

Q.1

a) **Convert $(45.3)_7$ to BCD, Excess-3 and gray code. State Demorgan's theorem** 05

Ans: $(45.3)_7 = 4 \times 7^1 + 5 \times 7^0 + 3 \times 7^{-1}$
 $= 28 + 5 + 0.436$
 $= (33.426)_{10}$

BCD form:

$(33.426)_{10}$
 $(0011\ 0011\ .\ 0100\ 0010\ 0110)_{BCD}$ 01

Excess-3:

$(33.426)_{10}$
 $+ \quad 3\ 3\ 3\ 3\ 3$

 $\begin{matrix} 6 & 6 & 7 & 5 & 9 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ (0110 & 0110 & 0111 & 0101 & 1001)_{\text{excess-3}} \end{matrix}$ 01

Binary code: $(33.426)_{10} = ()_2$

2	33	
2	16	1
2	8	0
2	4	0
2	2	0
1	0	1
	1	

$(33)_{10} = (100001)_2$

Decimal fraction	Base	= Answer	Recorded bit
0.426	X 2	= 0.852	0
0.852	X 2	= 1.704	1
0.704	X 2	= 1.408	1
0.408	X 2	= 0.816	0
0.408	X 2	= 0.816	0
0.816	X 2	= 1.632	1

$(0.426)_{10} = (0.011001)_2$
 $(33.426)_{10} = (100001.011001)_2$

Binary Number: 1 ExOR 0 ExOR 0 ExOR 0 ExOR 0 ExOR 1 ExOR 0 ExOR 1 ExOR 1 ExOR 0 ExOR 0 ExOR 1 02

Gray Code: $\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$
 $(100001.011001)_2 = (110001.110101)_{\text{gray code}}$

Demorgan's Theorem:

- 1) $\overline{A + B} = \overline{A} \cdot \overline{B}$
- 2) $\overline{A \cdot B} = \overline{A} + \overline{B}$

01

Statement:

- 1) The complement of a sum is equal to the product of the complements.
- 2) The complement of a product is equal to the sum of the complements.

b) **Write the hamming code for 1101.** 05

Ans: Consider for even parity
 Step 1: find the no. of parity bits required

Let $p=3$ then,

$$2^p = 2^3 = 8$$

$$X + P + 1 = 4 + 3 + 1 = 8$$

Three parity bits are sufficient

Therefore total code bits = $4 + 3 = 7$

Step 2: construct a bit location table

Bit designation	D ₇	D ₆	D ₅	P ₄	D ₃	P ₂	P ₁
Bit location	7	6	5	4	3	2	1
Binary location	111	110	101	100	011	010	001
Information Bits	1	1	0		1		
Parity bits				0		1	0

02

Step 3: Determine the Parity bits:

For P₁- check Bit locations 3,5,7

These bit locations have two one's there fore to have even parity P₁ must be 0

Bits			even
3	5	7	0
1	0	1	

01

For P₂- check Bit locations 3,6,7

These bit locations have three one's there fore to have even parity P₂ must be 1

Bits			even
3	6	7	1
1	1	1	

01

For P₄- check Bit locations 5,6,7

These bit locations have three one's there fore to have even parity P₄ must be 0

Bits			even
5	6	7	0
0	1	1	

Enter the parity bits into table

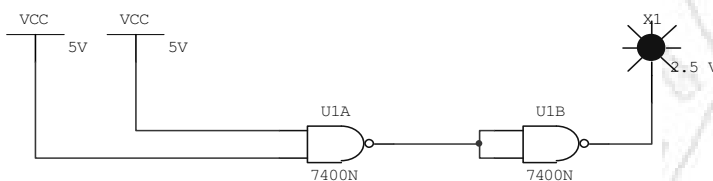
So Hamming code is **1100110**

c) **Justify, NAND gate is a Universal logic gate**

Ans: These **NAND gate** is commonly referred to as the **Universal Gates**, because it can be used to create the basic AND, OR and NOT gate.

Implementation of all the basic gates Using NAND Gate.

AND gate:



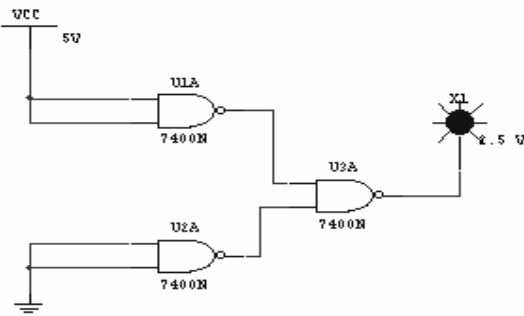
01

NOT gate:



01

OR gate:



02

Hence, NAND gate is a Universal logic gate

d) **Design a full adder using suitable Decoder**

05

Ans: Full Adder :

- A logical circuit for addition of two one bit numbers with lower order carry is called as full adder.

Truth table:

Inputs			Sum	Carry
A	B	C		
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

Write expression for sum and carry.

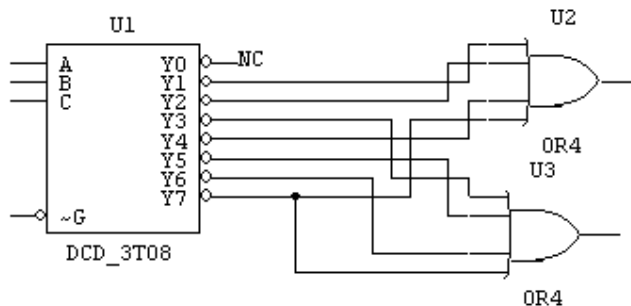
Sum: $\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$
 $\sum m(1,2,4,7)$

Therefore Y_1, Y_2, Y_4, Y_7 should be ORed to get sum output

Carry: $AC + BC + AB$
 $\sum m(3,5,6,7)$

Therefore Y_3, Y_5, Y_6, Y_7 should be ORed to get CARRY output

Circuit diagram:



03

02

Q.2 *Given the logic expression* 10

a) $AB + \overline{A}\overline{C} + C + AD + \overline{A}\overline{B}C + ABC$

- i) *Express in standard SOP & POS form*
- ii) *Draw the K-map for the equation*
- iii) *Minimize & realize using NAND gates only*

Ans: i) standard SOP & POS form

$$AB + \overline{A}\overline{C} + C + AD + \overline{A}\overline{B}C + ABC$$

$$AB(C+\overline{C})(D+\overline{D}) + AC(B+\overline{B})(C+\overline{C}) + C(A+\overline{A})(B+\overline{B})(D+\overline{D}) + AD(B+\overline{B})(C+\overline{C}) + ABC(D+\overline{D}) +$$

$$= ABC(D+\overline{D})$$

= 03

$$\overline{A}BCD + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}\overline{D} +$$

$$\overline{A}BCD + \overline{A}BC\overline{D}$$

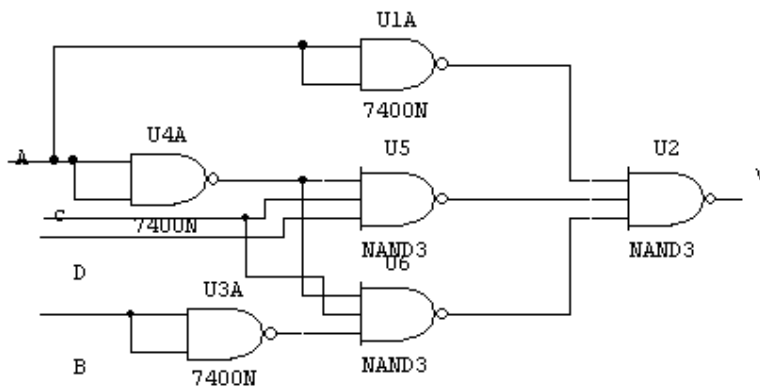
CD	0	0	1	1
	0	1	1	0
AB				
00	0	0	1	1
01	0	0	1	0
11	1	1	1	1
10	1	1	1	1

$$Y = A + \overline{A}CD + \overline{A}\overline{B}C$$

POS FORM:

$$Y = \overline{A}\overline{C} + \overline{A}B\overline{D}$$

iii) using NAND gates only



b) *Minimize using Quine Mc Cluskey's Method* 10

$$F(A,B,C,D) = \sum m(0,1,3,5,7,9,11) + d(2,14)$$

Ans:

Step1: list all the minterms in binary form 01

Step2: arrange the minterms according to number of 1's 01

Step3: compare each binary number with every term in the adjacent next high category & if they differ only by 1 position, put a check mark & copy the term in the next column with '_' in the position that they differ. 02

minterms	binary	minterms	binary	minterms	Binary	minterms	Binary
m ₀	0000√	m ₀	0000√	0,1	000_√	0,1,2,3	00__
m ₁	0001√	m ₁	0001√	0,2	00_0√	1,3,5,7	0__1
m ₃	0011√	m ₂	0010√	1,3	00_1√	1,3,9,11	_0_1
m ₅	0101√	m ₃	0011√	1,5	0_01√		
m ₇	0111√	m ₅	0101√	1,9	_001√		
m ₉	1001√	m ₉	1001√	2,3	001_√		
m ₁₁	1011√	m ₇	0111√	3,7	0__11√		
m ₂	0010√	m ₁₁	1011√	3,11	_011√		
m ₁₄	1110√	m₁₄	1110				
Step 1		Step 2		Step 3		Step 4	

02

Step 5: list the prime implicants:

prime implicants		
m₁₄	1110	$\overline{A}BC\overline{D}$
0,1,2,3	00__	$\overline{A}\overline{B}$
1,3,5,7	0__1	$\overline{A}D$
1,3,9,11	_0_1	$\overline{B}D$

01

STEP 6: Select minimum no. of prime implicants which must cover all the minterms except don't care minterms.

prime implicants	m ₀	m ₁	m₂	m ₃	m ₅	m ₇	m ₉	m ₁₁	m₁₄
14	•								•
0,1,2,3 √		•	•	•					
1,3,5,7 √		•		•	•	•			
1,3,9,11 √		•		•			•	•	

02

01

$$Y = \overline{A}\overline{B} + \overline{A}D + \overline{B}D$$

Q.3

a) **Design a two bit magnitude comparator circuit using gates.**

10

Ans: A binary comparator is a logical circuit which carries out the comparison between 2 generally noted binary numbers **A** and **B**.

It has 3 noted exits **A = B**, **A > B** and **A < B** which indicate the result of the comparison as follows :

01

▶ If number **A** is equal to the number **B** (**A = B**), the exit **A = B** passes to state **1** while the exits **A > B** and **A < B** pass to state **0**.

▶ If number **A** is strictly higher than the number **B**, only the exit **A > B** passes to state **1**.

▶ If number **A** is strictly lower than the number **B**, only the exit **A < B** passes to state **1**.

The truth table for 2-bit comparator:

Input				output		
A ₁	A ₀	B ₁	B ₀	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1

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

03

K-map simplification:

A>B

A=B

B_0B_1	0	01	11	10
A_1A_0	00	0	0	0
	01	1	0	0
	11	1	1	0
	10	1	1	0

B_0	00	0	1	1
B_1		1	1	0
A_1	00	1	0	0
A_0	01	0	1	0
	11	0	0	1
	10	1	0	0

03

A>B =

$$A_0 \bar{B}_1 \bar{B}_0 + A_1 \bar{B}_1 + A_1 A_0 \bar{B}_0$$

$$B_0) (A_1 \cdot B_1)$$

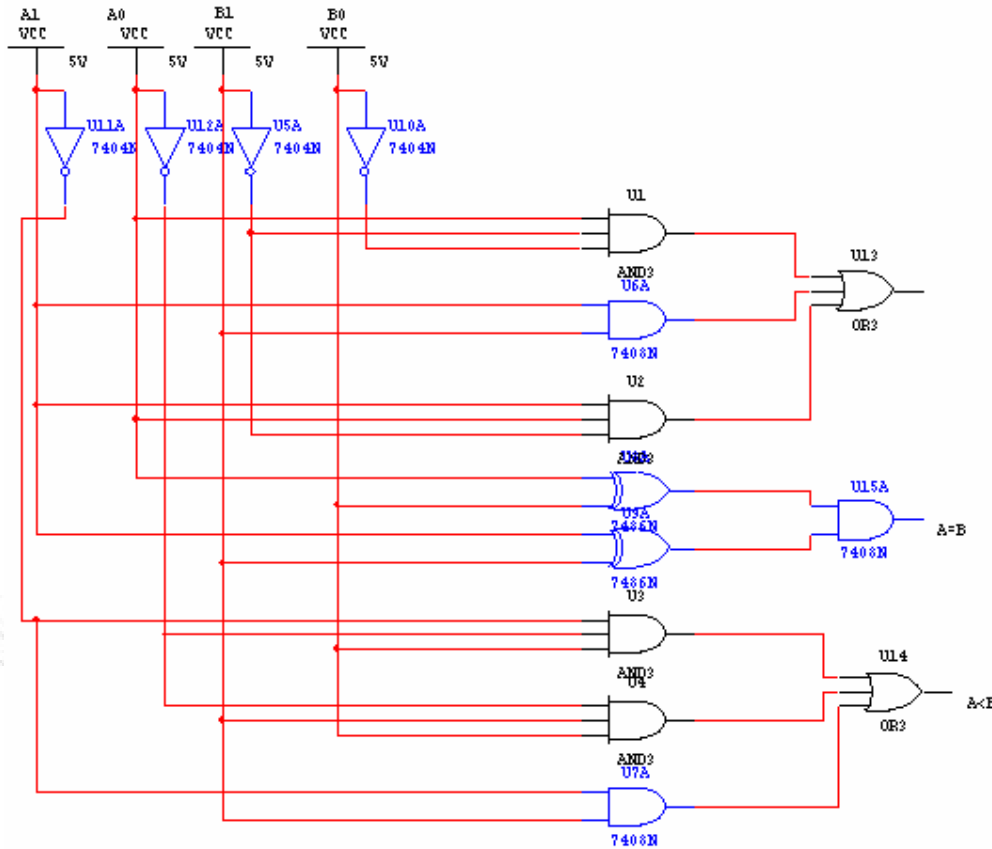
$$(A=B) = (A_0 \cdot$$

0

$$A < B = \bar{A}_1 \bar{A}_0 B_0 + \bar{A}_0 B_0 B_1 + \bar{A}_1 B_1$$

B_0	00	0	1	1
B_1		1	1	0
A_1	00	0	1	1
A_0	01	0	0	1
	11	0	0	0
	10	1	0	0

DIAGRAM:



03

b) **Design a BCD to seven segment Display decoder**

10

Ans; **BCD to seven-segment display decoder**

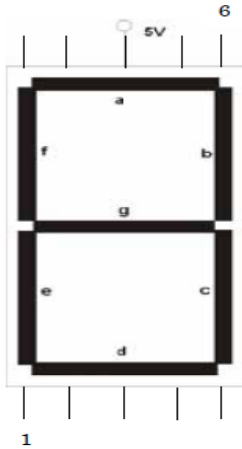
A **seven-segment display** may have 7, 8, or 9 leads on the chip. Usually leads 8 and 9 are decimal points. The figure below is a typical component and pin layout for a seven segment display.

IC-7447

- The digital display that consists of seven LED segment is commonly used to display decimal numerals in digital system.
- Most familiar examples are electronics calculators and watches where one 7 segments display device is used for displaying one numeral 0 to 9 for using this display device the data has to be converted form some binary code to the code required for display.
- In seven segments as name indicates we have seven LED segments with one decimal point LED the Arrangement is show in fig. the arrangement is like English 8. The reason for this is with this arrangement you can display digit (0-9) and same alphanumeric characters like A, B, C, D, E, F, G, H & so on.
- Truth table gives the Truth table of BCD to 7 segment decoder.

01

Here A, B, C, D is the natural BCD code for numerals 0 to 9.the k-map for each of output a through.9 are given in fig. the entries in k-map corresponding to the six binary combination not used in the truth table are X don't care.

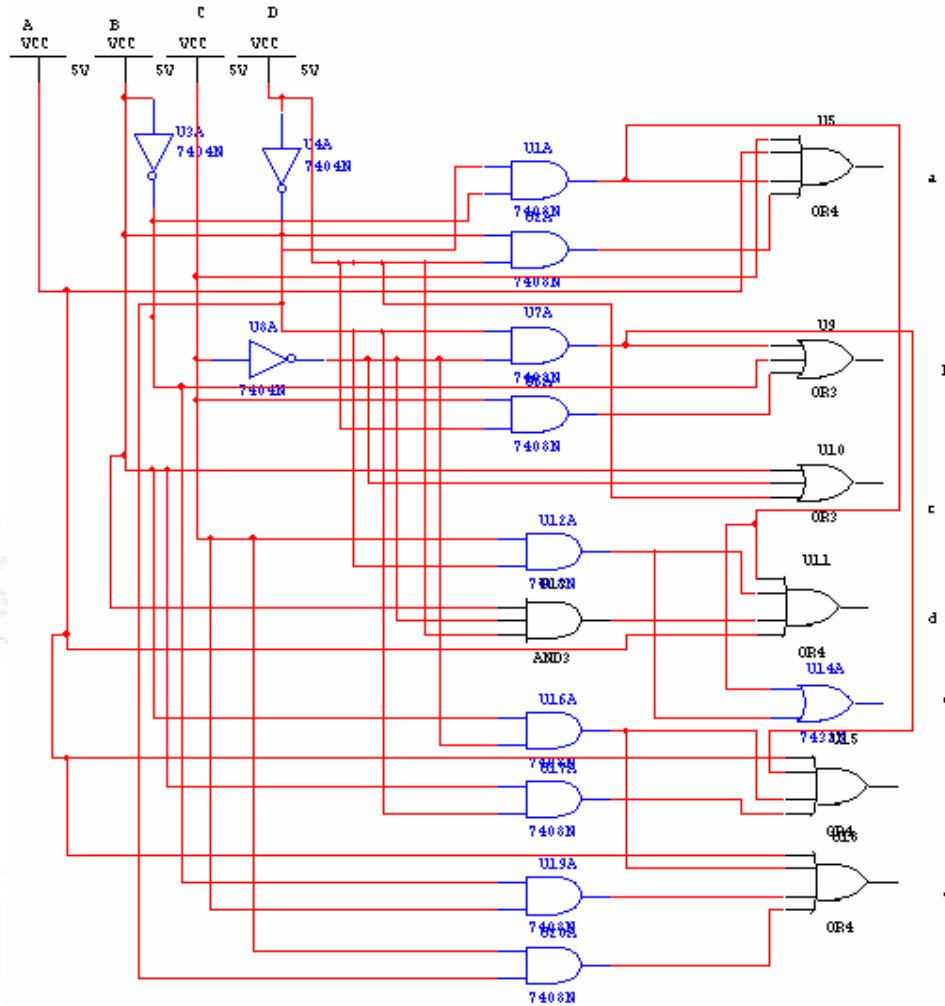


03

Q4	Q3	Q2	Q1	O/P	Display	Glowing LEDs
0	0	0	0	0	0	a,b,c,d,e,f
0	0	0	1	1	1	b,c
0	0	1	0	2	2	a,b,d,e,g
0	0	1	1	3	3	a,b,c,d,g
0	1	0	0	4	4	b,c,f,g
0	1	0	1	5	5	a,c,d,f,g
0	1	1	0	6	6	a,c,d,e,f,g
0	1	1	1	7	7	a,b,c
1	0	0	0	8	8	a,b,c,d,e,f,g
1	0	0	1	9	9	a,b,c,d,f,g

Truth table:

Digit	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1



Q.4 Implement the following using only one 4:1 MUX

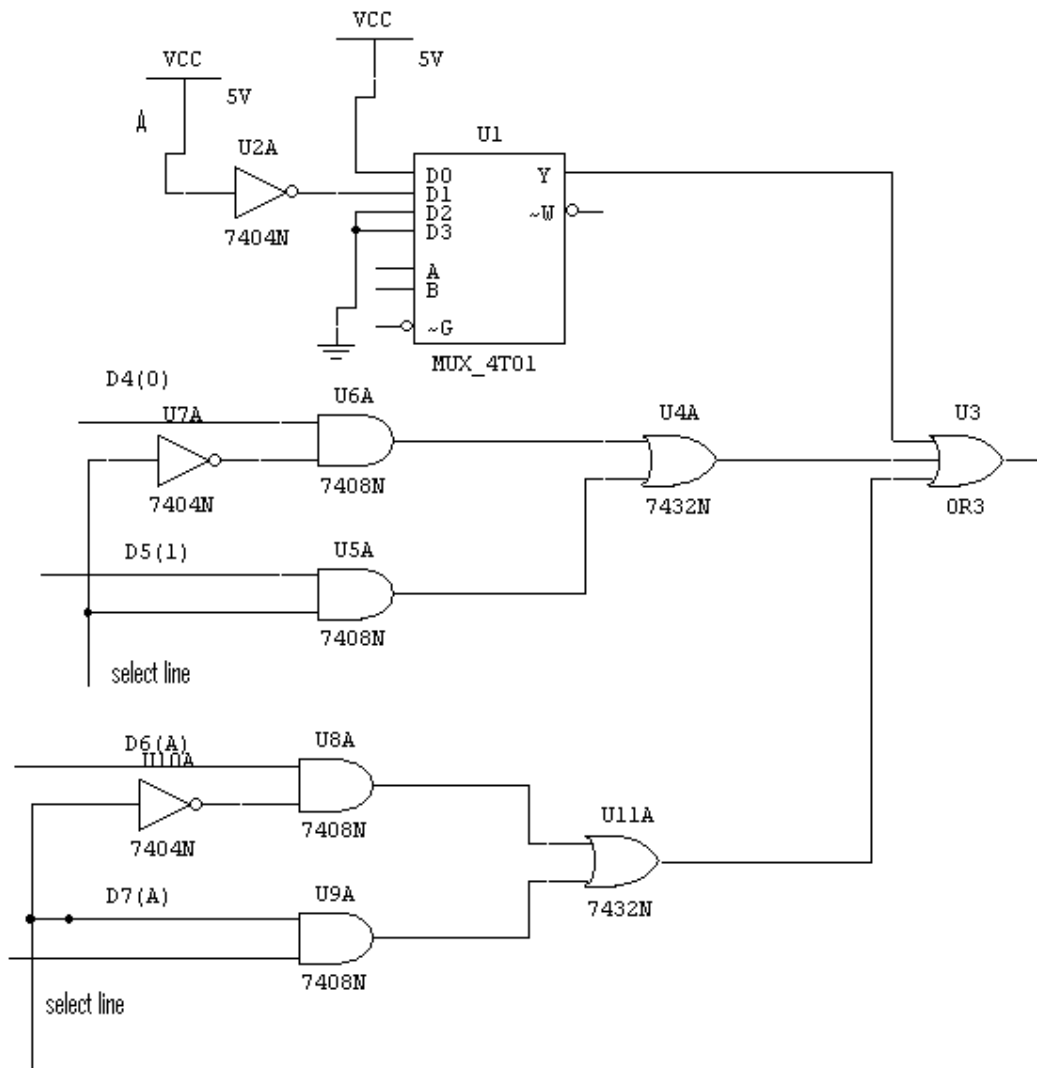
$$F(A,B,C,D) = \sum m(0,1,5,7,8,11,13,14)$$

Ans: Given function has 4-variables A,B,C,D.

I/p	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇
\overline{A}	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
I/P to MUX	1	\overline{A}	0	0	0	1	A	\overline{A}

10

04

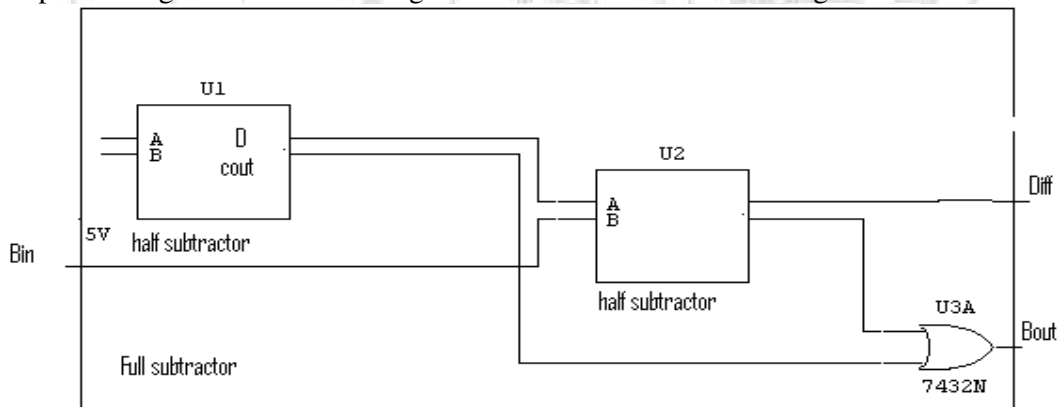


06

b) **Design full subtractor using two half subtractors**

10

Ans: Implementing full subtractor using two half subtractors and an OR gate.



05

Expression for difference out D as:

$$D_n = A_n \oplus B_n \oplus B_{n-1}$$

This is same as the expression for D output of full subtractor

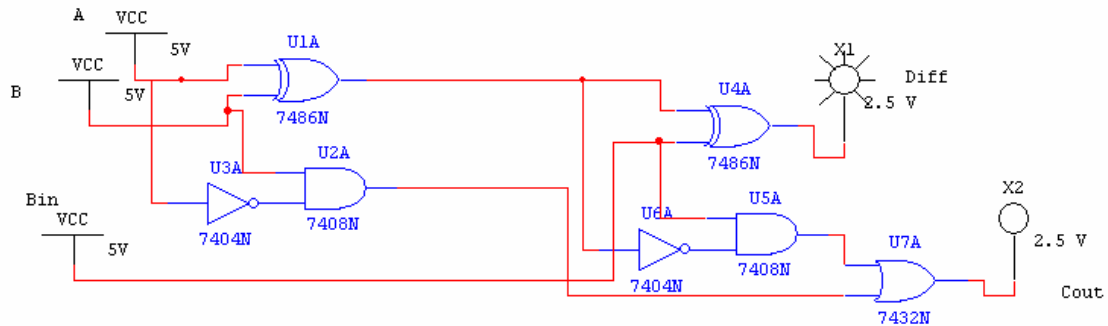
Now the expression for Borrow output B_{out}

$$\text{Borrow} = \overline{A_0}B_n + B_0B_n + \overline{A_0}B_0$$

Expression is exactly same as that for B_0 of the full subtractor.

Thus circuit diagram shown fig below acts as a full subtractor

Circuit diagram:



05

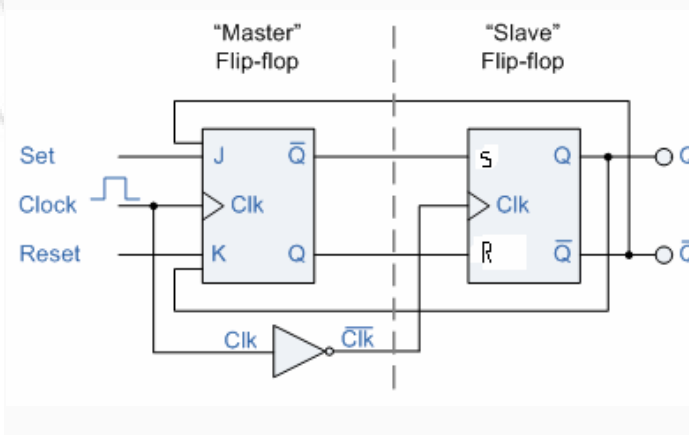
Q.5 Explain the working of master slave JK flip flop. Explain how race condition is overcome by it. 10

Ans: **Master-Slave JK Flip-flop**

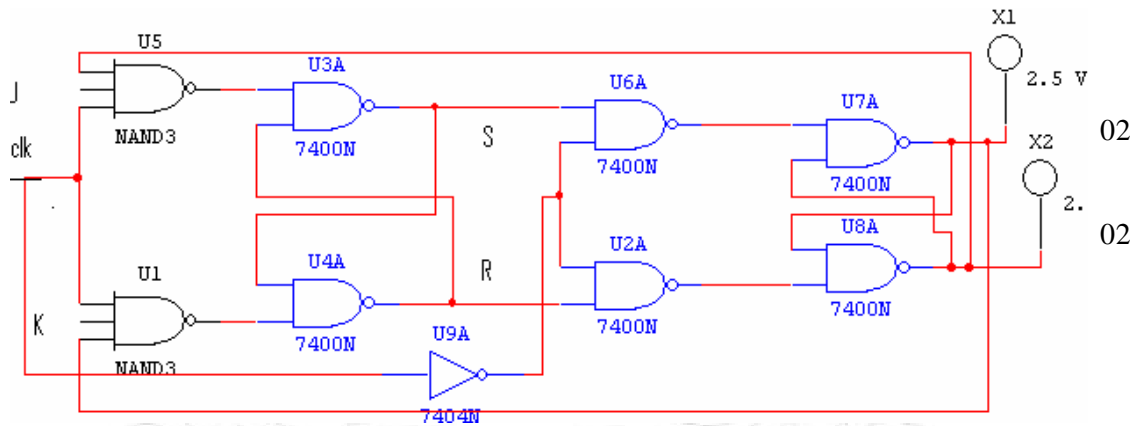
- **Master-Slave JK Flip-flop** eliminates all the timing problems by using two SR flip-flops connected together in series, one for the "Master" circuit, which triggers on the leading edge of the clock pulse and the other, the "Slave" circuit, which triggers on the falling edge of the clock pulse.
- The **Master-Slave Flip-Flop** is basically two JK bistable flip-flops connected together in a series configuration with the outputs from Q and \overline{Q} from the "Slave" flip-flop being fed back to the inputs of the "Master" with the outputs of the "Master" flip-flop being connected to the two inputs of the "Slave" flip-flop as shown below.
- Master is positive level triggered. But due to the presence of the inverter in the clock line, the slave will respond to the negative level.
- Hence when the clock = 1 the master is active and the slave is active and the master is inactive.

03

Master-Slave JK Flip-Flops



03



The input signals J and K are connected to the "Master" flip-flop which "locks" the input while the clock (Clk) input is high at logic level "1".

As the clock input of the "Slave" flip-flop is the inverse (complement) of the "Master" clock input, the outputs from the "Master" flip-flop are only "seen" by the "Slave" flip-flop when the clock input goes "LOW" to logic level "0".

Therefore on the "High-to-Low" transition of the clock pulse the locked outputs of the "Master" flip-flop are fed through to the JK inputs of the "Slave" flip-flop making this type of flip-flop edge or pulse-triggered.

Then, the circuit accepts input data when the clock signal is "HIGH", and passes the data to the output on the falling-edge of the clock signal. In other words, the **Master-Slave JK Flip-flop** is a "Synchronous" device as it only passes data with the timing of the clock signal.

race condition:

Thus as long as J=K=1 & E=1 the output will keep toggling indefinitely. This multiple toggling in the JK FF is called race around condition.

race around condition is avoided by:

- 1) Using edge triggered J K F/F
- 2) Using master slave JK F/F

b)

Convert :

- i) JK to D F/F
- ii) D to T F/F

10

Ans:

Excitation table of all flip flops:

Q_t	Q_{t+1}	S	R	J	K	D	T
0	0	0	x	0	x	0	0
0	1	1	0	1	x	1	1
1	0	0	1	x	1	0	1
1	1	x	0	x	0	1	0

05

i) JK to D F/F

Excitation table :

D	Q_t	Q_{t+1}	J	K
0	0	0	0	x
1	0	1	1	x
0	1	0	x	1
1	1	1	x	0

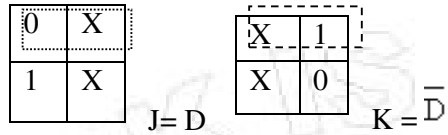
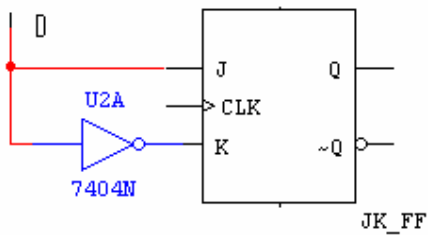
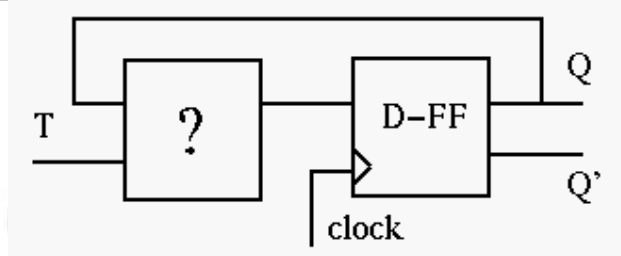


diagram:



ii) **D to T F/F**

05



We need to design the circuit to generate the triggering signal D as a function of T and Q:

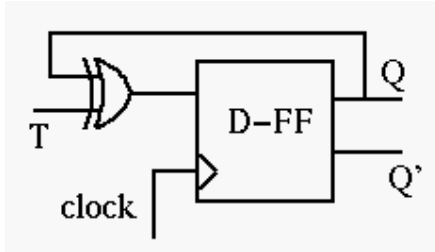
$$D = f(T, Q)$$

Consider the excitation table:

Q_t	Q_{t+1}	T	D
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

Treating D as a function of T and current FF state $Q(Q_t)$, we have

$$D = T'Q + TQ' = T \oplus Q$$



Q.6 **What is shift register? Explain 4 bit shift register.**

10

a)

Ans

SHIFT REGISTER

The binary data in a register can be moved within the register from one flip flop to the other or outside it with application of clock pulse.

The register that allow such data transfer are called shift register .

Shift Registers consists of a number of single bit "D-Type Data Latches" connected together in a chain arrangement so that the output from one data latch becomes the input of the next latch and so on, thereby moving the stored data serially from either the left or the right direction.

03

The number of individual Data Latches used to make up **Shift Registers** are determined by the number of bits to be stored with the most common being 8-bits wide. Shift Registers are mainly used to store data and to convert data from either a serial to parallel or parallel to serial format with all the latches being driven by a common clock (Clk) signal making them Synchronous devices.

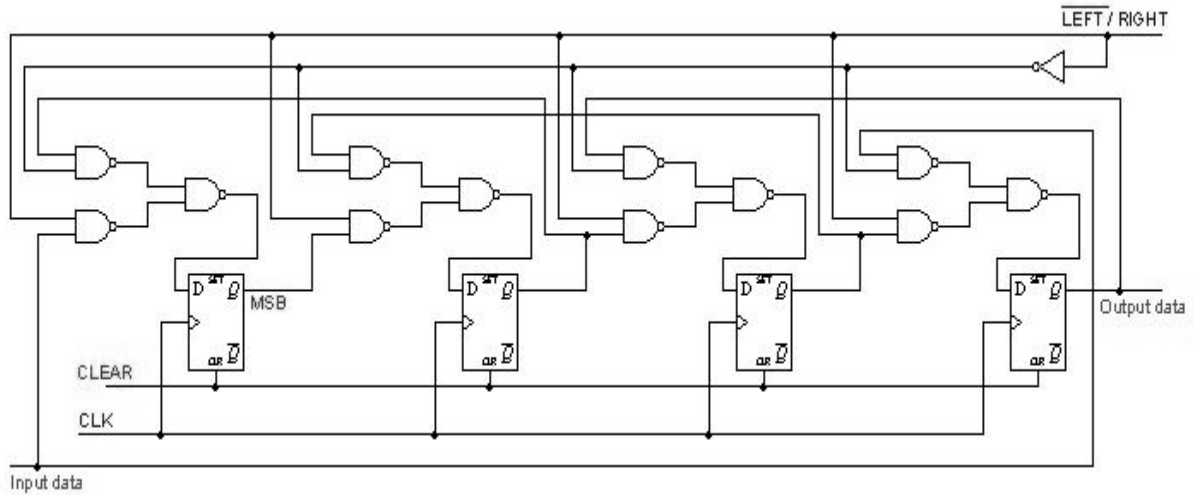
Bidirectional Shift registers.

This type of register allows shifting of data either left or right side. It can be implemented by using logic gate circuitry that enable transfer of data from one stage to the next stage the right or to the left depending on the level of control line.

02

The RIGHT/LEFT ' is the control i/p signal which allows data shifting either to right or left. a high on this line enable the shifting of data to the right and low enables to the left. When RIGHT/LEFT ' signal is high gates G1,G2,G3,G4 are enabled. The state of Q o/p of each flip flop is passed through the D i/p of the following flip flop.

When a clock pulse arrives the data shifted one place right . When the RIGHT/LEFT ' signal is low gates G5,G6,G7,G8 are enabled, the Q o/p of each flip flop is passed through the D i/p of the preceding flip flop. When clock pulse arrives the data are shifted one place left.



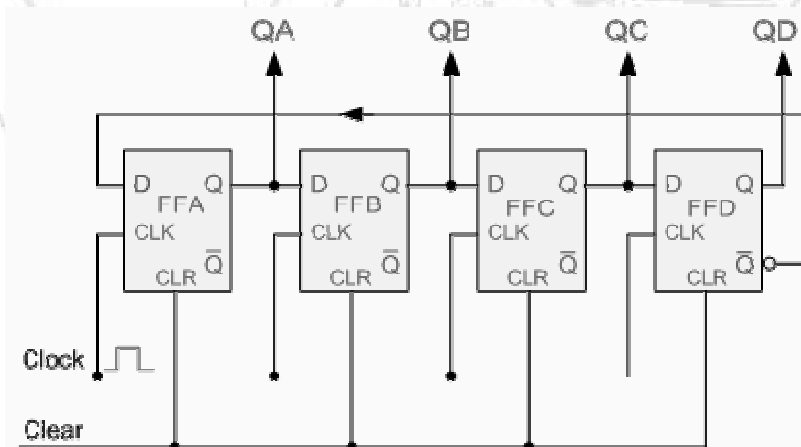
A bidirectional or reversible shift register is one in which the data can be shift either left or right. A four-bit bidirectional shift register using D flip-flops is shown below. Here a set of NAND gates are configured as OR gates to select data inputs from the right or left adjacent bistables, as selected by the LEFT/RIGHT control line. The animation below performs right shift four times, then left shift four times. Notice the order of the four output bits are not the same as the order of the original four input bits.

b) Draw a 4-bit Johnson Ring Counter.

Ans: Johnson Ring Counter

Johnson Ring Counter or "Twisted Ring Counter", are exactly the same idea as the *Walking Ring Counter* above, except that the inverted output Q of the last Flip-flop is connected back to the input D of the first Flip-flop as shown below. The main advantage of this type of ring counter is that it only needs half the number of Flip-flops compared to the standard walking ring counter then its Modulo number is halved.

4-bit Johnson Ring Counter



This inversion of Q before it is fed back to input D causes the counter to "count" in a different way. Instead of counting through a fixed set of patterns like the walking ring

01
10

02

03

counter such as for a 4-bit counter, "1000"(1), "0100"(2), "0010"(4), "0001"(8) etc, the Johnson counter counts up and then down as the initial logic "1" passes through it to the right replacing the preceding logic "0". A 4-bit Johnson ring counter passes blocks of four logic "0" and then four logic "1" thereby producing an 8-bit pattern. As the inverted output Q is connected to the input D this 8-bit pattern continually repeats. For example, "1000", "1100", "1110", "1111", "0111", "0011", "0001", "0000" and this is demonstrated in the table below.

FFA	FFB	FFC	FFD
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
Johnson Ring Counter, Count Sequence			

Standard 2, 3 or 4-stage Johnson Ring Counters can also be used to divide the frequency of the clock signal by varying their feedback connections and divide-by-3 or divide-by-5 outputs are also available

Q.7 **Write short notes on any three of the following**
a) **VHDL Features:**

Ans:

- VHDL is a hardware description language that can be use to model a digital system.
- VHDL is a language used for simulation and synthesis of digital logic.
- A VHDL description of a digital system can be transformed into a gate level implementation. This process is known as synthesis.

HDL describes the hardware of digital systems. This description is in textual form.

- The HDL provides the digital designers with a mean of describing a digital system at a wide range of levels of abstraction and at the same time, provides access to computer aided design tools.
- The HDL represents digital systems in the form of documentation, which can be understood by human.
- It allows hardware designers to express their designs with behavioral constructs.
- The HDL makes it easy to exchange the ideas between the designers.
- HDLs are used to describe hardware for the purpose of simulation, modeling, testing, design and documentation.

Structure of VHDL module:

The main components of a VHDL description consists of following kinds of declarations:

- Package(optional)
- Entity
- Architecture
- configuration(optional)

b) **PAL & PLA**

Ans: • **PLAs** (Programmable Logic Array) - offer flexible features for more complex designs

- **PAL/GALs** (Programmable Array Logic/Generic Array Logic) - offer good flexibility and are faster and less expensive than PLAs

	<i>Device type</i>	<i>AND array</i>	<i>OR array</i>
PLA	Programmable	Programmable	Programmable
PAL	Programmable	Fixed	

PLA is Programmable Logic Array (PLA). The PLA is a PLD that consists of a **programmable AND array and a programmable OR array.**

02

Types:

mask - programmable logic array:

With a mask programmable PLA, the user must submit a PLA program table to the manufacturer.

field programmable logic array:

The second type of PLA is called a field programmable logic array. The user by means of certain recommended procedures can program the EPLA.

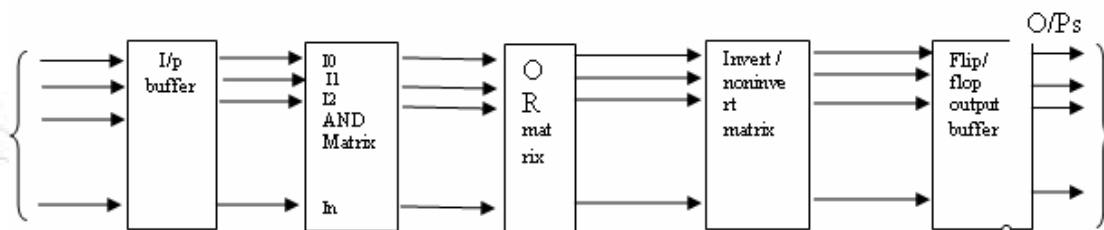
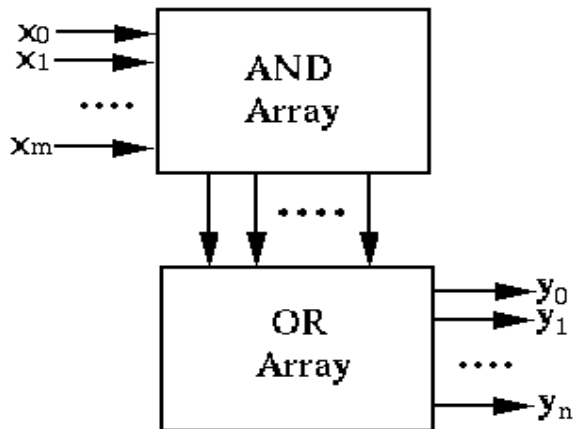


Fig: PLA block diagram



Programmable Logic Array (PLA)

PALs

PAL is Programmable Array Logic. PAL consists of a programmable AND array and a fixed OR array with output logic.

02

major differences between PLA and PAL

PLA:

Both AND and OR arrays are programmable and Complex
Costlier than PAL

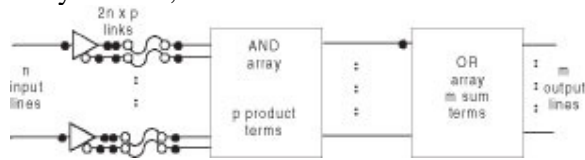
PAL

AND arrays are programmable OR arrays are fixed
Cheaper and Simpler

02

The general structure of this device is similar to PLA, but in a PAL device only AND gates are programmable. The OR array in this device is fixed by the manufacturer. This makes

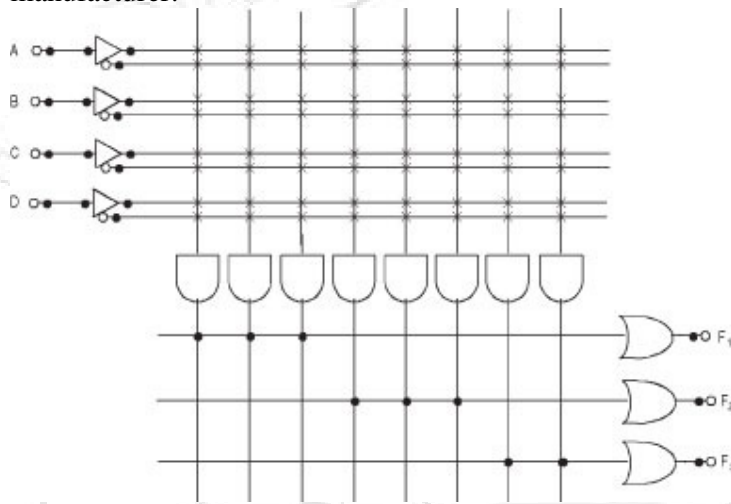
PAL devices easier to program and less expensive than PLA. On the other hand, since the OR array is fixed, it is less flexible than a PLA device.



01
1/2

Above Figure represents the general structure of a PAL device. It has n input lines which are fed to buffers/inverters. Buffers/inverters are connected to inputs of AND gates through programmable links. Outputs of AND gates are then fed to the OR array with fixed connections.

It should be noted that, all the outputs of an AND array are not connected to an OR array. In contrast to that, only some of the AND outputs are connected to an OR array which is at the manufacturer.



c) **Octal to binary encoder**

06
1/2
1/2

Ans: The octal number system uses EIGHT values to represent numbers. The values are,
0 1 2 3 4 5 6 7

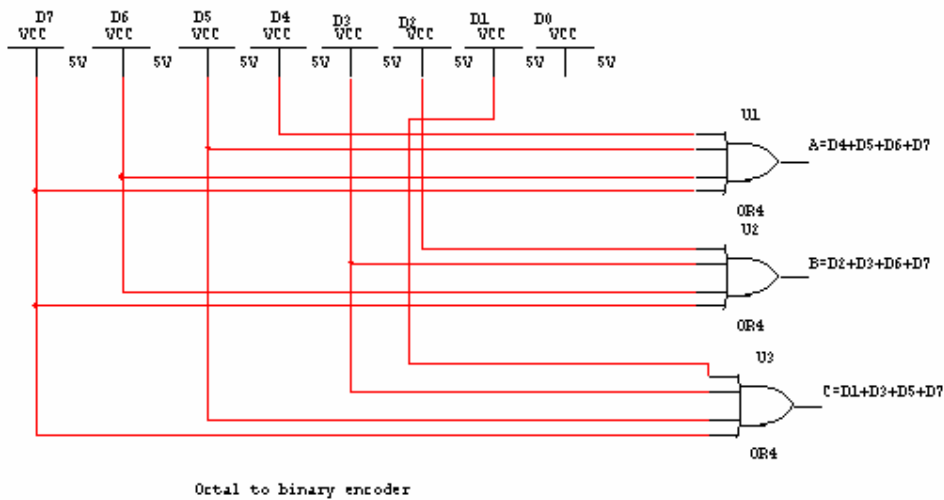
One octal digit is the equivalent value of three binary digits.

- following fig shows octal to binary encoder. It has eight inputs, one for each octal digit, and three outputs that generate the corresponding binary code.

Truth table for octal to binary converter:

Inputs								outputs		
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

03



03

d) **Parity generator**

06
1/2

Ans: A parity bit is used for the purpose of detecting errors during transmission of binary information. A parity bit is an extra bit included with a binary message to make no. of 1s either Odd or Even. The message including the parity bit is transmitted and then checked at the receiving end for errors. An error is detected if the checked parity does not correspond with the one transmitted. The circuit that generates the parity bit in the transmitted is called parity generator and the circuit that checks the parity in the receiver is called parity checker.

In Even parity the added parity bit will make the total number of 1s an even amount. In Odd parity bit will make total number of 1s an odd amount.

01
1/2

The three bits in the message together with the parity bit are transmitted to their destination, where they are applied to the parity checker circuit.

The parity checker circuit checks for possible errors in the transmission since the information was transmitted with even parity.

TRUTH TABLE OF PARITY CHECKER

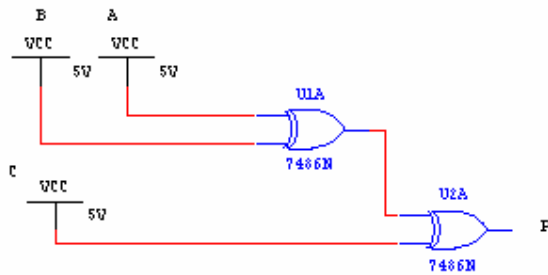
3 Bit Message			Odd parity Bit	Even Parity Bit
A	B			
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

02

This 3 Bit generator/checker features odd/even outputs and control inputs to facilitate operation in either odd- or even-parity applications.

Let us design even parity generator:

$$P = A \text{ ExOr } B \text{ ExOr } C$$



03

e) **Self complementing code**

06

1/2

Ans: There are many possible weights to write a number in BCD code. Some codes have desirable properties, which make them suitable for specific applications. Two such desirable properties are:

1. Self-complementing codes

2. Reflective codes

When we perform arithmetic operations, it is often required to take the “complement” of a given number. If the logical complement of a coded number is also its arithmetic complement, it will be convenient from hardware point of view. In a **self-complementing coded** decimal number, $(A)_{10}$, if the individual bits of a number are complemented it will result in $(9 - A)_{10}$.

01

1/2

Example: Consider the 2421 code.

- The 2421 code of $(4)_{10}$ is 0100.
- Its complement is 1011 which is 2421 code for $(5)_{10} = (9 - 4)_{10}$.

02

Therefore, 2421 code may be considered as a self-complementing code. A necessary condition for a self-complementing code is that the sum of its weights should be 9.

A self-complementing code, which is not weighted, is excess-3 code. It is derived from 8421 code by adding 0011 to all the 8421 coded numbers.

Another self-complementing code is 631-1 weighted code.

Three self-complementing codes are

Decimal Digit	Excess-3 Code	631-1 Code	2421 Code
0	0011	0011	0000
1	0100	0010	0001
2	0101	0101	0010
3	0110	0111	0011
4	0111	0110	0100
5	1000	1001	1011
6	1001	1000	1100
7	1010	1010	1101
8	1011	1101	1110
9	1100	1100	1111

03

References:

- 1) M. Morise Mano,"Digital Logic and computer design "
- 2) R.P.jain, "Modern Digital Electronics"
- 3) Samuel Lee,"Digital circuits and logic design"