ANSWERS FOR TMA 02

EEX5351

Digital Electronic System

By

S.A. Pushpitha Kavinda

Reg No: 617143597

Submitted to,

Department of Electrical and Computer Engineering

Faculty of Engineering Technology

The Open University of Sri Lanka

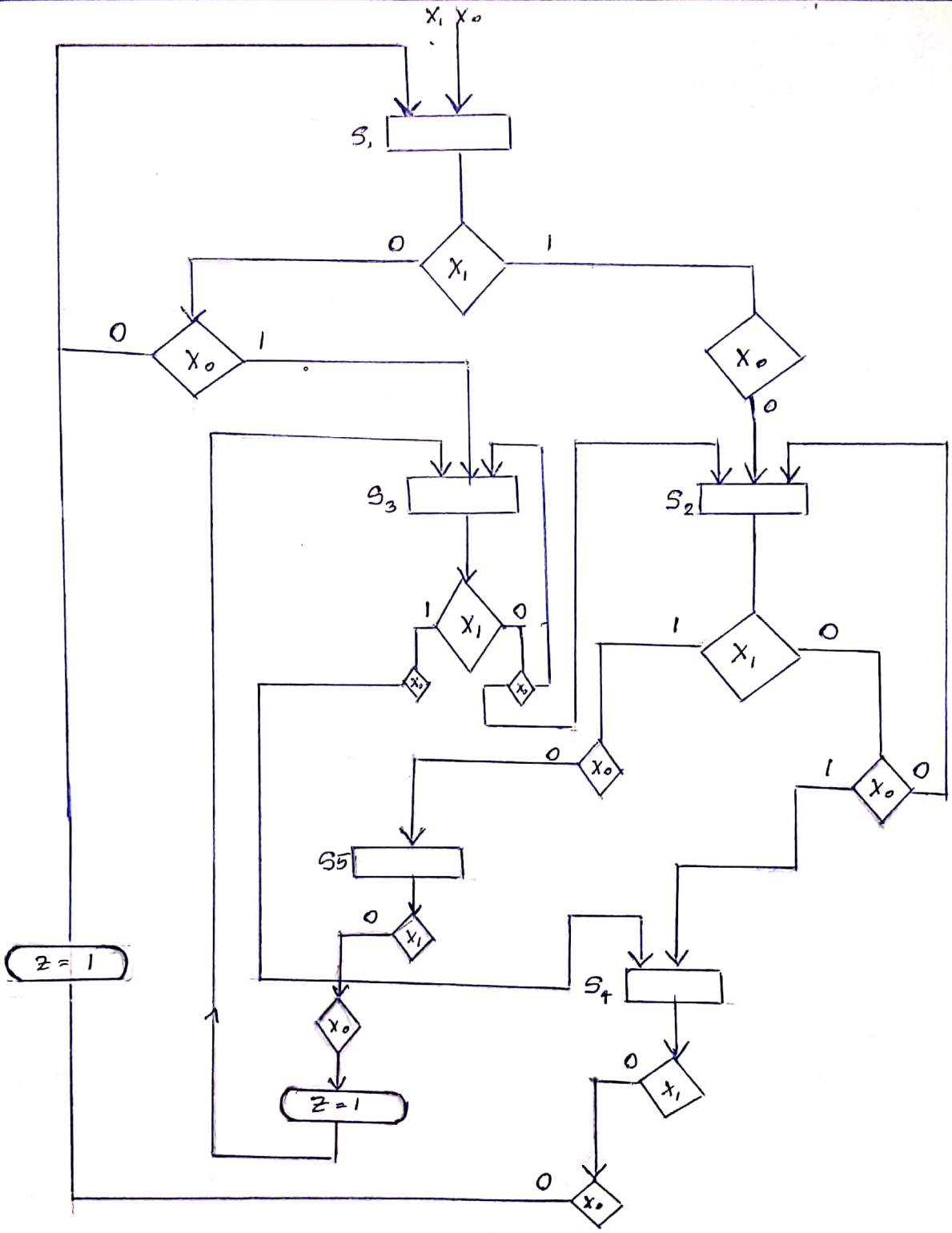
At

Colombo Regional Center

Due Date

15/09/2022

Q1)
a)


X1 X0
S1
0
X1
1
0
X0
1
X0
0
S3
S2
1
X1
0
1
X1
0
0
X0
1
X0
0
S5
0
X1
Z = 1
S4
X0
Z = 1
0
X1
0
X0

b)

Let states are defining by bits $Q_2Q_1Q_0$ and the relevent next states are $Q_2^+Q_1^+Q_0^+$.

Let, $S_1 = 000$ $S_2 = 001$ $S_3 = 010$

$S_4 = 011$ $S_5 = 100$

⇒ Q_2^+ is '1' when only next state at S_5. Therefor

$f(Q_2, Q_1, Q_0) \cup (x_1, x_0)$

$S_2 \xrightarrow{10} S_5$ 00110 $\therefore Q_2^+ = \overline{Q_2}\,\overline{Q_1}\,Q_0\,x_1\,\overline{x_0}$

⇒ Q_1^+ is '1' when only next state at S_3 or S_4.

$f(Q_2, Q_1, Q_0) \cup (x_1, x_0)$

$S_1 \xrightarrow{01} S_3$ ✓00001

$S_3 \xrightarrow{00} S_3$ 01000

$S_5 \xrightarrow{00} S_3$ 10000

$S_2 \xrightarrow{01} S_4$ ✓00101

$S_3 \xrightarrow{10} S_4$ 01010

00x01, ✓01000, 10000, ✓01010 → 00x01, 010x0, 10000

$\therefore Q_1^+ = \overline{Q_2}\,\overline{Q_1}\,\overline{x_1}\,x_0 + \overline{Q_2}\,Q_1\,\overline{Q_0}\,\overline{x_0} + Q_2\,\overline{Q_1}\,\overline{Q_0}\,\overline{x_1}\,\overline{x_0}$

⇒ Q_0^+ is '1' when only next state at S_2 and S_4

$f(Q_2, Q_1, Q_0) \cup (x_1, x_0)$

$S_1 \xrightarrow{10} S_2$ 00010

$S_2 \xrightarrow{00} S_2$ ✓00100

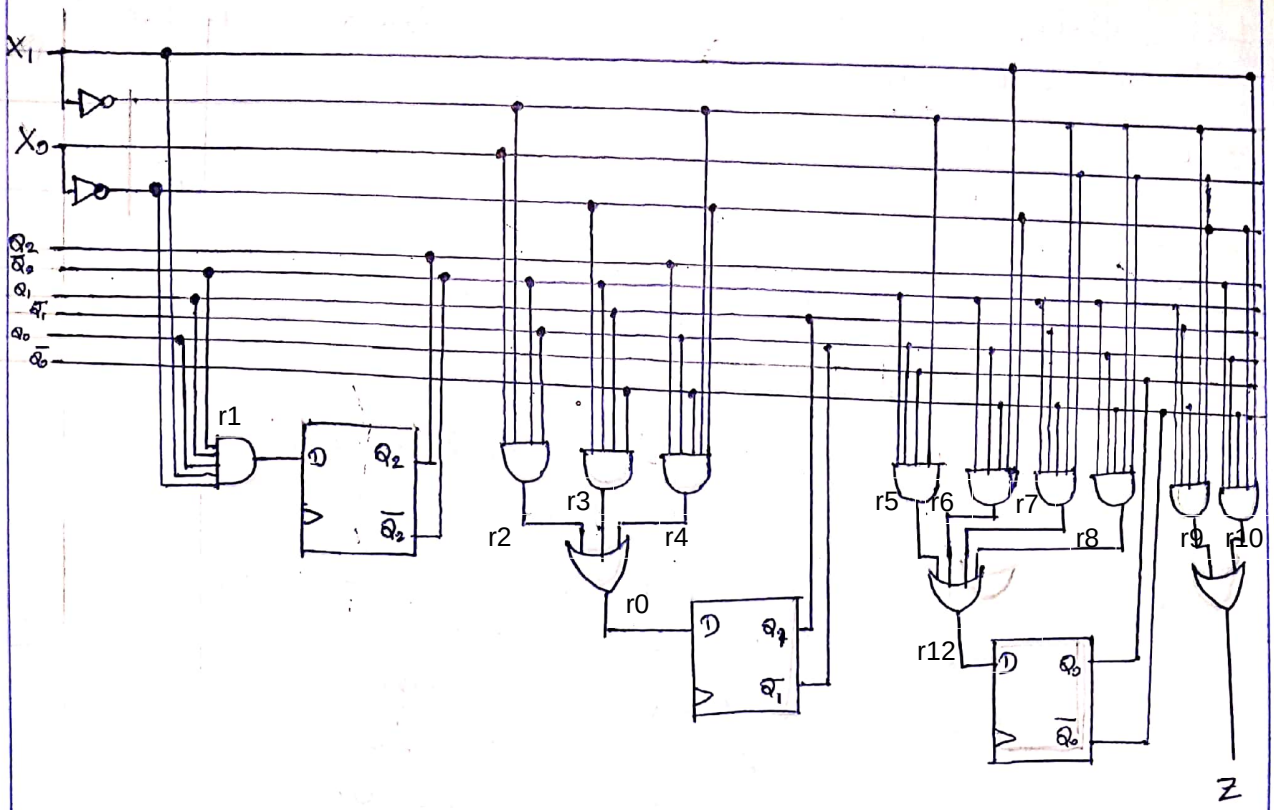
$S_3 \xrightarrow{01} S_2$ 01001

$S_2 \xrightarrow{01} S_4$ ✓00101

$S_3 \xrightarrow{10} S_4$ 01010

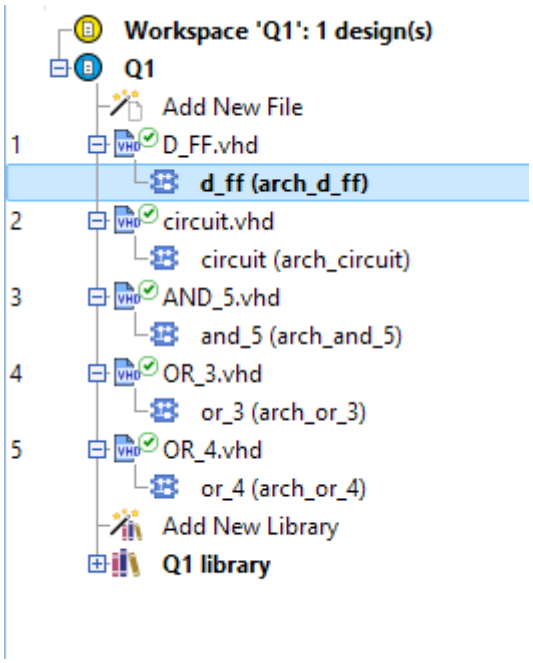
00010, ✓00100, 01001, ✓00101, 01010 → 0010x, 00010, 01001, 01010

$\therefore Q_0^+ = \overline{Q_2}\,\overline{Q_1}\,Q_0\,\overline{x_1} + \overline{Q_2}\,\overline{Q_1}\,\overline{Q_0}\,x_1\,\overline{x_0} + \overline{Q_2}\,Q_1\,\overline{Q_0}\,\overline{x_1}\,x_0 + Q_2\,\overline{Q_1}\,\overline{Q_0}\,\overline{x_1}\,x_0$



Z = Q2' Q1 Q0 X0' X1' + Q2 Q1' Q0' X0' X1'

C)



Code for 5 input AND Gate:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity AND_5 is
    port(X1,X2,X3,X4,X5 : in std_logic;
    Z : out std_logic);
end AND_5;

architecture arch_AND_5 of AND_5 is
begin
    Z <= X1 and X2 and X3 and X4 and X5;
end arch_AND_5;
```

Code for 4 input OR Gate:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity OR_4 is
    port(X1,X2,X3,X4 : in std_logic;
    Z : out std_logic);
end OR_4;

architecture arch_OR_4 of OR_4 is
begin
    Z <= X1 or X2 or X3 or X4;
end arch_OR_4;
```

Code for D Flip Flop

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity D_FF is
    port(
    clk,D : in std_logic;
    Q   : out std_logic;
    Q_bar : out std_logic);
end D_FF;

architecture arch_D_FF of D_FF is
begin
    process(clk,D) is
    begin
    if(rising_edge(clk)) then
        Q <= D;
        Q_bar <= ( not D);
    end if;

end process;

end arch_D_FF;
```

Code for Circuit:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity circuit is
    port(
    clk: in std_logic;
    X1: in std_logic;    --defining X1 pin
    X0: in std_logic;    -- defining X0 pin
    Q2,Q1,Q0,Q2_bar,Q1_bar,Q0_bar: inout std_logic;
    Z: inout std_logic
    );
end circuit;

architecture arch_circuit of circuit is
    component AND_5
    port(
        X1 : in STD_LOGIC;
        X2 : in STD_LOGIC;
        X3 : in STD_LOGIC;
        X4 : in STD_LOGIC;
        X5 : in STD_LOGIC;
        Z : out STD_LOGIC
    );
    end component;
    component OR_4
    port(
        X1 : in STD_LOGIC;
        X2 : in STD_LOGIC;
        X3 : in STD_LOGIC;
        X4 : in STD_LOGIC;
        Z : out STD_LOGIC
    );
    end component;

    component D_FF
    port(
        clk : in STD_LOGIC;
        D : in STD_LOGIC;
        Q : out STD_LOGIC;
        Q_bar : out STD_LOGIC
    );
    end component;


    signal rx1,rx0,r0,r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12 :std_logic;
    begin
        rx1<= not X1;
        rx0 <= not X0;
        FF2: D_FF port map (clk,r1,Q2,Q2_bar);
        FF1: D_FF port map (clk,r4,Q1,Q1_bar);
        FF0: D_FF port map (clk,r12,Q0,Q0_bar);

        AND1: AND_5 port map (Q2_bar,Q1_bar,Q0,X1,rx0,r1);
```

```
        AND21: AND_5 port map (Q2_bar,Q1_bar,rx1,rx0,'1',r2);
        AND22: AND_5 port map (Q2_bar,Q1,Q0_bar,rx0,'1',r3); --
        AND23: AND_5 port map (Q2,Q1_bar,Q0_bar,rx1,rx0,r4);
        OR1: OR_4 port map(r2,r3,r4,'0',r0);

        AND31: AND_5 port map (Q2_bar,Q1_bar,Q0,rx1,'1',r5);
        AND32: AND_5 port map (Q2_bar,Q1_bar,Q0_bar,X1,rx0,r6)    ;
        AND33: AND_5 port map (Q2_bar,Q1,Q0_bar,rx1,X0,r7);
        AND34: AND_5 port map (Q2_bar,Q1,Q0_bar,rx1,X0,r8);
        OR2: OR_4 port map(r5,r6,r7,r8,r12);

        AND41: AND_5 port map (Q2_bar,Q1,Q0,rx0,rx1,r9);
        AND42: AND_5 port map (Q2,Q1_bar,Q0_bar,rx0,rx1,r10) ;
        OR3: OR_4 port map(r9,r10,'0',Z)  ;


        end arch_circuit;
```

d)

Test Bench Code:

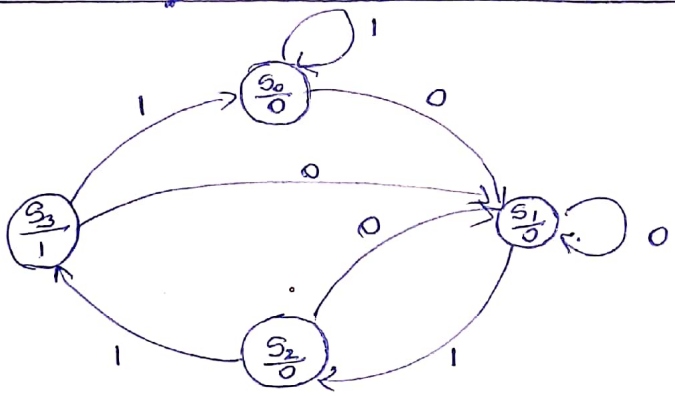
```
library IEEE;
use IEEE.std_logic_1164.ALL;

entity test_circuit is
end test_circuit;


architecture arch_test of test_circuit is

    component circuit
    port(
    clk: in std_logic;
    X1: in std_logic;    --defining X1 pin
    X0: in std_logic;    -- defining X0 pin
    Q2,Q1,Q0,Q2_bar,Q1_bar,Q0_bar: inout std_logic;
    Z: inout std_logic
    );
    end component;
    --for all: circuit use entity work.circuit(arch_circuit);

    signal clk,X1,X0,Q2,Q1,Q0,Q2_bar,Q1_bar,Q0_bar,Z : std_logic;

    begin
    U1: circuit port map( clk,X1,X0,Q2,Q1,Q0,Q2_bar,Q1_bar,Q0_bar,Z );
    --X1<= '0';X0<= '0';Q2<= '0';Q1<= '0';Q0<= '0';
    process
    begin

        clk <='0';
        wait for 5ns;
        clk <= '1';
        wait for 5ns;

    end process;

    process
    begin

    --case 1

    X0 <= '0';
    X1 <= '0';
    wait for 10ns;
    assert ( z = '0' and Q2='0' and Q1='0' and Q0='0' ) report "Failed Case
1" severity error;
    wait for 40ns;

    --case 2

    X0 <= '1';
    X1 <= '0';
    wait for 10ns;
    assert ( z = '0' and Q2='0' and Q1='1' and Q0='0' ) report "Failed Case
2" severity error;
```

```
    --case 3

    X0 <= '0';
    X1 <= '1';
    wait for 10ns;
    assert ( z = '0' and Q2='0' and Q1='0' and Q0='1' ) report "Failed Case
3" severity error;
    wait for 40ns;


    end process;

    end arch_test;
```

Q2)

* 011 pattern never be overlap. ∴ overlap and Non-overlap designs are identical

a) moore Sequential detector - moore machine



States: Code

S_0 - Reset (00)

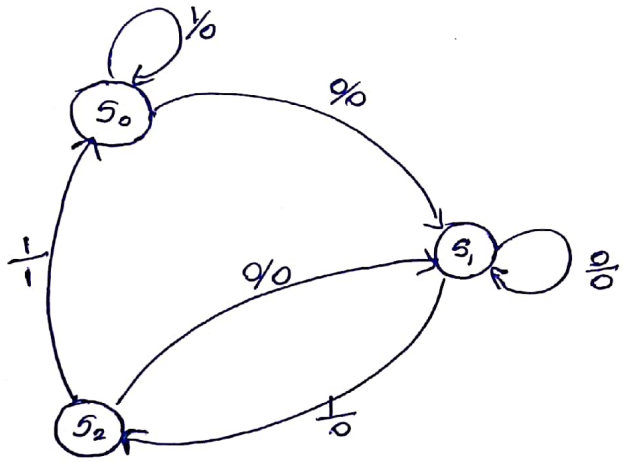
S_1 - 0 (01)

S_2 - 01 (10)

S_3 - 011 (11)

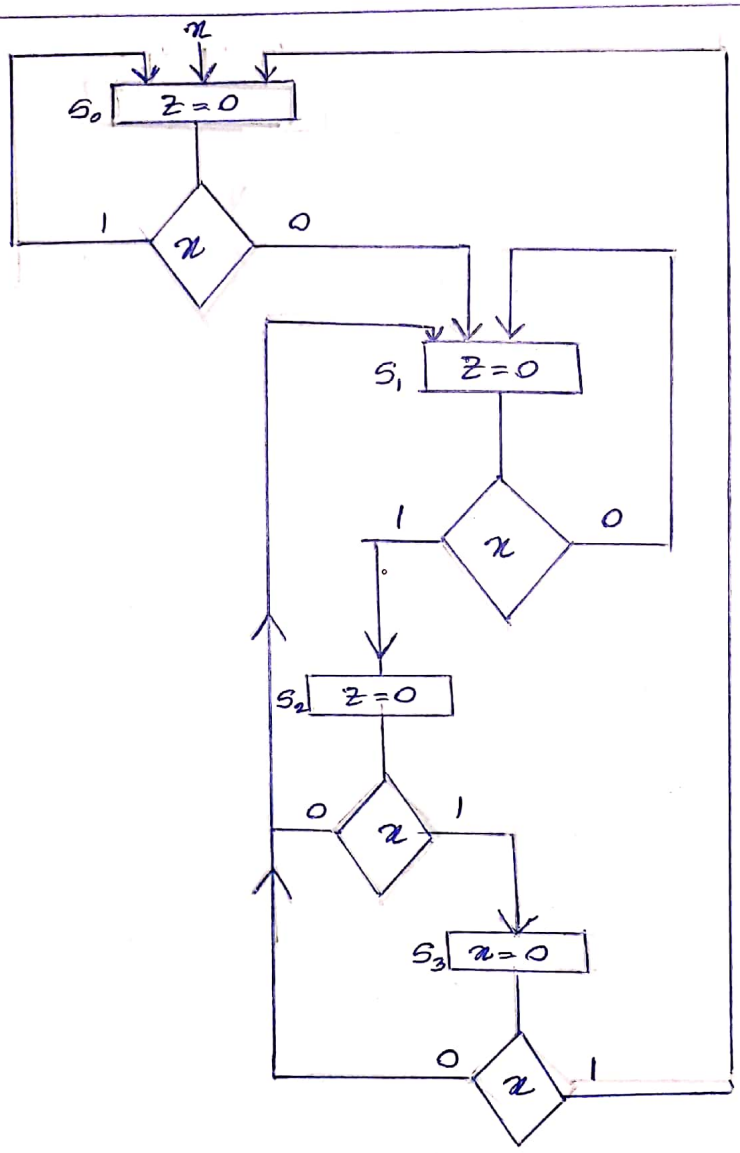
input - x

mealy - Sequential Detector

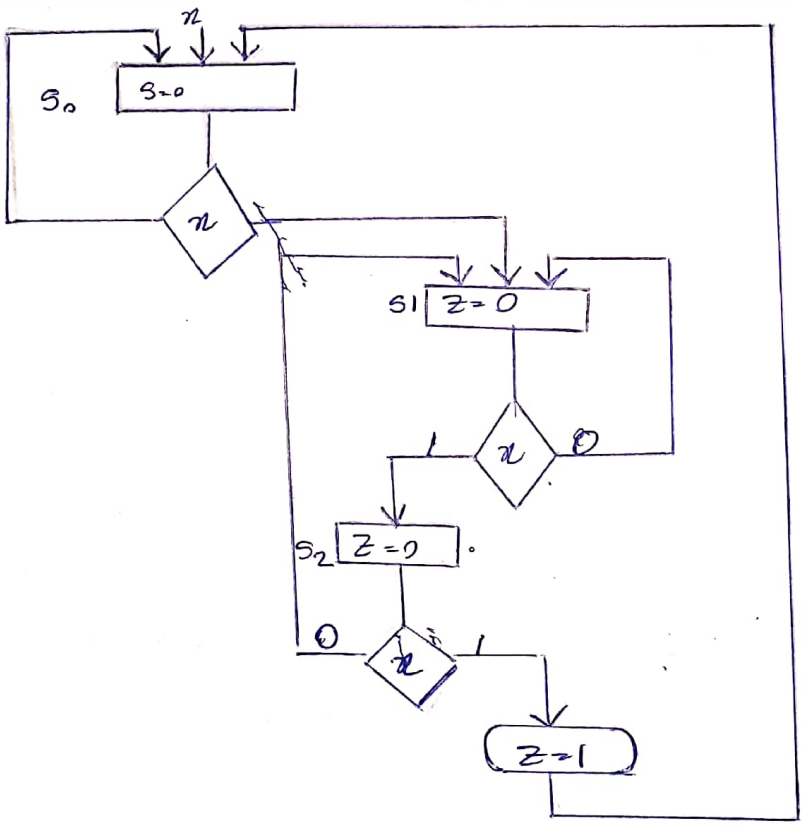


States:	Code
S_0 - Reset	00
S_1 - 0	01
S_2 - 01	10
~~S_3 - 011~~	~~11~~

b)



Asm chart for moore machine.



ASM chart fore mealy machine

c)

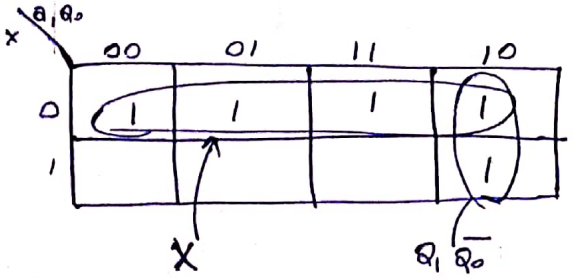
⇒ moore machine - state Table

Current state Q_1	Q_0	Input x	Next state Q_1^+	Q_0^+	output z	Flip Flop D_1	D_0
0	0	0	0	1	0	0	1
0	0	1	0	0	0	0	0
0	1	0	0	1	0	0	1
0	1	1	1	0	0	1	0
1	0	0	0	1	0	0	1
1	0	1	1	1	0	1	1
1	1	0	0	1	1	0	1
1	1	1	0	0	1	0	0

⇒ Mearly machine - state Table

Current state Q_1	Q_0	Input x	Next state Q_1^+	Q_0^+	output z	Flip Flop D_1	D_0
0	0	0	0	1	0	0	1
0	0	1	0	0	0	0	0
0	1	0	0	1	0	0	1
0	1	1	1	0	0	1	0
1	0	0	0	1	0	0	1
1	0	1	0	0	1	0	0
1	1	-	-		-	-	
1	1	-	-			-	

d) k-map for moore machine

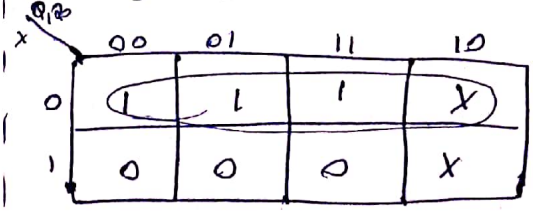


$Q_0^+ = x + Q_1\overline{Q_0}$

$Q_1^+ = \overline{Q_1}Q_0 + Q_1\overline{Q_0}$

$= Q_1 \oplus Q_0$

k-map for mearly machine



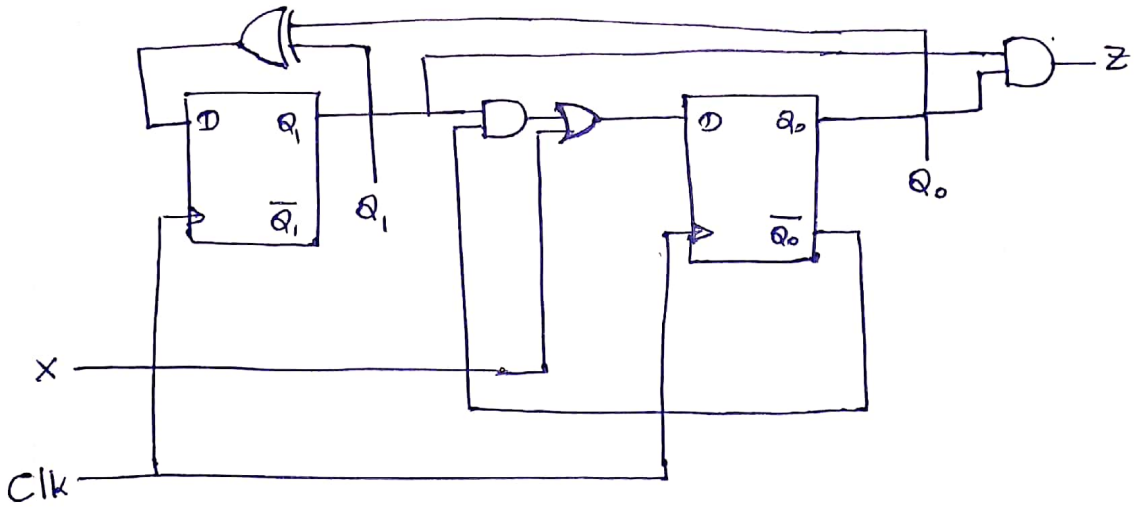
$Q_0^+ = \overline{x}$

$Q_1^+ = \overline{Q_1}\,Q_0 x$

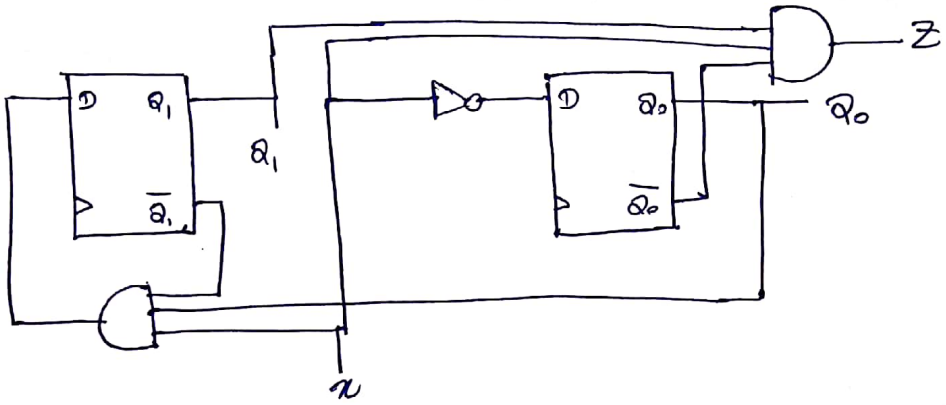
$Z = Q_1\overline{Q_0}\,x$

e)

Circuit Diagram for moore machine.

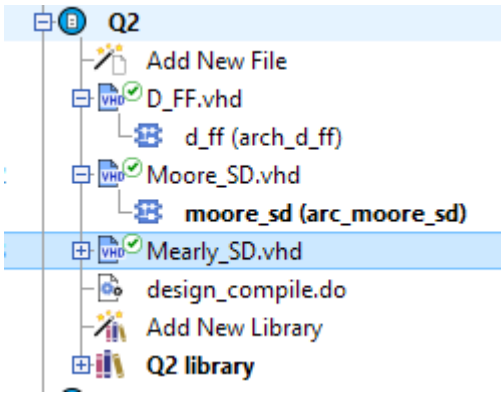


Circuit Diagram for mearly machine.



CamScanner

f)



VHDL Code for D- Flip Flop

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity D_FF is
    port(
    clk,D : in std_logic;
    Q   : out std_logic;
    Q_bar : out std_logic);
end D_FF;

architecture arch_D_FF of D_FF is
begin
    process(clk,D) is
    begin
    if(rising_edge(clk)) then
        Q <= D;
        Q_bar <= ( not D);
    end if;

end process;

end arch_D_FF;
```

VHDL Code For Sequential detector - Moore Machine

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Moore_SD is
    port(X,clk : in std_logic;
    Z : out std_logic);
end Moore_Sd;

architecture arc_Moore_SD of Moore_SD is


    component D_FF
    port(
        clk : in STD_LOGIC;
        D : in STD_LOGIC;
        Q : out STD_LOGIC;
        Q_bar : out STD_LOGIC
    );
    end component;


signal r1,r2,r3,r4,r5,r6 : std_logic;

begin
    FF1: D_FF port map(clk,r1,r2,open);
    FF2: D_FF port map(clk,r6,r3,r4);
    r1 <= r2 xor r3;
    r5 <= r2 and r4;
    r5 <= X or r5;
    Z <= r2 and r3;

end arc_Moore_SD;
```

VHDL Code for Sequential detector – mearly machine

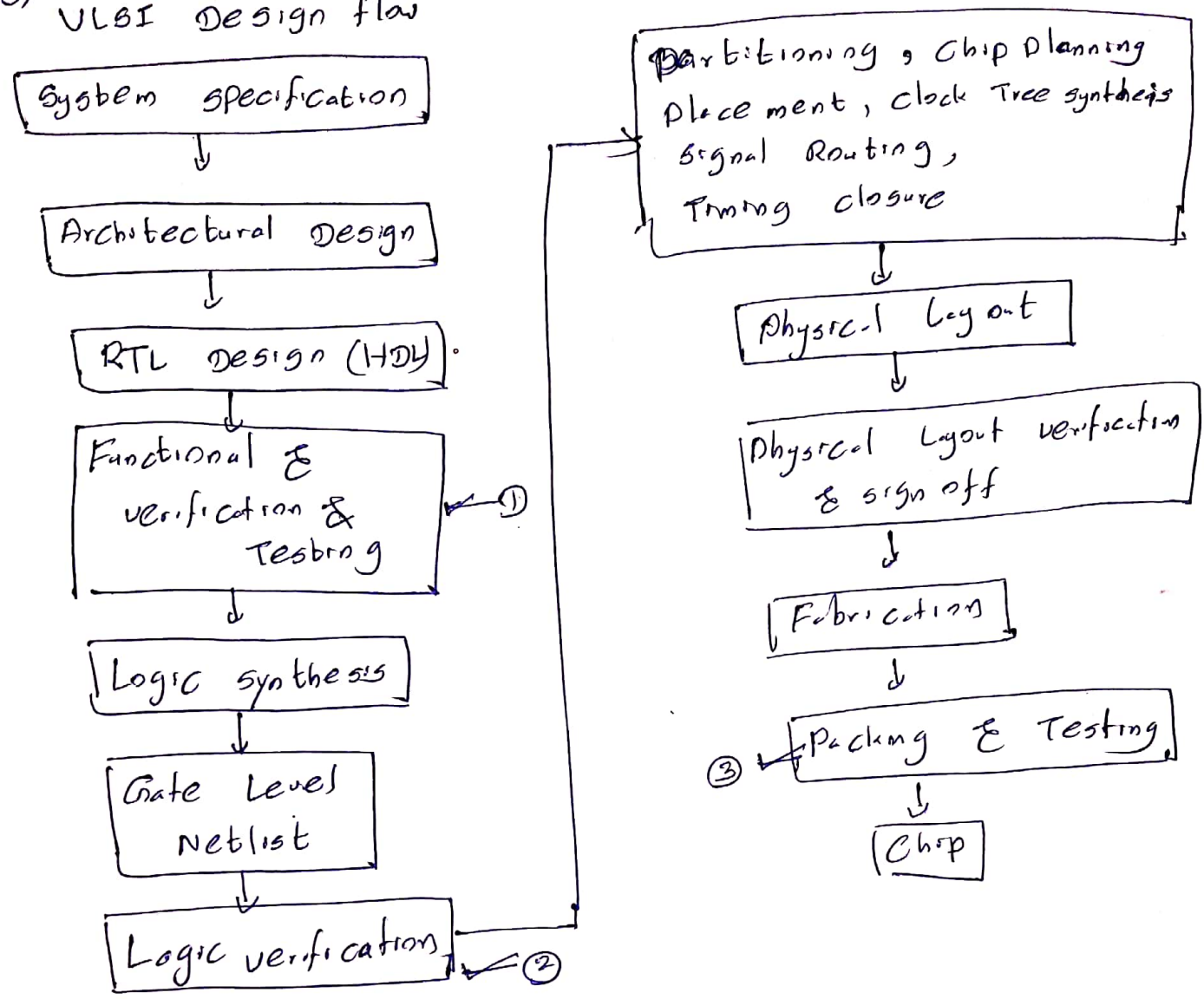
```
library IEEE;
use IEEE.std_logic_1164.all;


entity Mearly_SD is
    port(X,clk : in std_logic;
    Z : out std_logic);
end Mearly_SD;

architecture arc_Mearly_SD of Mearly_SD is


    component D_FF
    port(
        clk : in STD_LOGIC;
        D : in STD_LOGIC;
        Q : out STD_LOGIC;
        Q_bar : out STD_LOGIC
    );
    end component;

    signal r1,r2,r3,r4,r5,r6 : std_logic;

    begin
        FF1:D_FF port map(clk,r1,r5,r2);
        FF2:D_FF port map(clk,r4,r3,r6);
        r1 <= (r2 and r3) and X;
        r4 <= not X;
        Z <= (r5 and X) and r6;

    end arc_Mearly_SD;
```

g) Comparison:

- Moore machine required 4 states and Mealy machine required only 3 state for detect the sequence.
- Mealy machine required less number of components than Moore machine.
- Moore machine required 2 input components only. But mealy machine required 3 input components also.

Q3)

VLSI Design flow

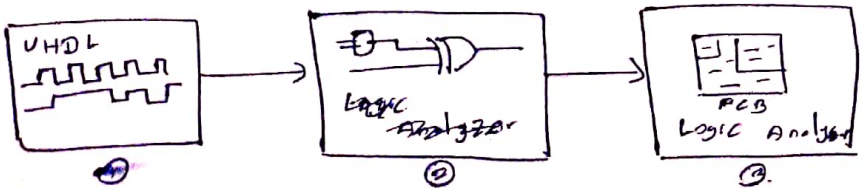


① Functional & verification & Testing.

Check whether the written description is correct or not. Testing using simulations.

② After synthesis the logics, verify the synthesised logical circuit.

③ After place the logic circuit on physical layouts, do a testing for verify the circuit behaviour in real time.



b) Functional Testing

In functional testing we put all possible input combinations and check the output.

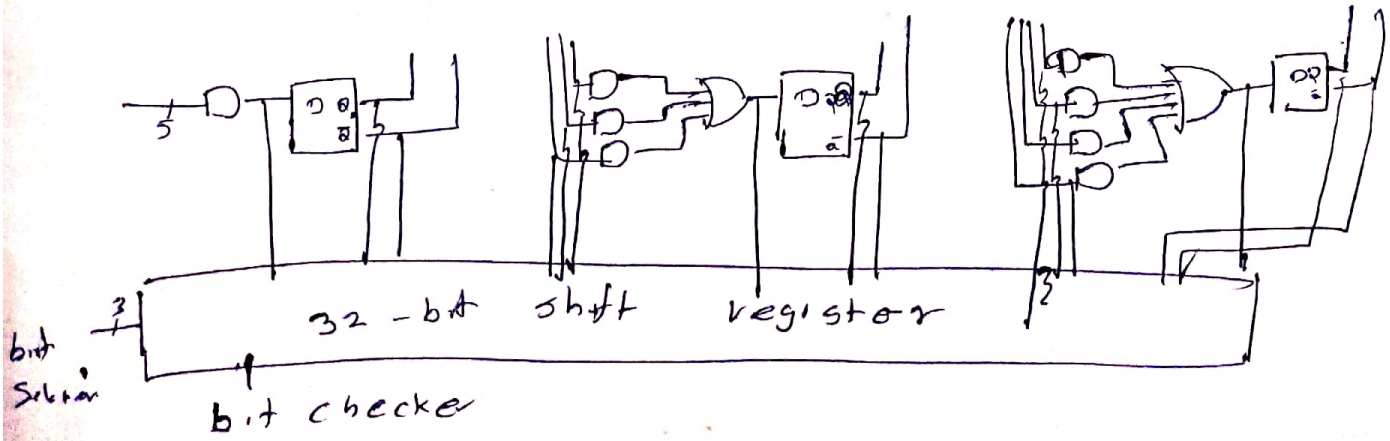
So Q1.(b) circuit has only 2 inputs X_1, X_0 and 4 outputs Q_2, Q_1, Q_0 and Z, we can put all possible Inputs variations to $X_1 X_0$ and see what happend on the state and the output (Z).

$$(X_1 X_0) \in \{(0,0), (0,1), (1,0), (1,1)\}$$

Structural Testing

* In structural testing we check each internal elements whether good or malfunctional. For this uses a 2 shift register for check sequentially each memory blocks and this way we can done the testing with the minimum Input/output pins.

For example if we want check A, AND Gate, we ~~want to~~ can select 5 inputs and check the result. Also we can disable unwanted Flip-flop s by using selector pin disable.



b)

⇒ Structural / Traditional Algorithms

* PODEM (Path-Oriented Design making)
* FAN (FAN-out-oriented test generation)
* TOPS (Topological search)
* Socrates
* EST (Equivalent STate Hashing)
* TG-LEAP (Test generation- Level dependent Analysis in Path sensitization)

⇒ satisfiability-based Algorithms

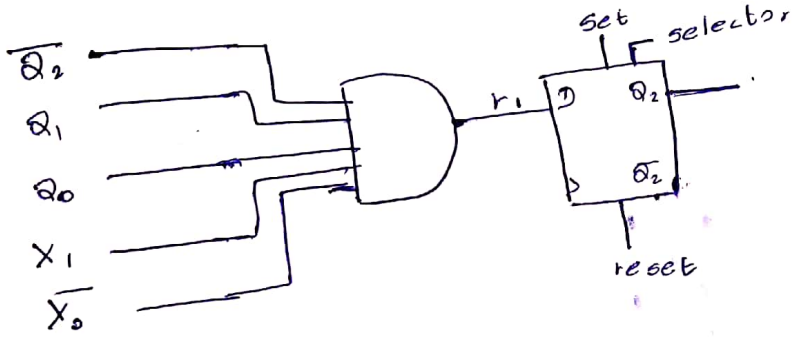
* SAT (Boolean satisfibility)
* TEGUS (Test Generation using satisfiability)
* TIG-GRASP (Test Generation using Generic search Algorithm for satisfiability Problems)

⇒ Heuristic Test set Compaction algorithms

* Compactest
* ROTCO (Reverse order Test Compaction)
* TSC (Test set Compaction
* MTFTG - multiple trget fault test generation
* TBO (Two-by-one)
* min test
* RVE (Redundant vector elimination).

24) a)

* we can enable and disable each Flip flop by using 'selector pin' of FF.
* Als we can use reset and set pins for set FF to known state.
* we can select isolate path from put 'logic'1' to each other pins for AND Gates and other logic 0 to each 'OR gates
* For example lets check $x_1 \rightarrow r_1 \rightarrow Q_2$ that.



In this case, For isolate X_1 we have to put values like this;

$X_0 = 0$

Q_2 Flip Flop reset = 1, set=0, selector = 1

Q_1 Flip Flop reset = 0, set = 1, selector = 1

Q_0 FlipFlop reset = 0, set = 1, selector = 1

So After done this experiment, if Q_2 not using with X_1, the fault will be have in and gate or D flip flop. (stack at 0 or stack at 1).