DD

Analog - Continuous
Digital - Discrete

Combination α Present Input
Sequential α Previous Inputs

① Number Systems

ⓐ Decimal 8756 0-9 Base 10
Radix 10

ⓑ Binary 10101 0-2 Base 2
Radix 2

ⓒ Binary to Decimal

2	49	1
2	24	0
2	12	0
2	6	0
2	3	1
	1	1

110001

0.125 to Binary

0.125 × 2	0.25	0
0.25 × 2	0.5	0
0.5 × 2	1	1

0.001

ⓓ Binary to Octal - Take sets of 3 digits
ⓔ Octal to Binary - Convert every digit to binary of 3 digits
ⓕ Signed Magnitude 00001 1
10001 -1
↳ Sign bit

ⓖ Diminished Radix Complement
Base r → r-1's Complement = $r^n - 1 - N$
(n digits)

ⓗ Radix Complement
Base r → r's Complement = $r^n - N$

ⓘ Addition & Subtraction in 1's Complement
4 + (-7) = 0100 + 1000 = 1100 = -3. When we have carry add it to LSB and we take it as +ve
(1's Complement)

In 2's Complement, we ignore the carry

① BCD

Each digit has 4 digits of binary. If the sum crosses 9, we add 6 to get proper BCD

② Gray Code

	Binary	Gray
0	0000	0000

Gray to Binary	Binary to Gray
0111	0101
↓↓↓↓	↓↓↓↓
0101	0111
XOR with current digit	XOR each digit

② Boolean Algebra

ⓐ
- $x + yz = (x+y)(x+z)$
- $x(y+z) = xy + xz$
- $x + xy = x$
- $(xy)' = x' + y'$
- $(x+y)' = x'y'$

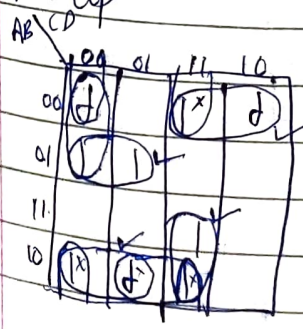
ⓑ

SOP – Min terms			POS – Min terms		
00	$a'b'$	m_0	00	$a+b$	M_0
01	$a'b$	m_1	01	$a+b'$	M_1
10	ab'	m_2	10	$a'+b$	M_2
11	ab	m_3	11	$a'+b'$	M_3

We represent function with $\Sigma m(3,5,6,7)$ or $\Pi M(0,1,2,3)$

To convert minterm to maxterm, all you have to do is take the nums not in minterm

③ KMap



We get $F = A'B'C + ACD + A'BC' + AB'C'$

Both don't care also here

④ QM Method

$F = \pi M(1,3,5,7,9,14,15) + d(6,12,13)$ Since POS → Group based on 0s

1	00001 ✓	6	00110 ✓
3	00011 ✓	12	01100 ✓
5	00101 ✓	13	01101 ✓
7	00111 ✓		
9	01001 ✓		
14	01110 ✓		
15	01111 ✓		

All these are PIs

	Col 1	Col 2		Col 3	
①	01111 15	15,7	0-111 ✓	3,1	000-1 ✓
②	00111 7	15,14	0111- ✓	5,1	00-01 ✓
	01110 14	15,13	011-1 ✓	9,1	0-001 ✓
	01101 13	7,3	00-11 ✓		
③	00011 3	7,5	001-1 ✓		
	00101 5	7,6	0011- ✓		
	01001 9	14,6	0-110 ✓		
	00110 6	14,12	011-0 ✓		
	01100 12	13,9	01-01 ✓		
④	00001 1	13,12	0110- ✓		
		13,5	0-101 ✓		

Col 3	
15,7,14,6	0-11-
15,14,13,12	011--
15,13,7,5	0-1-1
7,3,5,1	00--1
1,5,13,9	0--01

Dominant Rows are considered
Dominant Cols are deleted

Choose any one

	1	3	5	7	9	14	15	6	12	13
15 7 14 6				X		X	X			
15 14 13 12						X	X	X		
15 13 7 5			X	X		X	X		X	X
00--1 ✓ 7 3 5 1		X	X	X						
✓ 1 5 13 9	X		X		X					

→ a+b+e′ → 0--01 a+d+e′

(5) Combinational Circuits

(a) Parity Generator $A \oplus B \oplus C \oplus D$ Even

$\overline{A \oplus B \oplus C \oplus D}$ Odd

(b) Adders

(i) Half Adder

X	Y	S	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = X \oplus Y$

$C_{out} = X \cdot Y \rightarrow$ For Half Subtractor $\rightarrow X'Y$

(ii) Full Adder

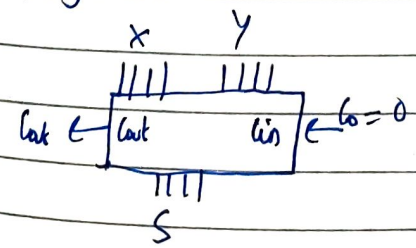
X	Y	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = X \oplus Y \oplus C_{in}$

$C_{out} = X(Y + C_{in}) + YC_{in}$

$= C(x \oplus y) + xy$

(iii) 4 bit Binary Adder



For Subtraction

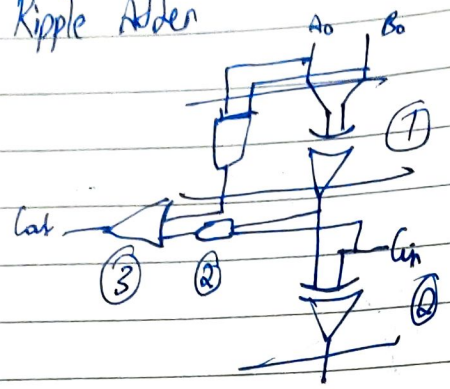
$X - Y = X +$ 2's Complement of Y

$= X +$ 1's Complement of $Y + 1$

$= X + Y' + 1$

(iv) BCD Adder, Check 9 and Add 6

(v) Ripple Adder



$S_0 \rightarrow 2$ gates
$S_1 \rightarrow 4$ gates
$C_{out} \rightarrow 5$ gates

$2n+1 \rightarrow$ CLAA

(vi) Fast Adder

$C_0 =$ Input Carry
$C_1 = G_0 + P_0 C_0$
$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0$

$P_i = A_i \oplus B_i$
$G_i = A_i \cdot B_i$

(vii) Multiplier

$$\begin{array}{ccc} & A_1 & A_0 \\ & B_1 & B_0 \\ \hline A_1B_1 & A_1B_0+A_0B_1 & A_0B_0 \end{array}$$

(viii) Magnitude Comparator

A_0	B_0	G_0	L_0	E_0
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

$G_0 = A_0 B_0'$
$L_0 = A_0' B_0$
$E_0 = (A_0 \oplus B_0)' = (G_0 + L_0)'$

(ix) For overflow detection we use XOR gate

(x) 3 bit Comparator

$G = G_2 + E_2 G_1 + E_2 E_1 G_0$
$L = L_2 + E_2 L_1 + E_2 E_1 L_0$
$E = E_2 E_1 E_0$

6) Combinational Devices

a) Decoder

X	Y	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	0	0
1	1	0	0	0	1

$D_0 = X'Y'$

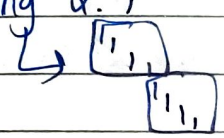
$D_1 = X'Y$

$D_2 = XY'$

$D_3 = XY$

Decoder gives you min terms

3:8 Using 2:4



$F = \Sigma m(1,2,4,7) = X'Y'Z + X'YZ' + XY'Z' + XYZ$

Use 3:8 Decoder

b) Encoder

D_0	D_1	D_2	D_3	X	Y
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

But if D_1 & D_2 are 1, then both X & Y will be 1 resulting in 1,1

$X = D_2 + D_3$

$Y = D_1 + D_3$

So, we use Priority Encoder

$X = D_2 + D_3$

$Y = D_1 D_2' + D_3$

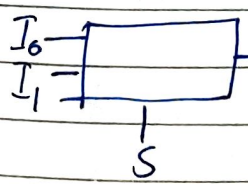
$V = D_0 + D_1 + D_2 + D_3$

Limitations:

- if $D_1 D_2$ are 1 → 1
- if all are zero → 00 → Indicate D_0

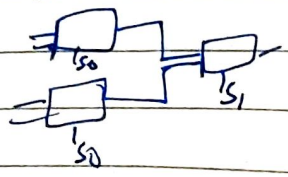
c) Multiplexes

2:1



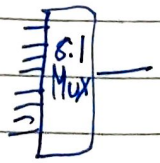
output $= I_0 S' + I_1 S$

4:1 using 2:1 → Use one of the Inputs as selector



S_1	S_0	O/P
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

8:1 Mux

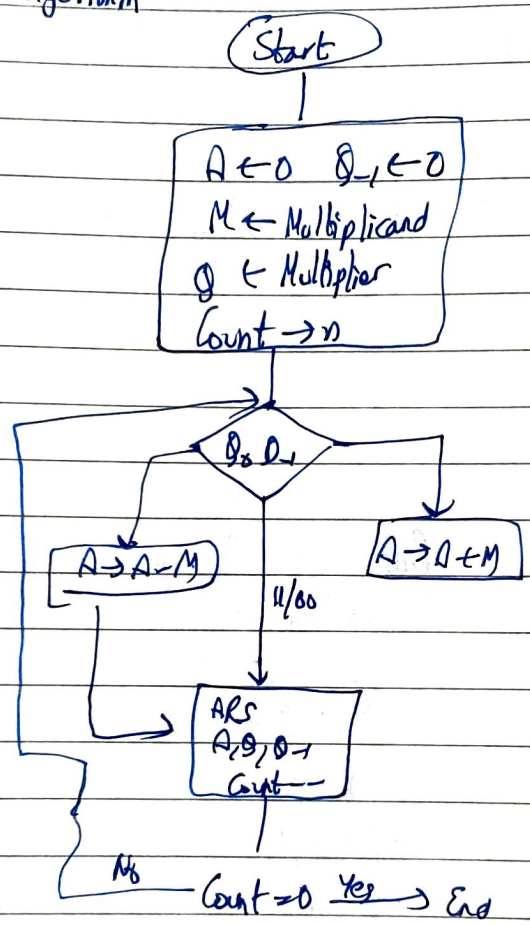


Hazards

Static 0 Hazard : Occurs when output is 0 and should remain at 0, but temporarily switches to a 1 due to a change in an input

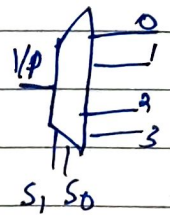
Static 1 Hazard : 1 instead of 0

Booth's Algorithm

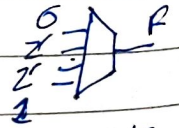


- ARS
  - Retain MSB and push others
  - Always have one bit extra
  - In the end if MSB is 1 we take 2's Complement for -ve

Demultiplexer



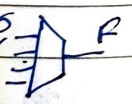
I/P	S_1	S_0	D_0	D_1	D_2	D_3
0	x	x	x	x	x	x
1	0	0	1	x	x	x
1	0	1	x	1	x	x
1	1	0	x	x	1	x
1	1	1	x	x	x	1



$F = x'y'I_0 + x'yI_1 + xy'I_2 + xyI_3$

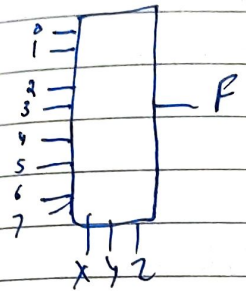
- 0 fact temporarily
input

push other
it extra
SB is 1
format for -ve



$= x'y'I_0 + x'yI_1 + xy'I_2 + xyI_3$

Mux based Implementations



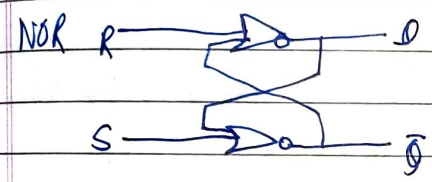
	$\bar{x}\bar{y}$	$\bar{x}y$	$x\bar{y}$	xy
z	1 (000)	3 (010)	5 (100)	7 (011)
$\bar{z}$	0 (001)	2 (110)	4	6

4:1 Mux

Solve and get Small Eqns

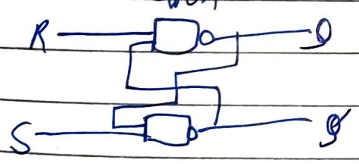
7 Sequential Circuits

SR latch



S	R	Q	Q̄
0	0	Same	
0	1	0	1
1	0	1	0
1	1	Invalid	

S̄R̄ latch



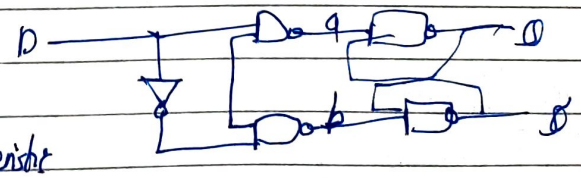
S	R	Q	Q̄
0	0	Not Allowed	
0	1	0	1
1	0	1	0
1	1	Same	

We use the concept of dominant inputs for example NAND gate Even if one of them is 0, we get 1 as answer

D latch

D	Q	Q'
0	0	1
1	1	0

← Characteristic Table



JK latch (Made to overcome 11 State)

JK →	J	K	Q(t+1)
	0	0	Q(t)
	0	1	0
	1	0	1
	1	1	Q(t)'

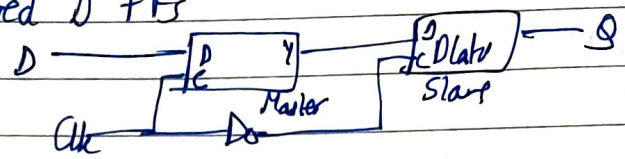
Problem with latch – They are transparent i.e. any change on inputs is seen on outputs immediately. This causes sync problems

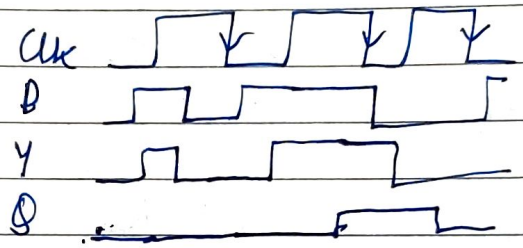
Soln: We use latches to create flipflops

Latches – Level Sensitive

Flipflops – Edge Sensitive – Made using
- Special Gate based Design
- Master Slave or Special Config of two latches

Ex: Edge Triggered D FFs





This is –ve edge FF

→ (a) D Flip Flop

CT:

D	Q(t+1)
0	0
1	1

ET:

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

$Q(t+1) = D$

→ (b) JK Flip Flop

CT:

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q(t)'

$Q(t+1) = JQ' + K'Q$

ET:

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

→ ⑦ T Flip Flop

$Q(t+1) = T'Q + TQ'$

T	$Q(t+1)$
0	$Q(t)$
1	$Q'(t)$

Flip Flop Conversions: Equate $Q(t+1)$ (s)

(i) $D = JQ' + K'Q$
For JK using D

(ii) T using D
$TQ(t)' + T'(Q(t)) = D$

(iii) T using JK
$JQ' + K'Q = T \oplus Q$

⑧ Counters

ⓐ Analysis given a circuit
- Derive State Eqn
- State Table
- State Diagram

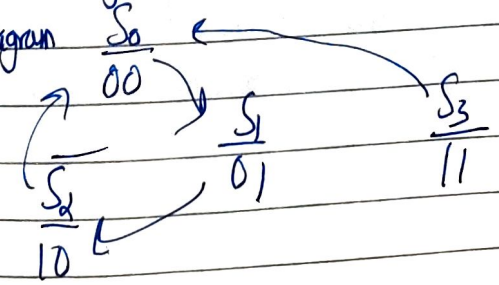
ⓑ Design Given a functionality
- Derive State Diagram
- State table.
- State Equations
- Circuit

Mealy Model
- Output depends on present state

Moore Model
- Output depends only on Present State

ⓐ Mod 3 Counter - Synchronous

ⓐ State Diagram



ⓑ State Table

PS	Q_B	Q_A	NS	$Q_B(t+1)$	$Q_A(t+1)$	D_B	D_A
S_0	0	0	S_1	0	1	0	1
S_1	0	1	S_2	1	0	1	0
S_2	1	0	S_0	0	0	0	0
S_3	1	1	S_0	0	0	0	0

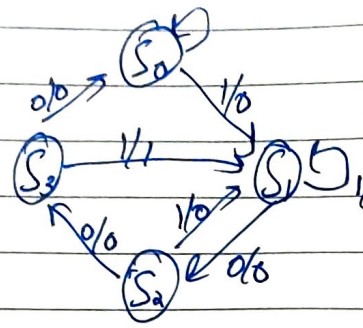
$D_A = Q_B'Q_A'$
$D_B = Q_B'Q_A$

(b) Sequence Detector 1001

Overlapping 1001001

0001001

Non Overlapping 0001000



PS Q_A	Q_B	In X	NS $Q_A(t+1)$	$Q_B(t+1)$	Out Y	D_A	D_B
S_0 0	0	0	S_0 0	0	0	0	0
S_0 0	0	1	S_1 0	1	0	0	1
S_1 0	1	1	S_1 0	1	0	0	1
S_1 0	1	0	S_2 1	0	0	1	0
S_2 1	0	1	S_1 0	1	0	0	1
S_2 1	0	0	S_3 1	1	0	1	1
S_3 1	1	0	S_0 0	0	0	0	0
S_3 1	1	1	S_1 0	1	1	0	1

Minimum
- JK

$D_A = Q_A' Q_B X' + Q_A Q_B' X$

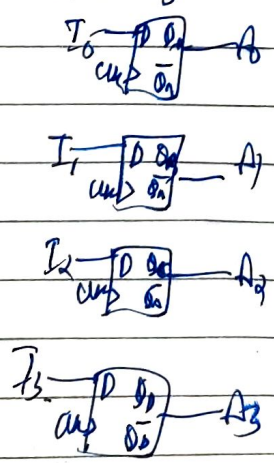
$D_B = X + Q_A Q_B'$

$Y = Q_A Q_B X$

Easy
- D

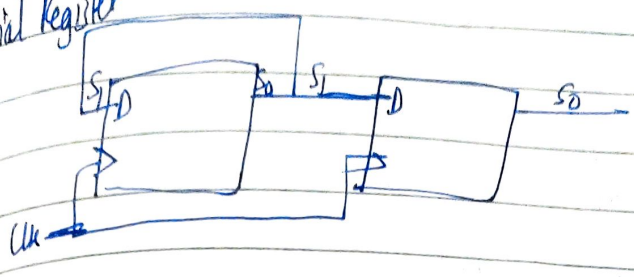
Registers

- Can store multiple bits
- ☆ Parallel Registers



To leave registers unchanged
- Inputs held Constant
- Avoid Clock from reaching register

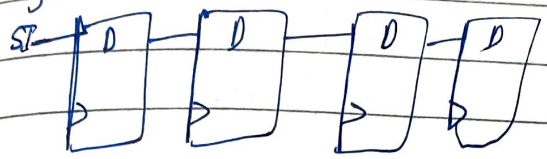
Serial Register



A	B
1011	0010
1101	1001
1110	1100
0111	0110
1011	1011

Will be same

Shift Register

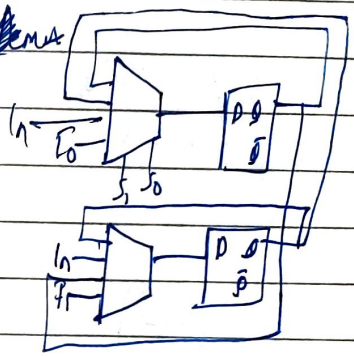


$1 \rightarrow 1010 = 1101$

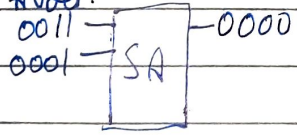
Universal Shift Register

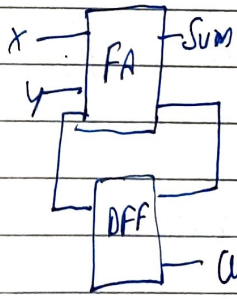
S_1	S_0	Op
0	0	Nothing
0	1	Right Shift
1	0	Left Shift
1	1	Parallel Load

Use ~~Demux~~

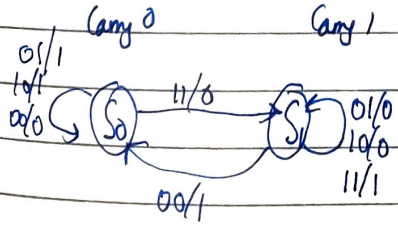


Serial Adder:





A	B	Din	Din(t+1)	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



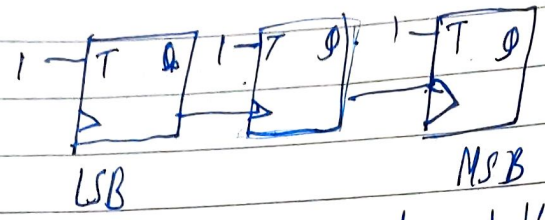
$Din(t+1) = Din(X \oplus Y) + XY$

$Sum = X \oplus Y \oplus Din$

10 Counters with Registers

(a) Async – Clock is not same

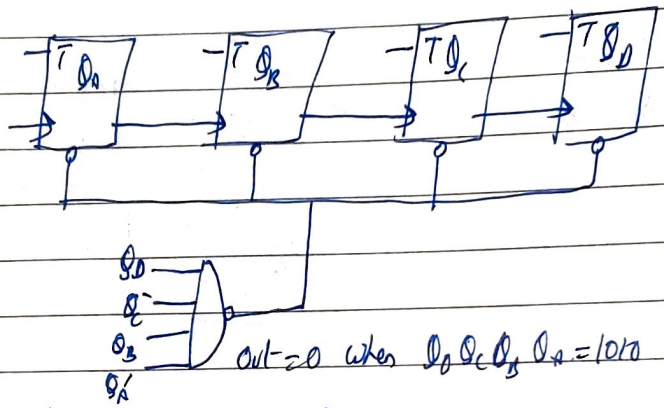
Sync – Clock is same



mod 8 Counter

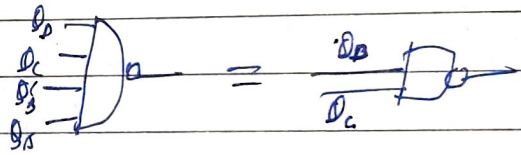
If we connect one more we get mod 16 counter.

If we have to convert a mod 16 counter to mod 10, as soon as 1010 comes, it should be reset

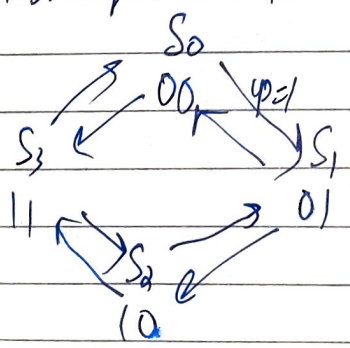


out = 0 when $Q_D Q_C Q_B Q_A = 1010$

Similarly for mod 12 →



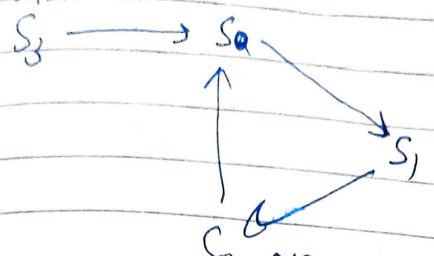
(b) 2 bit up down Counter



	PS		In	NS	
	Q_B	Q_A		Q_B	Q_A
S_0	0	0	1	S_1 0	1
S_1	0	1	1	S_2 1	0
S_2	1	0	1	S_3 1	1
S_3	1	1	1	S_0 0	0
S_0	0	0	0	S_3 1	1
S_3	1	1	0	S_2 1	0
S_2	1	0	0	S_1 0	1
S_1	0	1	0	S_0 0	0

Then we solve

⑥ Counters with unused States



3 bit Only 0 145

010 011 → S_0 ← 110 111

000

S_3 101

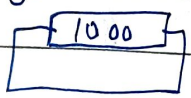
S_1 001

S_2 100

Self Correcting
↳ Goes to an allowed Sequence

⑦ Ring Counter

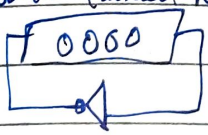
4 bit Shift Register



1000
0100
0010 4 Only
0001

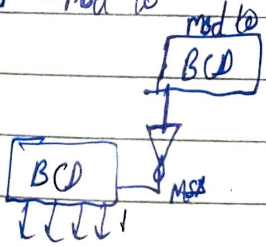
Ring Counter using 2 bit Counter, use Decoder

⑧ Johnson Counter (twisted Ring Counter)

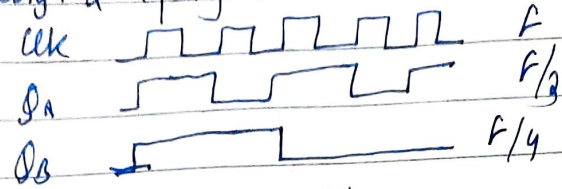


0000 1111
1000 0111
1100 0011
1110 0001

⑨ BCD Counter — mod 10

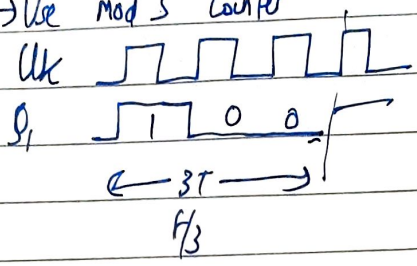


Q Can you design a frequency divider circuit to divide the f by 3



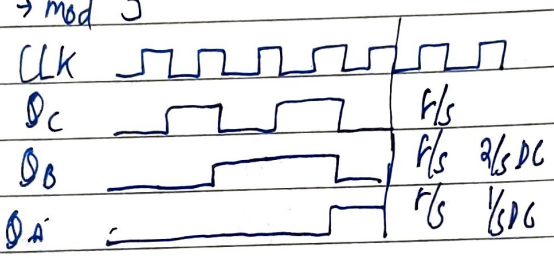
In 1Clk → 2Clk is up

For f/3 → Use mod 3 Counter



$\begin{bmatrix} 00 \\ 01 \\ 10 \\ 00 \end{bmatrix}$

Q f/5 → mod 5



$Q_A Q_B Q_C$
000
001
010
011
100

$\frac{T_{on}}{T_{tot}}$ = Duty Cycle

Up Counter

$J_0 = 1$ $J_1 = Q_0$ $J_2 = Q_0 Q_1$ $J_3 = Q_0 Q_1 Q_2$

$K_0 = 1$ $K_1 = Q_0$ $K_2 = Q_0 Q_1$ $K_3 = Q_0 Q_1 Q_2$

(11) Memory & Programmable logic

Consists of cells that store a large quantity of info

1. Memory Read - Transferring stored info out of memory
2. Memory Write - Storing new info into memory

★ Types of Memory

RAM	ROM
- Read & Write	- Only read
- Stored info is lost	- Info already stored
	- PLD

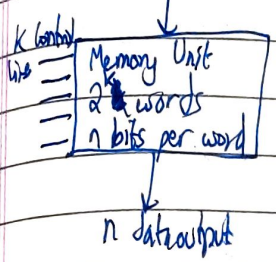
(12) PLDs

① ROM ② PLA ③ PAL

Programming involves blowing fuse along path to obtain desired function

(13) RAM

- Time it takes to transfer info out from any desired location is same — RAM
- In magnetic disks — time is not same for every location
- Stores info in groups of bits called <u>words</u>
  - ↳ entity of bits that works as a unit



$2^k \times n$ bits

For Stale : Ideal → No Read No Write

(14) Contents of a 1024 x 16 RAM

1024 → Words, 16 → Bits p/s

Memory Unit
1k Words 1 word = 16 bits

Access Lines = $\log_2 1024 = 10$ 10 bit Address per word

Capacity = 16 kb or 2 KB

(15) Contents of a 32K x 8 RAM

32K → Words, 8 → Bits per second

= 256 kb RAM

= 32 KB RAM

Address lines = $\log_2 32K = 15$

(16) Memory Operations

En	R/W		Write	Read
0	x	None	- Apply binary Address & Activate Memory	- Apply binary Address & Activate Memory
1	0	Write	- Apply binary data	- Activate Read
1	1	Read	- Activate Write	

- Access Time - Time reqd to select a word & read it
- Cycle Time - Time reqd to complete write operation
  - ↳ CPU has clock - Time Period = 20ns
    Access & Cycle Time → ≤ 50ns
    Minimum Clock Cycles reqd = Atleast 3 Clock Cycles

(iv) Types of RAM

SRAM - Cache - Consists of latches, store binary info and this remains as long as power is applied
- Easier to use and has short access and cycle time

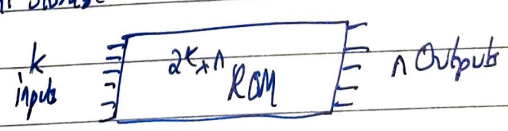
DRAM - Dynamic RAM - Info stored in form of electric charges on capacitors
- Lead to discharge and refreshing done by cycling

Register Memory - Very Fast but Expensive

Sequential Access Memory - Access Time is variable
Random Access Memory -
Volatile Memory and Non volatile

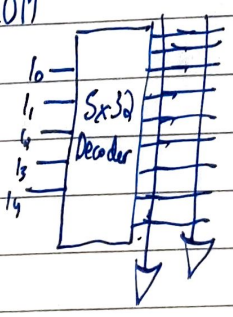
(B) ROM

- Permanent Storage



No Data Inputs
1/more Enable

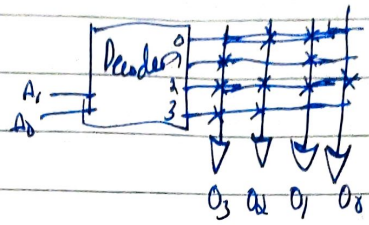
- 32x8 ROM



$2^k \times n$ ROM
X indicates a connection made

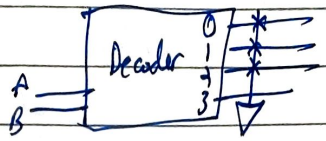
(i) 4x4 ROM

A_1 A_0	O_3	O_2	O_1	O_0
0 0	0	1	1	0
0 1	1	0	1	0
1 0	1	1	1	1
1 1	1	1	0	0

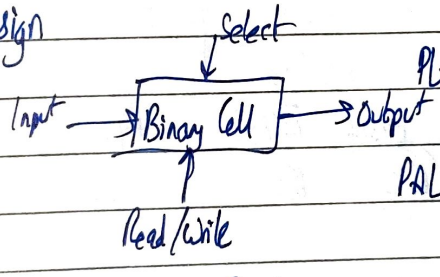


(ii) $F(A,B) = \Sigma m(0,1,2)$

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



(c) RAM Design

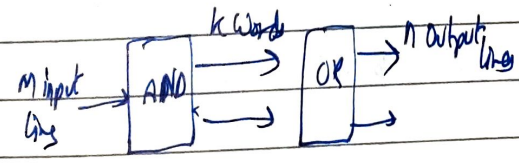


PLA: Inputs – Programmable AND Array – Programmable OR Array – Outputs

PAL: Inputs – Programmable AND Array – Fixed OR Array – Outputs

(B) Design Procedure for PLDs

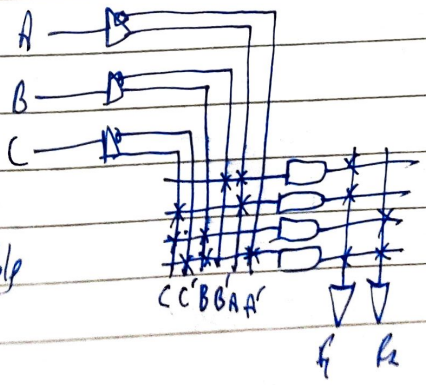
(A) PLA – Remember as Programmable logic – Full logic can be programmed



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

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

Term	Inputs A	B	C	Outputs F_1	F_2
AB'	1	0	–	1	–
AC	1	–	1	1	–
$B'C$	–	0	1	–	1
$A'BC'$	0	1	0	1	1



- Minimize SOP
- Fill Programming Table
- Mark Crosses

If we want negation in end, use XOR with 0/1

(B) PAL

- Minimize Boolean function
- Try to realize with single section
- If large use multiple section

$W(A,B,C,D) = \Sigma m(2,12,13) = ABC' + A'B'CD'$

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

$Y(A,B,C,D) = \Sigma m(1,2,8,12,13) = \underbrace{ABC' + A'B'CD'}_{W} + AC'D' + A'B'C'D$

Product	AND Inputs A	B	C	D	W	Outputs	
1	1	1	0			ABC'	Section 1
2	0	0	1	0		$A'B'CD'$	
3							
4	1					A	Section 2
5		1	1	1		BCD	
6							
7					1	W	Section 3
8	1		0	0		$AC'D'$	
9	0	0	0	1		$A'B'C'D$	

Cross on AND hate means 0

Q1

F_1

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

$F(SOP)$: $A'C'$ $\quad \bar{F}(POS)$: $C + A$ $\quad F(POS)$: $(A+C)'$

F_2

0	1	1	0	0
1	0	0	0	0

$F(SOP)$: $A'B'$ $\quad \bar{F}(POS)$: $A + B$ $\quad F(POS)$: $(A+B)'$

F_3

0	0	1	1	0
1	0	0	1	1

$F(SOP)$: $A'C + AB$ $\quad \bar{F}(POS)$: $A'C' + AB'$ $\quad F(POS)$: $(A'C' + AB')'$

F_4

0	1	0	0	0
1	1	1	0	0

$F(SOP)$: $B'C' + AB'$ $\quad \bar{F}(POS)$: $B + \bar{A}C$ $\quad F(POS)$: $(B + \bar{A}C)'$

	Term	A	B	C	F_1	F_2	F_3	F_4
1	$\bar{A}\bar{C}$	0	–	0	1		1	
2	A	1	–	–		1		
3	B	–	1	–		1		1
4	$A\bar{B}$	1	0	–			1	
5	$\bar{A}C$	0	–	1				1

We take negation using XOR hate

Q. PAL and PLA for $F_1 = \Sigma m(0,4,5,11,12) + d(6,8)$, $F_2 = \Sigma m(4,8,10,12) + d(0,14)$ } F(ABCD)

$F_1 =$

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

EPI EPI EPI

$F_1 = A\bar{B}\bar{C} + A\bar{B}CD + \bar{C}\bar{D}$

Similarly $F_2 = \bar{C}\bar{D} + A\bar{D}$

PLA Table

	AB	CD	F_1	F_2
$A\bar{B}\bar{C}$	0 1	0 -	1	-
$A\bar{B}CD$	1 0	1 1	1	-
$\bar{C}\bar{D}$	- -	0 0	1	1
$A\bar{D}$	1	0	-	1

PAL (4 IP → 2 OP)

	A	B	C	D	
$\bar{C}\bar{D}$	-	-	0	0	F_1
$A\bar{B}CD$	1	0	1	1	F_1
$A\bar{B}\bar{C}$	0	1	0	-	F_1
$\bar{C}\bar{D}$	-	-	0	0	F_2
$A\bar{D}$	1	-	-	0	F_2

(14) ASM Algorithmic State Machines

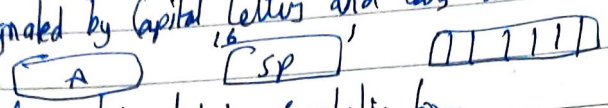
- Sometimes due to a large no. of gates, we go with modular approach
- Modules – Registers, MUXs, arithmetic circuits and control logic
- Interconnected with datapath (Op) & Control Signals

(a) Register Transfer Level Notation

- Modules defined by set of registers and op(s) where registers are basic elements
- Info flow & processing performed on data stored in registers
- RTL
  - Set of Registers
  - Operations
  - Control

(b) Inter Register Transfer

- Registers in digital System
  - Designated by Capital Letters and Cells are numbered

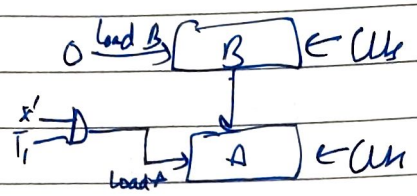


- Info transfer designated in symbolic form

  $A \leftarrow B$ Replacement Operator
- Contents of A replaced by contents of B
- Transfer shouldn't happen all the time $x'T_1 : A \leftarrow B$

  Transfer happens only if $x'T_1 = 1$

  ↳ Timing Variable



(c) Symbols

Letters – Registers A, PC

Subscript – bit of Register A_6

Parenthesis – Portion of Register $PC(4)$

Arrow ← Transfer $A \leftarrow B$

Colon : Terminate Control $x'T_1 :$

Comma, Separation Op $A \leftarrow B, C \leftarrow D$

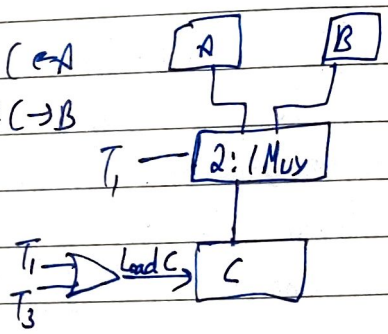
Shr R – Shift Right

Square Brackets [] Address of Mem Transfer

Ex:

$T_1 : C \leftarrow A$

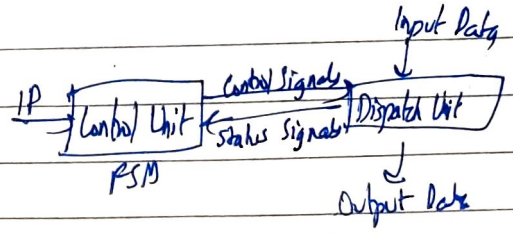
$T_3 : C \rightarrow B$



- ASMs store binary info
- Data : manipulated by performing operations (Arithmetic, Shift etc.)
- Control : Provides command signals that coordinate and execute various operations

Logic Design of Digital Systems

- Design of circuits for data
- Design of Control Circuits

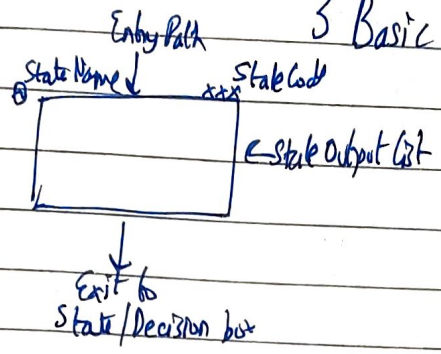


Control logic - Generates signals for sequencing the operations in datapath → FSM
- FSM → Finite State Machine

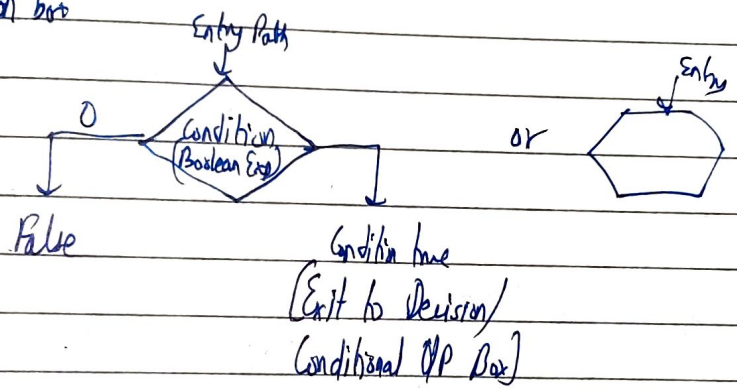
Control Sequence & Datapath tasks — Hardware Algo
↳ Flowchart → ASM

3 Basic Elements → State box, Decision box, Conditional box

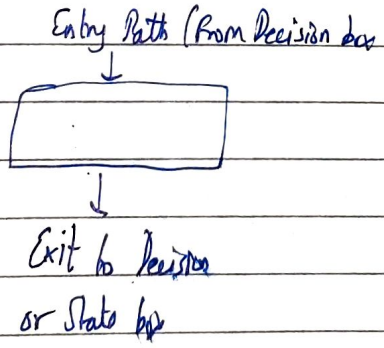
ⓐ State box



ⓑ Decision box

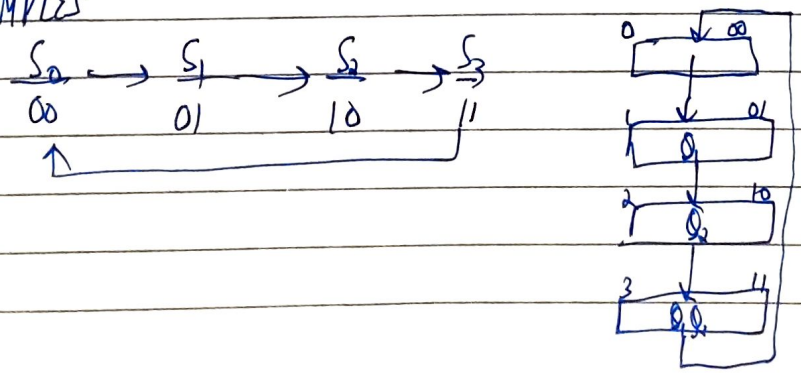


ⓒ Conditional box

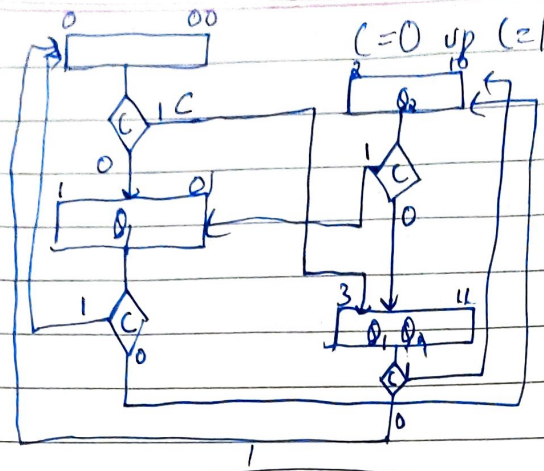


EXAMPLES

2 bit counter

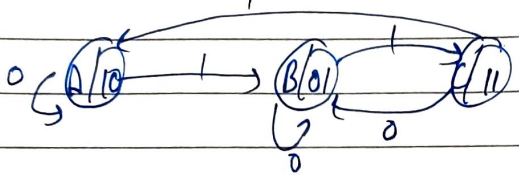


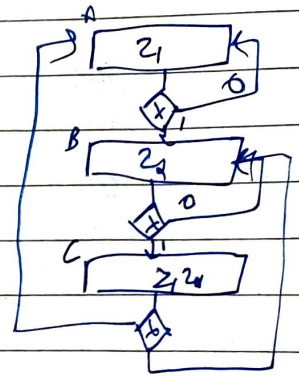
2 bit Up/down Counter



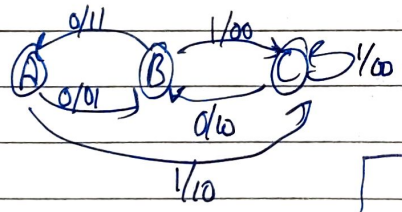
Up	Down
00	00
01	11
10	10
11	01
	00

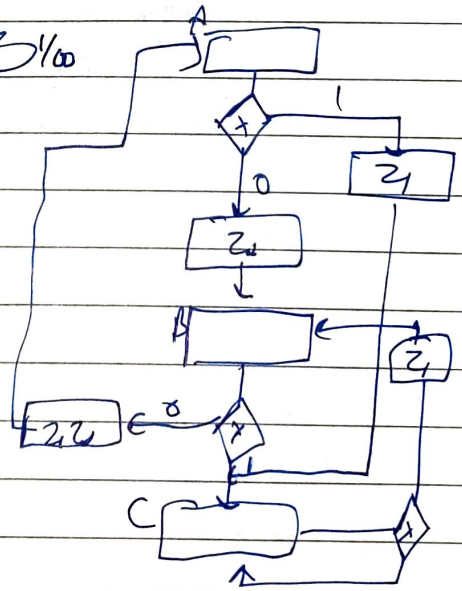
Moore Seq. Network



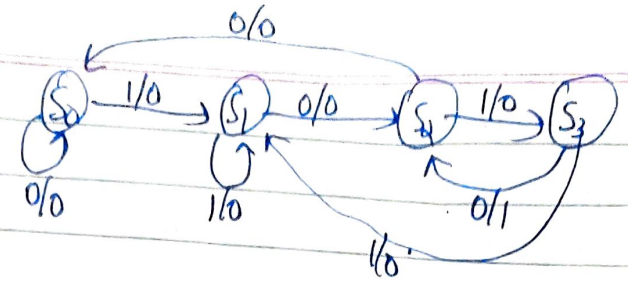


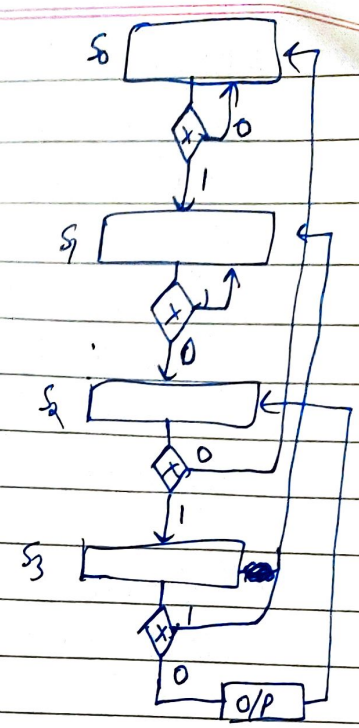
Mealy Seq. Network



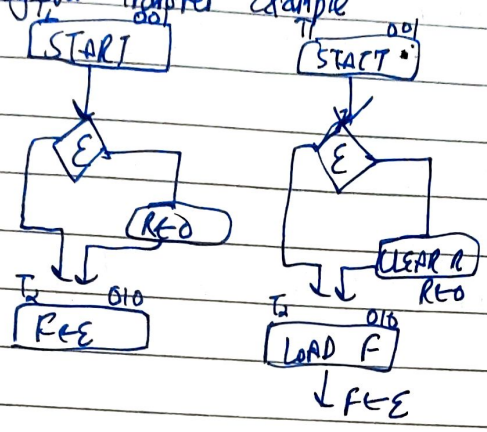


Sequence Detector 1010

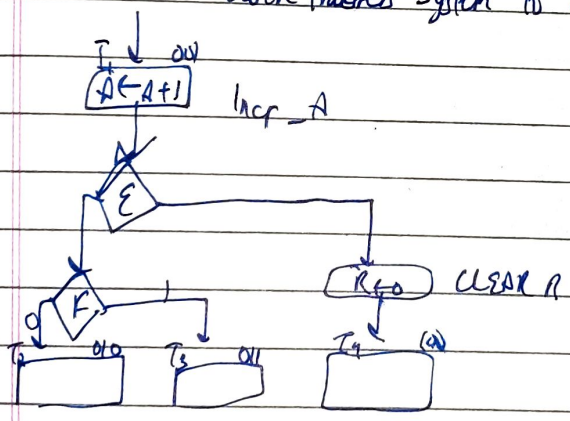




Ex: Register Transfer Example



ASM Block → State/Decision & Conditional Block
→ One Entrance Many Exits
→ Clock transfers system to one of the next states



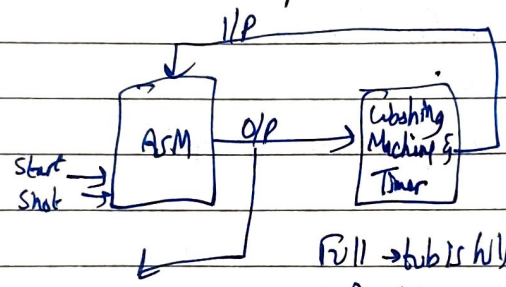
A is incremented, but R is cleared in next +ve edge cycle of Clock

- In Present State, a signal for incrementing A is generated and E and F are checked and clear R signal is generated if E=1
- In Next Cycle, A is incremented, R is cleared if E=1 and continue transfer to next state

Washing Machine Design Example

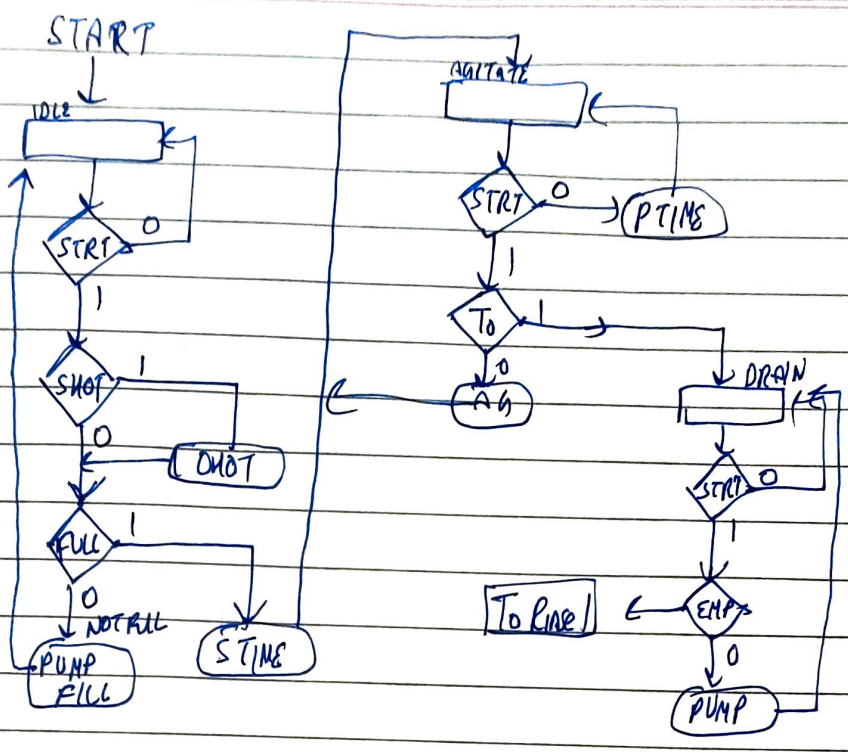
- Controller Begins when it receives STRT (start) from user
- It fills the washer with cold or hot water depending on user selection SHOT input
- Agitates until a timer indicates that the cycle is finished
- Controller then drains soppy water and fills the machine with cold water
- Washer agitates again till a time
- Controller drains the rinse water and enters the spin cycle, spinning the clothes dry until the timer indicates end of cycle
- If off button is ever pushed, the water holds the current state until start is pressed again

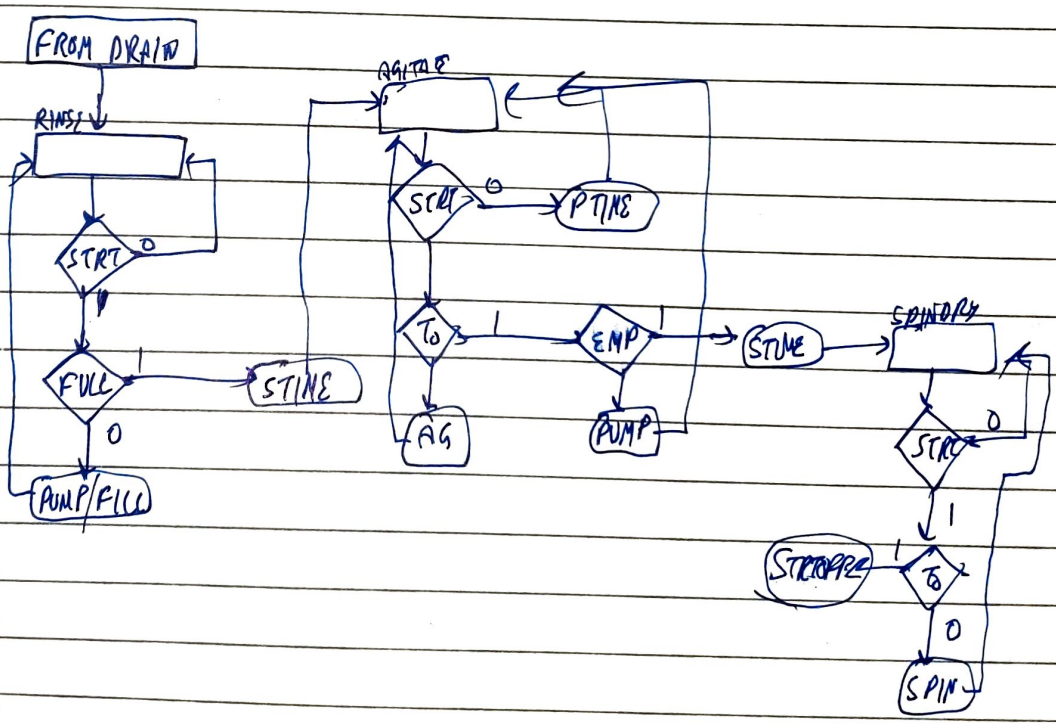
Fill Water → Wash Soap → Drain → Fill Water → Rinse



FULL → tub is full
EMP – tub is Empty
TO – Timer out

SPIN SPIN DRY
PTIME Pause
OHOT ON HOT
PUMP Turn on Pump
FILL Water IN/OUT
AG Agitate
STIME Start timer


START
IDLE
STRT
0
1
SHOT
1
0
DHOT
FULL
1
0
NOT FULL
PUMP FILL
STIME
AGITATE
STRT
0
PTIME
1
To
1
0
AG
DRAIN
STRT
0
1
EMP
0
PUMP
To Rinse


FROM DRAIN
RINSE
STRT
0
1
FULL
1
STIME
0
PUMP/FILL
AGITATE
STRT
0
PTIME
1
To
1
EMP
1
AG
PUMP
STIME
SPINDRY
STRT
0
1
To
1
STOPPER
0
SPIN

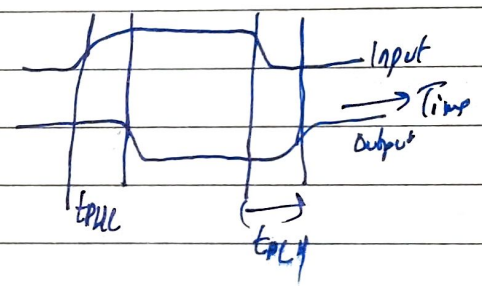
(15) Digital Logic

- Families

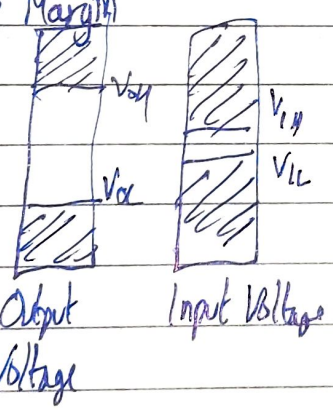
RTL – Resistor Transistor Logic
DTL – Diode Transistor Logic
TTL – Transistor-Transistor Logic
ECL – Emitter-Coupled Logic – High Speed
MOS – Metal Oxide Semiconductor – Compact
CMOS – Complementary MOS – low power / MOST Dominant
BiCMOS – Bipolar CMOS
GaAs : Gallium Arsenide – very high speed

Fan-in : # of gate inputs
Fan-out : # of standard loads a gate's output can drive
Noise margin : Max external noise tolerated
Power dissipation : Power consumed by gate dissipated as heat
Propagation Delay

$$\text{Fanout} = \frac{I_{OH}}{I_{IH}} \text{ or } \frac{I_{OL}}{I_{IL}}$$

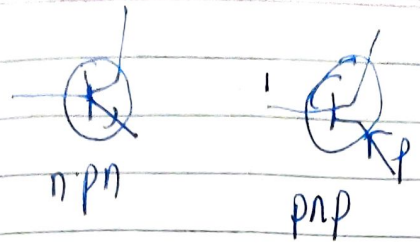


Noise Margin



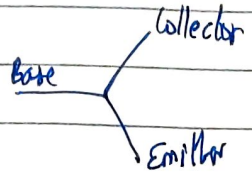
$V_{IH\,min} < V_{OH\,min}$
$V_{IL\,max} > V_{OL\,max}$

Power Dissipation
- milliwatts
- $V_{CC} * I_{CC}$
- $I_{CC}(avg) = \frac{I_{CCH} + I_{CCL}}{2}$



Transistor – Amplifier/Switching

Closed $V_{CE} = 0$ Practically $V_{CE} = 0.2V$



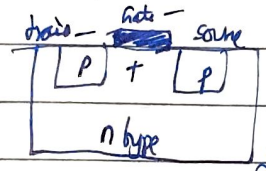
Using Base, we can control if the switch is closed or open

NOT Gate

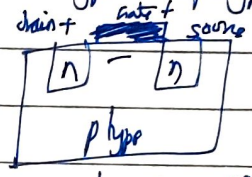
74 – Standard
74L → Low Power
H – High speed
S → Schottky
AS → Advanced Schottky
F → Fast

MOS Transistor

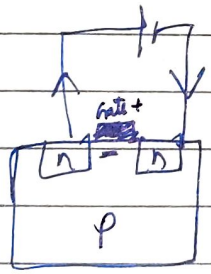
NMOS and PMOS – Channel is n type or p type resp.



PMOS



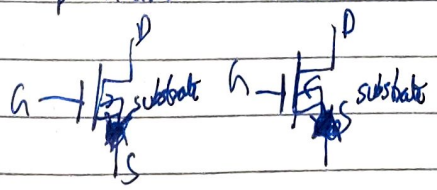
n channel NMOS

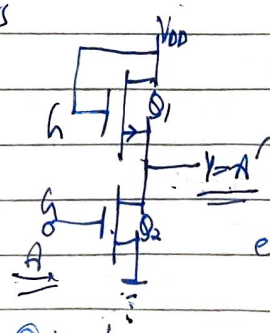


- Continuity is maintained only if n reaches n
- To create a channel, we apply voltage at gate, Here, the gate induces – and forces all the holes away

p channel n channel

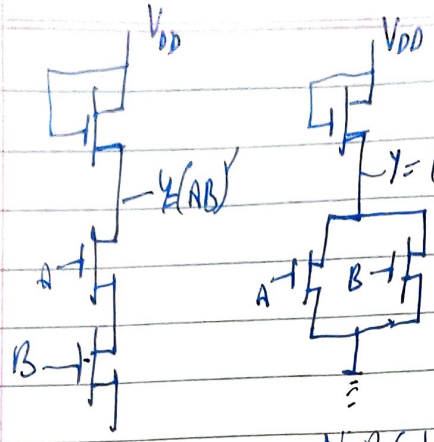
If substrate is not shown, reverse arrows



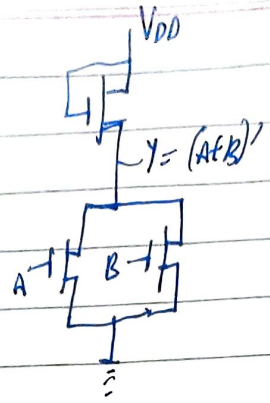


(a) Inverter

if we apply 5V ON $V_y = 0V$
ON
else ON $V_y =$
OFF



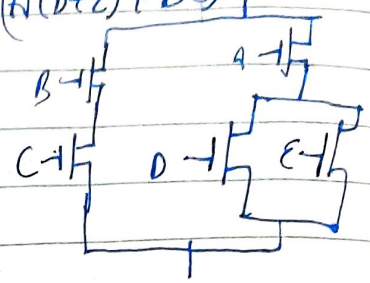
NAND Gate



NOR Gate

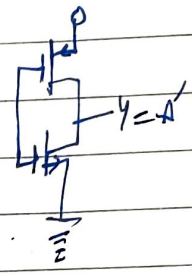
$Z = (A(D+E)+BC)'$

Don't Miss V_{CC} + Drain



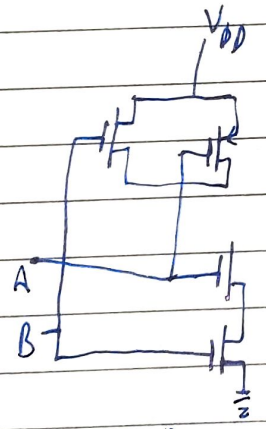
PMOS

Here Source is connected to V_{DD}
Not Base

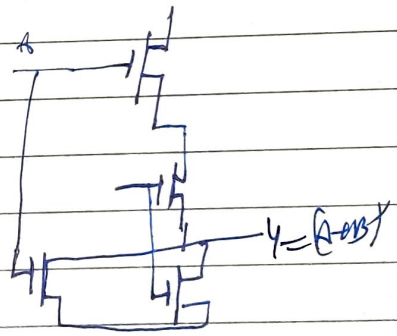


Dual

AND & OR are interchanged



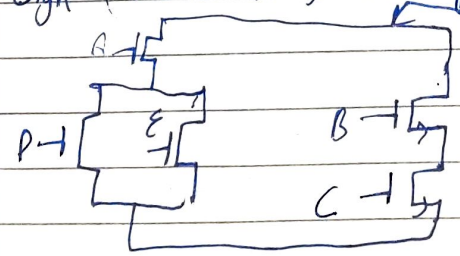
NAND



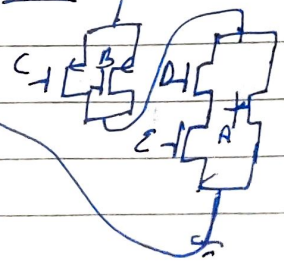
NOR

Design $(A(D+E)+BC)'$ CMOS

Draw n network first



n Network



Join n network & p network