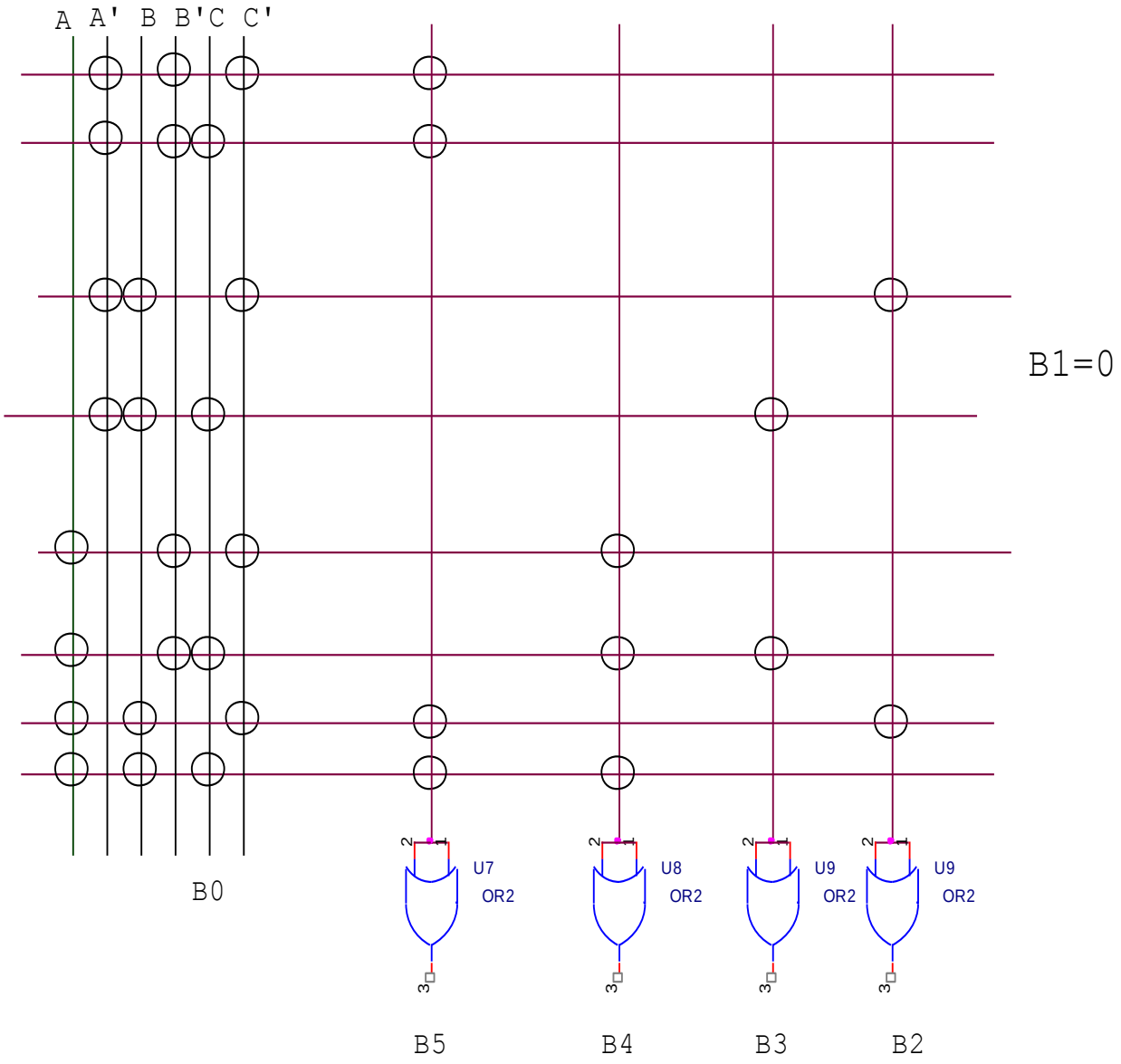
MODIFIED

INPUT			OUTPUT						DECIMAL
A	B	C	B5	B4	B3	B2	B1	B0	
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

TRUTH TABLE

INPUT			OUTPUT				DECIMAL
A	B	C	B5	B4	B3	B2	
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1
0	1	0	0	0	0	1	4
0	1	1	0	0	1	0	9
1	0	0	0	1	0	0	16
1	0	1	0	1	1	0	25
1	1	0	1	0	0	1	36
1	1	1	1	1	0	0	49

LOGIC DIAGRAM

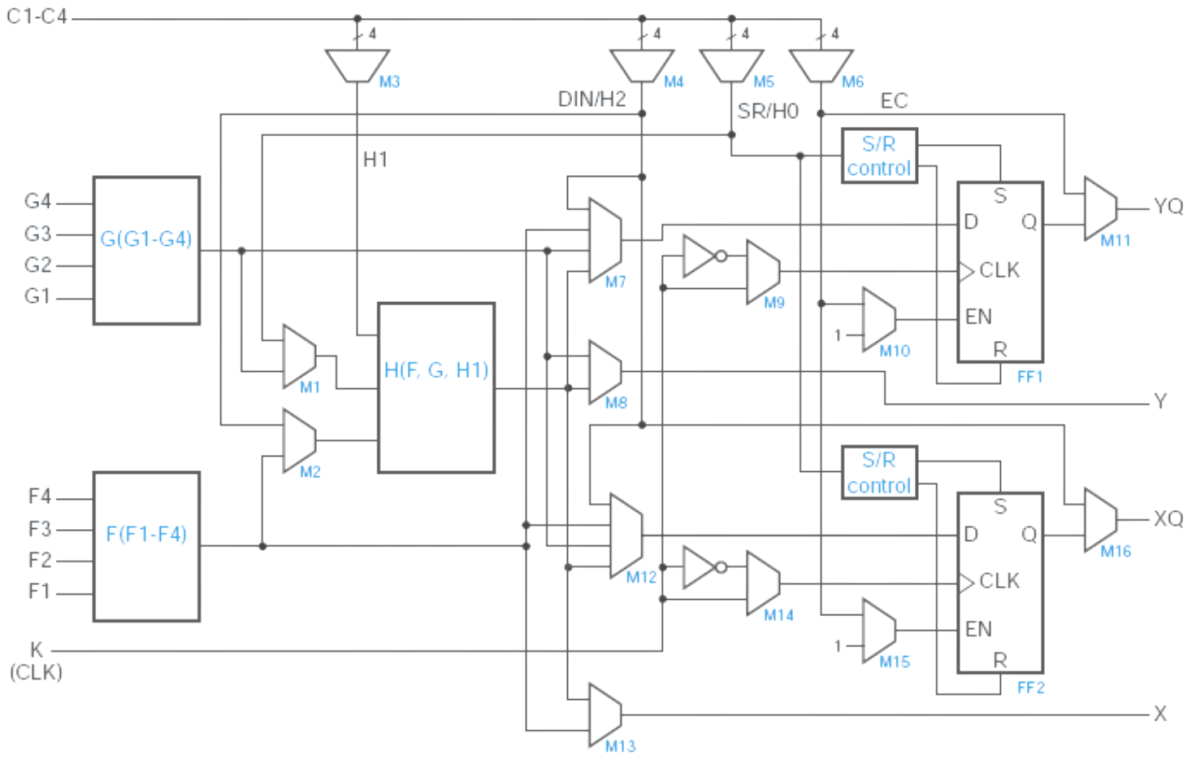


10. Draw the logic diagram of Xilinx 4000 CLB and I/O blocks and explain their function. (AUC NOV 2008)

The programmable logic blocks in the Xilinx XC4000E family of FPGAs arecalled *configurable logic blocks (CLBs)*. The smallest part, the XC3003E,contains a 10 ´ 10 array of CLBs, and the largest, the XC4025E, contains a32 ´ 32 array for a total of 1,024 CLBs.

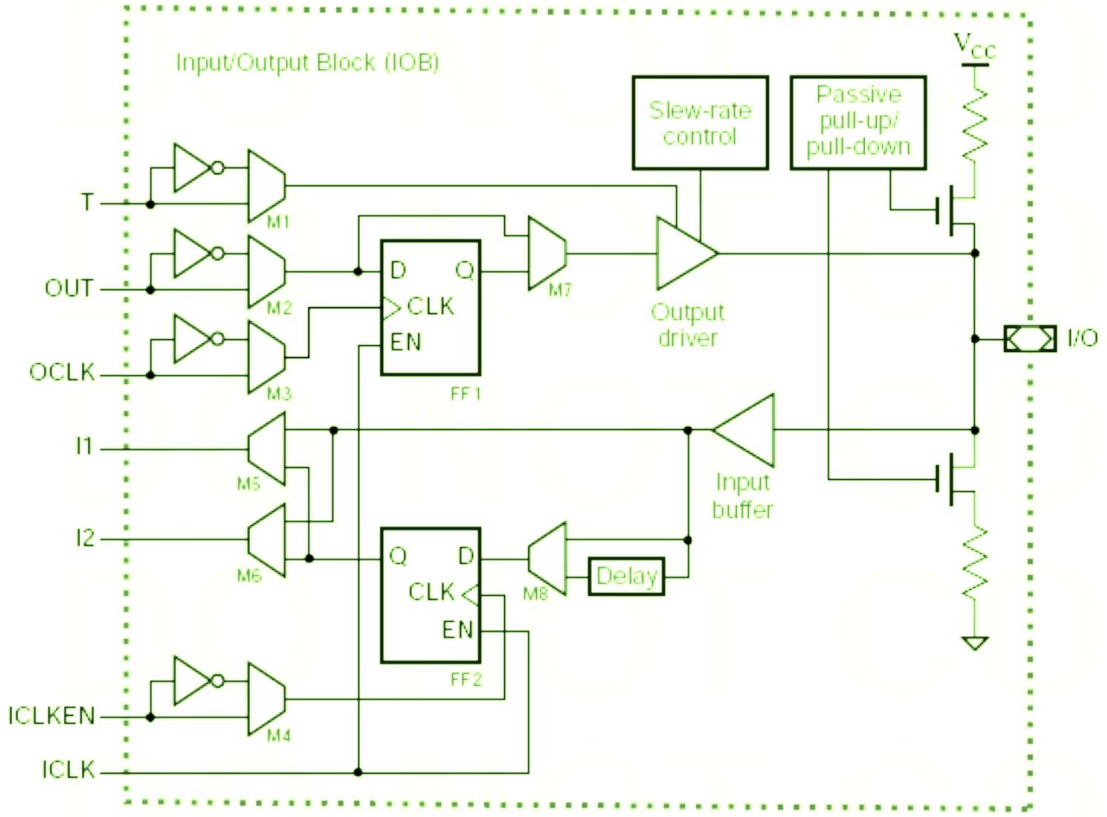
Configurable Logic Block

Figure shows the internal structure of an XC4000-series CLB.The CLBs most important programmable elements are the logic-functiongenerators F, G, and H. Both F and G can perform any combinational logicfunction of their four inputs, and H can perform any combinational logicfunction of its three inputs.



.

Input/Output Block

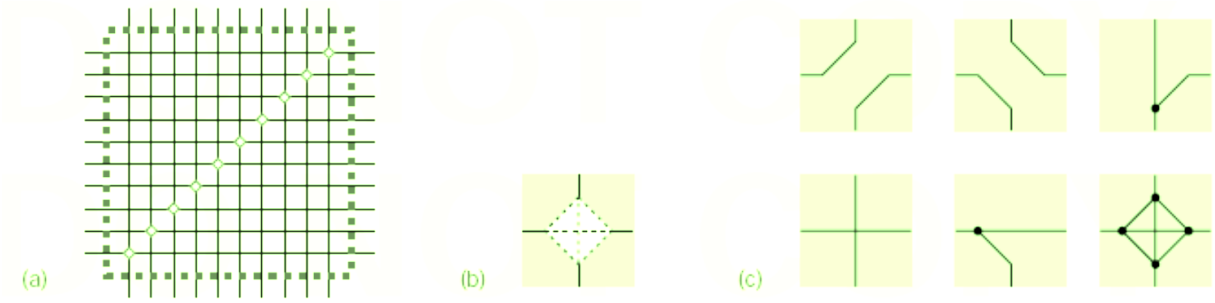


The structure of the XC4000 *I/O block (IOB)* is shown in Figure. An I/O pin can be used for input or output or both.

The XC4000 IOB has more “logic” controls . In particular, its input and output paths contain edge-triggered D flipflops selectable by multiplexers. Placing input and output flip-flops “upclose” to the device I/O pins is especially useful in FPGAs. On output, relativelylong delays from internal CLB flip-flop outputs to the IOBs can make it difficultto connect to external synchronous systems at very high clock rates. On input,long delays from the I/O pins to CLB flip-flop inputs can make it difficult tomeet external system setup and hold times if external inputs are clocked directlyinto a CLB flip-flop without being captured first by a flip-flop at the IOB pin. Ofcourse, using IOB flip-flops is possible only if the FPGA’s external interfacespecifications allow “pipelining” of inputs and outputs.

For pipelined inputs, the XC4000 IOB actually goes one step further byproviding a delayelement, selectable by M8, in series with the D input of inputflip-flop FF2. The effect of this element is to delay the D input relative to theFPGA’s internal copies of the system clock, guaranteeing that the input will havea zero hold-time requirement with respect to the external system clock. Thisbenefit comes at the expense of increased setup time, of course.

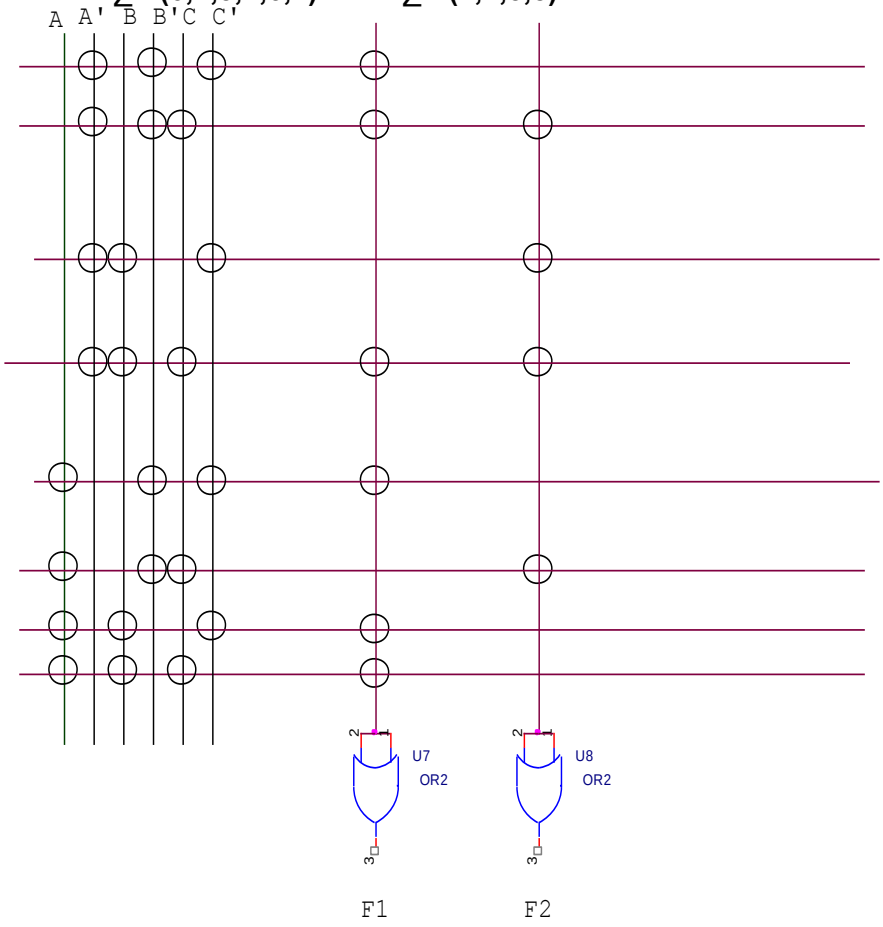
Figure shows a CLB and wires in detail. The smallsquares are programmable connections—a horizontal wire is connected or not toa vertical one, depending on the state of the programming bit (again, in a latch)for that switch. Additional, specialized programmable interconnect is provided at the edges of the CLB array for connections to the IOBs.In the figure, the shaded area in color is called a *programmable switchmatrix (PSM)*. The PSM is shown in more detail in Figure .



Each of the diamonds in (a) is a *programmable switch element (PSE)* that can connect anyline to any other, as shown in (b). With four lines, there are 6 possible pair wise connections as shown in (b), and the PSE has a transmission gate for each one of them. Some, none, or all of the transmission gates in a PSE may be enabled—again, by configuration bits stored in latches. Thus, many different connection patterns are possible, as shown in (c).The PSM is essential for hooking things up in the wiring structure. By enabling and disabling connections, PSEs extend or isolate wire segments in the “Single” and “Double” groups. More importantly, the PSM allows signals to “turn the corner” by connecting a horizontal wire to a vertical one. Without this, CLBs would not be able to connect to others in a different row or column of the array.

11. Implement the given functions using PROM (AUC NOV 2008)

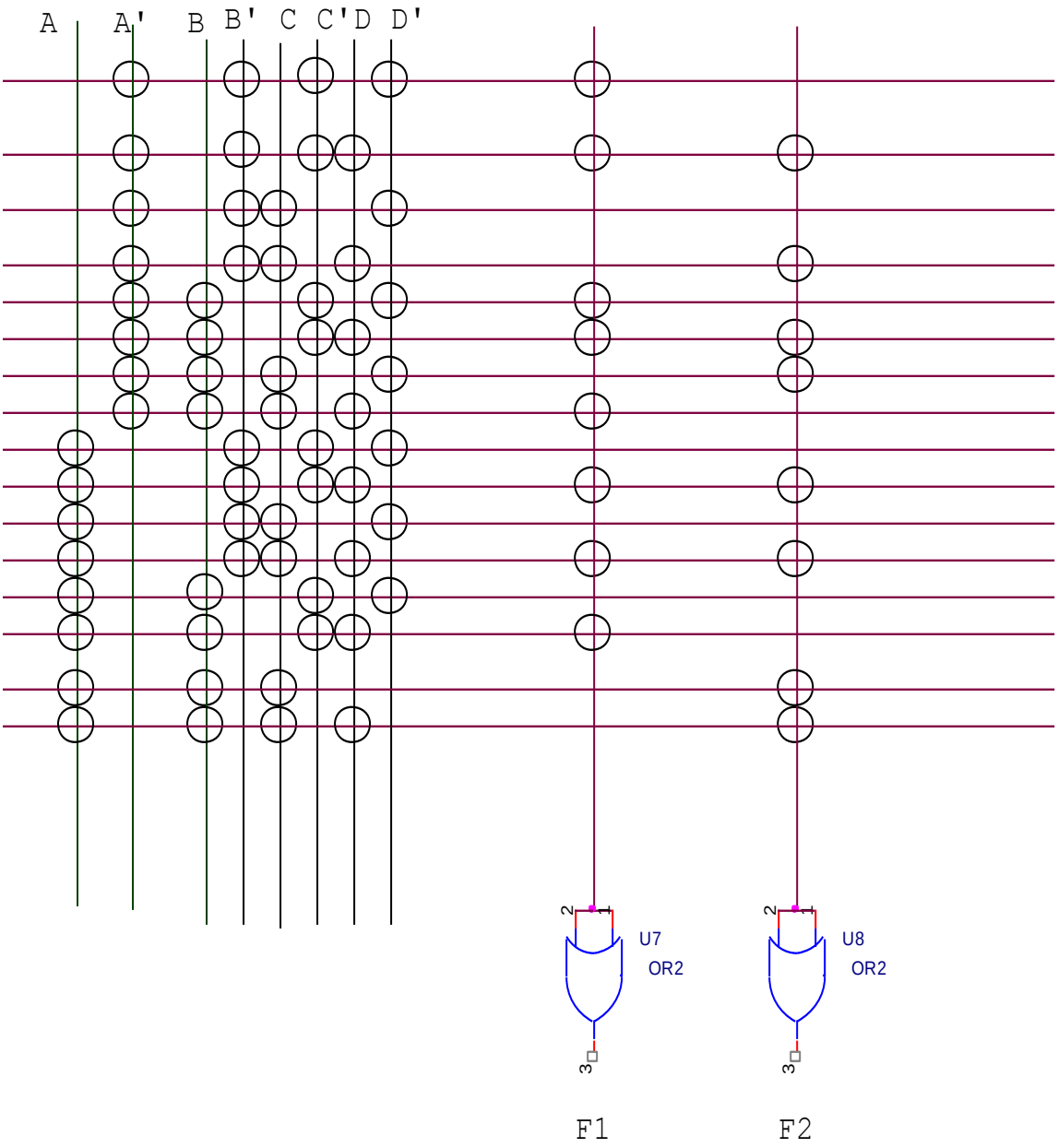
F1=∑m(0,1,3,4,6,7) F2=∑m(1,2,3,5)



12.Design the given functions using PAL and PROM (AUC NOV 2007)

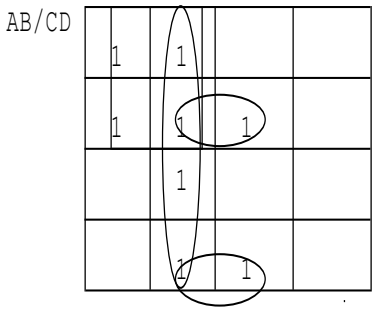
F1=∑m(0,1,4,5,7,9,11,13) F2=∑m(1,3,5,6,9,11,14,15)

Design Using Prom



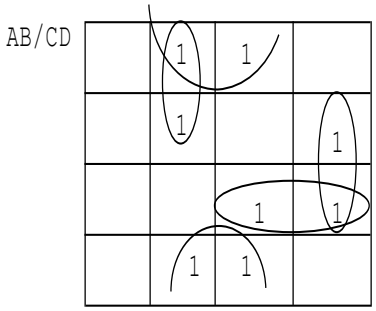
Design Using PAL

K MAP FOR F1

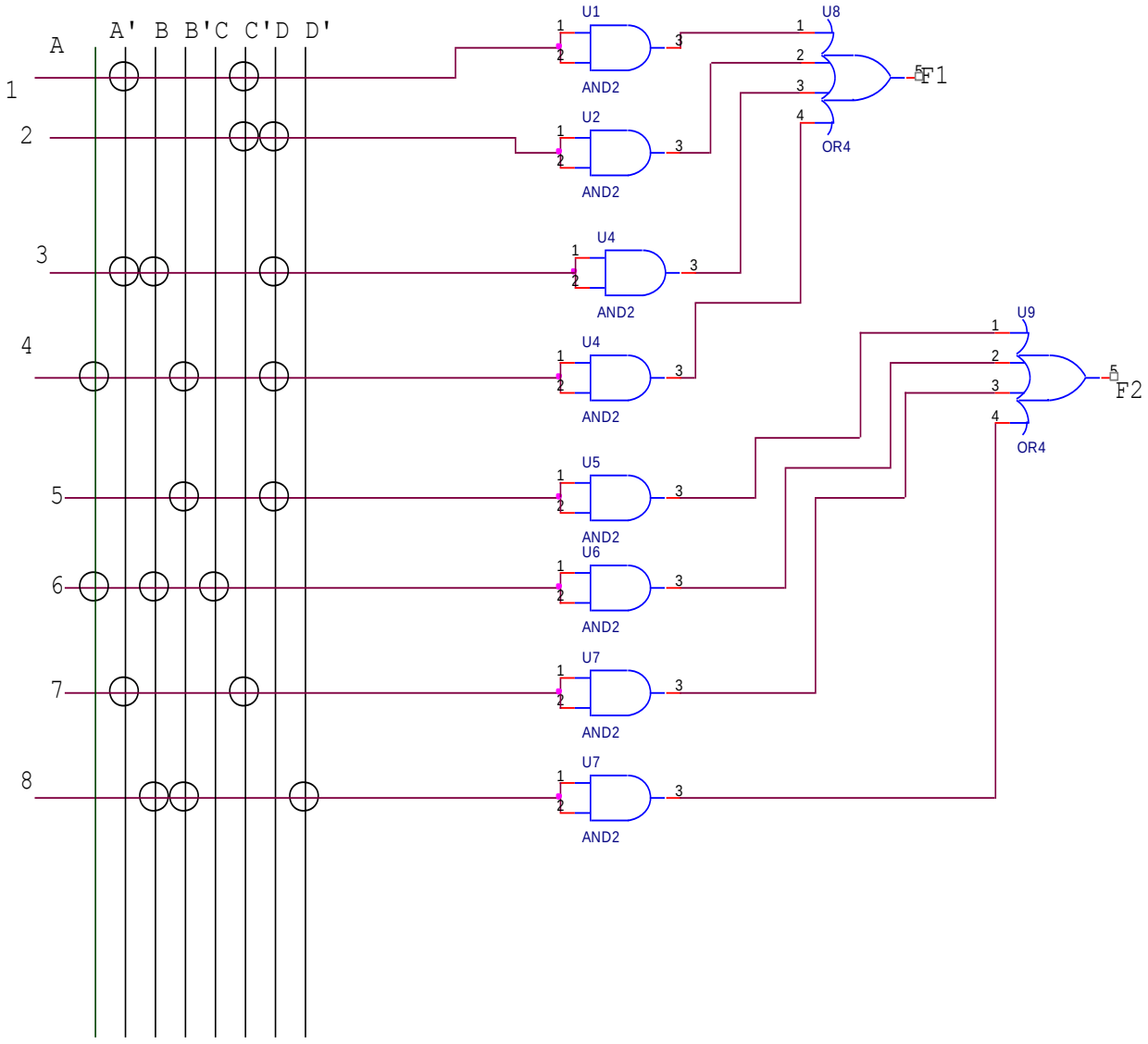


F1=A'D+B'C'D

K MAP FOR Y

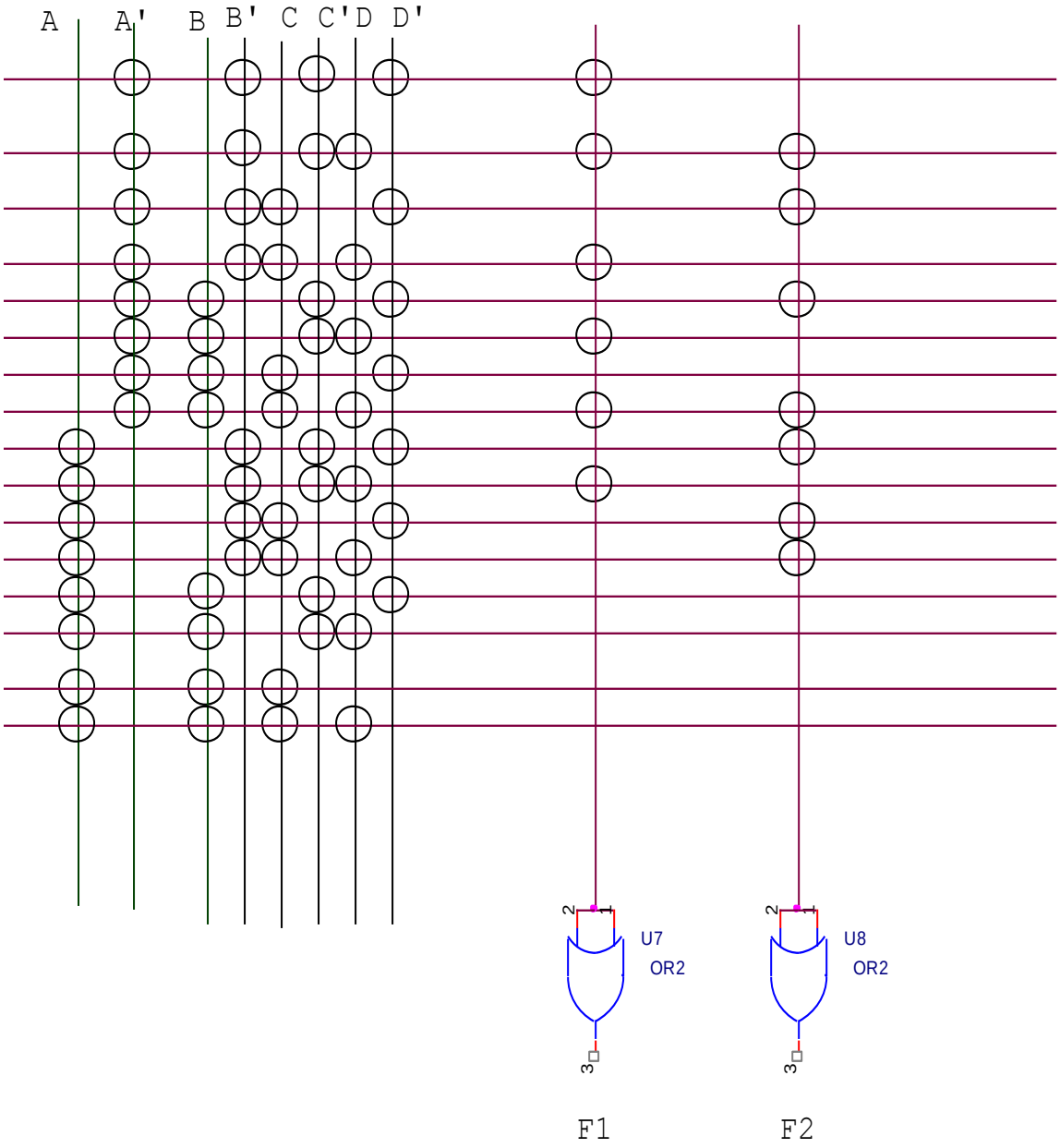


F2=B'D+ABC+A'C'+BCD'


A
A'
B
B'
C
C'
D
D'
1
2
3
4
5
6
7
8
U1
AND2
U2
AND2
U4
AND2
U4
AND2
U5
AND2
U6
AND2
U7
AND2
U7
AND2
U8
OR4
F1
U9
OR4
F2

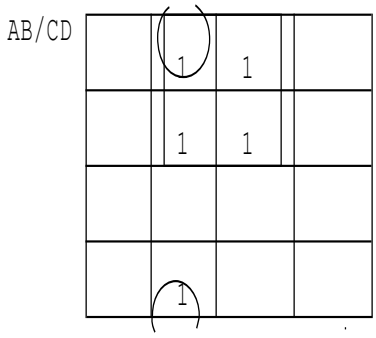
13. Implement the given function using PROM and PAL (AUC JUNE 2007)
F1=∑m(0,1,3,5,7,9) F2=∑m(1,2,4,7,8,10,11)

Design Using Prom



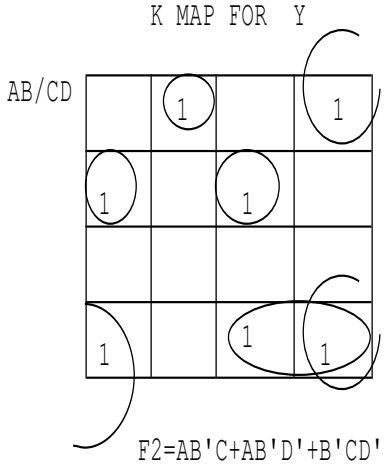
Design Using PAL

K MAP FOR F1

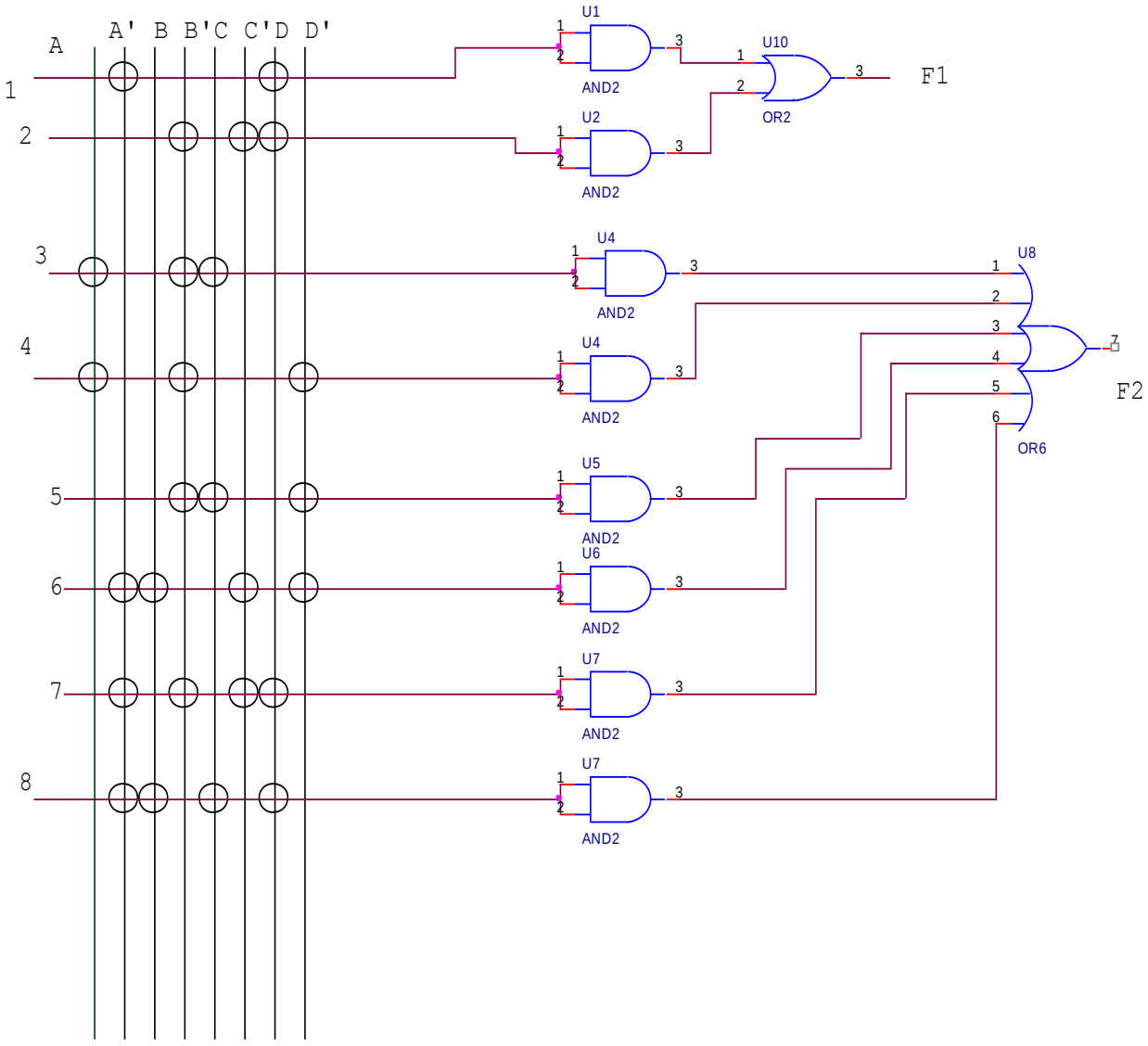


F1=A'D+B'C'D

K MAP FOR Y



F2=AB'C+AB'D'+B'CD'+A'BC'D'+A'B'C'D+A'BCD


A
A'
B
B'
C
C'
D
D'
1
2
3
4
5
6
7
8
U1
AND2
U2
AND2
U10
OR2
F1
U4
AND2
U4
AND2
U5
AND2
U6
AND2
U7
AND2
U7
AND2
U8
OR6
F2