ELL201

Experiment 6,7 Lab Reports

Name: Raunak Jain

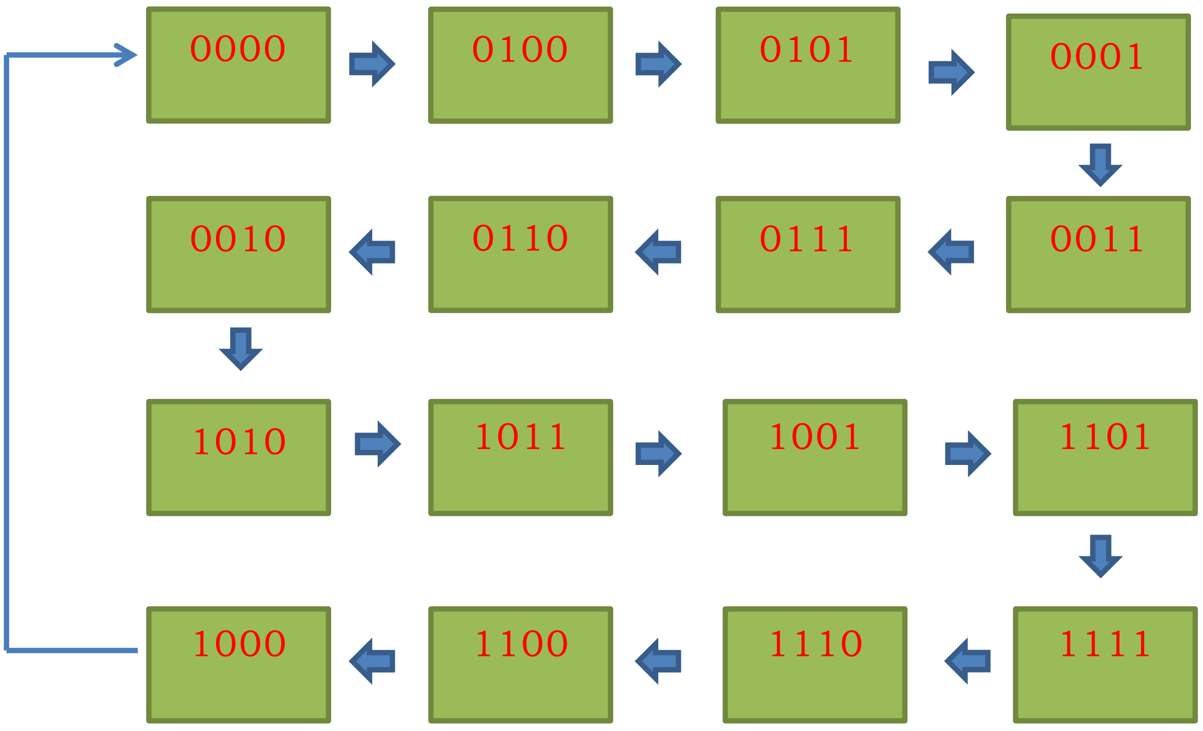
Entry Number: 2019MT10719

Experiment 6

1. Synchronous 4-bit Gray-Code Counter:

- Made counter :

0->4->5->1->3->7->6->2->10->11->9->13->15->14->12->8->0.



State Diagram

State Table:

A_n	B_n	C_n	D_n	A_{n+1}	B_{n+1}	C_{n+1}	D_{n+1}
0	0	0	0	0	1	0	0
0	0	0	1	0	0	1	1
0	0	1	0	1	0	1	0
0	0	1	1	0	1	1	1
0	1	0	0	0	1	0	1
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	1	1	0
1	0	0	0	0	0	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	0	1	1
1	0	1	1	1	0	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	1	1	1
1	1	1	0	1	1	0	0
1	1	1	1	1	1	1	0

- Since, it is a 4-bit counter, so we require **4 SR Flip-Flops**.

- **Values assigned to the inputs of Flip-Flops:**

A	B	C	D	S_A	R_A	S_B	R_B	S_C	R_C	S_D	R_D
0	0	0	0	0	X	1	0	0	X	0	X
0	0	0	1	0	X	0	X	1	0	X	0
0	0	1	0	1	0	0	X	X	0	0	X
0	0	1	1	0	X	1	0	X	0	X	0
0	1	0	0	0	X	X	0	0	X	1	0
0	1	0	1	0	X	0	1	0	X	X	0
0	1	1	0	0	X	0	1	X	0	0	X
0	1	1	1	0	X	X	0	X	0	0	1
1	0	0	0	0	1	0	X	0	X	0	X
1	0	0	1	X	0	1	0	0	X	X	0
1	0	1	0	X	0	0	X	X	0	1	0
1	0	1	1	X	0	0	X	0	1	X	0
1	1	0	0	X	0	0	1	0	X	0	X
1	1	0	1	X	0	X	0	1	0	X	0
1	1	1	0	X	0	X	0	0	1	0	X
1	1	1	1	X	0	X	0	X	0	0	1

- ❖ S_A and R_A are inputs of Flip-Flop A (Outputs: A and A')
- ❖ S_B and R_B are inputs of Flip-Flop B (Outputs: B and B')
- ❖ S_C and R_C are inputs of Flip-Flop C (Outputs: C and C')
- ❖ S_D and R_D are inputs of Flip-Flop D (Outputs: D and D')

- **Karnaugh Maps:**

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

S_A

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

R_A

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

S_B

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

R_B

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

S_C

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

R_C

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

S_D

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

R_D

- <u>**Minimized Expressions for inputs are:**</u>

❖ For SR Flip-Flop which gives A,A’ as outputs:

S_A = B’CD’ and R_A = B’C’D’

❖ For SR Flip-Flop which gives B,B’ as outputs:

S_B = A’C’D’+A’CD+AC’D and R_B = AC’D’+A’C’D+A’CD’

❖ For SR Flip-Flop which gives C,C’ as outputs:

S_C = A’B’D+ABD and R_C = AB’D+ABD’

❖ For SR Flip-Flop which gives D,D’ as outputs:

S_D = A’BC’+AB’C and R_D = BC

- **<u>Verilog Code:</u>**

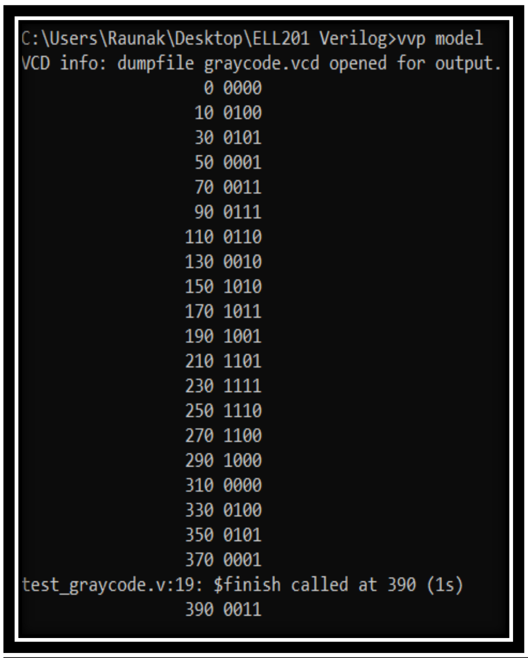
❖ Generating Counter code:

```
module graycode(A,B,C,D,Acomp,Bcomp,Ccomp,Dcomp,clk,rst);

input clk,rst;
output A,B,C,D,Acomp,Bcomp,Ccomp,Dcomp;

wire SA,RA,SB1,SB2,SB3,SB,RB1,RB2,RB3,RB,SC1,SC2,SC,RC1,RC2,RC,SD1,SD2,SD,RD;

and(SA,Bcomp,C,Dcomp);
and(RA,Bcomp,Ccomp,Dcomp);
and(SB1,Acomp,Ccomp,Dcomp);
and(SB2,Acomp,C,D);
and(SB3,A,Ccomp,D);
or(SB,SB1,SB2,SB3);
and(RB1,A,Ccomp,Dcomp);
and(RB2,Acomp,Ccomp,D);
and(RB3,Acomp,C,Dcomp);
or(RB,RB1,RB2,RB3);
and(SC1,Acomp,Bcomp,D);
and(SC2,A,B,D);
or(SC,SC1,SC2);
and(RC1,A,Bcomp,D);
and(RC2,A,B,Dcomp);
or(RC,RC1,RC2);
and(SD1,Acomp,B,Ccomp);
and(SD2,A,Bcomp,C);
or(SD,SD1,SD2);
and(RD,B,C);

SRFlipFlop SRA(A,Acomp,rst,SA,RA,clk);
SRFlipFlop SRB(B,Bcomp,rst,SB,RB,clk);
SRFlipFlop SRC(C,Ccomp,rst,SC,RC,clk);
SRFlipFlop SRD(D,Dcomp,rst,SD,RD,clk);

endmodule

module SRFlipFlop(Q,Qcomp,rst,S,R,clk);

input S,R,rst,clk;
output reg Q,Qcomp;

always @(clk | rst)
begin
  if(rst == 1)
    begin
      Q = 0;
      Qcomp = 1;
    end
end

always @(posedge clk)
begin
   Q = S|(~R&Q);
   Qcomp = ~Q;
end

endmodule
```

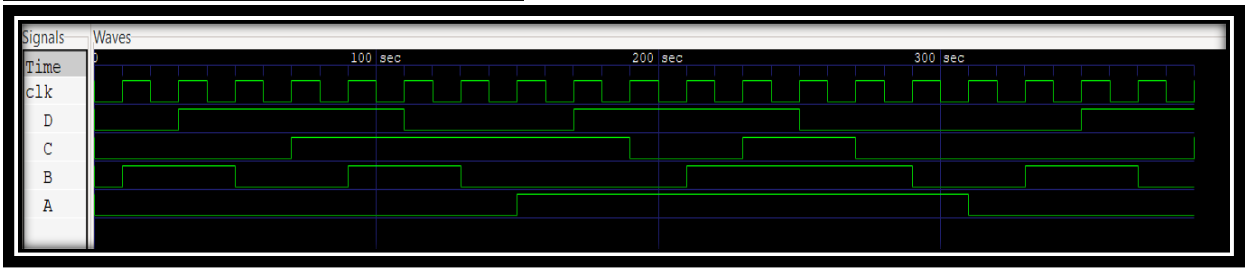
❖ Testing Code:

```
module test_graycode;

reg clk,rst;
wire A,Acomp,B,Bcomp,C,Ccomp,D,Dcomp;

graycode DUT(A,B,C,D,Acomp,Bcomp,Ccomp,Dcomp,clk,rst);

always #10 clk = ~clk;

initial
 begin
  $dumpfile("graycode.vcd");
  $dumpvars(0,test_graycode);
  $monitor ($time," %b%b%b%b", A, B, C, D);
  clk = 0;
  rst = 1;
  #2; rst = 0;
  repeat(20) @(posedge clk);
  $finish;
 end

endmodule
```

❖ Output:



```
C:\Users\Raunak\Desktop\ELL201 Verilog>vvp model
VCD info: dumpfile graycode.vcd opened for output.
                   0 0000
                  10 0100
                  30 0101
                  50 0001
                  70 0011
                  90 0111
                 110 0110
                 130 0010
                 150 1010
                 170 1011
                 190 1001
                 210 1101
                 230 1111
                 250 1110
                 270 1100
                 290 1000
                 310 0000
                 330 0100
                 350 0101
                 370 0001
test_graycode.v:19: $finish called at 390 (1s)
                 390 0011
```



2. Synchronous Ring Counter:

- We require **4 D Flip-Flops** as there are 4 outputs (Q_0,Q_1,Q_2,Q_3).

- **Values assigned to the inputs:**

Q_0	Q_1	Q_2	Q_3	D_0	D_1	D_2	D_3
0	0	0	0	X	X	X	X
0	0	0	1	0	0	0	1
0	0	1	0	1	0	0	1
0	0	1	1	1	0	0	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	1
0	1	1	0	1	1	0	1
0	1	1	1	1	1	0	0
1	0	0	0	0	0	1	0
1	0	0	1	0	0	1	1
1	0	1	0	1	0	1	1
1	0	1	1	1	0	1	0
1	1	0	0	0	1	1	0
1	1	0	1	0	1	1	1
1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	0

- **Karnaugh Maps:**

Q_3Q_2 \ Q_1Q_0	**00**	**01**	**11**	**10**
00	X	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

D_0

Q_3Q_2 \ Q_1Q_0	**00**	**01**	**11**	**10**
00	X	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

D_1

Q_3Q_2 \ Q_1Q_0	**00**	**01**	**11**	**10**
00	X	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

D_2

Q_3Q_2 \ Q_1Q_0	**00**	**01**	**11**	**10**
00	X	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

D_3

On Minimizing Input Expression using Karnaugh Map:

- **$D_0 = Q_1$**
- **$D_1 = Q_2$**
- **$D_2 = Q_3$**
- **$D_3 = Q_1'Q_0+Q_1Q_0' = Q_1$ xor Q_0**

- My Entry Number is 2019MT10719 and hence, X4 = 9 which is 1001. So, my counter will start from 1001 i.e. $Q_3 = 1$, $Q_2 = 0$, $Q_1 = 0$ and $Q_0 = 1$.

- <u>**Verilog Code:**</u>

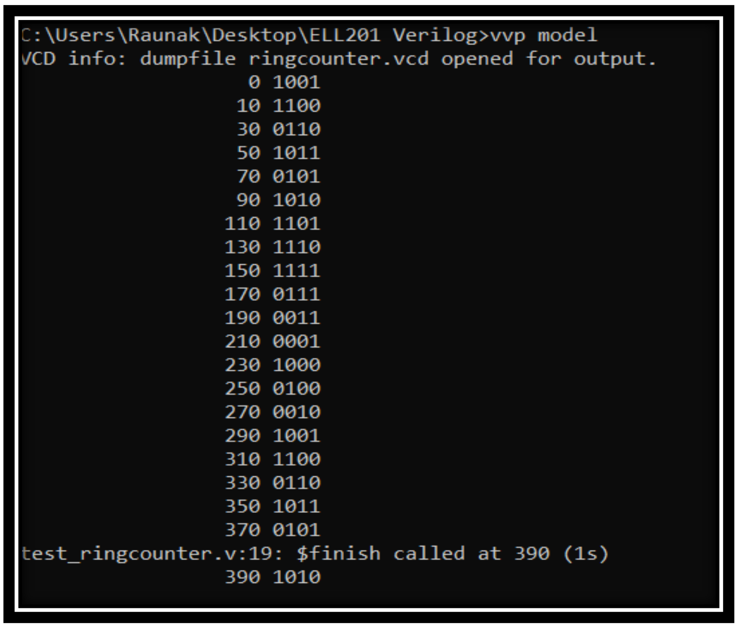
❖ Ring counter Code:

```
module ringcounter(Q0,Q1,Q2,Q3,Q0comp,Q1comp,Q2comp,Q3comp,clk,rst);

input clk,rst;
output Q0,Q1,Q2,Q3,Q0comp,Q1comp,Q2comp,Q3comp;
wire D0,D1,D2,D3,a,b;

assign a = 1;
assign b = 0;

buf(D0,Q1);
buf(D1,Q2);
buf(D2,Q3);
xor(D3,Q0,Q1);

DFlipFlop DQ0(Q0,Q0comp,D0,clk,rst,a,b);
DFlipFlop DQ1(Q1,Q1comp,D1,clk,rst,b,a);
DFlipFlop DQ2(Q2,Q2comp,D2,clk,rst,b,a);
DFlipFlop DQ3(Q3,Q3comp,D3,clk,rst,a,b);

endmodule

module DFlipFlop(Q,Qcomp,D,clk,rst,a,b);

input D,clk,rst,a,b;
output reg Q,Qcomp;

always @(clk | rst)
begin
   if(rst == 1)
     begin
       Q = a;
       Qcomp = b;
     end
end

always @(posedge clk)
begin
   Q = D;
   Qcomp = ~Q;
end

endmodule
```

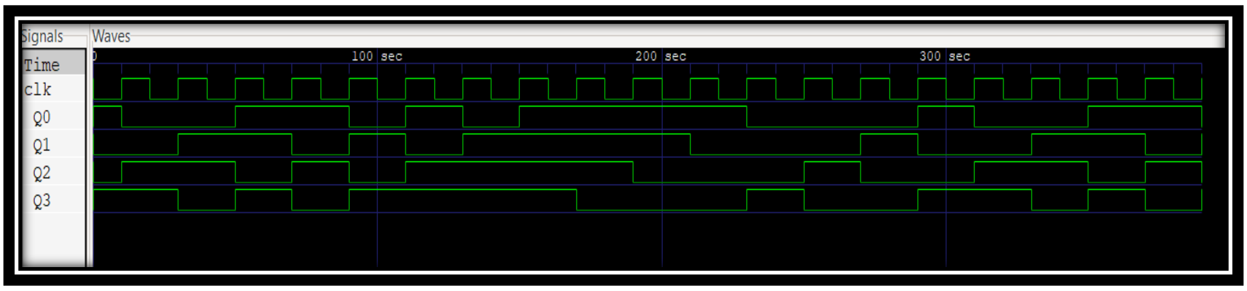
Tester Code:

```
module test_ringcounter;

reg clk,rst;
wire Q0,Q0comp,Q1,Q1comp,Q2,Q2comp,Q3,Q3comp;

ringcounter DUT(Q0,Q1,Q2,Q3,Q0comp,Q1comp,Q2comp,Q3comp,clk,rst);

always #10 clk = ~clk;

initial
 begin
  $dumpfile("ringcounter.vcd");
  $dumpvars(0,test_ringcounter);
  $monitor ($time," %b%b%b%b", Q3, Q2, Q1, Q0);
  clk = 0;
  rst = 1;
  #2; rst = 0;
  repeat(20) @(posedge clk);
  $finish;
 end

endmodule
```

Output:



```
C:\Users\Raunak\Desktop\ELL201 Verilog>vvp model
VCD info: dumpfile ringcounter.vcd opened for output.
                     0 1001
                    10 1100
                    30 0110
                    50 1011
                    70 0101
                    90 1010
                   110 1101
                   130 1110
                   150 1111
                   170 0111
                   190 0011
                   210 0001
                   230 1000
                   250 0100
                   270 0010
                   290 1001
                   310 1100
                   330 0110
                   350 1011
                   370 0101
test_ringcounter.v:19: $finish called at 390 (1s)
                   390 1010
```

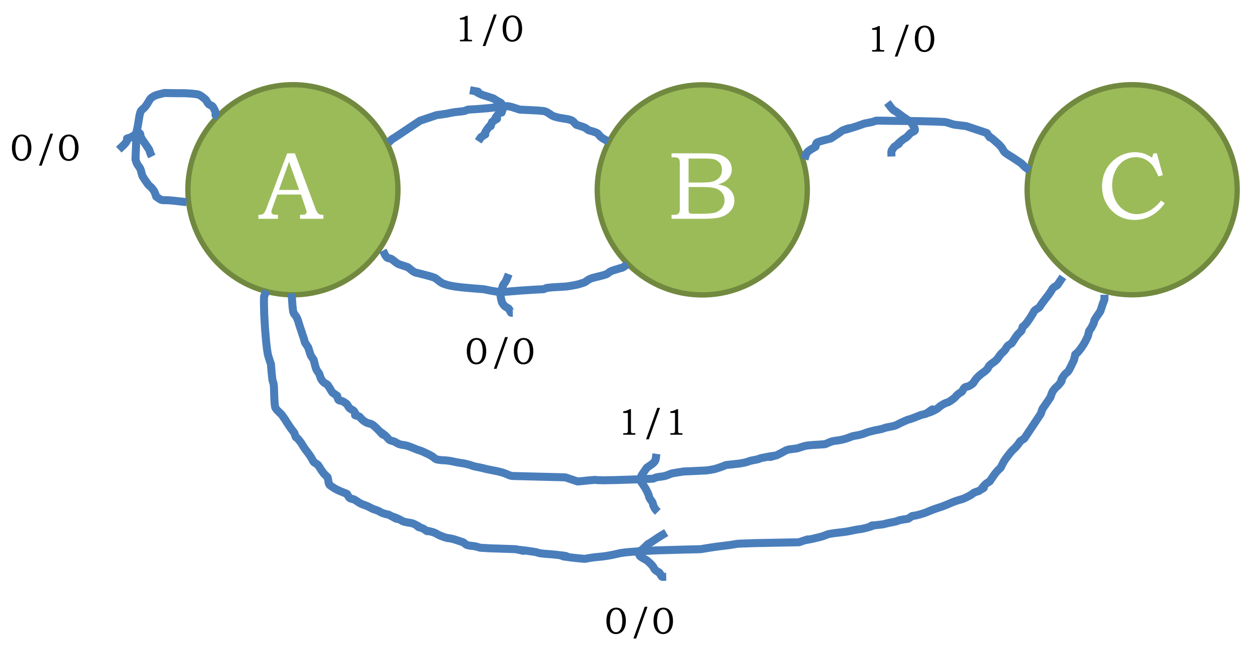


- No, all 16 possible states are not covered here. (0000 is not covered here). Here, ring counter depends on initial state as the counter moves in loop starting from the present state. The above counter works as **Pseudo Random Sequence Generator**.

Experiment 7

- My Entry Number is 2019MT10719 and so, I have to generate sequence 0,0,1,0,0,1.

- **State Diagram:**



- Here as there are 3 states, so we require at least **2 D Flip-Flops**.

- Values assigned to the states:

❖ **A = 00**
❖ **B = 01**
❖ **C = 10**

- **State Table:**

A_n	B_n	X	A_{n+1}	B_{n+1}	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	X	X	X
1	1	1	X	X	X

A_n	B_n	X	D_A	D_B
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	X	X
1	1	1	X	X

- **Karnaugh Maps:**

BX A	00	01	11	10
0	0	0	1	0
1	0	0	X	X

D_A

BX A	00	01	11	10
0	0	1	0	0
1	0	0	X	X

D_B

BX A	00	01	11	10
0	0	0	0	0
1	0	1	X	X

Y

Minimized Expressions are:

- **D_A = BX**
- **D_B = A’B’X**
- **Y = AX**

- **Verilog Code:**

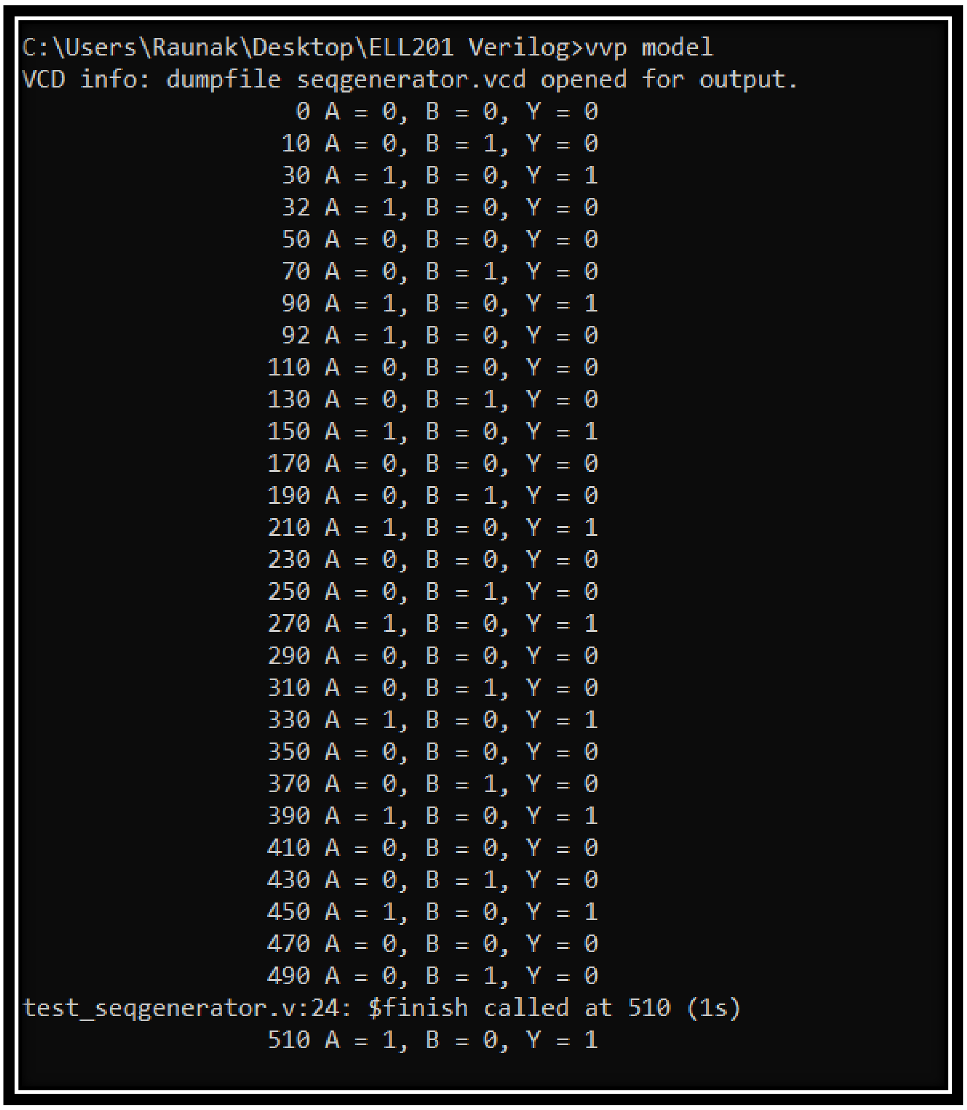
❖ Code for FSM :

```
module seqgenerator(A,B,Acomp,Bcomp,Y,X,clk,rst);

input clk,rst,X;
output A,B,Acomp,Bcomp,Y;
wire DA,DB;

and(DA,B,X);
and(DB,Acomp,Bcomp,X);
and(Y,A,X);

DFlipFlop DF1(A,Acomp,DA,clk,rst);
DFlipFlop DF2(B,Bcomp,DB,clk,rst);

endmodule

module DFlipFlop(Q,Qcomp,D,clk,rst);

input D,clk,rst;
output reg Q,Qcomp;

always @(clk | rst)
begin
   if(rst == 1)
     begin
       Q = 0;
       Qcomp = 1;
     end
end

always @(posedge clk)
begin
   Q = D;
   Qcomp = ~Q;
end
```

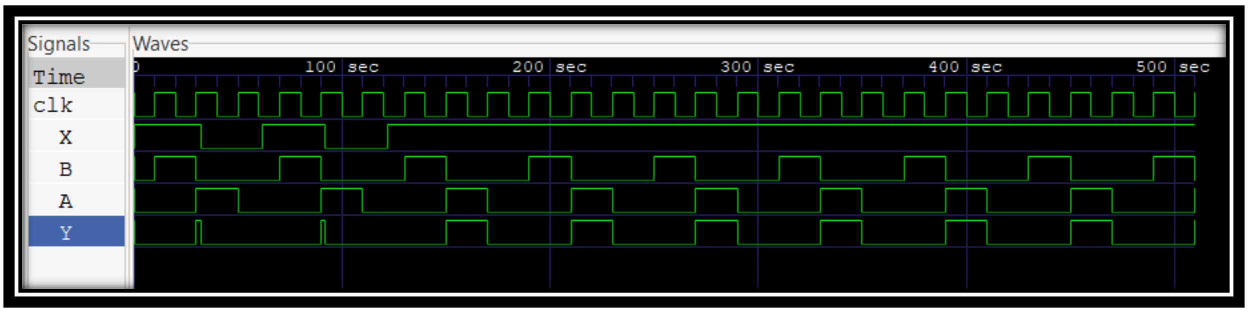
❖ Tester Code:

```
module test_seqgenerator;

reg clk,rst,X;
wire A,Acomp,B,Bcomp,Y;

seqgenerator DUT(A,B,Acomp,Bcomp,Y,X,clk,rst);

always #10 clk = ~clk;

initial
 begin
  $dumpfile("seqgenerator.vcd");
  $dumpvars(0,test_seqgenerator);
  $monitor ($time," A = %b, B = %b, Y = %b", A,B, Y);
  clk = 0;
  rst = 1;
  X = 1;
  #2; rst = 0;
  #30; X = 0;
  #30; X = 1;
  #30; X = 0;
  #30; X = 1;
  repeat(20) @(posedge clk);
  $finish;
 end

endmodule
```

❖ Output:



```
C:\Users\Raunak\Desktop\ELL201 Verilog>vvp model
VCD info: dumpfile seqgenerator.vcd opened for output.
                   0 A = 0, B = 0, Y = 0
                  10 A = 0, B = 1, Y = 0
                  30 A = 1, B = 0, Y = 1
                  32 A = 1, B = 0, Y = 0
                  50 A = 0, B = 0, Y = 0
                  70 A = 0, B = 1, Y = 0
                  90 A = 1, B = 0, Y = 1
                  92 A = 1, B = 0, Y = 0
                 110 A = 0, B = 0, Y = 0
                 130 A = 0, B = 1, Y = 0
                 150 A = 1, B = 0, Y = 1
                 170 A = 0, B = 0, Y = 0
                 190 A = 0, B = 1, Y = 0
                 210 A = 1, B = 0, Y = 1
                 230 A = 0, B = 0, Y = 0
                 250 A = 0, B = 1, Y = 0
                 270 A = 1, B = 0, Y = 1
                 290 A = 0, B = 0, Y = 0
                 310 A = 0, B = 1, Y = 0
                 330 A = 1, B = 0, Y = 1
                 350 A = 0, B = 0, Y = 0
                 370 A = 0, B = 1, Y = 0
                 390 A = 1, B = 0, Y = 1
                 410 A = 0, B = 0, Y = 0
                 430 A = 0, B = 1, Y = 0
                 450 A = 1, B = 0, Y = 1
                 470 A = 0, B = 0, Y = 0
                 490 A = 0, B = 1, Y = 0
test_seqgenerator.v:24: $finish called at 510 (1s)
                 510 A = 1, B = 0, Y = 1
```


Signals
Waves
100 sec
200 sec
300 sec
400 sec
500 sec
Time
clk
X
B
A
Y