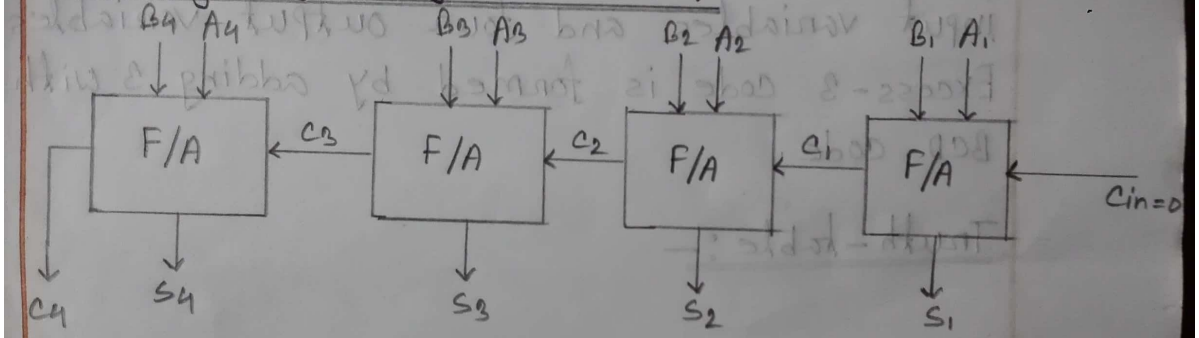
8. Define & Draw the binary Parallel adder (Full 4-bit adder).

Sol^{n} :-

Binary Parallel Adder :- A binary Parallel adder is a digital function that produces the arithmatic sum of two binary numbers in Parallel.

Drawing figure given below :



If consist of full adder connected in casecade with the output canny from one full adder connected to the input canny of the next full adder.

Q. Design BCD to excess-3 code convention.

Ans:-

Code converter/convenesion :- A code converter is a circuit that converts one binary code to another.

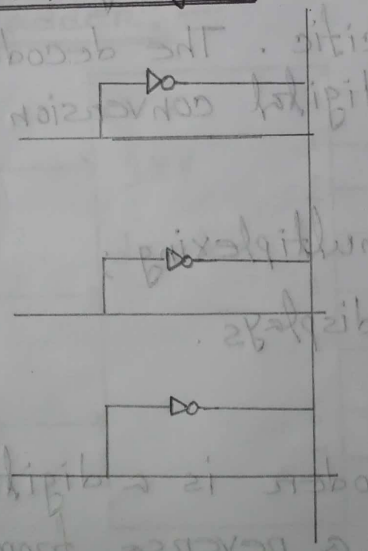
BCD to excess-3 code converter :- In BCD to code and excess-3 code it must have four input variables and four output variables Excess-3 code is formed by adding 3 with BCD code.

Truth-table :-

2016

* **Decoder :-** A decoder is a combinational logic circuit that converts binary information from n input lines to a maximum of 2^n unique output lines.

Circuit Design :-



Application of Decoder:

Decoders are used to analyze data ~~structure~~ ems for a certain data code and given an output if the data is present. Decoders are used whenever an output or a groups of outputs is to be activated only on the occurence of a specific. The decoders were used in analog to digital conversion in analog decoders.

(i) Including data demultiplexing;

(ii) seven segment displays.

*Encoder :— An encoder is a digital function that produces a reverse from that of a encoder.

Application of Encoder :—

Encoders are used to translate rotary or linear motion into a digital signal, encoders use with external electronics, such as counters, to accumulate data to determine position.

(P.T.O.) →

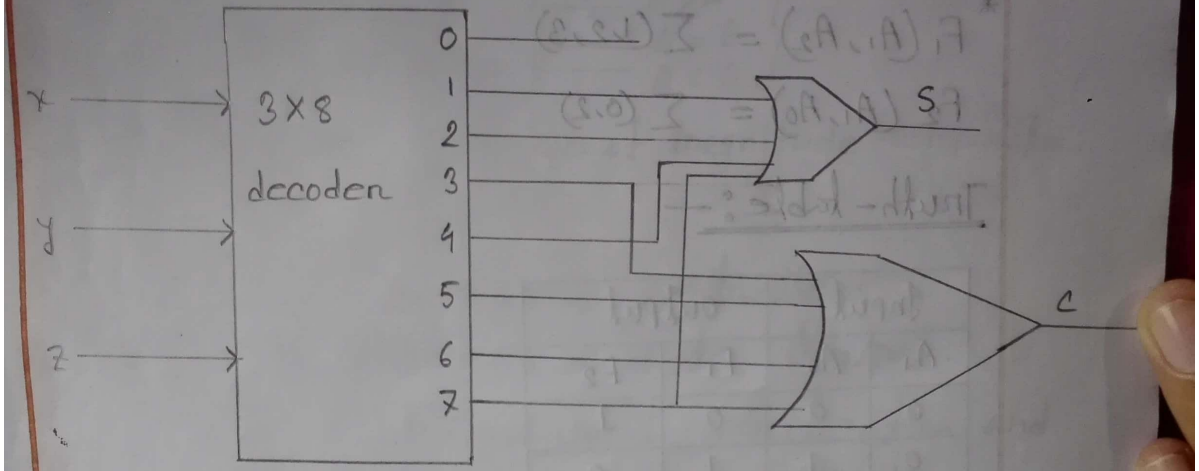
8. Implement a full adder circuit with a decoder and two OR gate.

Ans:—

$$S(x,y,z) = \sum(1,2,4,7)$$

$$C(x,y,z) = \sum(3,5,6,7)$$

Full-adder:—



*ROM Design:—

32 × 08 bit

↓ output = OR gate

↙ input → decoder

2 to 4 bit

2048 bit ROM

$512 \times 4 = 2^9$

$2^n = 2^9$

(P.T.O) →

in = 9

out = 512

32 to X 8 bit

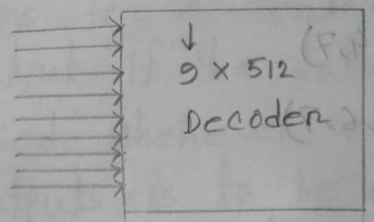
output = 2^n

$= 2^5$

$= 32$

$= 2^n$

$= 25$

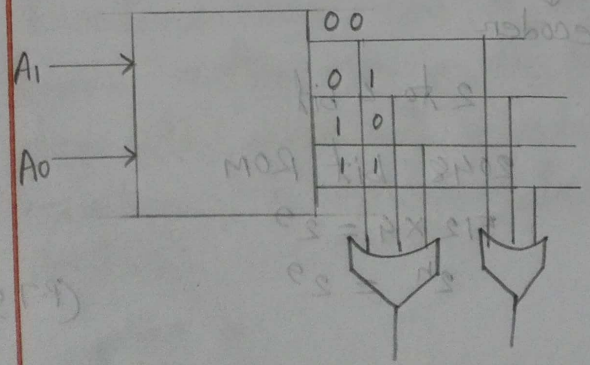


* $F_1(A_1, A_2) = \sum(1,2,3)$

$F_2(A_1, A_0) = \sum(0,2)$

Truth-table :—

Input		output	
A_1	A_0	F_1	F_2
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0



Magnitude comparator:- A magnitude comparator is a combinational logic circuit that compares two binary numbers and determine their relative magnitude.

Block-diagram:

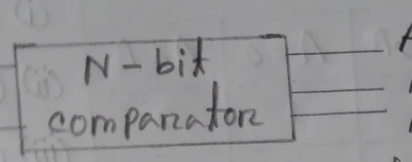


fig: block-diagram of magnitude comparator.

Applications:-

(i) Process controllers,
(ii) Servo-motor control.
(iii) Password verification and biomatric application.

Q. Explain 1-bit magnitude comparator circuit?

Ans:-

1-bit comparator:- A comparator used to compare two-bit (2-bit) called a single bit comparator (1-bit comparator). It consist of two

(P.T.O) →

input each of two singles bit numbers and three outputs to generate less that equal to, greater than between two binary numbers

Truth-table:-

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

(i) $(A>B) = AB'$

(ii) $(A<B) = A'B$

(iii) $(A=B) = A'B'+AB$
$= \overline{(A \oplus B)}$

Above this (i) & (ii) expression;

$(A>B) + (A<B) = AB' + AB'$

$((A>B) + (A<B))' = (AB' + AB')'$

$= (AB')' \cdot (A'B)'$

$= (A'+B) \cdot (A+B')$

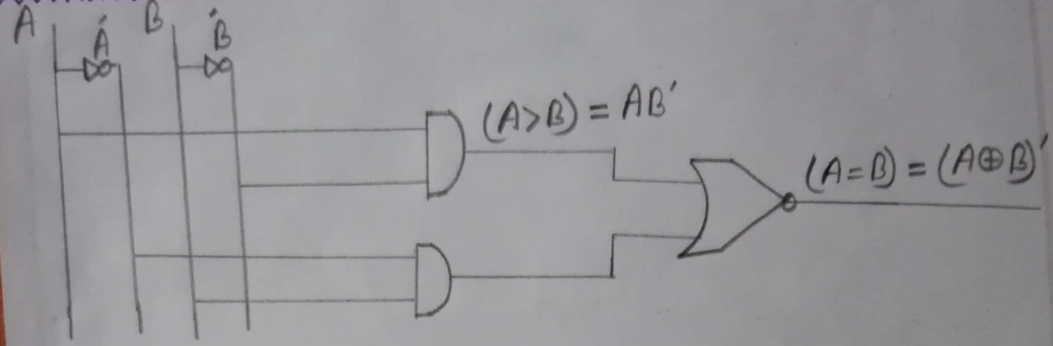
$= AA' + A'B' + AB + BB'$

$= 0 + A'B' + AB + 0$

$= (A \oplus B')$

$= (A = B)$.

circuit-design :—



8. Explain 2- bit magnitude comparator ?

Ans :—

*chap:- "Registers"...

① ***Register :-** Register is a group of binary storage cells suitable for holding binary information. A group of flip-flop consitutes a register. Since each flip-flop is a binary cell capable of storing one bit of information.

2015 5.(c) **Shift-Register :-** A register capable of shifting its binary information either to the right or to the left is called a shift register.

The logical configuration of a shift register consists of a chain of flip-flops connected in casecade, with the output of one flip-flop connected to the input of the next flip-flop. And a common clock pulse which causes the shift from one stage to the next.

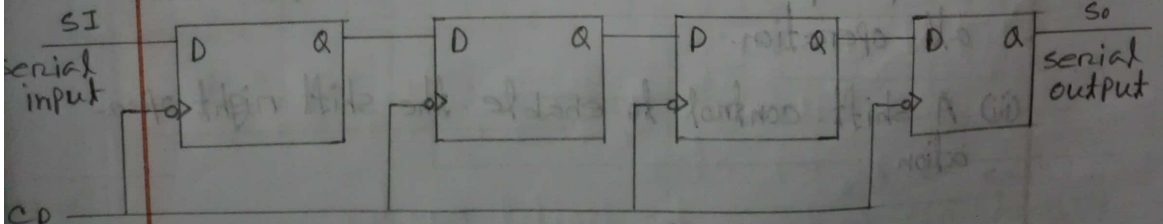


Fig: shift Register.

V.V.Q
2017, 15
5.(a)

Q. Design a 4 bit right shift register with parallel load of data / Design a 4 bit bidimentional shift-register with parallel load.

Ans:—

Shift register can be used for converting serial data to parallel data and vice-versa.

A register capable of shifting both right and left is called bidimentional shift register. If the register has both shift and parallel-load capabilities, it is called a shift register with parallel load.

It consist of 4D flip-flops, 4 mux, A clear control, A clock pluse (CP). S1, S0 section line, Parallel left-shift, right-shift control and 4 Parallel input and 4 output line.

(i) A clear control to clear the register to '0'.

(ii) A cp input for clock pluses to synchronize all operation.

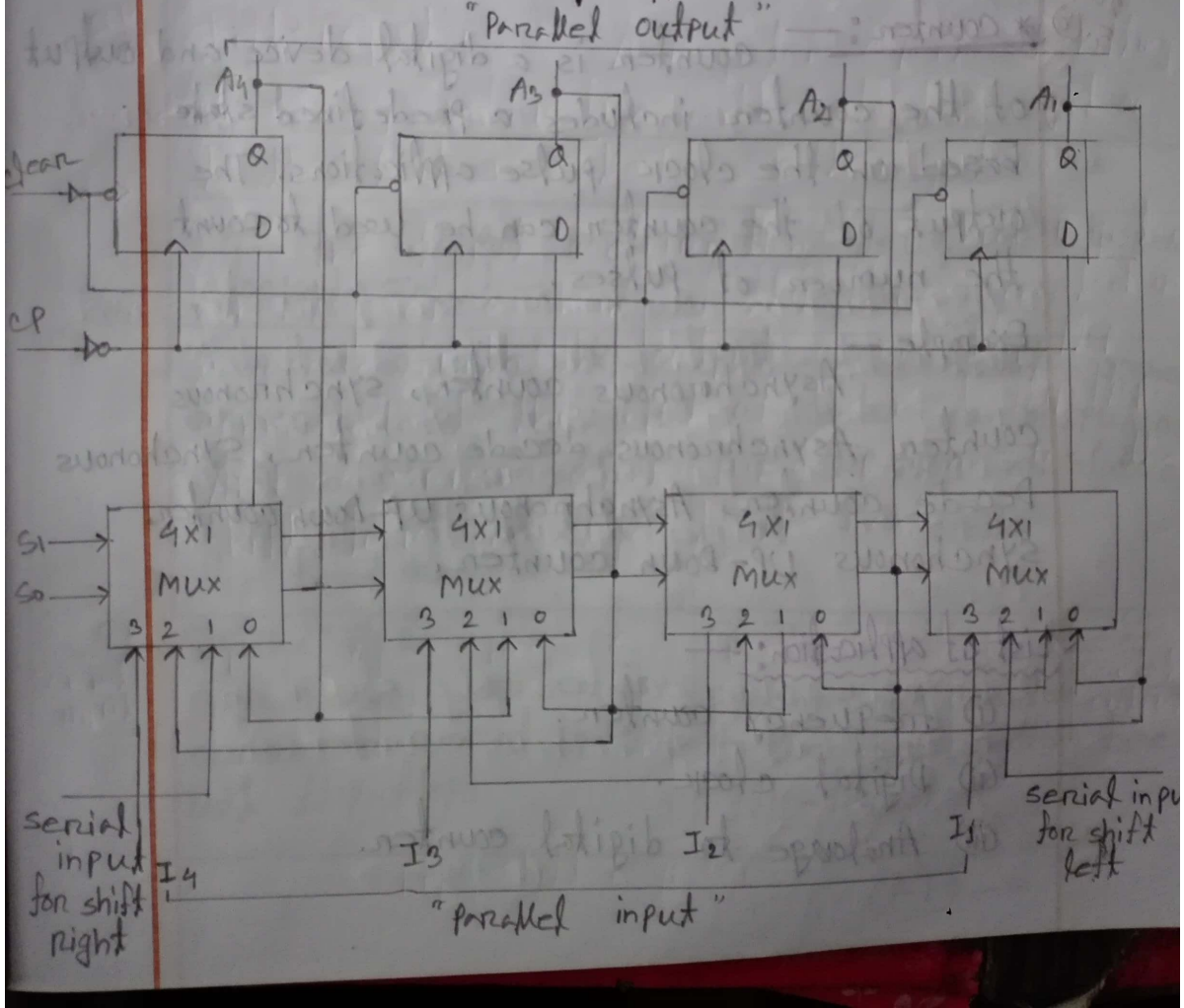
(iii) A shift control to enable the shift right operation.

(P.T.O) →

(iv) A parallel load control to enable a parallel transfer and the n input lines associated with the parallel transfer.

(v) n parallel output lines.

(vi) A control state that leves the information in the register unchanged even through clock pluses are continuously applied.



Mode control		register
S_1	S_0	
0	0	No change
0	1	shift Right
1	0	shift left
1	1	Parallel load

2017
5.(b) * counter :— counter is a digital device and output of the counter includes a predefined state based on the clock pulse applications. The output of the counter can be used to count the number of pulses.

Example — Asynchronous counter, synchronous counter Asynchronous decade counter, synchonous Decade counter, Asynchrionous up-Down counter. synchonous up-Down counter.

List of application :—

(i) frequency counter.
(ii) Digital clock.
(iii) Analouge to digital counter.

(iv) With some changes in their design counter can be used as frequency deviden circuit input frequency /2.

(v) We can design digital triangular wave generation by using counter.

2017 5.(b) Define Asynchronous & synchoronous counter.

8. Distuingusih between Asynchronous & synchoronu counter.

Ans :—

"Asynchronous"	"synchoronous"
(i) All flip-flop are not clocked simultaneously.	(i) All flip-flop are clocked simultaneously.
(ii) Output of first flip-flop drives the clock of the second flip-flop, the output of second drives the thind and so on.	(ii) There is no inten connection between an output of one flip-flop and clock of next flip-flop.
(iii) The stetting time of asynchronous counter is cumutative sum of individual flip-flops.	(iii) The stetting time of synchronous counter is equ to the higest setting time of flip-flops.

(P.T.O) →

(iv) It is also known as a serial counter.	(iv) It is known as a parallel counter.
(v) It's design and implement are very simple.	(v) Its design and implementation become tedious and complex as the number of states increases.
(vi) It is also in speed.	(vi) It is fasten in speed.

2013, 14, 18
5.(a) 8. Advantage of synchronous counter over synchorous counter.

Ans:—

(i) Asynchronous counter are serial counters and different flip-flop are triggred with different clock pulse while synchronous counter are paralled counter and all flip-flops are triggered with the same clock pulse.

(ii) Asynchronous counters are slower in nature but synchronous counters are fasten.

(P.T.O) →

(iii) Only fixed number of count sequence is possible with asynchronous counter while any number of count sequence is possible with synchronous counter.

(iv) There are problems with decoding errors with asynchronous counter while there are no such problems with synchronous counter

↗ Tocci - 362 Page

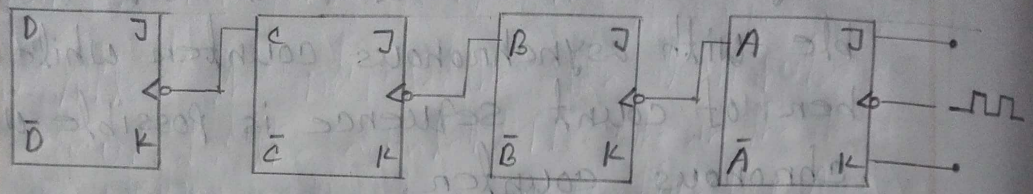
(2) 2013, 14 5.(c) * MOD Number :— MOD Number is generally equal to the number of states that the counter goes through in each complete cycle before it recycles back to its starting state. MOD Number can be increased by adding more to the counter.

∴ MOD Number = 2^N (N is number of ff).

2017 5.(c) Truth-table, logic diagram of a MOD-10 ripple counter

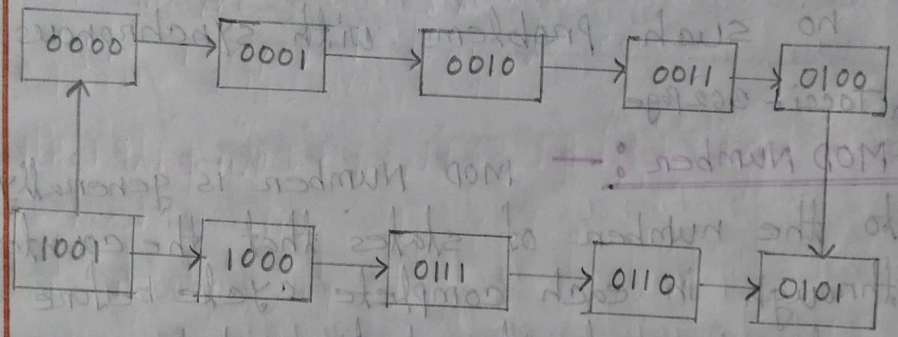
Timing diagram :—

(P.T.O) →



All J & K input assumed to be 1

Fig :- MOD-10 Ripple counter.



Truth-table :-

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
0	0	0	0

It is a MOD-10 counter. so, it can count 10 distinct number and than going back to the starting state.

Explain :—

(i) The clock pulses are applied only to the clock input of flip-flop A. Thus, flip-flop A will toggle (change to its opposite state) each time the clock pules make a negative (HIGH to LOW) transition. Note that J=K=1 to all FFs.

(ii) The normal output of flip-flop acts as the CLK input for flip-flop B, and so flip-flop B will toggle each time the A output goes from 1 to 0. Similarly, flip-flop C will toggle when B goes from 1 to 0 and flip-flop D will toggle when C goes from 1 to 0.

(iii) FF outputs D, C, B & A represent a 4bit binary number D as the MSB. Lets assume that all FFs have been cleaned to the 0 state. (clear inputs are not shown). The wave froms in figure-1 show that a binary counting sequence from 0000 to 1001 is followed as clock

(P.T.O) →

Pulses are continuously applied.

(iv) After the NEXT to the Ten clock pulse has occured, the counter FFs are in the 1001 condition. On the Ten NEXT, flip-flop A goes from 1 to 0, which causes flip-flop B to go from 1 to 0 and so on unit the counter is in the 0000 state. The counter has gone recycled back to 0000.

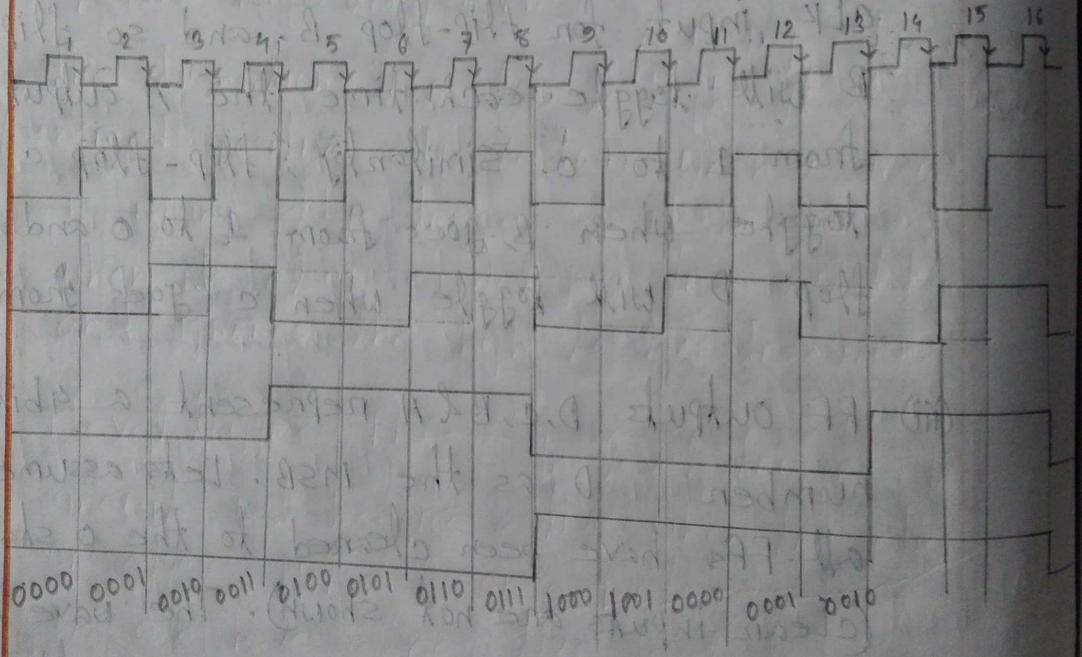


Fig 1: MOD-10 clock pulse.

* Binary Ripple counter / 4-bit Ripple counter :—

A binary Ripple counter consist of a series connections of complementing flip-flops (T or JK) with the output of each flip-flop connected to the cp input of the next higher orden flip-flop. The flip-flop receives the least significant bit from the incoming count pulse.

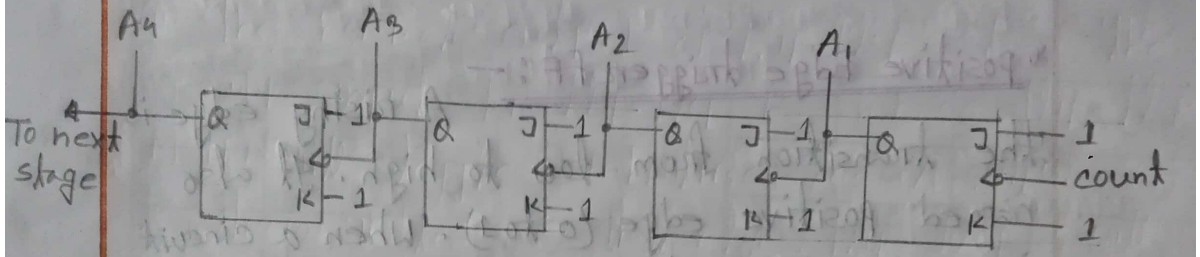


Fig: 4-bit binary Ripple counter.

*Negative Edge triggen FF:— A falling is the high to low transition. It also known as the negative edge (1 to 0). When a circuit is falling edge-triggered, it becomes active when the clock signals goes from high to low and ignores the low to high transition.

*positive Edge triggen FF:— A rising edge is the transition from low to high. It also named positive edge (0 to 1). When a circuit is rising edge-triggered, it becomes active when its clock signal goes from low to high and ignores the high to low transition.

*Level triggned ff:— In level triggned the circuit will become active when the gating on clock pulse is on a particular level. We can have a negative level triggening in which the circuit active when the clock signal is low on a positive level

(P.T.O) →

triggering in which the circuit is active when the clock signal is high.

2015-16 (4)

* **Excitation table :-** An excitation table shows the minimum input that are necessary to generate a particular next state when the current state is known.

current state Q	Next state Q^+	Required input S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

Fig :- E-table for SR FF.

C	J	K	Q(n+1)
0	0	0	Q(n)
1	0	0	Q(n)
1	0	1	0
1	1	0	1
1	1	1	Q(n)

Fig : E-table for JK FF.

C	D	Q(n+1)
0	x	Q(n)
1	0	0
1	1	1

Fig : E-table for DFF.

Q(n)	Q(n+1)	T	Operation
0	0	0	No change
0	1	1	complement
1	0	1	"
1	1	0	No change

Fig : E-table for TFF.

2017 6.(c)

* Presettable :- When LD input is high then whatever binary value is present on LOAD INPUT, will be immediately copied to the outputs and stay that way untill LD goes low. This only works if the RESET input is low.

2018, 17 6.(b) 5.(c)

8. MOD-8 synchronous up/Down counter with logic diagram its timing operation. [Tocci - 378, 377].

Ans:-

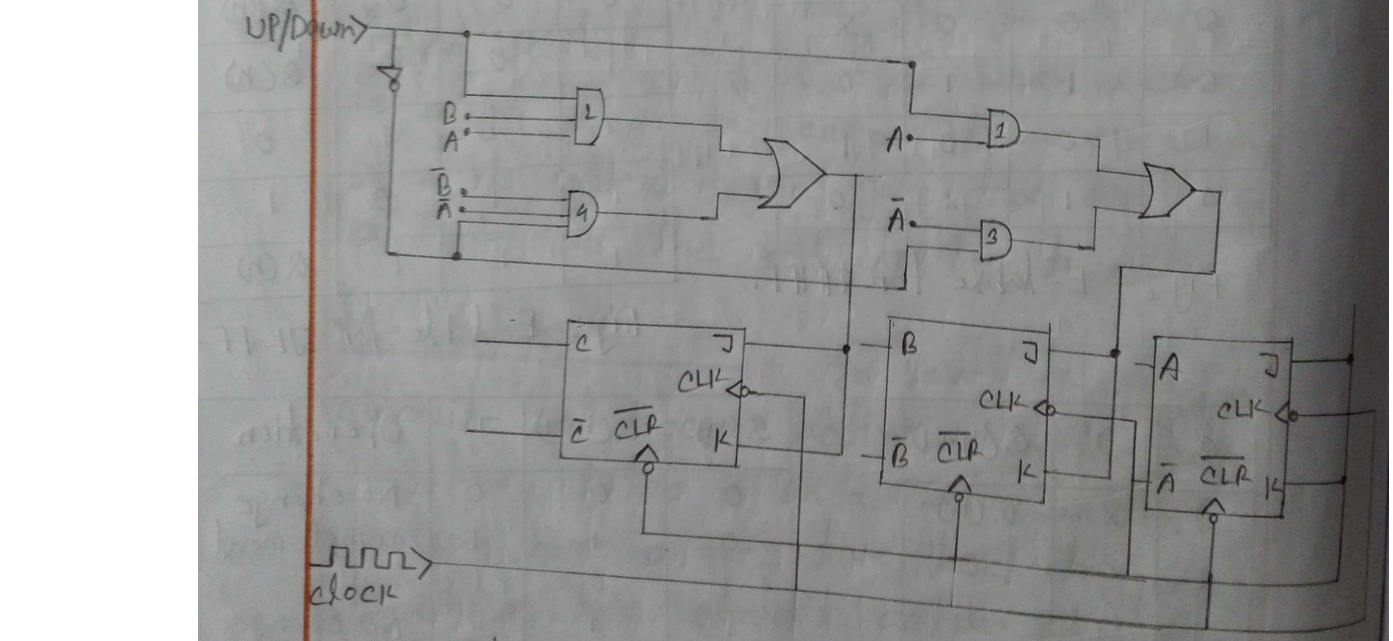


Fig (a): Logic diagram of MOD-8 synchronous up/Down counter.

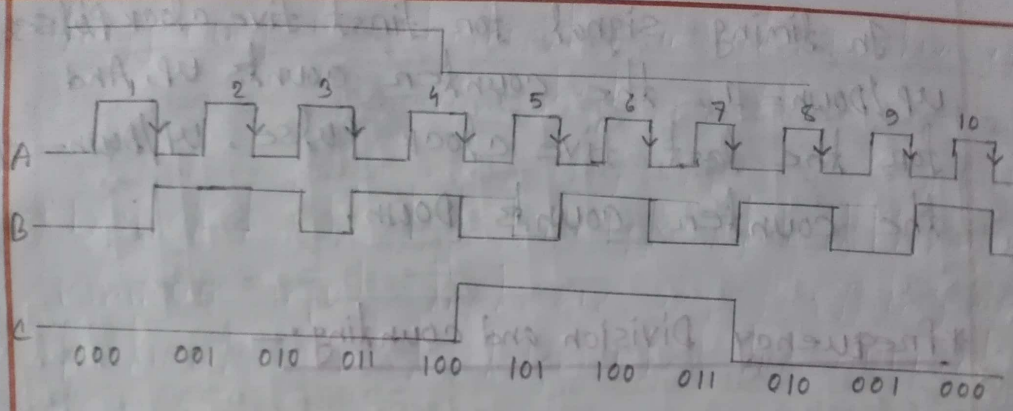


Fig (b): Timing operation of MOD-8.

Fig (a) show to from a parallel UP/Down counter: The control input UP/Down control whether the normal FF outputs or the inverted FF outputs are fed to the J and K inputs of the successsive ffs. When UP/Down is held HIGH, AND gates 1 and 2 are enabled while AND gates 3,4 are disabled. This allows the A and B output through gates 1 and 2 to the J and K inputs of ffs B and C.

When UP/Down is held LOW AND gates 1,2 are disable and 3,4 are enable. This allows the inverted A and B outputs through gates 3 and 4 into J and K inputs of ffs B and C.

(P.T.O) →

In timing signal for first five clock pulses, UP/Down = 1, the counter counts UP, And for the last five clock pulse, UP/Down = 0 the counter counts Down.

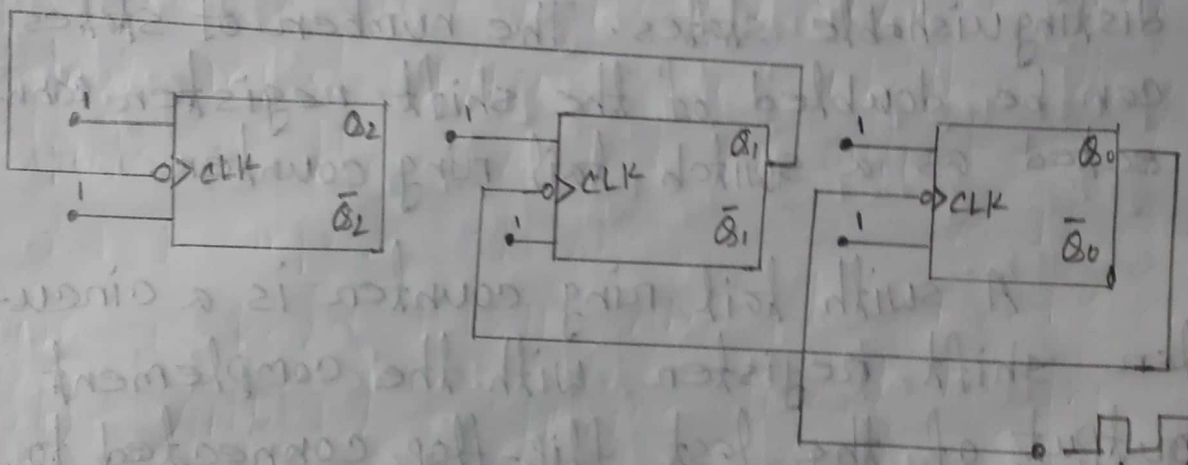
*Frequency Division and counting :-

2018
5.(a).

6.4 KHZ to 8 KHZ:- N ff would produce an output frequency from the last ff which is equal to 1/2 N of the input frequency.

$$\frac{8}{64} = \frac{1}{8} = \frac{1}{2^3}$$

We need three ffs to construct binary counter that will convert a 64 KHz pulse signal into a 8-KHz square wave.

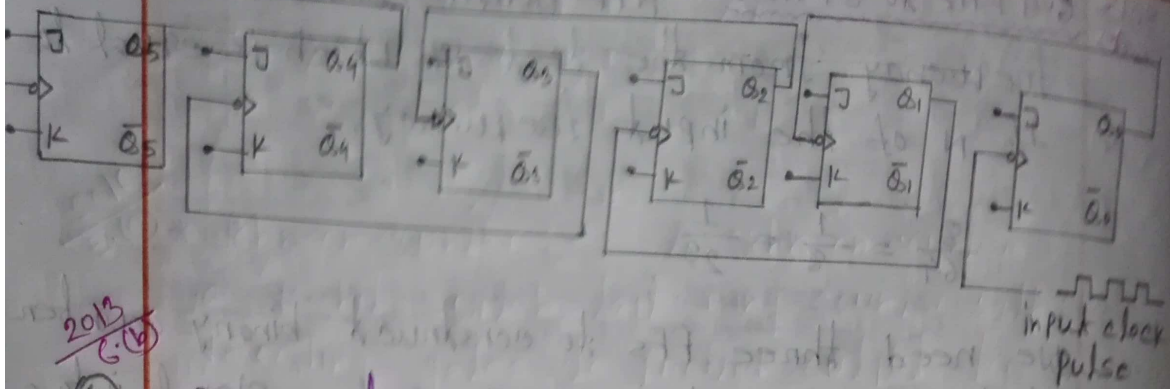


2013
5.(b)

A counter circuit that will convert 64KHz to 1Hz square wave:

$\frac{1}{64} = \frac{1}{2^6}$ that means we need 6ffs to construct binary counter that will convert a 64KHz pulse signal to 1 KHz square wave.

(P.T.O) ⟶



2013
C.U

(4) * Ring counter: → A k bit ring counter circulates a single bit among the flip-flops to provide k distinguishable states. The number of states can be doubled of the shift register connected as a switch tail ring counter.

A swith tail ring counter is a circular shift register with the complement output of the last flip-flop connected to the input of the first flip-flop. The circular connection made from the complement output of the rightmost flip-flop to the input of the leftmost flip-flop. The register shifts its contents once to the right with every clock pulse, and at the same time the complement value of E flip-flop. in transfered into

(P.T.O) →

A flip-flop. starting from a cleared state. the switch tail ring counter goes through eight states.

A K bit switch tail ring counter go through a sequence of 2K state. starting from all 0's each shift operation inserts 1's from the left until the register is filled with all 1's. 0's are inserted from the left until the register is again filled with all 0's.

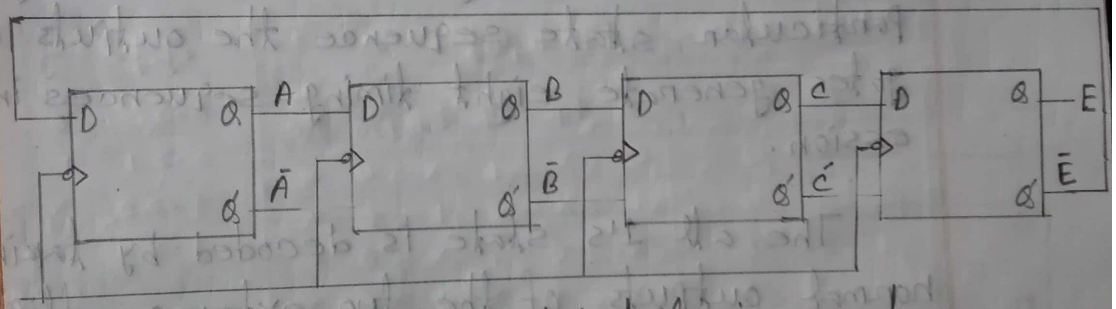


Fig: 4 state switch tail ring counter.

Sequence Number	FF output A B C E	AND gate required for output
1.	0 0 0 0	A'E'
2.	1 0 0 0	AB'
3.	1 1 0 0	BC'
4.	1 1 1 0	CE'
5.	1 1 1 1	AE

6.	0	1	1	1	A'B
7.	0	0	1	1	B'C
8.	0	0	0	1	C'E

* Johnson counter :- A Johnson counter is a k-bit switch tail ring counter with 2k decoding gates to provid outputs for 2k timing signal. The eight AND gates listed in the table Johnson counter complete when connected to the circuit. Each gate is enabled during one particular state sequence the outputs of the gates generate eight timing sequences in succession.

The all 1's state is decoded by taking the normal outputs of the two extreme flip-flops All other states are decoded from an adjacent 1,0 and 0,1 pattern in the sequence. For example, sequence 7 has an adjacent 0,1 pattern in flip-flops B and C. The denoted output is then obtained by taking the complement of B and the normal output of C, on B'C.

(P.T.O) →

6.	0	1	1	1	A'B
7.	0	0	1	1	B'C
8.	0	0	0	1	C'E

2016
5. (c) * Johnson counter :— A Johnson counter is a k-bit switch tail ring counter with 2k decoding gates to provid outputs for 2k timing signal. The eight AND gates listed in the table Johnson counter complete when connected to the circuit. Each gate is enabled during one particular state sequence the outputs of the gates generate eight timing sequences in succession.

The all 1's state is decoded by taking the normal outputs of the two extreme flip-flops All other states are decoded from an adjacent 1,0 and 0,1 pattern in the sequence. For example, sequence 7 has an adjacent 0,1 pattern in flip-flops B and C. The denoted output is then obtained by taking the complement of B and the normal output of C, on B'C.

(P.T.O) →

6.	0	1	1	1	A'B
7.	0	0	1	1	B'c
8.	0	0	0	1	c'E

2016
5.(c) * Johnson counter :— A Johnson counter is a k-bit switch tail ring counter with 2k decoding gates to provid outputs for 2k timing signal. The eight AND gates listed in the table, Johnson counter complete when connected to the circuit. Each gate is enabled during one particular state sequence the outputs of the gates generate eight timing sequences in succession.

The all 1's state is decoded by taking the normal outputs of the two extreme flip-flops. All other states are decoded from an adjacent 1,0 and 0,1 pattern in the sequence. for example, sequence 7 has an adjacent 0,1 pattern in flip-flops B and c. The denoted output is then obtained by taking the complement of B and the normal output of c, on B'c.

(P.T.O) →

counter with J-K flip-flops that count the sequence 0, 2, 4, 6 and 0, 1, 3, 7, 5, 2, 0.
(247 Morris Mano & 308 Tocci).

The logic diagram of the counter is shown in fig-1. since there are four unused states (0, 2, 4, 6) / two unused state (0, 2, 3, 6, 5, 1, 0) we analyze the circuit to determine thein effect. If the circuit goes to an invalid state the next count pulse transfers it to one of the valid state and it is continues to count correctly. Thus the counter is self-starting.

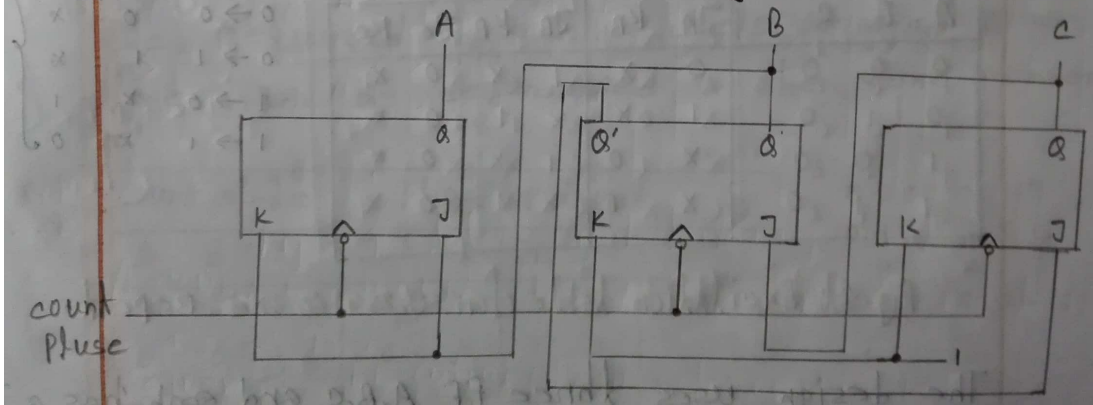


Fig: 1 Logic diagram of counter.

(P.T.O) →

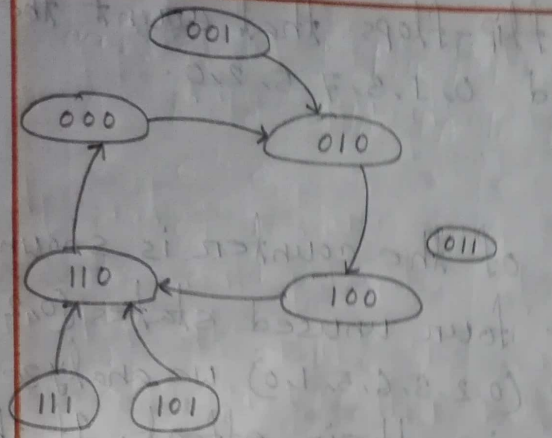


Fig: 2 state diagram for 0, 2, 4, 6 and Repeat.

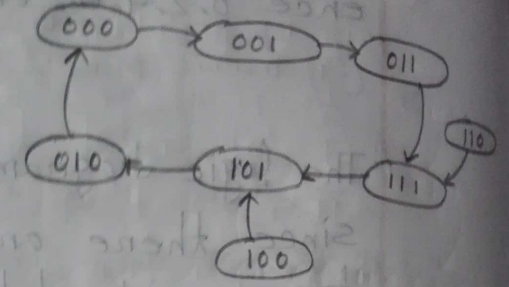


Fig 2: state diagram for 0, 1, 3, 7, 5, 2, 0.

count sequence			FF input					
A	B	C	J_A	K_B	J_B	K_B	J_C	K_C
0	0	0	0	X	1	X	0	X
0	1	0	1	X	X	1	0	X
1	0	0	X	0	1	X	0	X
1	1	0	X	1	X	1	0	X

state	J	K
0 → 0	0	X
0 → 1	1	X
1 → 0	X	1
1 → 1	X	0

Fig 3: Excitation table for 0, 2, 4, 6 and repeat.

The design uses three FF A, B, C and each has a J and a K input. The "PRESENT" state 000 go to the NEXT state 010 on the occurance of a clock pulse. For this state transition A Flip-flop goes

(P.T.O) →

from '0' to '0'. From the J-K excitation table J_A must be at '0' and K_A at 'x' for the transition occur. The B flip-flop also goes from '0' to '1' and so J_B = '1' and K_B = 'x'. The C flip-flop goes from '0' to '0' and so J_C = '0' and K_C = 'x'.

In line two "PRESENT" state '010' go to the NEXT state '100'. So the FF A go to '0' to '1' and so J_A = '1' and K_A = 'x'. B FF go to '1' to '0'. So, J_B = 'x' & K_B = C FF go to '0' to '0'. So, J_C = '0' & K_C = 'x'.

The required J and K levels for all other lines can be determined in the same manner.

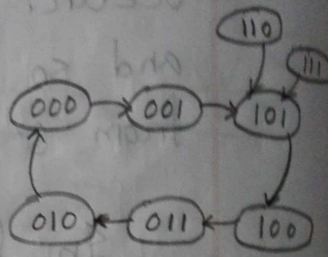
Count state			Flip-Flop input					
A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	X	0	X	1	X
0	0	1	0	X	1	X	X	0
0	1	1	1	X	X	0	X	0
1	1	1	X	0	X	1	X	0
1	0	1	X	1	1	X	X	1
0	1	0	0	X	X	1	0	X

0→0 0 X
0→1 1 X
1→0 X 1
1→1 X 0

Fig 3: Excitation table for 0,1,3,7,5,2,0.

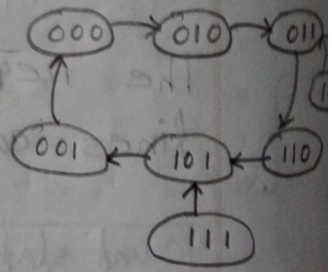
(P.T.O)→

count state			Flip-flop input					
A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	X	0	X	1	X
0	0	1	1	X	0	X	X	0
1	0	1	X	0	0	X	X	1
1	0	0	X	0	1	X	0	X
1	1	0	X	1	X	0	0	X
0	1	0	0	X	X	1	0	X



Excitation table for 0, 1, 5, 4, 3, 2, 0.

count state			Flip-flop input					
A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	X	1	X	0	X
0	0	1	0	X	X	0	1	X
1	0	1	1	X	X	0	X	1
1	0	0	X	0	X	1	1	X
1	1	0	X	1	0	X	X	0
0	1	0	0	X	0	X	X	1



Excitation table for 0, 2, 3, 6, 5, 1, 0.

"Chap:- Memory":

Q. Difference between DRAM & SRAM.

Ans:-

	"SRAM"	"DRAM"
1. Speed	fasten	slower
2. Size	small	langer
3. cost	expensive	cheap
4. Used in	cache memory	main memory
5. Density	less dense	high dense
6. construction	complex and uses capacitor and transistor.	simple and uses capacitons and very few transistor.
7. Power consumption	low	high.

Q. Define Memory?

Ans:-

Memory:-

Q. Difference between FPGA & CPLD:

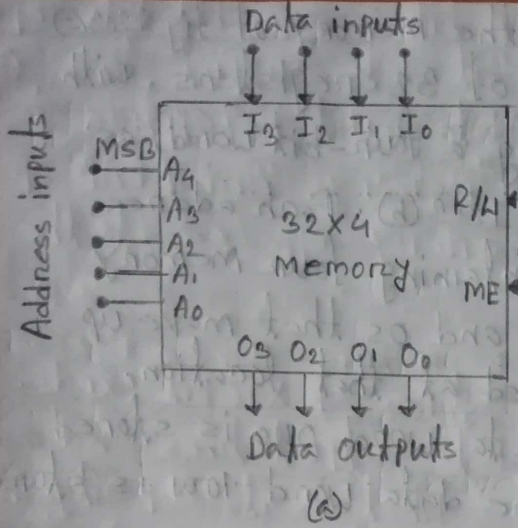
Ans:—

"FPGA"	"CPLD"
(i) In contain up to 1,00000 of tiny logic blocks.	(i) In contain logic blocks up to a few thousand.
(ii) FPGA as fine-grain architecture.	(ii) It course-grain architecture.
(iii) FPGA are great for more complex applications.	(iii) CPLD are better for simple one.
(iv) FPGA is RAM base digital logic chip.	(iv) CPLD is EEPROM based logic chip.
(v) More expensive	(v) Much expensive
(vi) Field programmable gate arrays.	(vi) complex programmable logic Device.

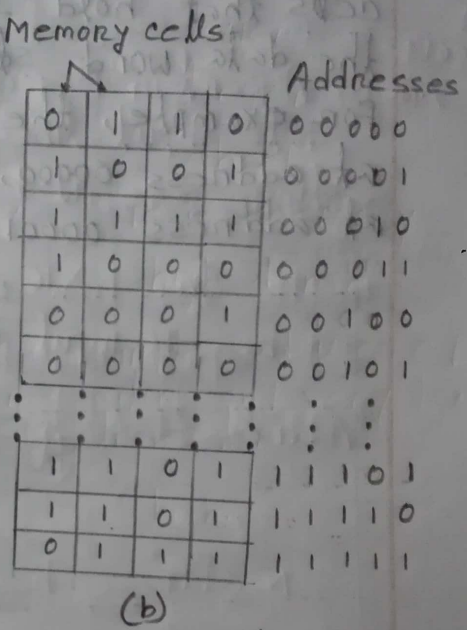
*Address Input:—

Because this memory stores 32 words, it has 32 different ~~binary~~ storage location and therefore 32 different binary addresses ranging from 00000 to 11111 (0 to 31 in decimal).

(P.T.O) →



(a)



(b)

FIGURE 12·3: (a) Diagram of a 32×4 memory;
(b) virtual arrangement of memory cells into 32 four-bit words.

Thus, there are five address inputs A_0 to A_4. To access one of the memory locations for a read or a write operation, the five-bit address code for that particular location is applied to the address inputs. In general, N address inputs are required for a memory that has a capacity of 2^N words.

(P.T.O) →

We can visualize the memory of fig 12.3(a) as an arrangement of 32 registers, with each register holding a four-bit word, as illustrated in figure 12.3(b). Each address location is shown containing four memory cells that hold 1s and 0s that make up the data word stored at that location. For example, the data word 0110 is stored at address 00000, the data word 1001 is stored at address 00001, and so on.

*RAM Architecture:-

Internal organization of a 64x4 RAM.:-

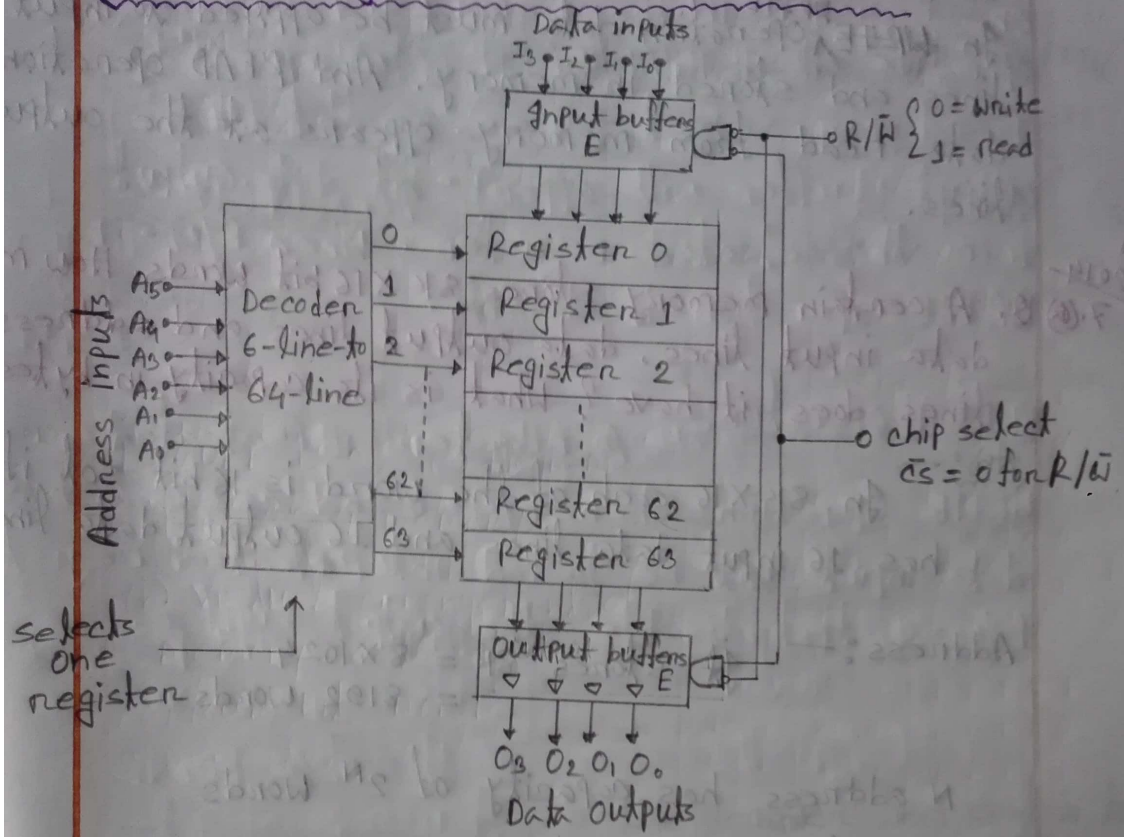


Fig: internal organization of a 64x4 RAM.

64x4 memory that stores 64 four bit words. It has 64 different storage locations ranging from 000000 to 11111 (0 to 63 decimal). There are six address input A0 to A5. In general N address input required a memory that has capacity 2^N words.

(P.T.O) →

Each of the word four bit so four bit data input lines I_0 to I_3 and four bit data output lines O_0 to O_3. In ~~WRITE~~ WRITE operation data must be applied to input lines and stored in memory. And READ operation data read from memory appears at the output lines.

2014
7.(a) 8. A certain memory stores 8K × 16 bit words. How many data input lines, data output lines and address lines does it have? What is its capacity in bytes?

Ans:—

In 8K × 16 each of the word is 16 bit so it has 16 input data lines and 16 output data lines.

Address:— It stores $8K = 8 \times 1024$
$= 8192$ words

N address has capacity of 2^N words

So, $2^{13} = 8192$ where $N = 13$

13 Address lines are required.

Capacity in bytes:— Each of the word 16 bits that means 2 byte. so the total capacity in bytes are, $8192 \times 2 = 16384$ bytes.

2016
6.(c)

* RAM using 2×4 decoder (2 input & 2 output bit):

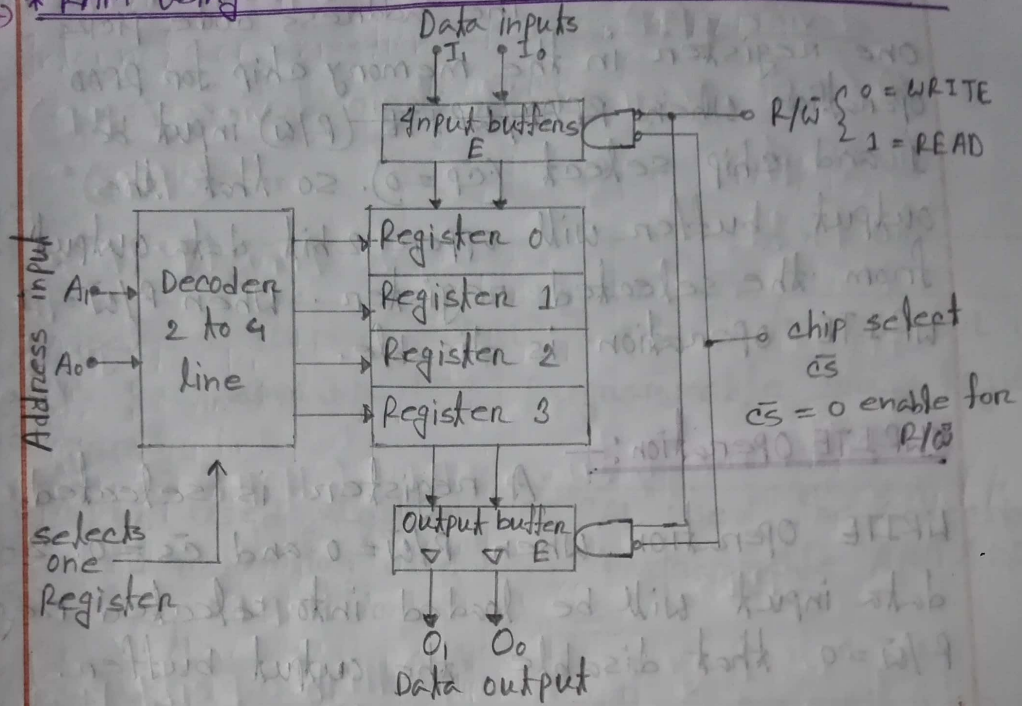


Fig: 2×4 decoder RAM.

The RAM used 2 to 4 line decoder that contain two Address lines A_0 A_1 and four Register (register 0 to register 3). The decoder used two input lines I_0 I_1 and two output line O_0 O_1 that means each of the words is 2-bit.

*READ Operation:— The address code picks one register in the memory chip for READ operation when READ/WRITE ($R/\bar{W}$) input be 1 and chip select ($\bar{CS}=0$). so that the output buffer will appear 2 bit data output from the selected register. When $R/\bar{W}=1$ WRITE operation is disable.

*WRITE Operation:— A register is selected for WRITE Operation when $R/\bar{W}=0$ and $\bar{CS}=0$. 2-bit data input will be loaded into selected regis $R/\bar{W}=0$ that disable the output buffer.

*chap:- "Programmable logic Device Architectures"..

2013: 8 (a) 7.(b) Defn

Q. Difference between PLA & PAL.

Ans:

"PLA"	"PAL"
(i) The full form of PLA is programmable logic array.	(i) The full form of PAL is Programmable Array logic.
(ii) PLA used programmable collection of AND & fined collection of OR gates.	(ii) PAL used the collection of Programmable AND & OR gates.
(iii) The availability of PLA is more.	(iii) The availability of PAL is less prolific.
(iv) Flexibility of PLA is less.	(iv) Flexibility of PAL is more.
(v) cost of PLA is middle ~~(v)~~ range.	(v) cost of PAL is expensive.
(vi) The number of functions implemented in PLA is limited	(vi) The number of functions implemented in PAL is large.
(vii) The speed of PLA is high.	(vii) The speed of PAL is slow.

* There are three type of programmable logic device (PLD).

(i) simple programmable logic devices (SPLDs)

(ii) complex " " " (CPLDs) ┐

(iii) Field " gate Arrays (FPGAs) ┘ are high capacity programmable logic device. (HCPLDs)

Q. Difference between SPLDs & HCPLDs:

A:—

"SPLDs"	"HCPLDs"
(i) simple programmable logic devices.	(i) High capacity programmable logic devices.
(ii) It typically contain 600 or fewer gates.	(ii) It contains hundreds of thousands of gates.
(iii) Less complecated.	(iii) Much complicated.
(iv) Much cheaper than HCPLDs.	(iv) Expensive than SPLDs.

* K-map :—

C'D \ A'B'	A'B	AB	AB'
C'D'	0000 0	0001 1	0011 3	0010 2
C'D	0100 4	0101 5	0111 7	0110 6
CD	1100 12	1101 13	1111 15	1110 14
CD'	1000 8	1001 9	1011 11	1010 10

C'D' \ A'B'	A'B'	AB	AB'
C'D'	0000 0	0100 4	1100 12	1000 8
C'D	0001 1	0101 5	1101 13	1001 9
CD	0011 3	0111 7	1111 15	1011 11
CD'	0010 2	0110 6	1110 14	1010 10

A'B' \ C'D'	C'D'	C'D	CD	CD'
A'B'	0000 0	0100 4	1100 12	1000 8
A'B	0001 1	0101 5	1101 13	1001 9
AB	0011 3	0111 7	1111 15	1011 11
AB'	0010 2	0110 6	1110 14	1010 10

A'B' \ C'D'	C'D'	C'D	CD	CD'
A'B'	0000 0	0001 1	0011 3	0010 2
A'B	0100 4	0101 5	0111 7	0110 6
AB	1100 12	1101 13	1111 15	1110 14
AB'	1000 8	1001 9	1011 11	1010 10

All combination is possible in K-map we can use any of them.

$x+x' = 1$ $x \cdot x' = 0$ complement law

sum of product (SOP) :— $ABC + A'C + BD'$

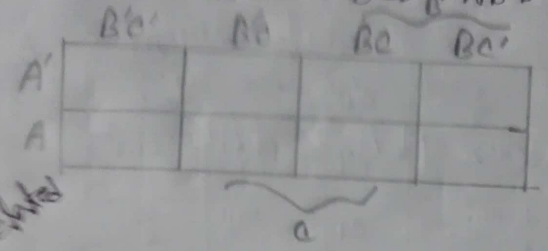
product of sum (POS) :— $(A+B+C)(A'+C)(B+D')$

K-map simplification

A'B' \ C'D'	C'D'	C'D	CD	CD'
A'B'	0	1	0	0
A'B	0	0	1	1
AB	0	0	1	1
AB'	1	1	1	1

(P.T.O) →

$X = B'C'D + ABC + BCD + AB'D' + ACD$

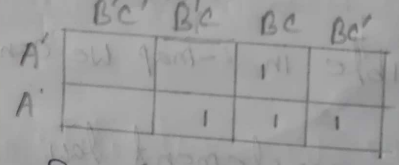


* PLA implementation :—

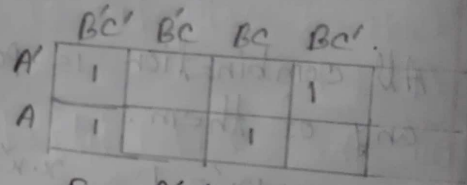
$F_1(A,B,C) = \sum(3,5,6,7)$

$F_2(A,B,C) = \sum(0,2,4,7)$

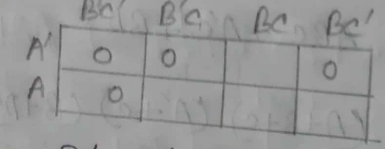
implement PLA circuit with a PLA having three inputs, four product terms and two outputs.



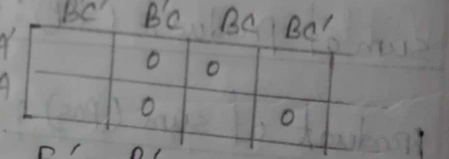
$F_1 = AC + AB + BC$



$F_2 = B'C' + A'C' + ABC$



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



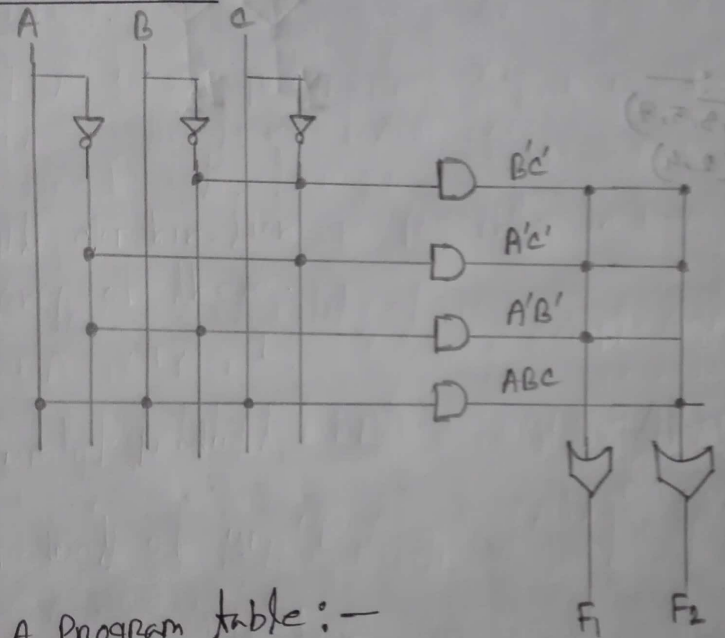
$F_2' = B'C + A'C + ABC'$

The PLA have three inputs A, B, C and two outputs F_1' and F_2 have four product terms $B'C'$, $A'C'$, $A'B'$, ABC where $F_1' = (B'C' + A'C' + A'B')$ & $F_2 = (BC' + A'C' + ABC)$. The output of F_2 is normal complemented.

Program PLA table :—

	Product term	inputs A	B	C	output F_1	F_2
$B'C'$	1	–	0	0	1	1
$A'C'$	2	0	–	0	1	1
$A'B'$	3	0	0	–	1	–
ABC	4	1	1	1	–	1

PLA circuit :—

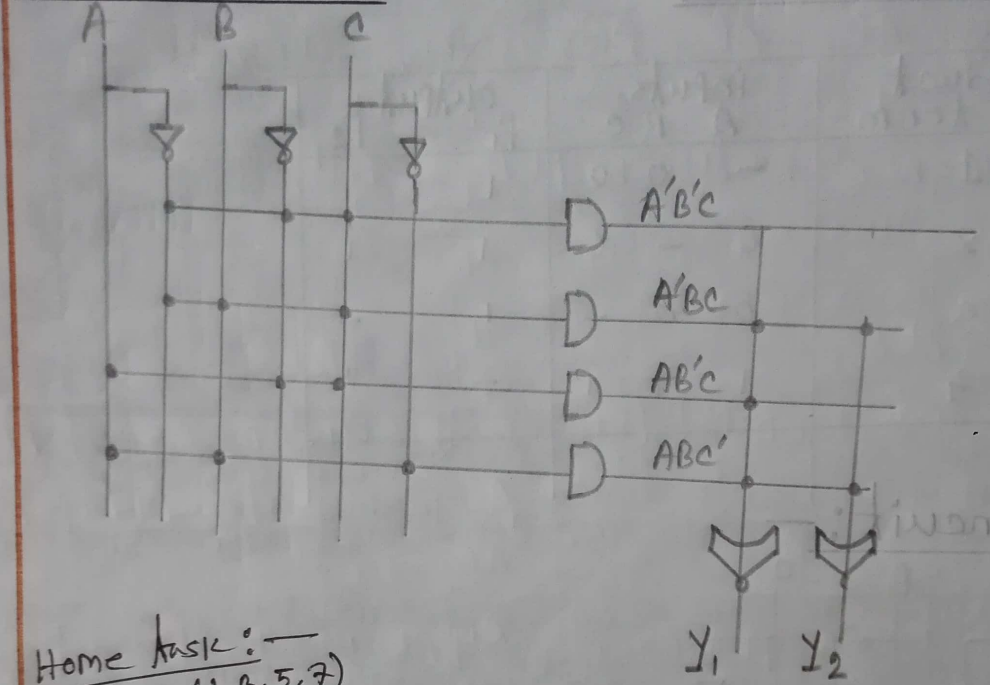


PLA Program table :—

	Product term	inputs A	B	C	outputs Y_1	Y_2
$A'B'C$	1	0	0	1	1	–
$A'BC$	2	0	1	1	1	1
$AB'C$	3	1	0	1	1	–
ABC'	4	1	1	0	1	1

(P.T.O) →

PLA circuit:-



Home task:-

$Y_1 = \Sigma(1,3,5,7)$

$Y_2 = \Sigma(2,4)$