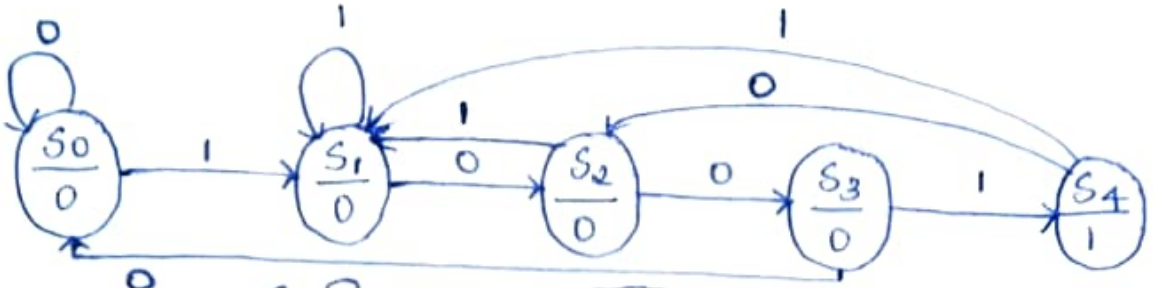
Moore Machine (overlapping)

1 0 0 1

4 + 1 ⇒ 5 states



$S_1 = \{1, 11\}$

$S_2 = \{10, 101\}$

$S_3 = \{100, 1000\}$

$S_4 = \{1001, 0011\}$

1 0 0 **1 0** → S_2

RTL CODE:

```
//-----------------------------Moore OverLapping_1001 -----------------------------
//                                                                                --
// VLSI Aspirant : RAHUL REDDY DB                                                 --
//                RTL                                                             --
// Learn without limitations...                                                   --
//                                                                                --
//--------------------------------------------------------------------------------

// Code your design here
module e_ee (clk,rst,in,out);

  //inputs and outputs
  input clk,rst,in;
  output reg out;

  //Parameters
  parameter s0=3'b000,
            s1=3'b001,
            s2=3'b010,
            s3=3'b011,
            s4=3'b100;

  //internal reg
  reg [2:0]state;

  //sequential always block
  always @(posedge clk or posedge rst)
      if(rst)
        begin
         out<=1'b0;
          state<=s0;
        end
      else
        begin
          case(state)
            s0:begin
              out<=1'b0;
              if(in)
                state<=s1;
              else
                state<=s0;
            end

            s1:begin
              out<=1'b0;
              if(~in)
                state<=s2;
              else
                state<=s1;
            end

            s2:begin
              out<=1'b0;
              if(~in)
                state<=s3;
              else
                state<=s1;
            end
```

```
59
60              s3:begin
61                out<=1'b0;
62                if(in)
63                  state<=s4;
64                else
65                  state<=s0;
66              end
67
68
69              s4:begin
70                out<=1'b1;
71                if(in)
72                  state<=s1;
73                else
74                  state<=s2;
75              end
76            endcase
77
78          end
79  endmodule
```

TB CODE:

testbench.sv

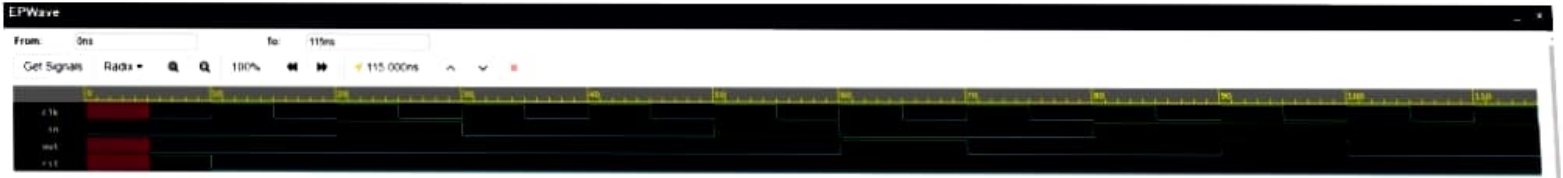
```
//-----------------------------Moore OverLapping_1001 -------------------------------
//                                                                                  --
// VLSI Aspirant : RAHUL REDDY DB                                                   --
//                TB                                                                --
// Learn without limitations...                                                     --
//                                                                                  --
//----------------------------------------------------------------------------------


// Code your TB here
module e_ee_tb ();

  //set up variables
  reg clk,rst,in;
  wire out;

// Instantiate the moore over lapping rtl
  e_ee DUT (.clk(clk),.rst(rst),.in(in),.out(out));

  //clock generation
  always begin
    #5 clk=1'b0;
    #5 clk=~clk;
  end

  //stimulate inputs
  initial begin
    in=1'b0;
    #5 rst=1'b1;
    #5 rst=1'b0;
    #10 in=1'b1;
    #10 in=1'b0;
    #10 in=1'b0;
    #10 in=1'b1;
    #10 in=1'b0;
    #10 in=1'b0;
    #10 in=1'b1;
    #10 in=1'b1;
  end

  //set up the monitoring fot the signal values
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
    $monitor ("[%0t]:  clk=%b, rst=%b,in=%b, out=%b",$time,clk,rst,in,out);
    #120 $finish();
  end
endmodule
```

Result from Display statement:

Log | Share

```
1 module and 0 UDP read.
recompiling module e_ee_tb
rm -f _cuarc*.so _csrc*.so pre_vcsobj_*.so share_vcsobj_*.so
if [ -x ../simv ]; then chmod a-x ../simv; fi
g++  -o ../simv      -rdynamic  -Wl,-rpath='$ORIGIN'/simv.daidir -Wl,-rpath=./simv.daidir -Wl,-r
../simv up to date
CPU time: .353 seconds to compile + .344 seconds to elab + .362 seconds to link
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP2_Full64; Runtime version U-2023.03-SP2_Full64;  Aug  8 05:58 2024
[0]:  clk=x, rst=x,in=0, out=x
[5]:  clk=0, rst=1,in=0, out=0
[10]:  clk=1, rst=0,in=0, out=0
[15]:  clk=0, rst=0,in=0, out=0
[20]:  clk=1, rst=0,in=1, out=0
[25]:  clk=0, rst=0,in=1, out=0
[30]:  clk=1, rst=0,in=0, out=0
[35]:  clk=0, rst=0,in=0, out=0
[40]:  clk=1, rst=0,in=0, out=0
[45]:  clk=0, rst=0,in=0, out=0
[50]:  clk=1, rst=0,in=1, out=0
[55]:  clk=0, rst=0,in=1, out=0
[60]:  clk=1, rst=0,in=0, out=1
[65]:  clk=0, rst=0,in=0, out=1
[70]:  clk=1, rst=0,in=0, out=0
[75]:  clk=0, rst=0,in=0, out=0
[80]:  clk=1, rst=0,in=1, out=0
[85]:  clk=0, rst=0,in=1, out=0
[90]:  clk=1, rst=0,in=1, out=1
[95]:  clk=0, rst=0,in=1, out=1
[100]:  clk=1, rst=0,in=1, out=0
[105]:  clk=0, rst=0,in=1, out=0
[110]:  clk=1, rst=0,in=1, out=0
[115]:  clk=0, rst=0,in=1, out=0
$finish called from file "testbench.sv", line 46.
$finish at simulation time                  120
          V C S   S i m u l a t i o n   R e p o r t
Time: 120 ns
CPU Time:      0.360 seconds;       Data structure size:   0.0Mb
Thu Aug  8 05:58:52 2024
Done
```

EP Wave form:


EPWave
From:
0ns
to:
115ns
Get Signals
Radix
100%
115.000ns
clk
in
out
Note: To revert to EPWave opening in a new browser window, set that option on your profile page