Dated: 03-09-2024

DLD (Ms. Zareen)

Quiz : Thursday of 5TH week

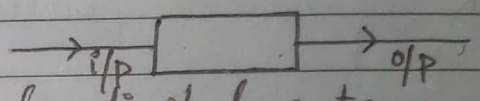
★ Number systems ★ Conversion

★ Signed/unsigned no. representation.

- 1's complement was rejected because it has two notations of zero (positive and negative)
- Registers are of 64-bits and for comparison it requires 64 exclusively NOR gate.

★ overflow (if both the operands has same sign and the result has opposite sign)

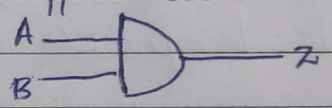
Logic Gates



↳ fundamental building blocks of digital systems.

AND

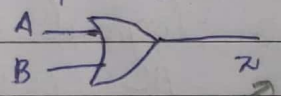
o/p is 1 iff all i/ps are 1's



$Z = A \cdot B$

OR

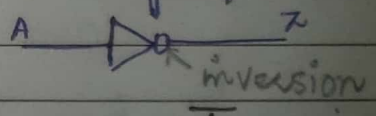
o/p is 1 iff any i/ps is 1



$Z = A + B$ diff than arithmetic add.

NOT

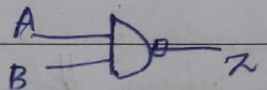
Inverter (Unary operator)



$Z = \overline{A}$

⇒ Derived gates

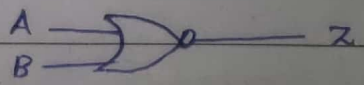
NAND



$Z = \overline{A \cdot B}$

o/p is 1 iff any of the i/p is 0.

NOR



$Z = \overline{A + B}$

o/p is 1 iff all the i/ps are 0.

Both of them are called as Universal gates.

1- Both can mimic the functionality the basic gates.

2- When fabricating, AND/OR expressions are never implemented they are converted to equivalent NAND (NOR) logic.

↳ Because NAND/NOR gates are energy efficients.

CamScanner

for even no of function: XOR = $\overline{XNOR}$

$A \oplus B = \overline{A \odot B}$

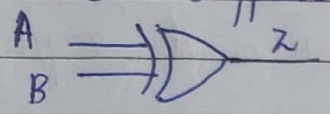
Dated: 03-09-2024

⇒ Secondary gates. for odd no of function: XOR = XNOR

$A \oplus B \oplus C = A \odot B \odot C$

XOR

o/p is 1 iff the i/ps are different. (for 2 i/p)



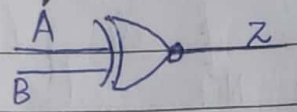
$z = A \oplus B$

$= \bar{A}B + A\bar{B}$

★ XOR is an odd function of 1's

XNOR

★ XNOR is an even function of 0's.

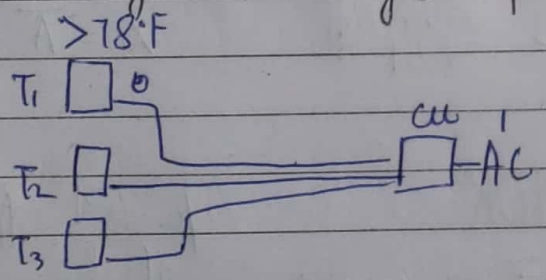


$z = A \odot B$

o/p is 1 iff the i/ps are same (for 2 i/ps)

Problem:

1- The AC unit in a building is to be turned on by a logical 1 at its control unit if the temp. at any of three location exceeds 78°F. Assuming that the temp. sensing devices produce a logical 0 when the temp. exceeds 78°F. Design the logic for AC control.



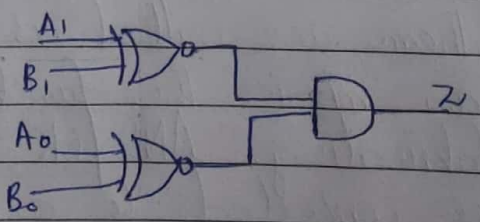
$AC = \overline{T_1 \cdot T_2 \cdot T_3}$

or $= \overline{T_1} + \overline{T_2} + \overline{T_3}$

2- Derive a logical expression that equals 1 only when the two 2-bit binary nos. A & B have the same value.

A $A_1 A_0$

B $B_1 B_0$



$z = (A_1 \odot B_1) \cdot (A_0 \odot B_0)$

9 8
↓ ↓
9 8

Dated: 05-09-2024

3- A logic circuit is required to produce 1 iff the binary no. $A_3 A_2 A_1 A_0$ has value greater than $3)_{10}$ but less than $8)_{10}$

$z = \Sigma(4,5,6,7)$ standard sum of mintesms exp.

A_3	A_2	A_1	A_0	z	mintesms (m)
0	0	0	0	0	$m_0 = \bar{A}_3 \bar{A}_2 \bar{A}_1 \bar{A}_0$
0	0	0	1	0	$m_1 = \bar{A}_3 \bar{A}_2 \bar{A}_1 A_0$
0	0	1	0	0	$m_2 = \bar{A}_3 \bar{A}_2 A_1 \bar{A}_0$
0	0	1	1	0	
0	1	0	0	1	
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	1	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	0	
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	0	

$$F(A_3, A_2, A_1, A_0) = \Sigma(4,5,6,7)$$
$$= \bar{A}_3 A_2 \bar{A}_1 \bar{A}_0 + \bar{A}_3 A_2 \bar{A}_1 A_0 + \bar{A}_3 A_2 A_1 \bar{A}_0 + \bar{A}_3 A_2 A_1 A_0$$

Simplified: $\bar{A}_3 A_2$ (All A_3 are 0 and A_2 are 1)

Dated: 05-09-2024

4- An alarm buzzer is to sound if the ignition (I) is turned on and the (G) gearshift is engaged, provided that the either of the front seats is occupied and the corresponding seat belt is not fastened. Develop the alarm's logic

left seat L
" " belt B_L

Right seat R
" " belt B_R

$$A = I \cdot G \cdot (L \cdot \overline{B_L} + R \cdot \overline{B_R})$$

5- Design a circuit that produces 0 iff the i/p binary no. $A_2A_1A_0$ has max possible value.

$$Z = \overline{A_2 A_1 A_0} \quad \text{(NANG gate)}$$

Karnaugh's Maps (K-MAP)

↳ graphical solution to the simplification problem

Constructing maps:

i/p variables : i # cells : 2^i

2-variable K-maps: # cells : $2^2 = 4$

F: A \ B	0	1
0	00	01
1	10	11

★ In K-maps, adjacent blocks should have 1 bit distance.
★ It uses greyreflected

Dated: 05-09-2024

3-variable K-map: # cells : $2^3 = 8$

F: A \ BC	00	01	11	10
0	000	001	011	010
1	100	101	111	110

* Right most and left most cells are always adjacent.
* diagnols are not adjacent.

4-variable K-map: # cells : $2^4 = 16$

F: AB \ CD	00	01	11	10
00	0000 0	0001 1	0011 3	0010 2
01	0100 4	0101 5	0111 7	0110 6
11	1100 12	1101 13	1111 15	1110 14
10	1000 8	1001 9	1011 11	1010 10

① $F(A, B, C) = \Sigma(0, 2, 3, 7)$

F: A \ BC	00	01	11	10
0	0 1	1	3 1	2 1
1	4	5	7 1	6

$$= BC + \bar{A}\bar{C}$$

Dated: 06-09-2024

Grouping in K-maps

1- Encircle groups of adjacent 1's (not diagonals)
2- Size of group should be power of 2.
   e.g: (1, 2, 4, 8, ...)
3- form the largest possible group.
4- right most and left most cells are adjacent.
5- upper most and lower most cells are adjacent.
6- Least no. of groups should cover all 1's.
7- Groups may overlap.
8- Atleast a single 1 should be unique in every group.
9- Note the non-changing variables for each group.

② $F(A, B, C) = \Sigma(0, 2, 4, 5, 6, 7)$

F: A \ BC	00	01	11	10
0	0: 1	1	3	2: 1
1	4: 1	5: 1	7: 1	6: 1

$= A + \bar{C}$

③ $F(A, B, C) = \Pi(0, 1, 5, 6)$ (Product of max-terms)

F: A \ BC	00	01	11	10
0	0	1	3: 1	2: 1
1	4: 1	5	7: 1	6

$= A\bar{B}\bar{C} + BC + \bar{A}B$

Dated:

(4) $F(A, B, C) = AC + \bar{B}\bar{C} + AB\bar{C}$

(5) $F(A, B, C, D) = \Sigma(0, 2, 4, 7)$

F: AB \ CD	00	01	11	10
00	0 1	1	3	2 1
01	4 1	5	7 1	6
11	12	13	15	14
10	8	9	11	10

$F(A, B, C, D) = \bar{A}BCD + \bar{A}\bar{B}\bar{D} + \bar{A}\bar{C}\bar{D}$

(6) $F(W, X, Y, Z) = \Sigma(3, 7, 11, 15)$

F: AB \ CD	00	01	11	10
00	0	1	3 1	2
01	4	5	7 1	6
11	12	13	15 1	14
10	8	9	11 1	10

$F(W, X, Y, Z) = YZ$

(7) $F(W, X, Y, Z) = \Sigma(1, 3, 5, 7, 11, 15)$

F: AB \ CD	00	01	11	10
00	0	1 1	3 1	2
01	4	5 1	7 1	6
11	12	13	15 1	14
10	8	9	11 1	10

$F(W, X, Y, Z) = YZ + \bar{W}Z$

Dated: 10-09-2024

(8) $F(W, X, Y, Z) = \Sigma(1, 3, 5, 7, 10, 11, 14, 15)$

(9) $F(W, X, Y, Z) = \Sigma(0, 2, 4, 5, 7, 10)$

F: AB \ CD	00	01	11	10
00	0 1	1	3	2 1
01	4 1	5 1	7 1	6
11	12	13	15	14
10	8	9	11	10 1

$F(W, X, Y, Z) = \bar{W}\bar{Y}\bar{Z} + \bar{W}XZ + \bar{X}Y\bar{Z}$

(10) $F(A, B, C, D) = \Pi(12, 13, 15)$

F: AB \ CD	00	01	11	10
00	0 1	1 1	3 1	2 1
01	4 1	5 1	7 1	6 1
11	12	13	15	14 1
10	8 1	9 1	11 1	10 1

$F(A, B, C, D) = \bar{A} + \bar{B} + C\bar{D}$

(11) $F(A, B, C, D) = \Sigma(8, 11, 12, 14, 15)$

F	00	01	11	10
00				
01				
11	1		1	1
10	1		1	

$F(A, B, C, D) = A\bar{C}\bar{D} + AB\bar{D} + ACD$

Dated: 10-09-2024

(12) $F(A,B,C,D) = (\bar{A}+\bar{B}+C+D)(A+\bar{B}+C+D)(A+B+C+\bar{D})$
$(A+B+\bar{C}+\bar{D})(\bar{A}+B+C+\bar{D})(A+B+\bar{C}+D)$

POS = Product of Sums (or) Product of maxterms

F	00	01	11	10
00	0 1	1	3	2
01	4	5 1	7 1	6 1
11	12	13 1	15 1	14 1
10	8 1	9	11 1	10 1

$= \Pi(12,4,1,3,2,9)$

$F(A,B,C,D) = BD + BC + AC + \bar{B}\bar{C}\bar{D}$

(13) $F(A,B,C,D) = \Sigma(0,1,2,3,6,7,12,13,14,15)$

F	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

(14) $= \Sigma(0,2,5,7,8,9,10,11)$

F, AB \ CD	00	01	11	10
00	1			1
01		1	1	
11				
10	1	1	1	1

$= \bar{A}BD + A\bar{B} + \bar{B}\bar{D}$

Dated: 10-09-2024

Dont care conditions

1- These are combination of input variables that we know will never occur.
2- They are marked as 'd' or 'x'.
3- In K-maps, these can be mapped treating them as 1's to form larger groups or can be disregarded entirely treating them as 0.

Example:

1- An 8421 BCD is a 4-bit coded system used to define decimal digits 0-9. (0000 - 1001)
2- The remaining 6 - combinations from 1010 - 1111 are don't cares.

Q: Design a logic circuit that produces 1 when input BCD digit is an odd number.

$F(A, B, C, D) = \Sigma(1, 3, 5, 7, 9) + d(10, 11, 12, 13, 14, 15)$

F: AB \ CD	00	01	11	10
00	0	1 1	3 1	2
01	4	5 1	7 1	6
11	12 d	13 d	15 d	14 d
10	8	9 1	11 d	10 d

$F(A, B, C, D) = D$

② $F(A, B, C, D) = \Sigma(0, 2, 4, 6, 8) + d(10, 11, 12, 13, 14, 15)$
$= \bar{D}$

* Directly write ans through LSB, for odd nos LSB is 1 and for even LSB is 0 so
for even : $\bar{D}$
for odd : D

Dated: 12-09-2024

③ $F(A,B,C,D) = \Sigma(1,3,7,11,15) + d(0,2,5)$

F	00	01	11	10
00	x	1	1	x
01		x	1	
11			1	
10			1	

$= CD + \bar{A}D$

④ $F(A,B,C,D) = \Sigma(0,1,2,9,11) + d(8,10,14,15)$

	00	01	11	10
00	1	1		1
01			x	x
11			1	x
10	x	1		

$= A\bar{B} + \bar{B}\bar{C} + \bar{B}\bar{D}$

5-variable K-map:

1- Make 2 maps of 16 cells each.
2- Map them separately
3- Now combine groups from the two maps which are at adjacent locations.

$F(A,B,C,D,E) = \Sigma(0,2,4,6,9,11,13,15,17,21,25,27,29,31)$

F AB \ CDE	000	001	011	010	110	111	101	100
00	0 1	1	3	2 1	6 1	7	5	4 1
01	8	9 1	11 1	10	14	15 1	13 1	12
11	24	25 1	27 1	26	30	31 1	29 1	28
10	16	17 1	19	18	22	23	21 1	22

$$\boxed{F = BE + A\bar{D}E + \bar{A}\bar{B}\bar{E}}$$

(All 1's) for full filled K-map => F = 1

Dated: 12-09-2024 (All 0's) for empty K-map => F = 0

method 2:

F: AB \ CDE	000	001	011	010	110	111	101	100
00	0: 1	1	3	2: 1	4: 1	5	7	6: 1
01	8	9: 1	11: 1	10	12	13: 1	15: 1	14
11	24	25: 1	27: 1	26	28	29: 1	31: 1	30
10	16	17: 1	19	18	20	21: 1	23	22

$= BE + A\bar{D}E + \bar{A}\bar{B}\bar{E}$

Standard SOP

② $F(A,B,C,D,E) = \Sigma(15, 23, 27, 29, 30, 31)$

AB \ CDE	000	001	011	010	110	111	101	100
00								
01						1		
11			1		1	1	1	
10						1		

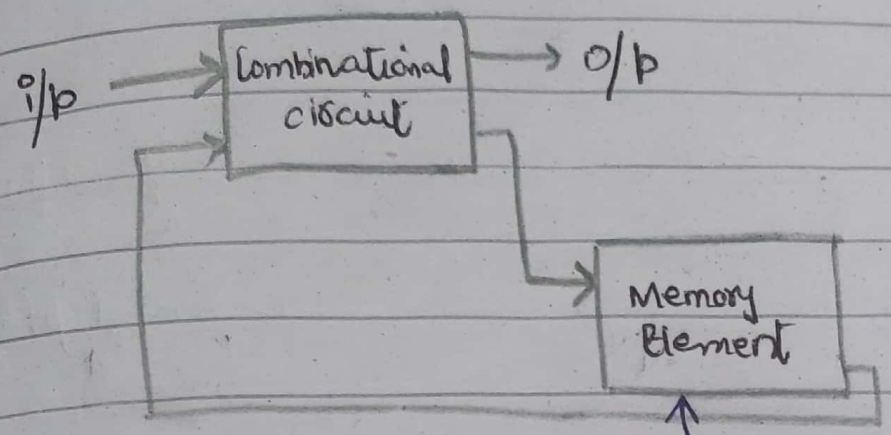
$= ABDE + BCDE + ACDE + ABCD + ABCE$

③ $F(A,B,C,D,E) = \Sigma(1, 5, 6, 7, 14, 15, 16, 17, 20, 22, 23, 30, 31)$

$F(A,B,C,D,E) =$

behaviors
Expression
Logic diagram
Truth Table

SEQUENTIAL CIRCUITS



state of the circuit
1 bit → 2 states < 0, 1
2 bits → 4 states < 00, 01, 10, 11

* In combinational circuits, o/p depends on the present i/ps
* In sequential ckt, o/p depend both on present state as well as current i/ps
* Behaviour of sequential circuits is described by a state table or state diagram.

Finite State Machine (FSM)

Behaviour of logic circuit
↳ logic expression
↳ ckt diagram
↳ Truth table

Memory Element :-

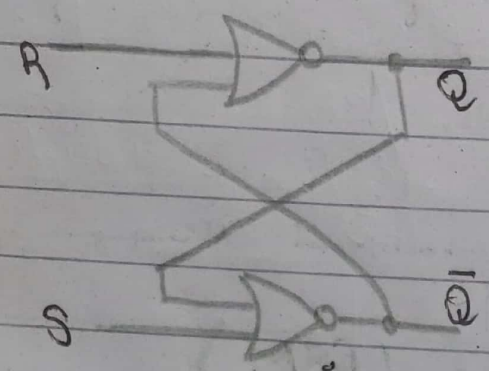
* Latches & FlipFlops :

↳ Bistable devices That store 1 bit of info
Latch.

↳ Basic unit of storage are Latch and FlipFlop

Latch :-

↳ Latch is a memory element by which we fetch bit from Mem.

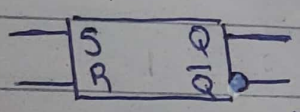


cross compiled configration

State Table

Q	S	R	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0 (invalid)
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0 (invalid)

S → Set → (means storing 1)
R → Reset → (means storing 0)



Active high i/p latch

Bistable latch
↓
bcz it has 2 stable o/p

Characteristic table

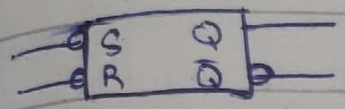
S	R	Q_{t+1}
0	0	Q_t (no change)
0	1	0
1	0	1
1	1	0 (invalid)

In Table,

* Set & Reset → both inactive (0) → o/p will not change
* Resest → active (1) → o/p is 0
* Set → active (1) → o/p is 1
* set & Reset → both active (R=1, S=1) → o/p is invalid

bcz of this reason (invalid o/p) we don't use Latch

bcz of Q=0 & $\bar{Q}$=0 which makes the hardware confused

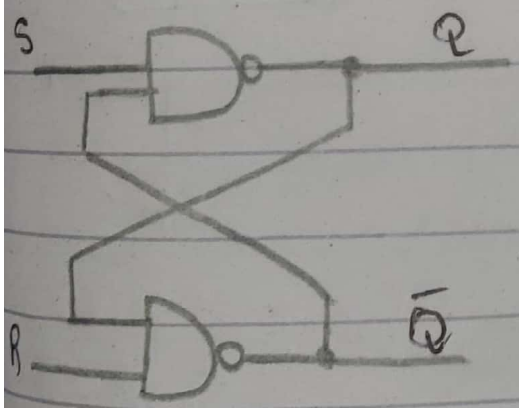


active low i/p latch

S → set → (means storing 0)
R → Reset → (means storing 1)

Characteristic table

S	R	Q_{t+1}
0	0	1 (Invalid)
0	1	1
1	0	0
1	1	Q_t (NC)



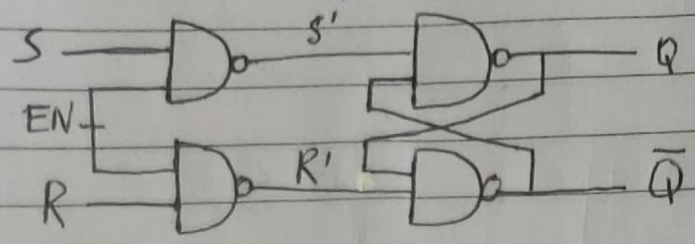
State Table

Q_t	S	R	Q_{t+1}
0	0	0	1 (invalid)
0	0	1	1
0	1	0	0
0	1	1	0 (NC)
1	0	0	1 (invalid)
1	0	1	1
1	1	0	0
1	1	1	1 (NC)

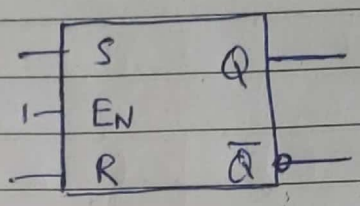
Dated: 19-09-2024

Gated SR Latch

NAND implementation



E	Q_t	S	R	Q_{t+1}
1	0	0	0	0 (NC)
1	0	0	1	0
1	0	1	0	1
1	0	1	1	inv.
1	1	0	0	1 (NC)
1	1	0	1	0
1	1	1	0	1
1	1	1	1	inv.
0	Q_t	x	x	Q_t

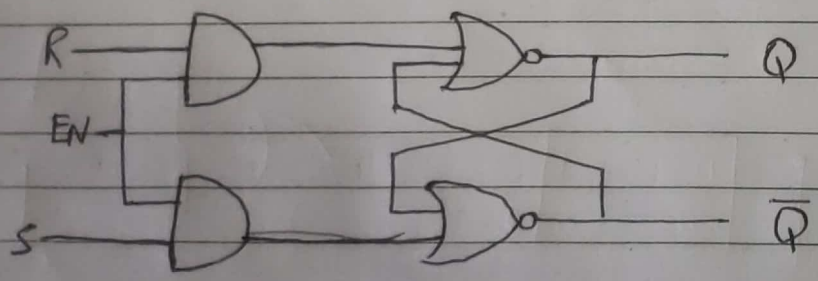


Characteristic eqn:

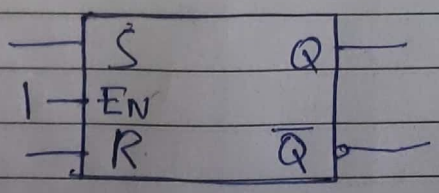
Q_t \ SR	00	01	11	10
0			x	1
1	1		x	1

$Q_{t+1} = S + Q_t \bar{R}$

NOR implementation

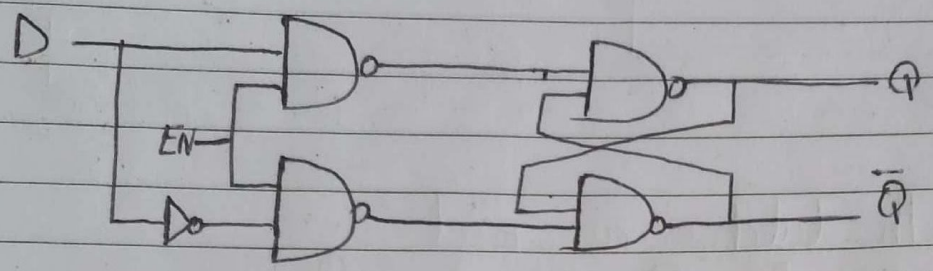


E	Q_t	S	R	Q_{t+1}
1	0	0	0	0 (NC)
1	0	0	1	0
1	0	1	0	1
1	0	1	1	inv
1	1	0	0	1 (NC)
1	1	0	1	0
1	1	1	0	1
1	1	1	1	inv
0	Q_t	x	x	Q_t



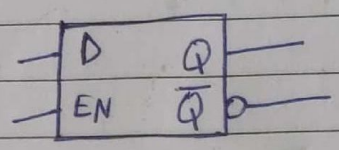
Dated: 19-09-2024

D-Latch (Transparent latch)



EN	Q_t	D	Q_{t+1}
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1
0	Q_t	x	Q_t

$$\boxed{Q_{t+1} = D}$$



* Transparent latch because the input is reflected directly to next state, irrespective of the present state.

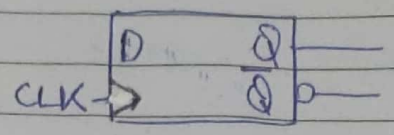
* Data latch because whatever input is given at D, it stores in latch.

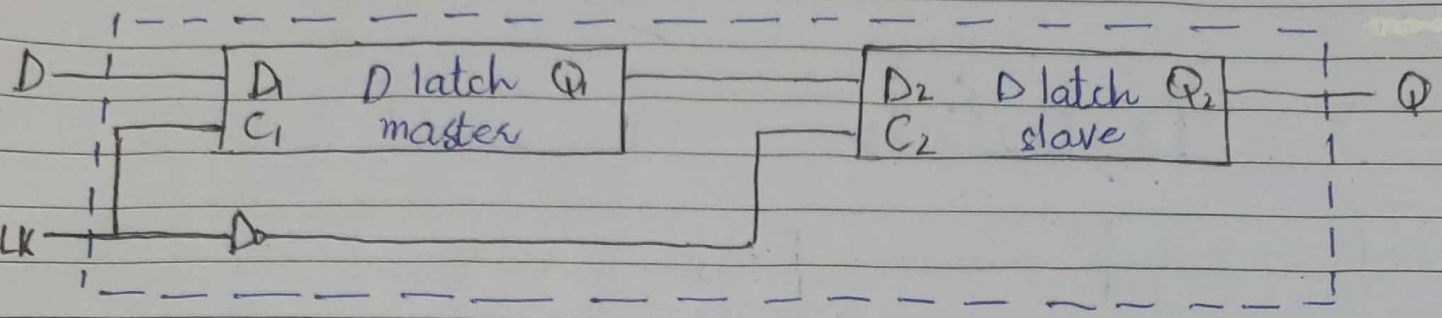
-ve edge trigger : bubble on CLK

Dated: 24-09-2023

DFFP:

Master slave DFFP:

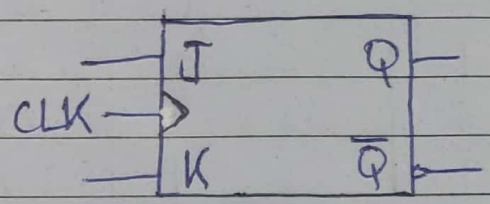


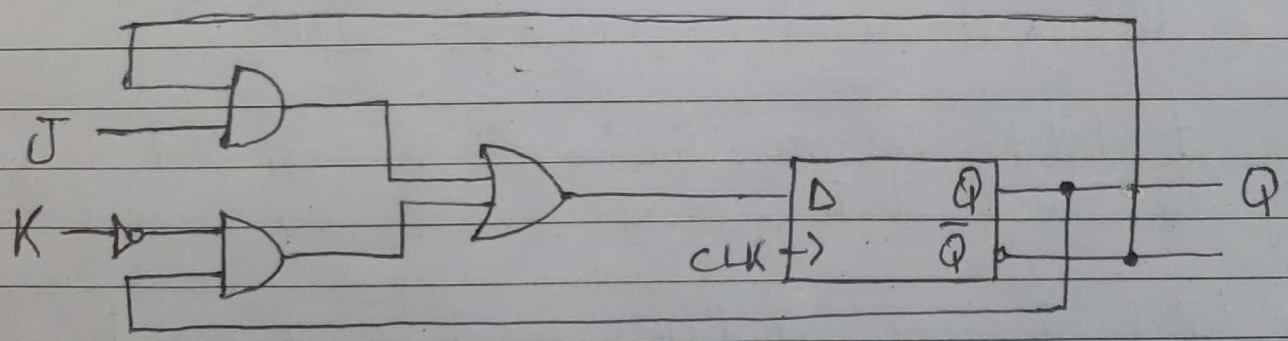


Initially both latches are reset.

	CLK	C_1	D_1	Q_1	C_2	D_2	Q_2
Initially	0	0	x	0	1	0	0

JK FPP:





Characteristic table:

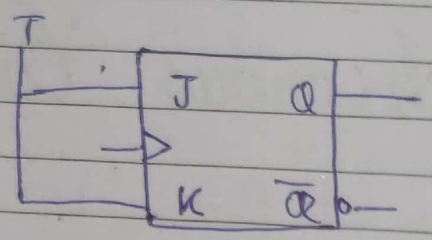
J	K	Q_{t+1}
0	0	Q_t (NC)
0	1	0 (reset)
1	0	1 (set)
1	1	$\overline{Q_t}$ (toggle)

Dated: 24-09-24

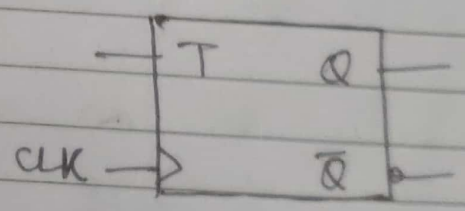
State table:

CLK	Q_t	J	K	Q_{t+1}
↑	0	0	0	0
↑	0	0	1	0
↑	0	1	0	1
↑	0	1	1	1
↑	1	0	0	1
↑	1	0	1	0
↑	1	1	0	1
↑	1	1	1	0
↓/1/0	Q_t	x	x	$\overline{Q_t}$

T ffp:



CLK	Q_t	T	Q_{t+1}
↑	0	0	0
↑	0	1	1
↑	1	0	1
↑	1	1	0
↓/1/0	Q_t	x	$\overline{Q_t}$

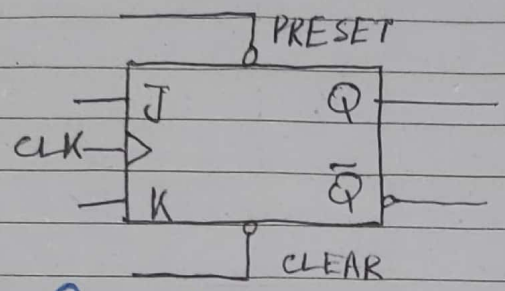


Dated: 26-09-24

Asynchronous Inputs

Preset and Clear

* Preset when enabled, stores 1 in the ffp.
  Clear when enabled, stores 0 in the ffp.
* These inputs function irrespective of the clock.



Counters

Binary Asynchronous Counters:

for an n-bit counter

# of stages / # ffps = n

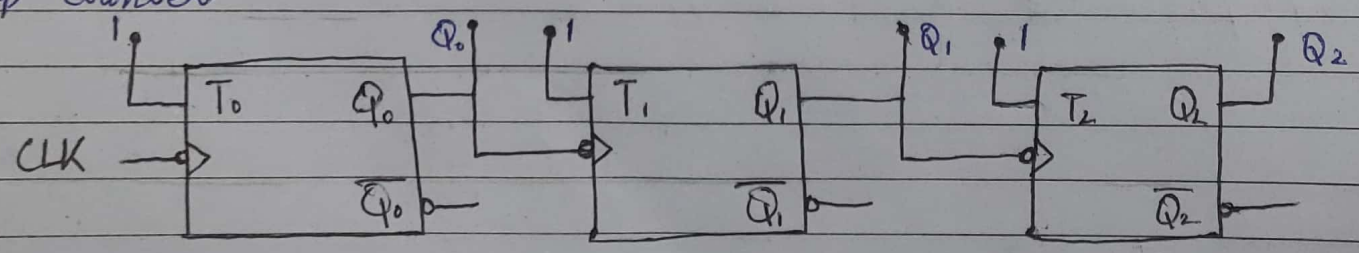
max count = $2^n - 1$

Total # of counts = 2^n

* Made only with JK or T ffp.

3-bit Counter

Up Counter

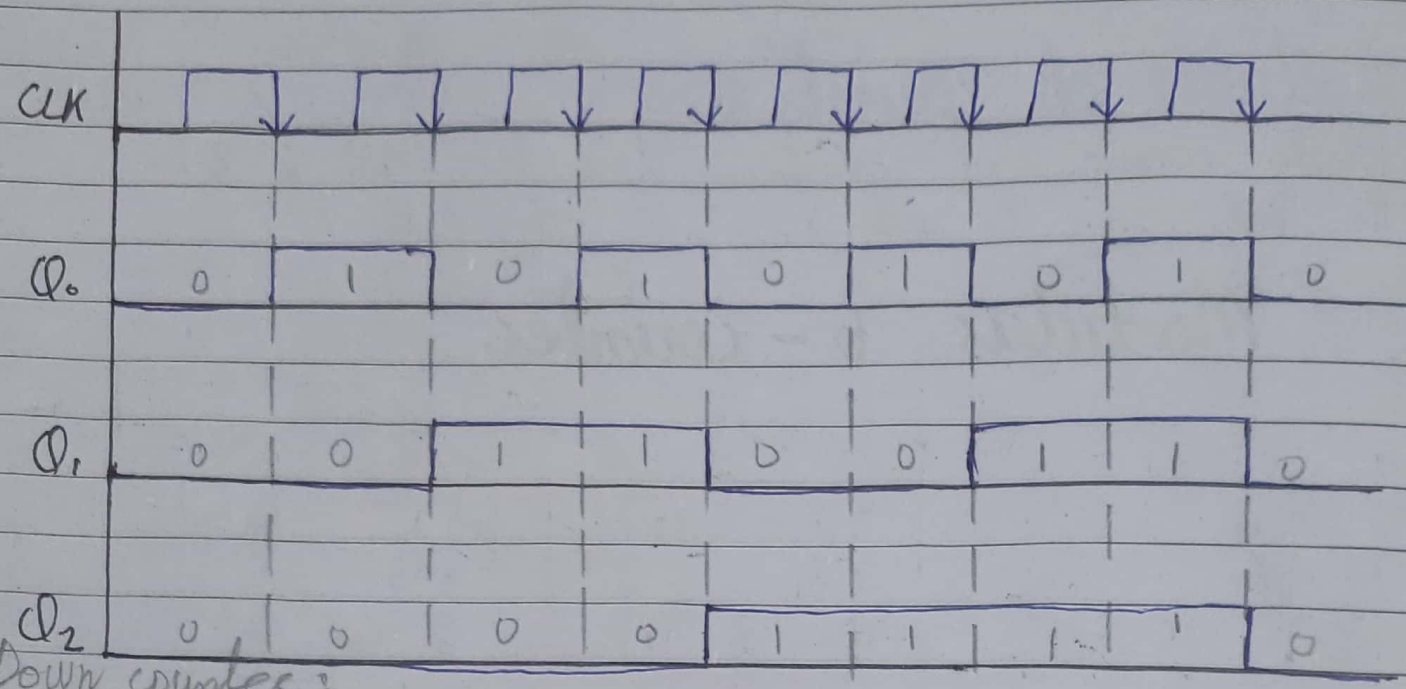


all initially reset.

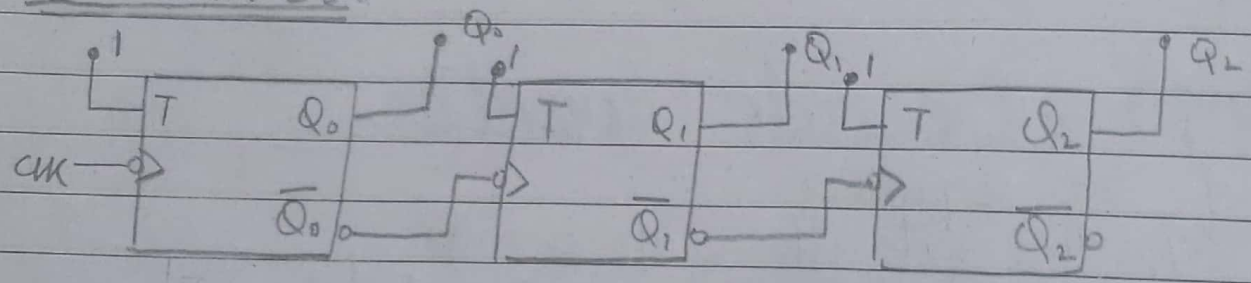
* all have different clk pulses thats why asynchronous.

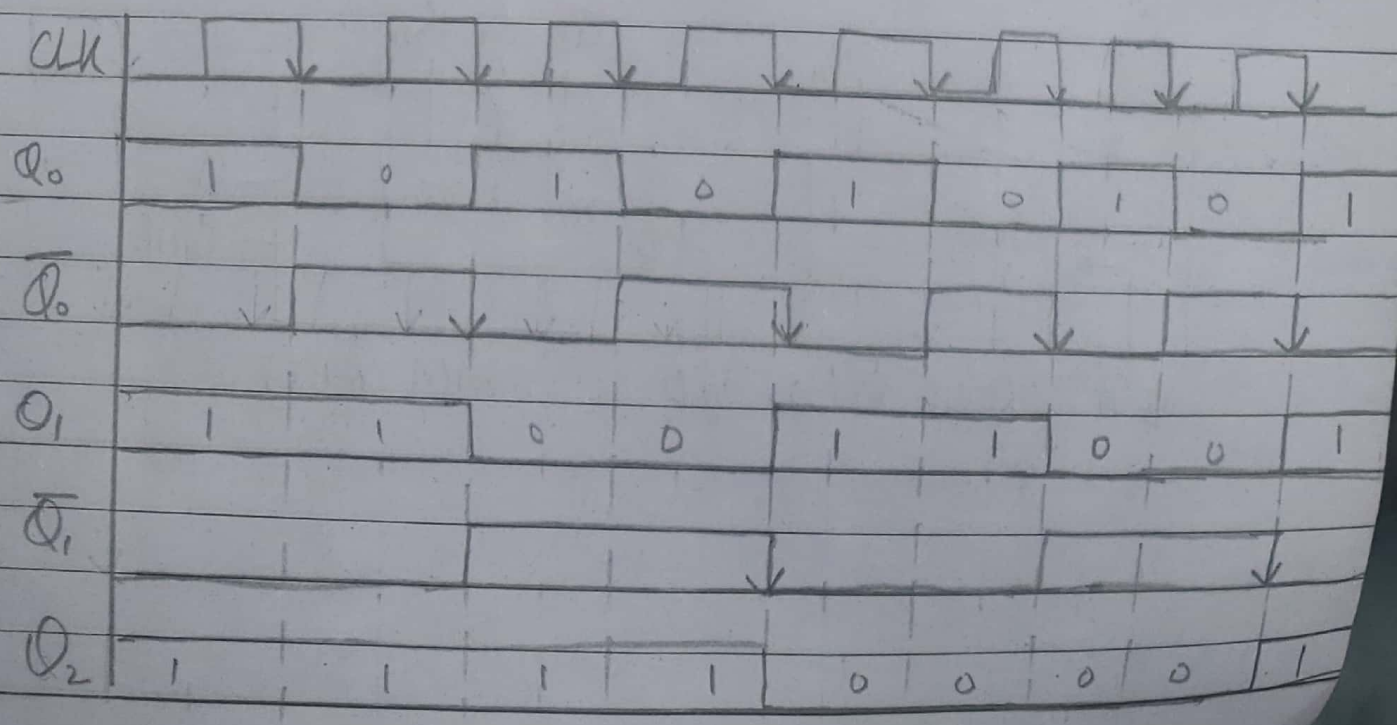
Dated: 26-09-2024

* Modulus 8-counter also Divide by 8 counter.



Down counter:





Dated: 01-10-2024

Divide by Modulus Counter

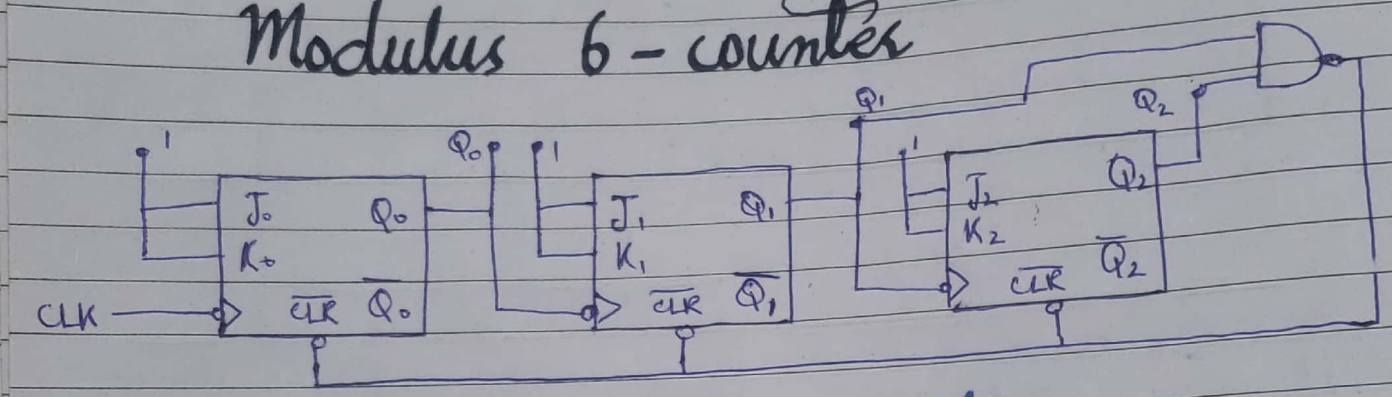
freq of MSB = $\frac{\text{freq. of CLK}}{\text{mod. value}}$

Modulus of counter:

↳ Total counts of counter. (i.e for 3-bit is 8)

* the freq. of Q_2 will be 1/8th of CLK.

Modulus 6 - counter

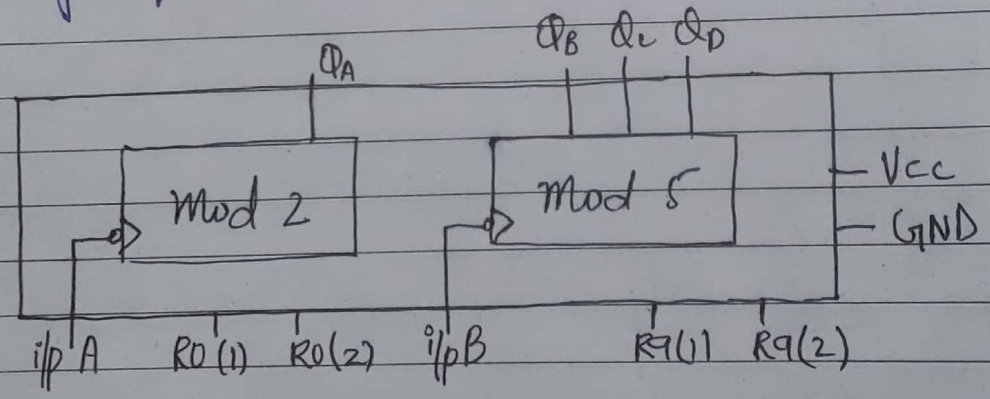


7490 - Decade Counter

* Capable of two sequences

BCD seq: i/p A : ext. clk , i/p B : Q_A

Biquinary seq: i/p B : ext. clk , i/p A : Q_D



$\underbrace{R_0(1), R_0(2), R_9(1), R_9(2)}_{\text{Active high}}$

BCD seq.

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
0	0	0	0	0

↳ from (0-9) then it will switch to zero again

↳ Here Q_A changes its transition as the pulse occurs

Biquinary seq.

0	0	0	0	0
0	0	1	0	2
0	1	0	0	4
0	1	1	0	6
1	0	0	0	8
0	0	0	1	1
0	0	1	1	3
0	1	0	1	5
0	1	1	1	7
1	0	0	1	9
0	0	0	0	0

↳ Bcz of MOD-5 all will be reset and Then cycle will again continue untill zero appeared

↳ Here Q_B changes its transition as the pulse occurs.

i/p R(0)1 R(0)2 i/p R9(1) R9(2)

Reset i/ps

R(0)1	R(0)2	R9(1)	R9(2)
1	1	0	X
1	1	X	0
X	X	1	1
0	X	0	X
X	0	0	X
0	X	X	0
0	X	X	0

Output.

Q_D	Q_C	Q_B	Q_A
0	0	0	0
0	0	0	0
1	0	0	1

CamScanner

Dated: 03-10-2024

Practice Problem

1- Design a MOD-11 asynchronous up counter.
2- Implement MOD-6 counter using 7490.
3- " MOD-7 " " " 7490.
4- Design seconds portion of a digitial clock using 7490.
5- " hours " " " " " " "
Assume 12 hr clock.
6- Same as 5. Assume 24 hr clock.

IC:

7492 - Divide by 12 counter (divide by 2 & divide by 6)
7493 - 4-bit binary counter.

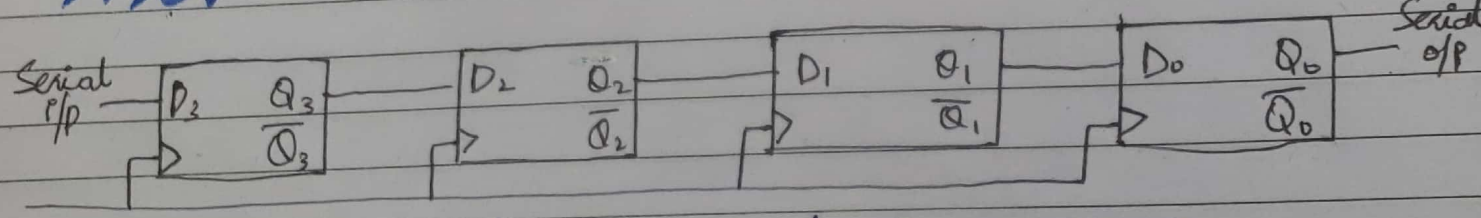
Shift Registers

★ always synchronous.

★ Registers: set of flip flops.
↳ parallel loading
↳ serial loading.
for 64-bit serial o/p, 63 pulses are required (due to shift) →

4 types:
1- Serial in serial out SISO
2- Serial in parallel out SIPO
3- Parallel in parallel out PIPO
4- Parallel in serial out PISO

SISO:



after clock pulse	Serial i/p	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4		1	1	1	0

CamScanner

Dated: 03-10-2024

Practice Problem

1- Design a MOD-11 asynchronous up counter.
2- Implement MOD-6 counter using 7490:
3- " MOD-7 " " "
4- Design seconds portion of a digitial clock using 7490.
5- " hours " " " " " " "
Assume 12 hr clock.
6- Same as 5. Assume 24 hr clock.

IC:

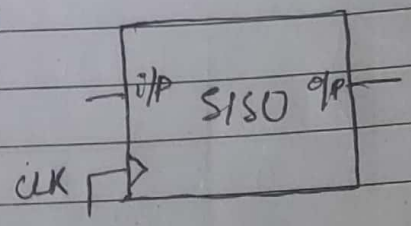
7492 - Divide by 12 counter (divide by 2 & divide by 6)
7493 - 4-bit binary counter.

Shift Registers

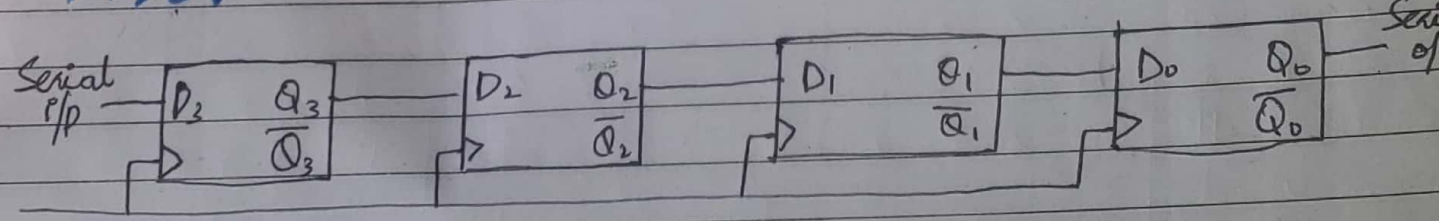
★ always synchronous.

★ Register: set of flip flops.
↳ parallel loading
↳ serial loading.
for 64-bit serial o/p, 63 pulses are required (due to shift) →

4 types:
1- Serial in serial out SISO
2- Serial in parallel out SIPO
3- Parallel in parallel out PIPO
4- Parallel in serial out PISO



SISO:

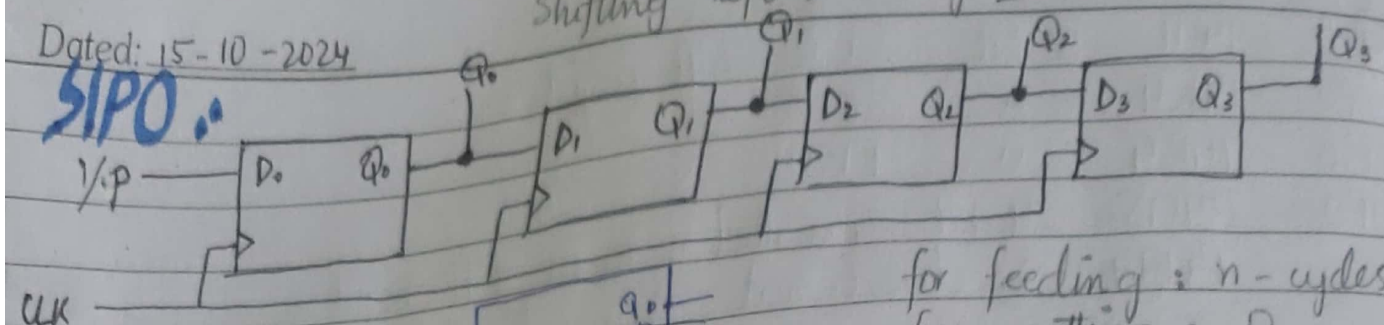


after clock pulse	Serial i/p	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4		1	1	1	0

Shifting right : ÷ by 2
Shifting left : × by 2^n

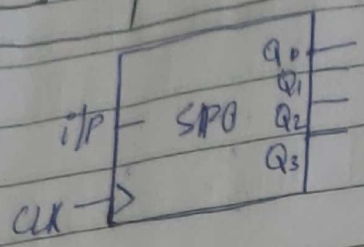
Dated: 15-10-2024

SIPO :

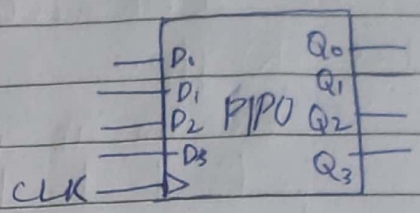


⇓

Shift left (to MSB)

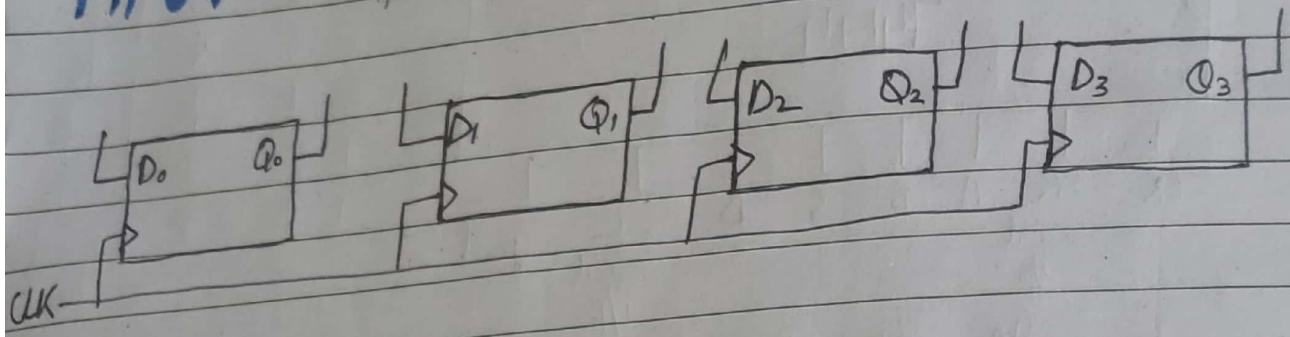


for feeding : n-cycles
for getting : 0



PIPO :

for feeding : 1 cycle
for getting : 0



PISO :

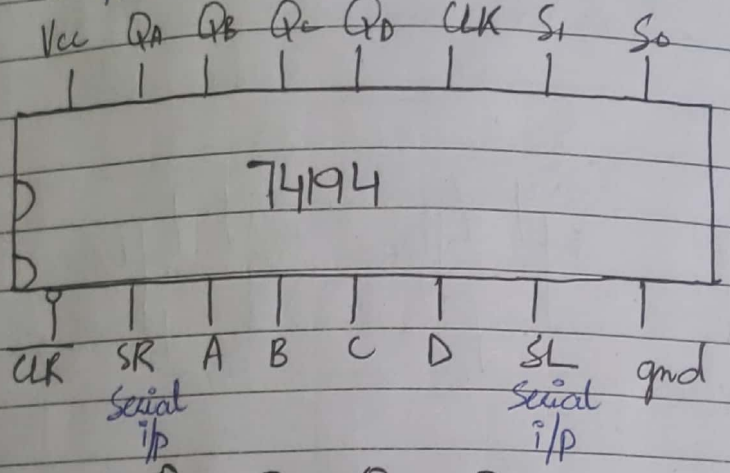
Setup time: As soon as the clk edge arrives, just consider the i/p which was just before the edge.

Dated: 17-10-2024

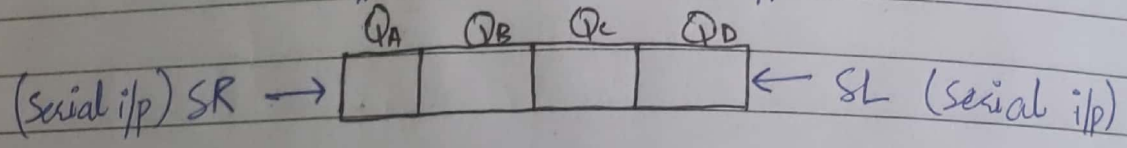
SISO	7491	8-bit	SISO	shift register	
SIPO	74164	"	SIPO	"	"
PISO	74165	"	PISO	"	"
PIPO	74195	4-bit	parallel	access shift register	
	74194	bi-directional	universal	shift register	

74194: 4-bit bidirectional universal shift registers.

↳ uses SR ffp.



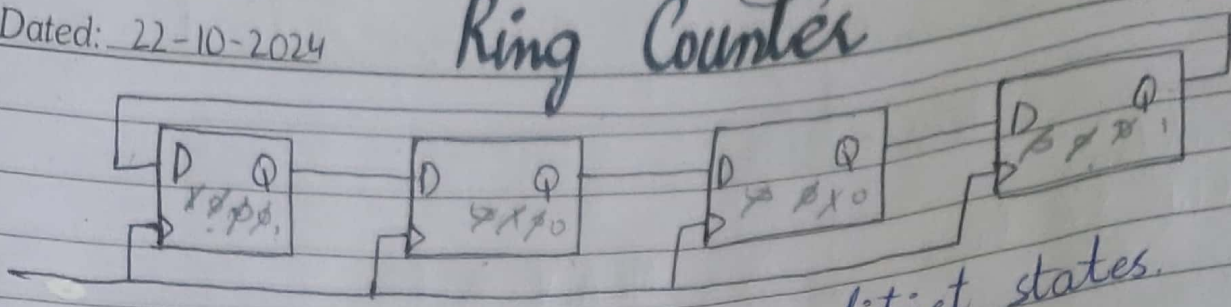
S_1	S_0	CLK	Action
0	0	X	NC
0	1	↑	SR
1	0	↑	SL
1	1	↑	parallel load



CLR	MODE S_1	S_0	CLK	Serial left	right	Parallel A	B	C	D	o/ps Q_{A+1}	Q_{B+1}	Q_{C+1}	Q_{D+1}
L	x	x	x	x	x	x	x	x	x	0	0	0	0
H	x	x	L/H	x	x	x	x	x	x	Q_{A+}	Q_{B+}	Q_{C+}	Q_{D+}
H	H	H	↑	x	x	a	b	c	d	a	b	c	d
H	L	H	↑	x	H	x	x	x	x	H	Q_{A+}	Q_{B+}	Q_{C+}
H	L	H	↑	x	L	x	x	x	x	L	Q_{A+}	Q_{B+}	Q_{C+}
H	H	L	↑	H	x	x	x	x	x	Q_{B+}	Q_{C+}	Q_{D+}	H
H	H	L	↑	L	x	x	x	x	x	Q_{B+}	Q_{C+}	Q_{D+}	L
H	L	L	x	x	x	x	x	x	x	Q_{A+}	Q_{B+}	Q_{C+}	Q_{D+}

Dated: 22-10-2024

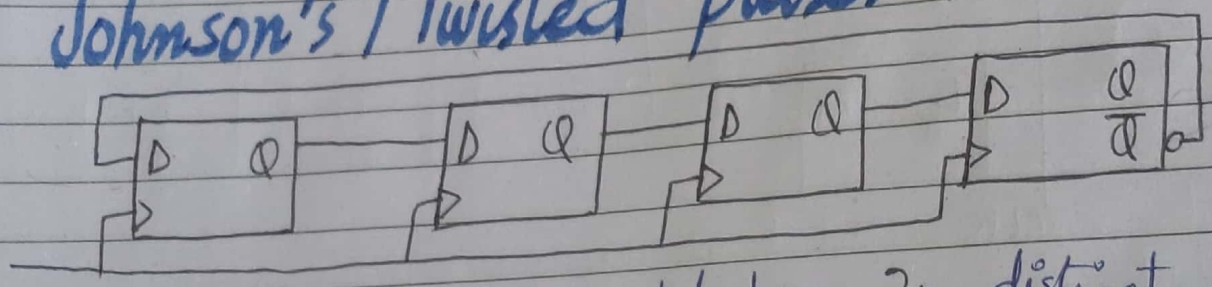
Ring Counter



- n-bit ring counter would have n distinct states.
- used in fancy displays.
- Decoding gates are not required for decimal conversion
- Used as rotate register.

CLK	O/P
0	1 0 0 0
1	0 1 0 0
2	0 0 1 0
3	0 0 0 1
4	1 0 0 0

Johnson's / Twisted pulses



- n-bit Johnson counter would have 2n distinct states.
- can be used for obtaining 1's complement of a number (after n pulses)

CLK	O/P
0	1 1 0 0
1	1 1 1 0
2	1 1 1 1
3	0 1 1 1
4	0 0 1 1
5	0 0 0 1
6	0 0 0 0
7	1 0 0 0
8	1 1 0 0

⇒ 1's complement

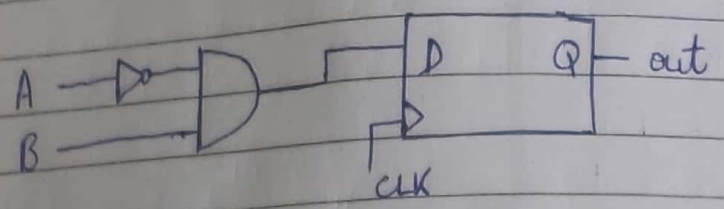
Dated: 22-10-2024

Design of a Sequential machine

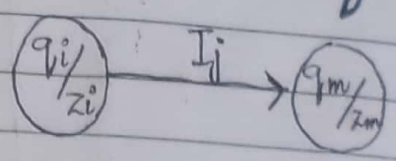
Two models.
1- MOORE machine 2- MEALY machine

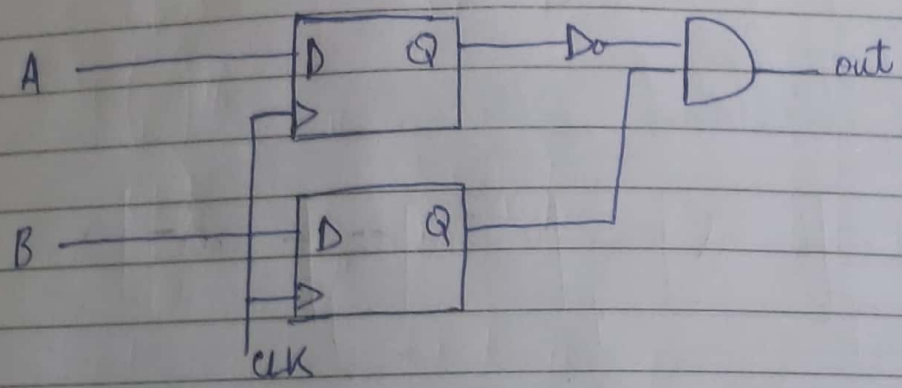
1- MOORE machine:

↳ o/p depends only on the present states.
↳ o/p change is synchronous, i.e it occurs at predictable time.



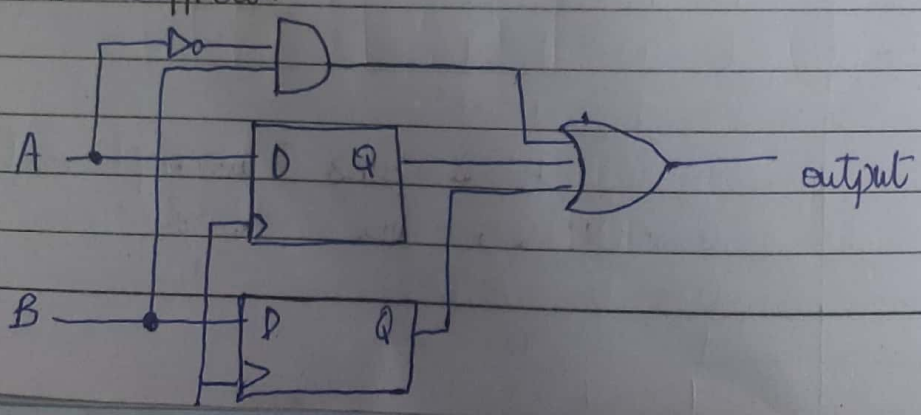
State diagram



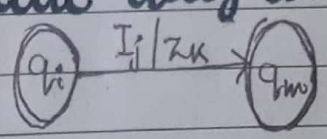


2- MEALY machine:

↳ o/p depends both on the present state as well as current i/ps.
↳ o/p change is irrespective of the CLK, the time analysis is difficult.



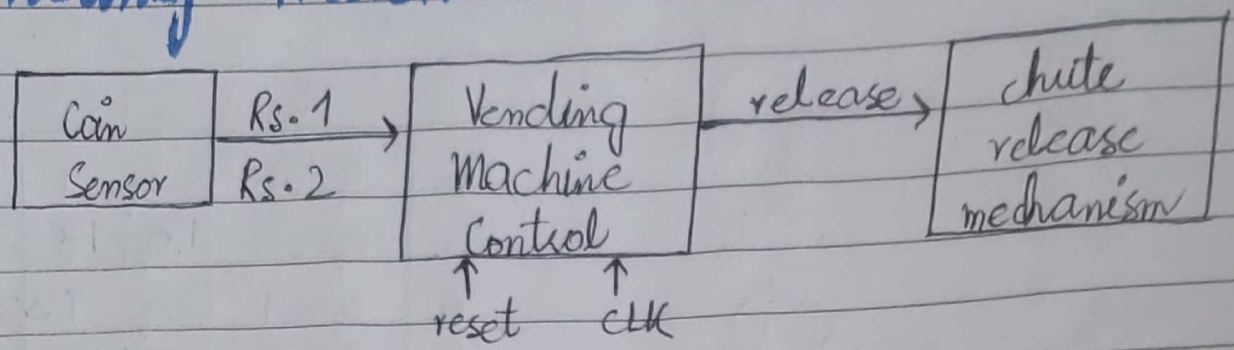
State diagram



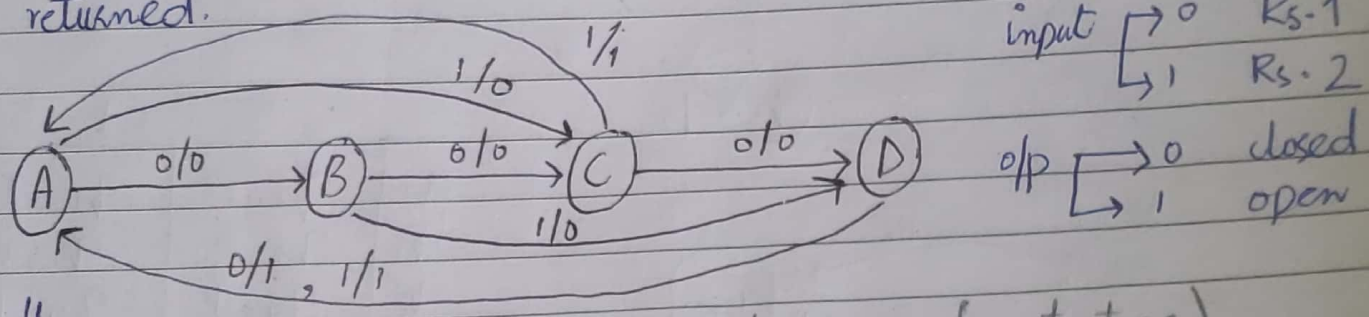
Dated: 24-10-2024

FSM example

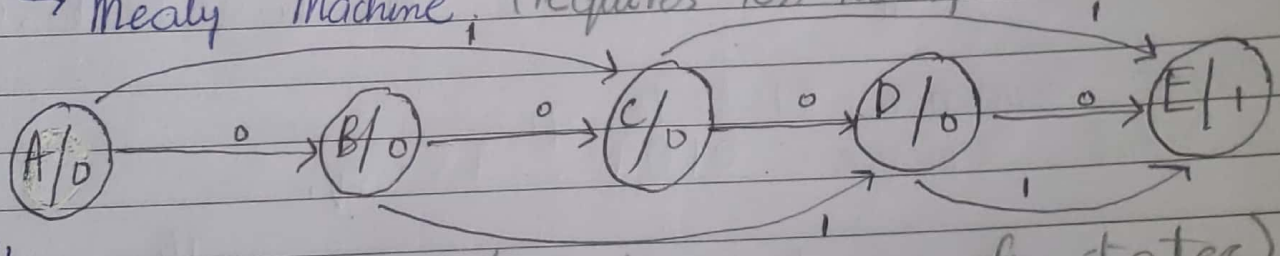
Vending Machine Controller:



Design an FSM that accepts Rs.1 and Rs.2 coins. When the value of money equals 4, the machine vends the item.
* Assume that the user enters fixed amount and no change is returned.

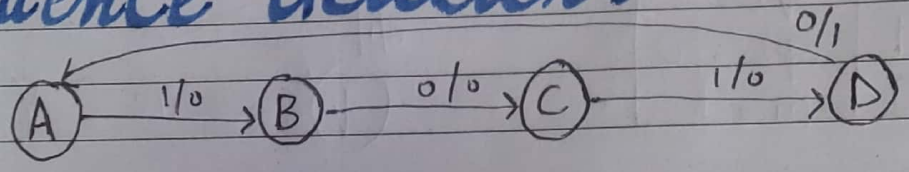


↳ Mealy Machine (requires less no of states)



↳ Moore Machine (usually more no of states)

Sequence detector: 1010



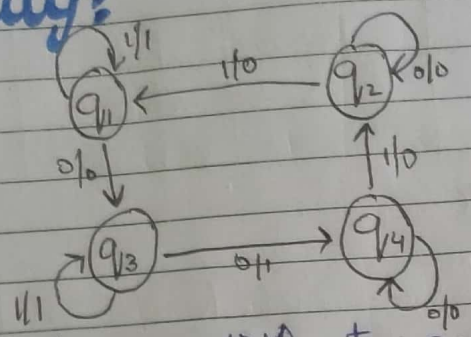
IS: 0111010101111011 ⇒ 2 ways can be made, one is like if you store the previous one and start the remaining after that. The other one is like, dont store the prev, and every sequence would be unique.

Dated: 29-10-2024

State diagram and State tables

Mealy:

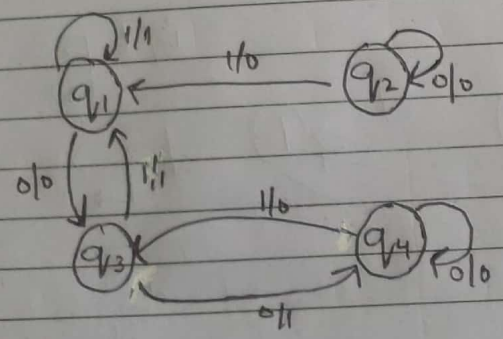
(1)



	0	1
q_1	$q_3, 0$	$q_1, 1$
q_2	$q_2, 0$	$q_1, 0$
q_3	$q_4, 1$	$q_3, 1$
q_4	$q_4, 0$	$q_2, 0$

Input Sequence: 1010 to q_3
Output Sequence: 1100 FS: q_2

(2)

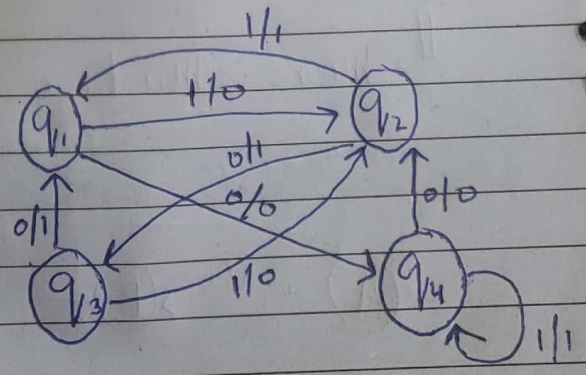


	0	1
q_1	$q_3, 0$	$q_1, 1$
q_2	$q_2, 0$	$q_1, 0$
q_3	$q_4, 1$	$q_1, 1$
q_4	$q_4, 0$	$q_3, 0$

Input Sequence: 0101 to q_1
Output Sequence: 0101 FS: q_1

(3)

	0	1
q_1	$q_4, 0$	$q_2, 0$
q_2	$q_3, 1$	$q_1, 1$
q_3	$q_1, 1$	$q_2, 0$
q_4	$q_2, 0$	$q_4, 1$

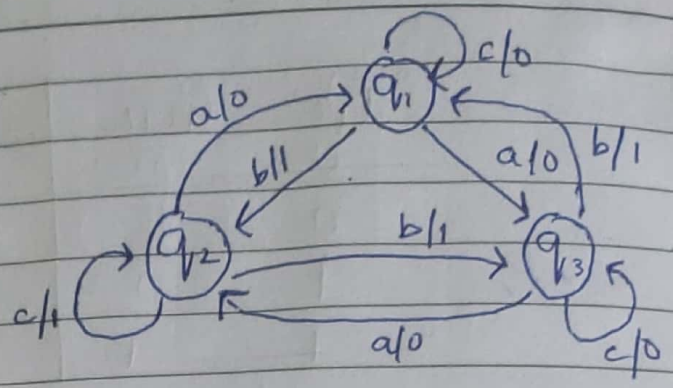


Input Sequence: 0110 to q_2
Output Sequence: 1010 FS: q_4

Dated: 29-10-2024

(4)

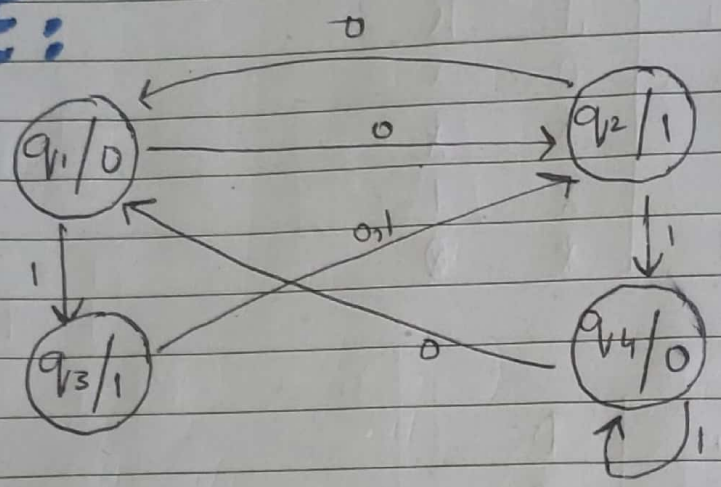
	a	b	c
q_1	q_3, 0	q_2, 1	q_1, 0
q_2	q_1, 0	q_3, 1	q_2, 1
q_3	q_2, 0	q_1, 1	q_3, 0



Input Seq: aab to q_3
OS : 001 FS : q_2

MOORE :

(1)



	0	1	
q_1	q_2	q_3	0
q_2	q_1	q_4	1
q_3	q_2	q_2	1
q_4	q_1	q_4	0

IS : 011 to q_1
OS : 100

★ In MOORE model, when input is applied make the transition first and mention the next state's o/p.

(2)

	0	1	
q_1	q_3	q_5	0
q_2	q_4	q_1	1
q_3	q_5	q_2	0
q_4	q_1	q_3	1
q_5	q_2	q_4	0

IS : 111 to q_5, OS : 101
IS : 010 to q_3, OS : 010

(3)

	0	1	
q_1	q_2	q_2	0
q_2	q_5	q_4	0
q_3	q_3	q_1	0
q_4	q_1	q_5	1
q_5	q_4	q_4	1

IS : 000 to q_4
OS : 001

CamScanner

Excitation

Dated: 29-10-2024

SR:

Q_t	Q_{t+1}	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

JK:

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

D:

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

T:

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

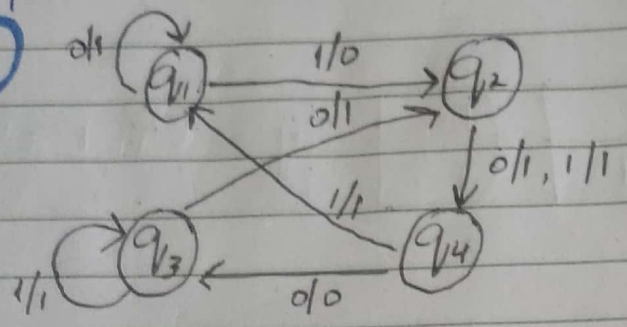
CamScanner

Dated: 31-10-2024

FSM Design Procedure

1- Understand the problem
2- Develop state table / state diagram
3- Perform state minimization
4- Perform state assignment
   # state variables (sv) = $\lceil \log_2 N \rceil$
   N : # states
5- Choose a ffp
6- Implement the FSM.

Ex 1



IS: 1011 at q_1
OS: 0110

State assignment:

$q_1 \Rightarrow 00$ $q_2 \Rightarrow 01$ $q_3 \Rightarrow 10$ $q_4 \Rightarrow 11$

State table:

FFP i/p column

x	$Q_1 t$	$Q_0 t$	$Q_1 t+1$	$Q_0 t+1$	y	T_1	T_0
0	0	0	0	0	1	0	0
0	0	1	1	1	1	1	0
0	1	0	0	1	1	1	1
0	1	1	1	0	0	0	1
1	0	0	0	1	0	0	1
1	0	1	1	1	1	1	0
1	1	0	1	0	1	0	0
1	1	1	0	0	1	1	1

CamScanner

Dated:

Dated: 05-1

Ex: 2.

T_1 Q_1Q_0 / x	00	01	11	10
0		1		1
1		1	1	

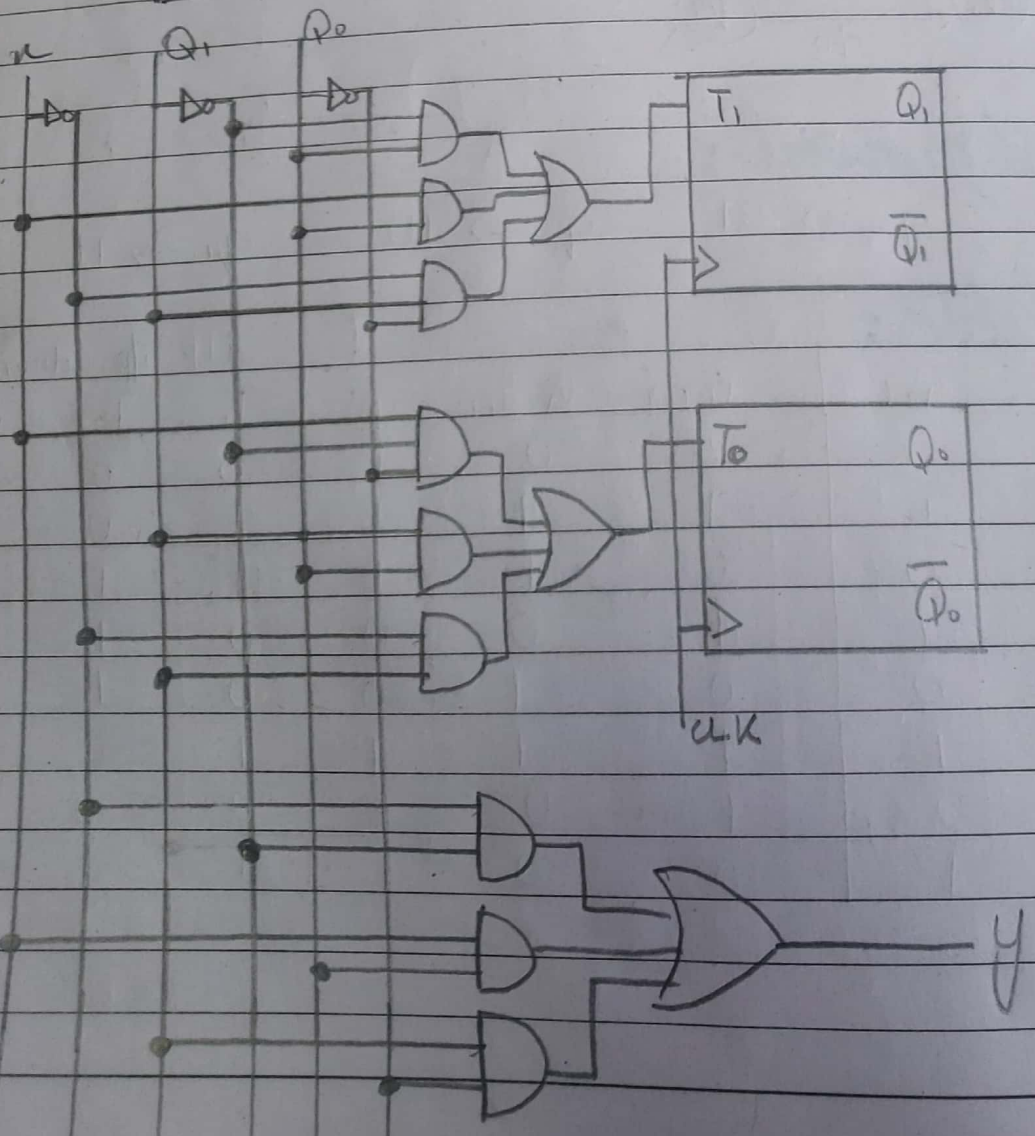
$$T_1 = \overline{Q_1}Q_0 + xQ_0 + \bar{x}Q_1\overline{Q_0}$$

T_0	00	01	11	10
0			1	1
1	1		1	

$$T_0 = x\overline{Q_1}\,\overline{Q_0} + Q_1Q_0 + \bar{x}Q_1$$

y	00	01	11	10
0	1	1		1
1		1	1	1

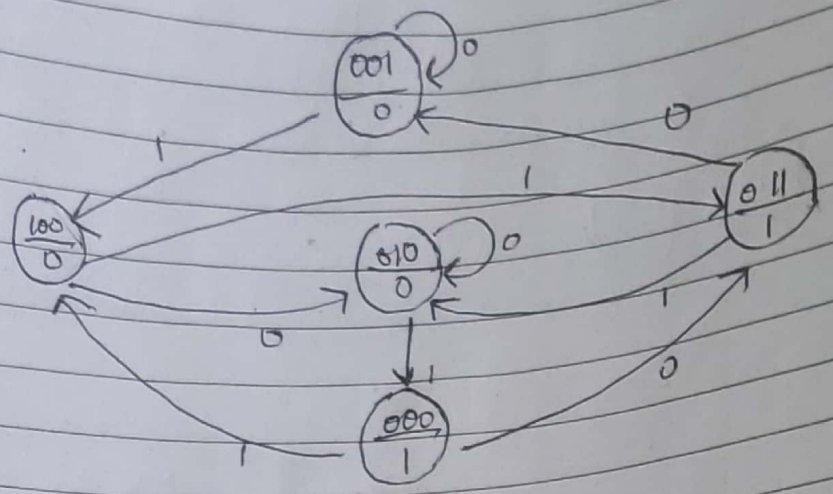
$$y = \bar{x}\,\overline{Q_1} + xQ_0 + Q_1\overline{Q_0}$$



St

Dated: 05-11-2024

Ex: 2.



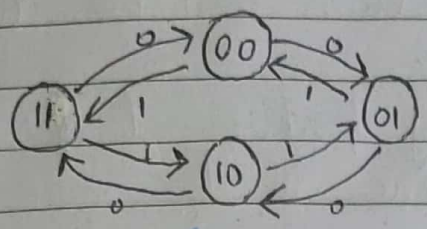
State Table:

$Q_2(t)$	$Q_1(t)$	$Q_0(t)$	x	$Q_2(t+1)$	$Q_1(t+1)$	$Q_0(t+1)$	y	T_2	T_1	T_0
0	0	0	0	0	1	1	1	0	1	1
0	0	0	1	1	0	0	1	1	0	0
0	0	1	0	0	0	1	0	0	0	0
0	0	1	1	1	0	0	0	1	0	1
0	1	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0
0	1	1	0	0	0	1	1	0	1	0
0	1	1	1	0	1	0	1	0	0	1
1	0	0	0	0	1	0	0	1	1	0
1	0	0	1	0	1	1	0	1	1	1
1	0	1	0	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x

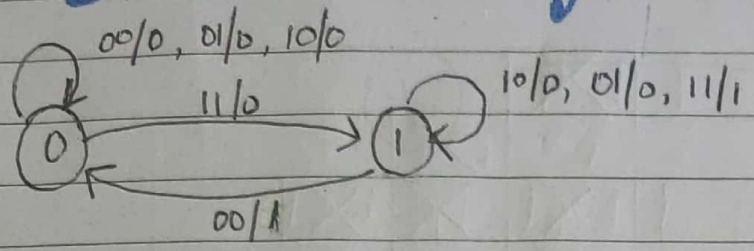
Dated: 07-11-2024

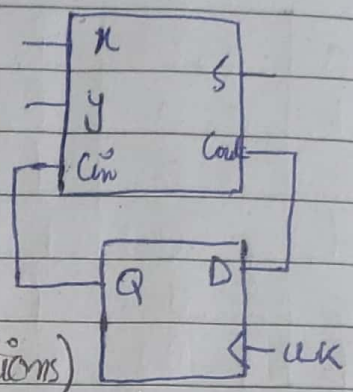
2-bit up/down counter

$x \Rightarrow$ 0 : up
1 : down



Serial binary adder





Input variables = 2 (x and y) (i.e 4 combinations)
Output indicates Sum.
State indicates Carry bit.
*1 ffp, so 1 state variable. (i.e 2 combinations) ⇒ (0 and 1

State minimization of FSMs :-

↳ removing extra states

Redundant states : Those whose functions can be accomplished by other states

Two states are equivalent if every possible set of i/ps generate the same o/p & take the machine to the same next state.

eg

	0	1
a	b, 0	d, 1
b	e, 0	f, 1
c	b, 0	d, 1
d	e, 0	b, 1
e	e, 0	f, 1
f	f, 0	c, 0

Pair chart method

b	be/df				
c	✓	be/df			
d	be/bd	bf	be/bd		
e	be/df	✓	be/df	bf	
f	X	X	X	X	X
	a	b	c	d	e

o/p compatible,

ac

be

a Δ c

b Δ e

Reduced stateable

	0	1
a	b, 0	d, 1
b	b, 0	f, 1
d	b, 0	b, 1
f	f, 0	a, 0

Q3 Δ02

	0	1	
q_1	q_3	q_0	0
q_2	q_5	q_4	0
q_3	q_5	q_6	0
q_4	q_5	q_1	1
q_5	q_1	q_2	0
q_6	q_6	q_4	0

q_2	~~24~~				
q_3	~~35~~/26	~~35~~/46			
q_4	×	×	×		
q_5	13	~~15~~/23	~~15~~/46	×	
q_6	~~35~~/23	35	~~46~~	×	~~15~~/24
	q^1	q^2	q^3	q^4	q^5

15, 13, 26, 35

2 Δ 6

1 Δ 3 Δ 5

	0	1	
q_1	q_1	q_2	0
q_2	q_1	q_4	0
q_4	q_1	q_1	1

Ex#03

	0	1
1	3, 0	1, 0
2	2, 0	1, 0
3	4, 1	3, 1
4	4, 0	2, 1

2	~~32/4~~		
3	X	X	
4	X	X	X
	1	2	3

can't be reduced further.

Ex#4

	0	1
1	2, 0	1, 1
2	4, 0	6, 0
3	6, 1	1, 1
4	3, 1	2, 0
5	5, 0	4, 0
6	8, 1	7, 0
7	5, 0	8, 1
8	7, 0	3, 0

2	X						
3	X	X					
4	X	X	X				
5	X	~~45/64~~	X	X			
6	X	X	X	35/27	X		
7	~~35/18~~	X	X	X	X	X	
8	X	47/63	X	X	57/43	X	X
	1	2	3	4	5	6	7

28, 46, 58

4 △ 6

2 △ 8 △ 5

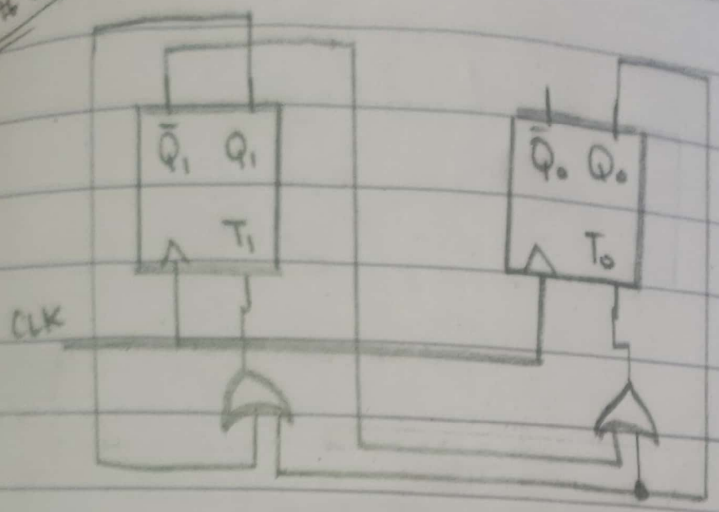
	0	1
1	2, 0	1, 1
2	4, 0	6, 0
3	6, 1	1, 1
4	3, 1	2, 0
7	5, 0	8, 1

CamScanner

Analysis of Sequential circuit

↳ Reverse Engineering → (Solution → prob.)

Ex # 01



$T_1 = Q_1 + Q_0$

$T_0 = \bar{Q}_1 + Q_0$

$$Q_{t+1} = T\bar{Q}_t + \bar{T}Q_t$$

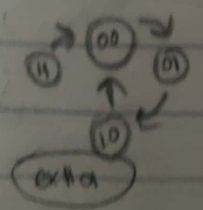
For Q_{0t+1} :

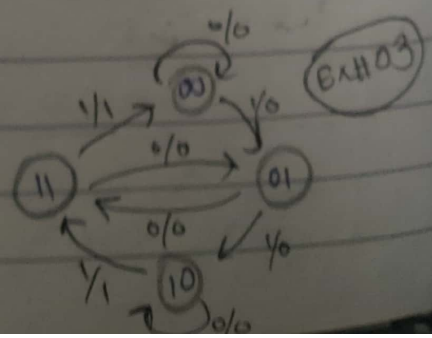
$$Q_{0t+1} = T_0\bar{Q}_{0t} + \bar{T}_0 Q_0$$
$$= (\bar{Q}_1 + Q_0)\bar{Q}_0 + \overline{(\bar{Q}_1 + Q_0)}Q_0$$
$$= \bar{Q}_1\bar{Q}_0 + 0 + \bar{Q}_1\bar{Q}_0(Q_0)$$
$$= \bar{Q}_1\bar{Q}_0 + 0$$
$$= \bar{Q}_0\bar{Q}_1$$

For Q_{1t+1} :

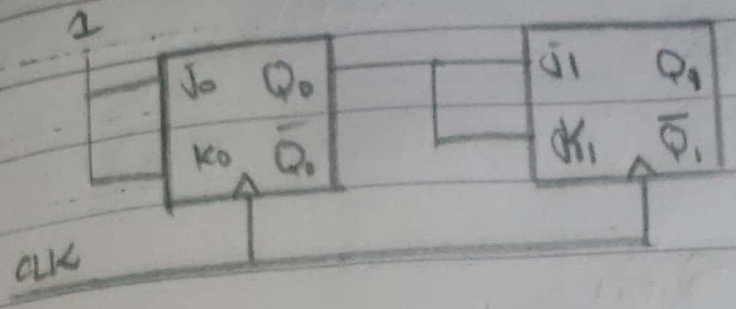
$$Q_{1t+1} = T_1\bar{Q}_{1t} + \bar{T}_1 Q_{1t}$$
$$= (Q_1 + Q_0)\bar{Q}_1 + \overline{(Q_1 + Q_0)}Q_1$$
$$= Q_1\bar{Q}_1 + Q_0\bar{Q}_1 + (\bar{Q}_1\bar{Q}_0)Q_1$$
$$= 0 + Q_0\bar{Q}_1 + 0$$
$$= Q_0\bar{Q}_1$$

Q_1Q_0	Q_1	Q_0
00	0	1
01	1	0
10	0	0
11	0	0





Ex # 02



$J_0 = K_0 = 1$

$J_1 = K_1 = Q_0$

$$Q_{t+1} = J\bar{Q}_t + \bar{K}Q_t$$

For Q_{0t+1} :

$Q_{0t+1} = J_0\bar{Q}_0 + \bar{K}Q_0$

$= (1)\bar{Q}_0 + (0)Q_0$

$\boxed{Q_{0t+1} = \bar{Q}_0}$

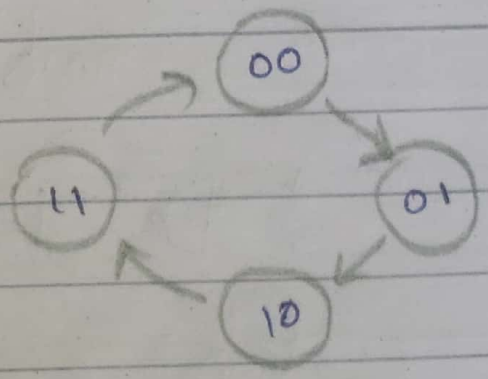
For Q_{1t+1} :

$Q_{1t+1} = J_1\bar{Q}_1 + \bar{K}_1Q_1$

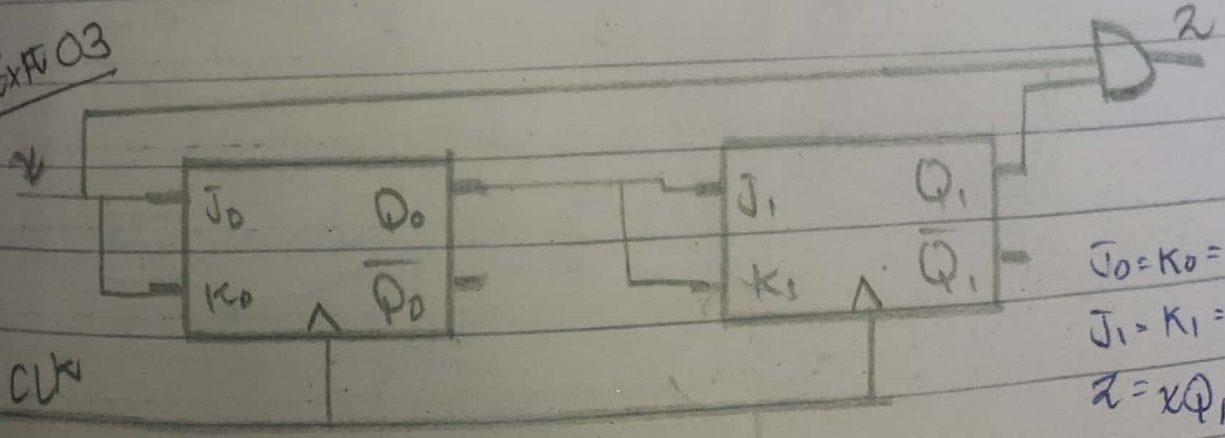
$= Q_0\bar{Q}_1 + \bar{Q}_0Q_1$

$\boxed{Q_{1t+1} = Q_0 \oplus Q_1}$

Q_1Q_0	Q_{1t+1}	Q_{0t+1}
00	0	1
01	1	0
10	1	1
11	0	0



Ex # 03



$J_0 = K_0 = x$

$J_1 = K_1 = Q_0$

$z = xQ_1$

$$Q_{t+1} = J\bar{Q}_t + \bar{K}Q_t$$

A	B	Z
0	0	1
1	1	1
1	1	0
1	0	1

For $Q_{1\,t+1}$:

$$Q_{t+1} = \bar{Q}_1 \bar{J}_1 + \bar{K}_1 Q_1$$
$$= \bar{Q}_1 Q_0 + \bar{Q}_0 Q_1$$
$$Q_{t+1} = Q_1 \oplus Q_0$$

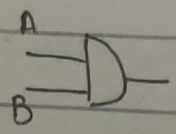
For $Q_{0\,t+1}$:

$$Q_{0\,t+1} = \bar{Q}_0 \bar{J}_0 + \bar{K}_0 Q_0$$
$$= \bar{Q}_0 x + \bar{x} Q_0$$
$$Q_{0\,t+1} = x \oplus Q_0$$

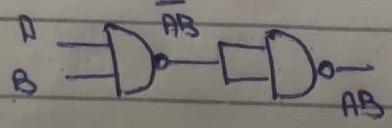
$Q_1 Q_0$	x = 0: $Q_1 Q_0$	x = 1: $Q_1 Q_0$
0 0	0 0, 0	0 1, 0
1 0	1 1, 0	1 0, 0
1 0	1 0, 0	1 1, 1
1 1	0 1, 0	0 0, 1

NAND / NOR Implementation :-

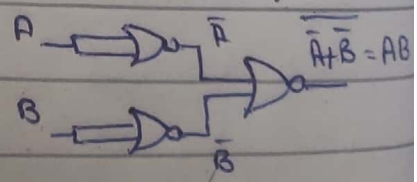
AND



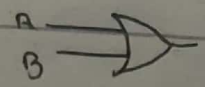
NAND

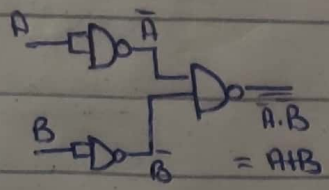


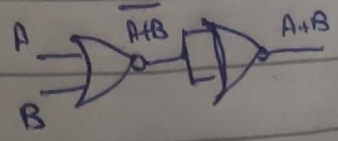
NOR



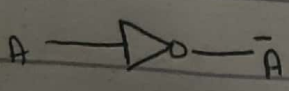
OR

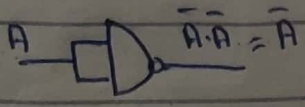


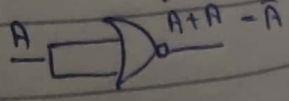




NOT :



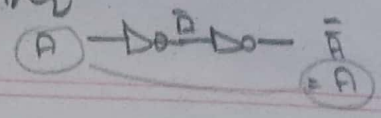




we don't count not gate as lvl bcz the i/p is stored in FFp which are bistable

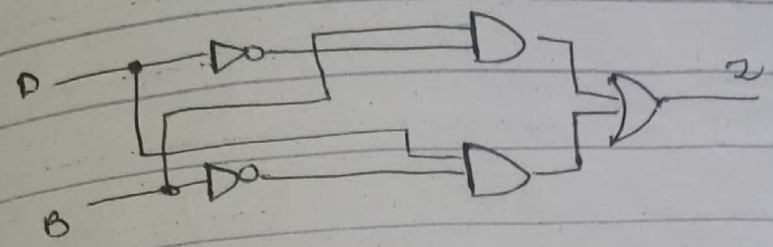
Q $\bar{Q}$

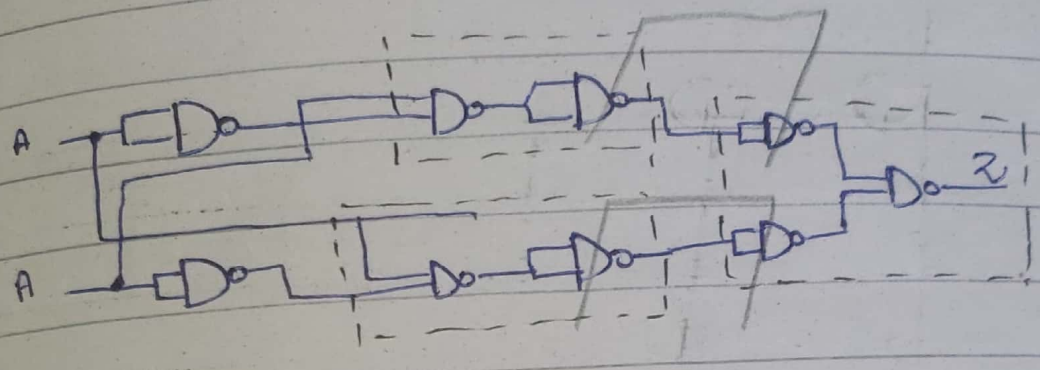
Inverters in series cancle out each other

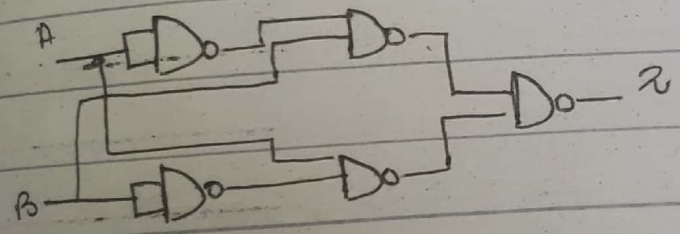


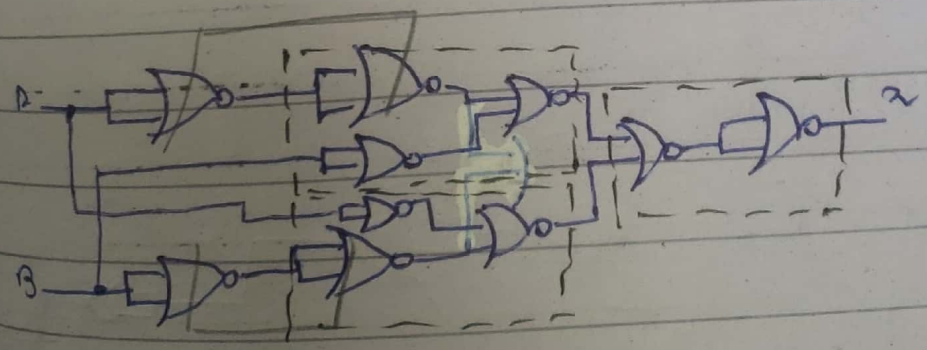
Eg

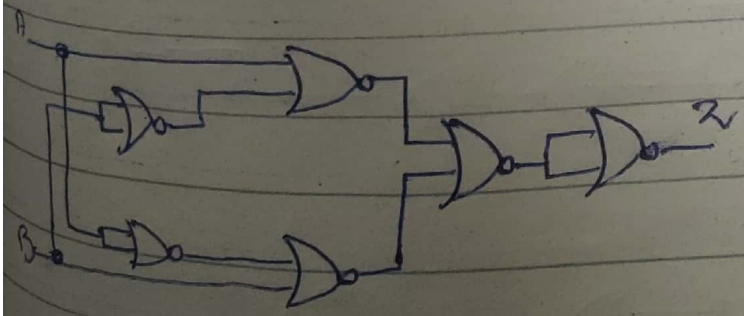
(1)



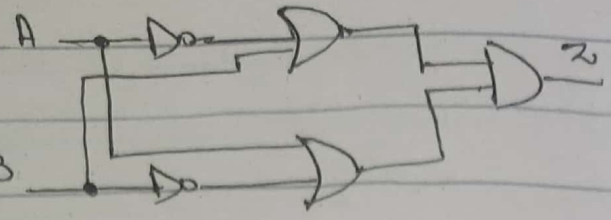




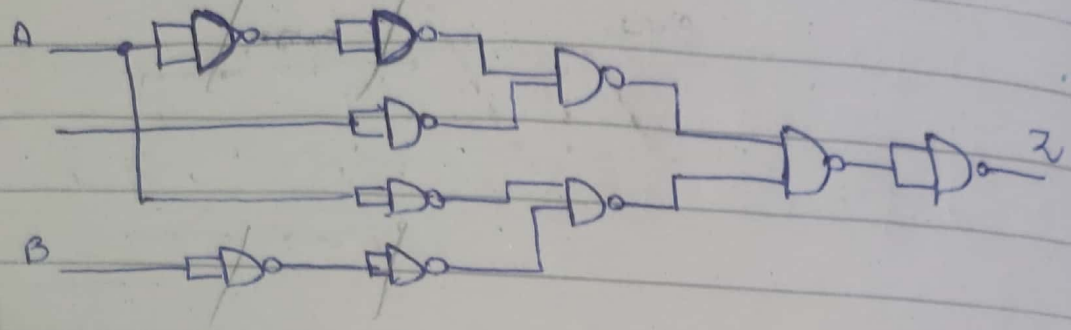




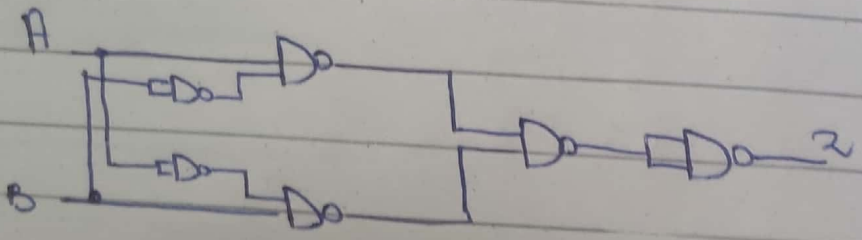
(2)



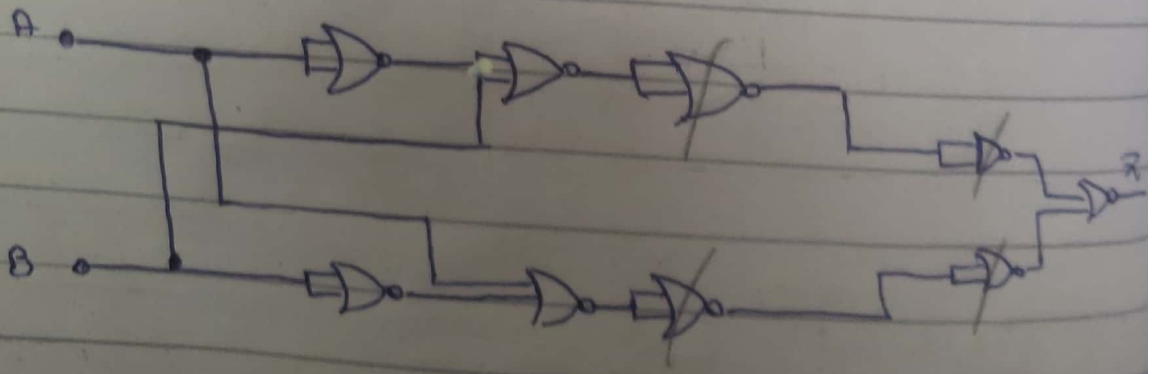
NAND Implementation



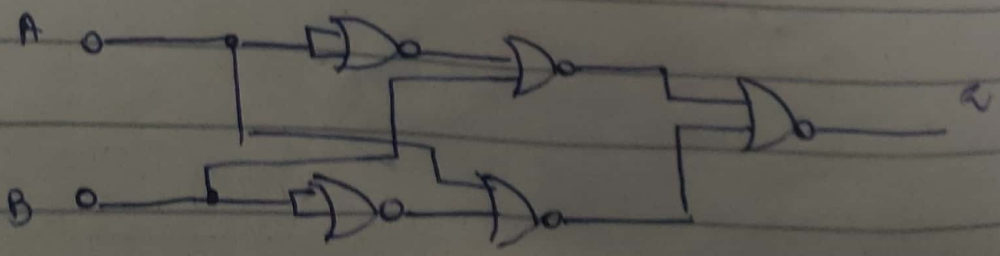
OR



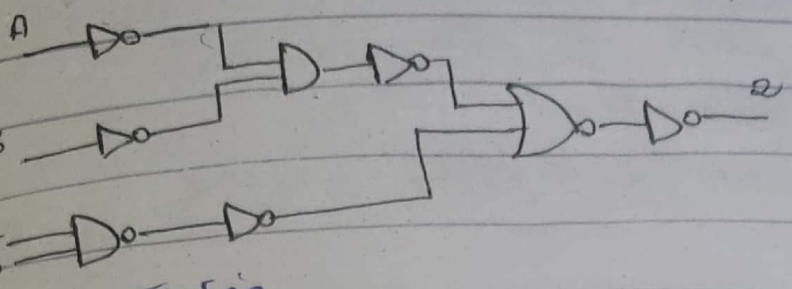
NOR Implementation



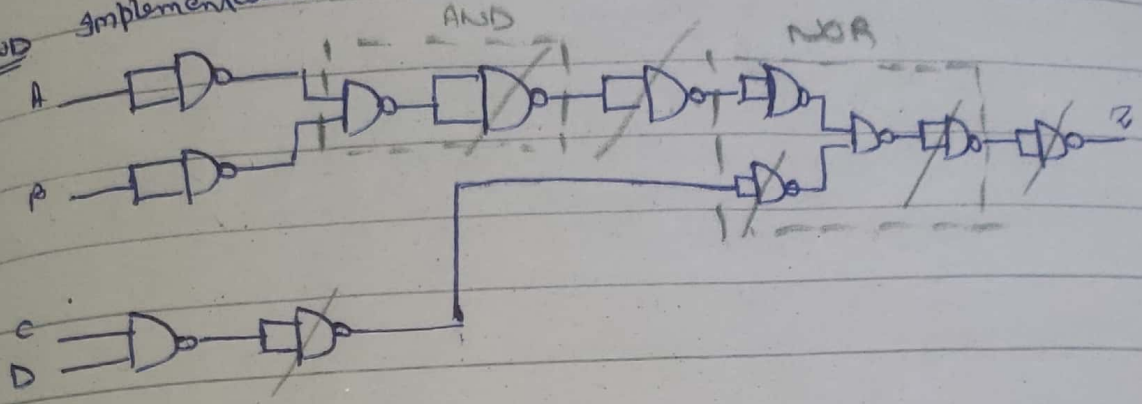
OR



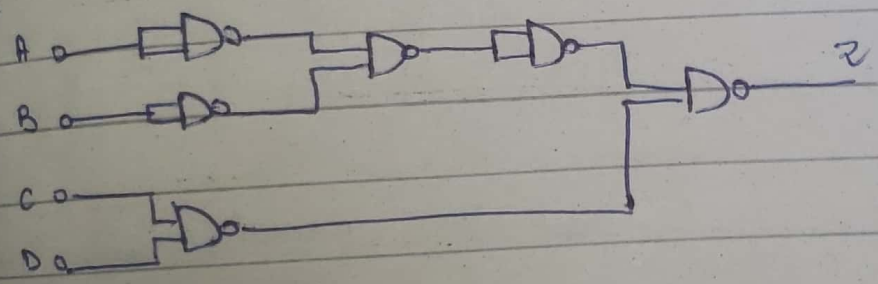
(3)



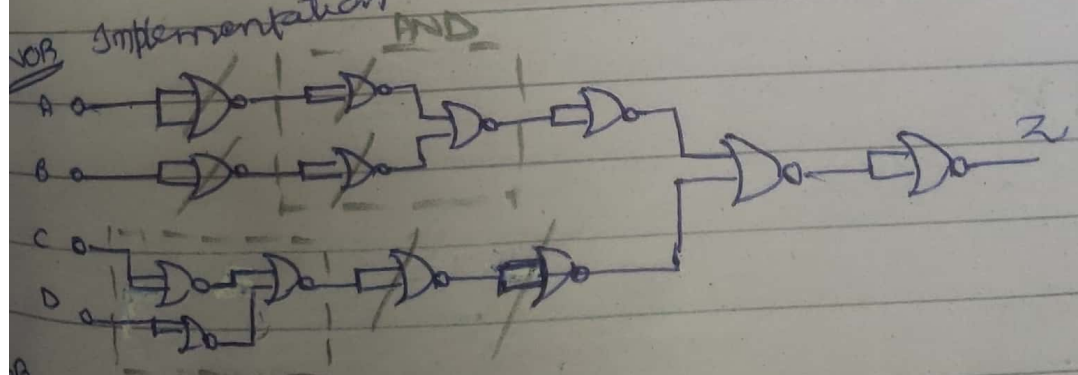
NAND Implementation



OR

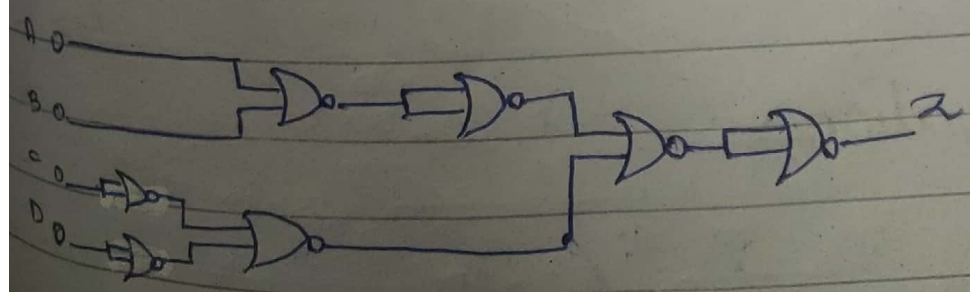


NOR Implementation

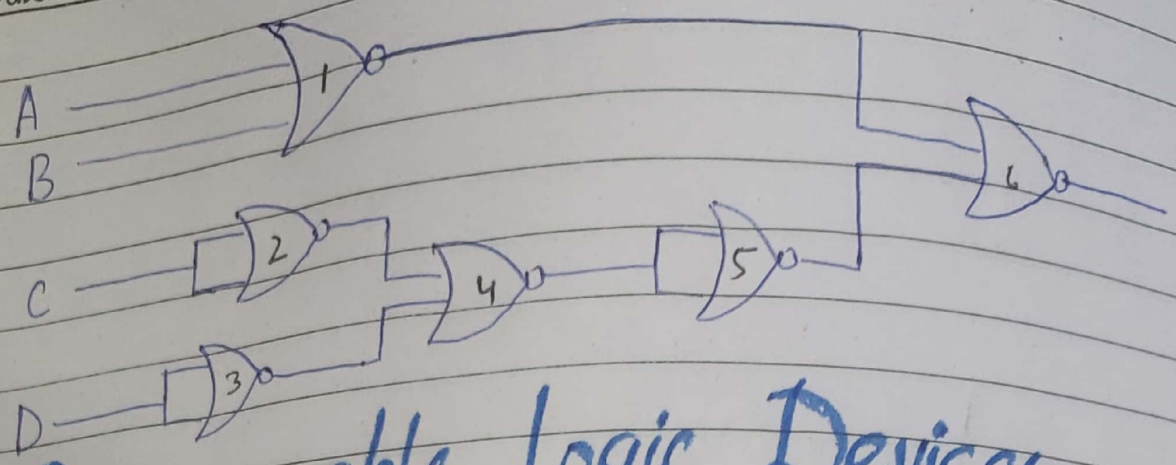


OR

NAND



Dated: 19-11-2024



Programmable Logic Devices

Three approaches to logic design.

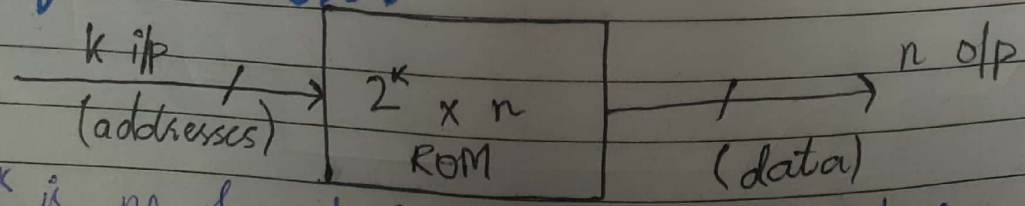
1- Using standard ICs.
2- ASICs (Application Specific Integrated Circuits)
3- PLDs (Programmable Logic Devices)

⇒ user configurable
⇒ capable of implementing logic function.
⇒ contains a regular structure and allows user to customize it.
⇒ PLDs are ICs with programmable AND array & OR array.
⇒ 3 types:
PROMs: fixed AND array, programmable OR array
PALs: programmable AND array, fixed OR array.
PLAs: programmable AND array, programmable OR array.

ROM: Types:

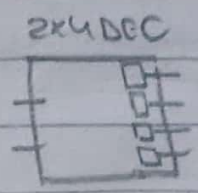
MASK ROM, PROM, EPROM, EEPROM

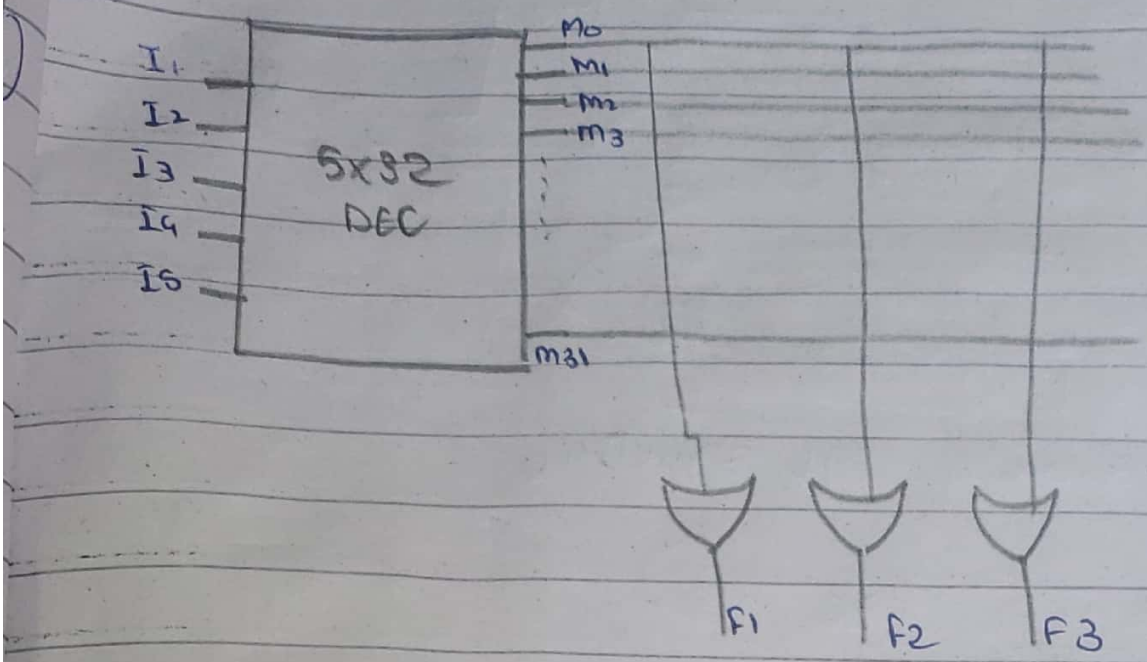
Organization:



2^k is no. of words in a ROM. n is word size.

Eg: 32 x 8 PROM :

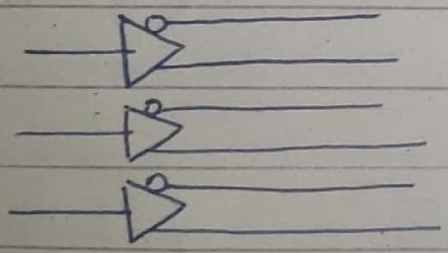




Ex # 7.1 (Pg # 319) For Practice

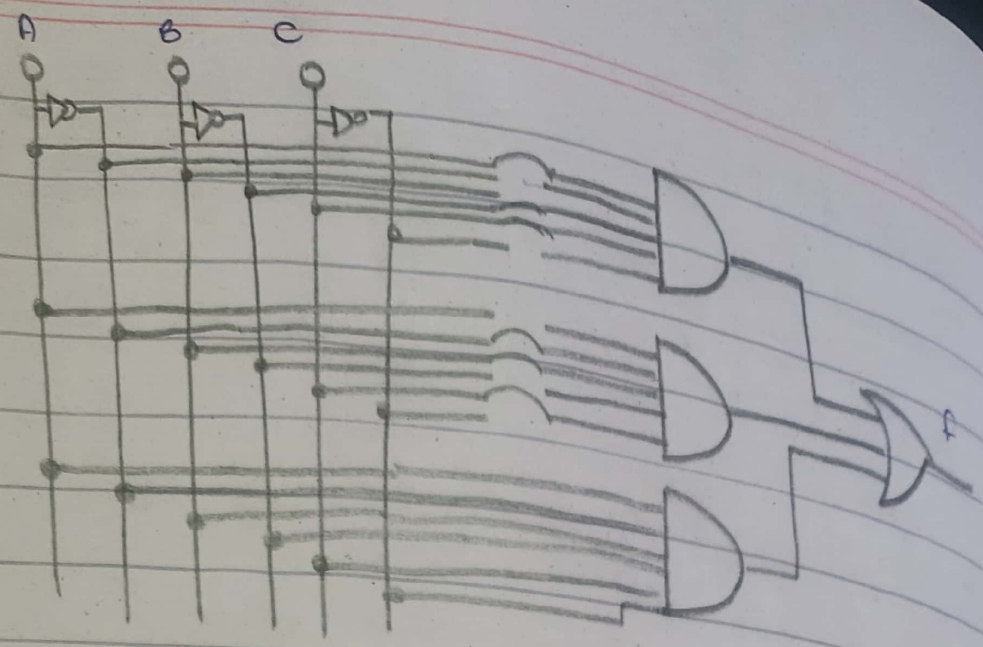
give The logic implementation of 32 x 4 bit ROM

Programmable Array Logic (PAL)



Ex,

$$f = A\bar{B}C + \bar{A}BC$$



3 i/p , 3 wide PAL structure

$f_1 = AB' + AC + A'BC'$

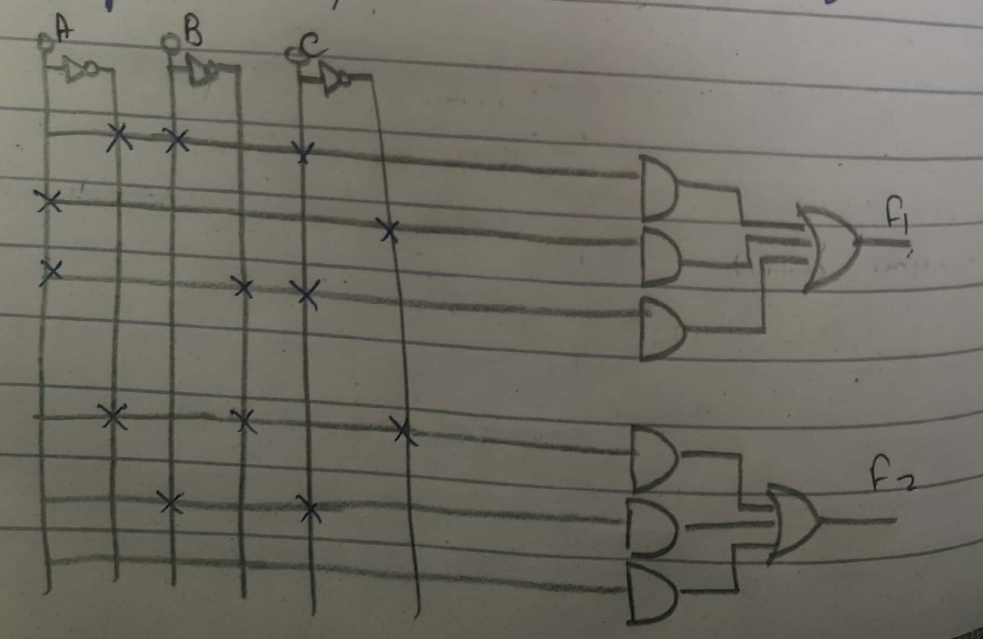
$f_2 = (AC + BC)'$

PAL : Implement :

$f_1 = \bar{A}BC + A\bar{C} + A\bar{B}C$

$f_2 = \bar{A}\bar{B}\bar{C} + BC$

using a 3 i/p , 3 wide dual PAL



Practice

1) Ex # 7.2 pg # 324

2) Use a quad 4 i/p 3 wide PAL

$f_1(A,B,C,D) = \Sigma(2,12,13)$

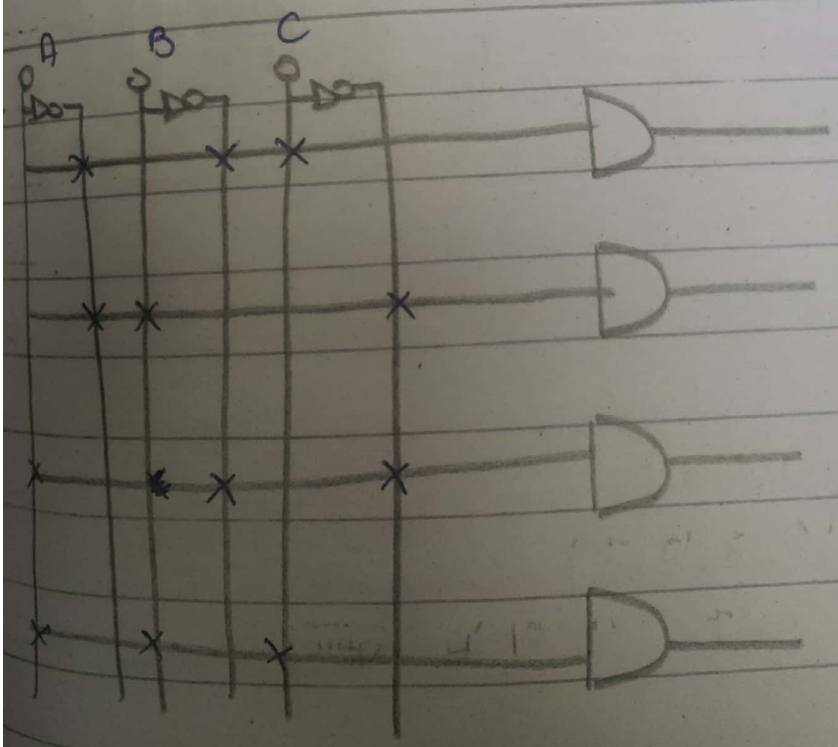
$f_2(A,B,C,D) = \Sigma(7,8,9,10,11,12,13,14,15)$

$f_3(A,B,C,D) = \Sigma(0,2,3,4,5,6,7,8,10,11,15)$

Programmable logic Array: (PLA)

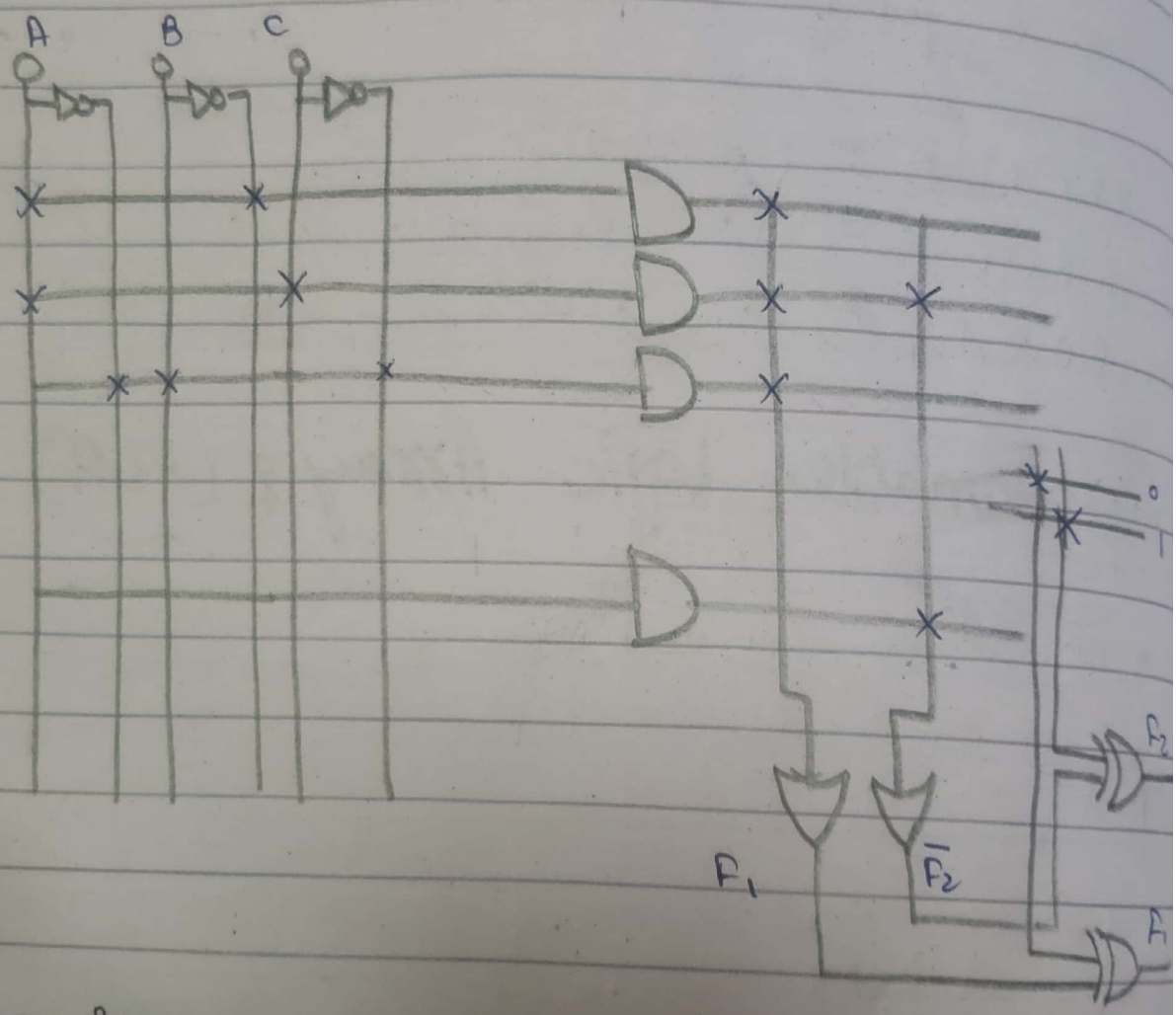
$f_1 = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC = S$

$f_2 = AB + AC + BC = C_{out}$



$F_1 = AB' + AC + A'BC'$

$F_2 = (AC + BC)'$



Practice

Implement

$F_1(A,B,C) = \Sigma(2,3,4,6)$

$F_2(A,B,C) = \Sigma(0,3,3,7)$

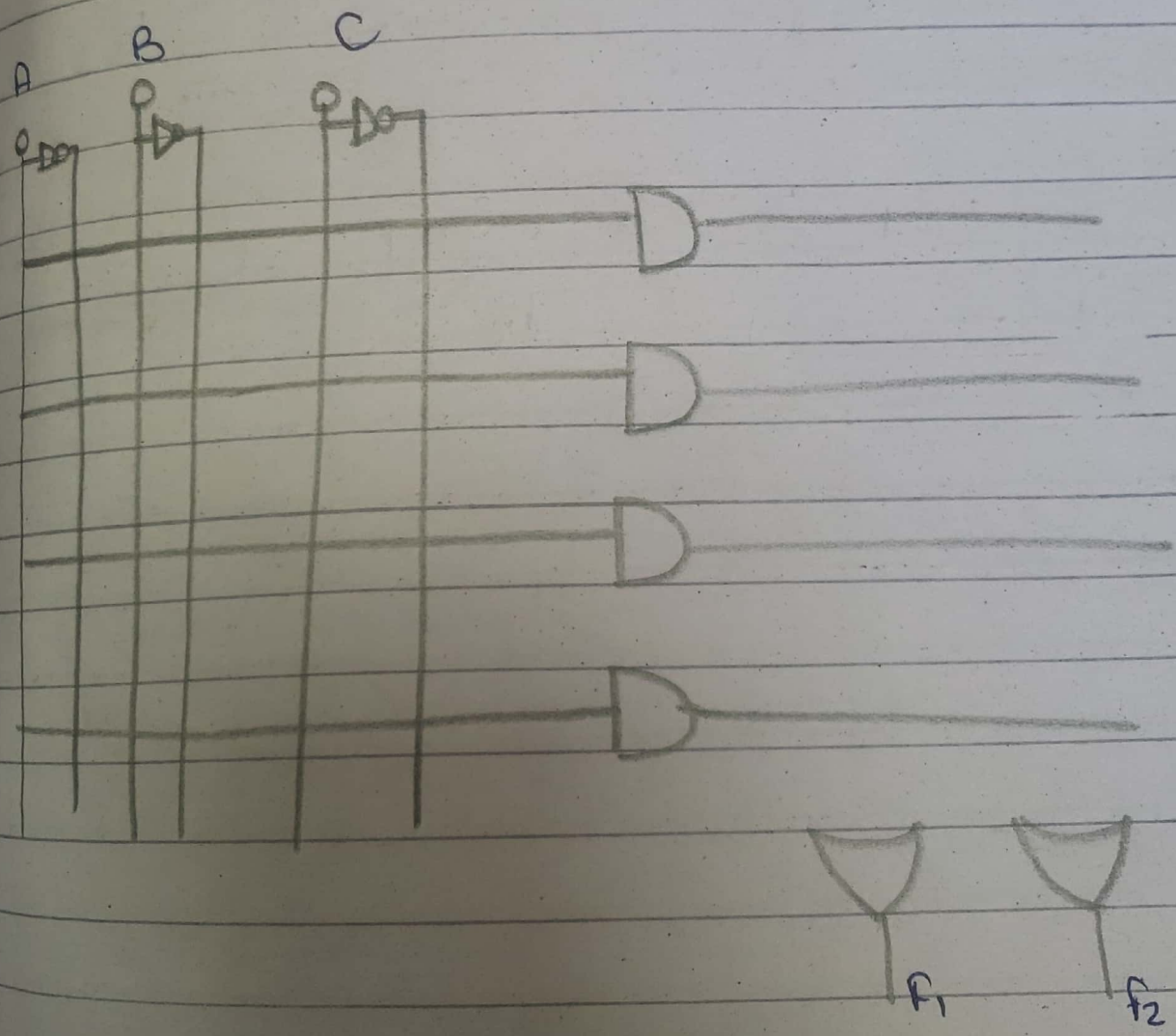
Using a PLA with 4 AND GATES and 2 OR Gates

f_1 A \ BC	00	01	11	10
0			1	1
1	1			1

f_2 A \ BC	00	01	11	10
0	1		1	
1		1	1	

$f_1 = \bar{A}B + A\bar{C}$

$f_2 = \bar{A}\bar{B}\bar{C} + B\bar{C} + AC$

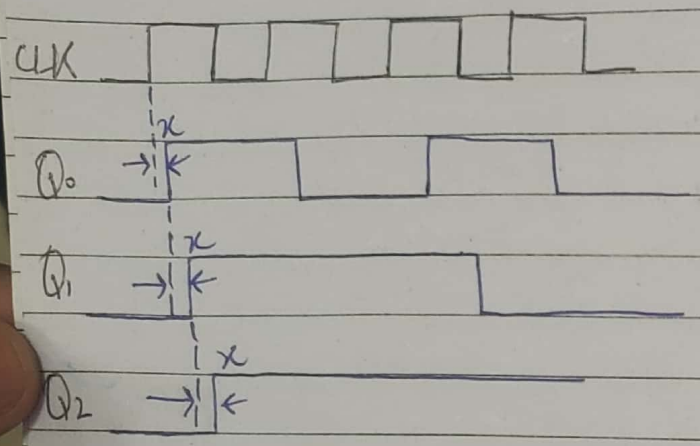


Dated: 26-11-2024

Comparing Synchronous and Asynchronous Counters

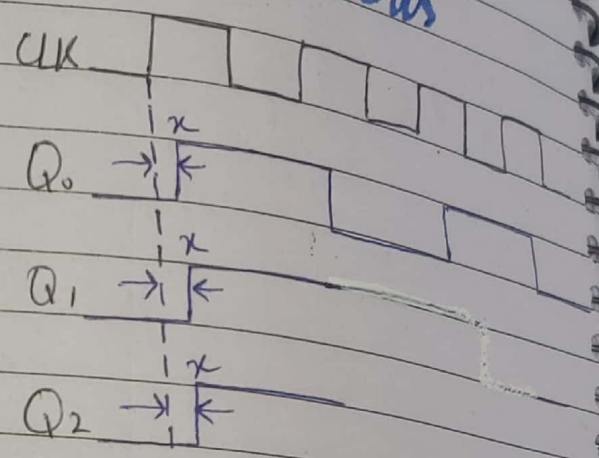
↳ Asynchronous are enable to design
↳ Asynchronous only can take few sequences (up/down)
↳ Asynchronous propagation delay ripples through.

Asynchronous



n: # of ffps
propagation delay of 1 ffp: x ns
propagation delay of counter: nx ns

Synchronous



propagation delay of the counter: x ns

Practical Consideration in logic circuits

⇒ propagation delay: time taken to respond to i/ps.

⇒ fan out: # of standard loads that the o/p of a gate can drive without impairing its normal operation.

Logic families

BJTs: fast, speed crucial application technologies.
⇒ TTL: Transistor - Transistor Logic
ECL: Emitter - coupled Logic (fastest)
I^2L: Integrated Injection Logic (small)

Dated: 26-11-2024

⇒ FETs: Low power consumption small (5:1 BJTs) technologies
↳ CMOS ↳ PMOS ↳ NMOS

TTL:

→ high 5V
low 0V
⇒ 74/54 series
⇒ float high
⇒ best delay: 1.7 ns (AS)
worst delay: 33 ns (L)
⇒ best power dissipation: 1 mW (L)
worst power dissipation: 22 mW (H)
74ALS04

CMOS:

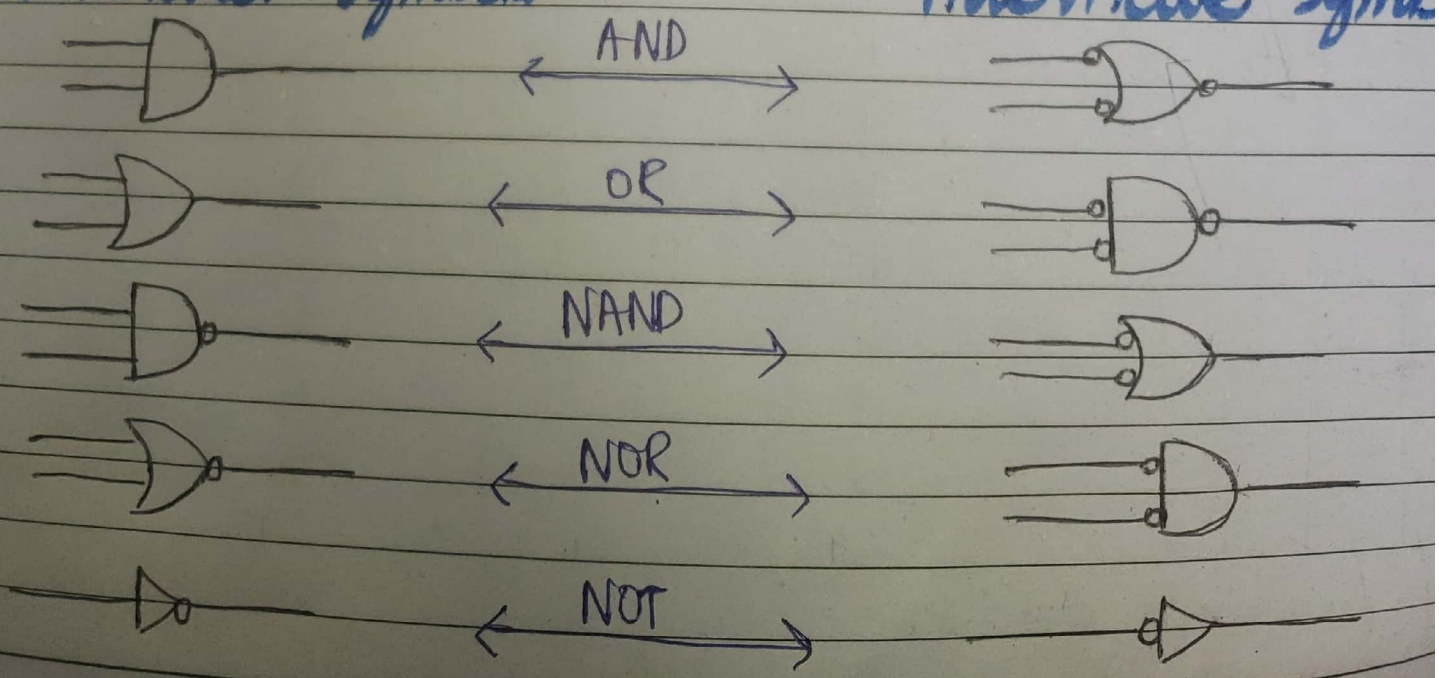
⇒ high 3V to 12V
low 0V
⇒ 40/54 series
⇒ smaller than TTL
⇒ improved noise immunity
⇒ less power consumption
best 0.1 mW
worst 0.17 mW

Speed power product:

measure of performance of an IC, FOM

Active low symbols

Standard symbols | Alternate symb



Dated: 28-11-2024

Digital Codes

Error Detection Codes

Parity codes:

⇒ even parity: concatenate a bit that makes total # of 1's even.

⇒ odd parity: concatenate a bit that makes total # of 1's odd.

* A given system operates with even or odd parity bit not both.
* Odd parity is prefessed over even parity.

BCD				Even parity bit	Odd parity bits
A	B	C	D		
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	1

Parity even = $A \oplus B \oplus C \oplus D$

Parity check even = $P_{even} \oplus A \oplus B \oplus C \oplus D$

Dated: 28-11-2024

$P_{odd} = \overline{A \oplus B \oplus C \oplus D} = A \odot B \odot C \odot D$

Parity Check odd $= \overline{P_{odd} \oplus A \oplus B \oplus C \oplus D}$

Block parity:

↳ When several words are transmitted in succession, parity bits can be assigned to both rows and columns.

↳ This makes single error correction possible.

CamScanner