Some sample submission for “PARITY CHECKER”

Verilog file and Testbench

Sample format for submission:

Student Name= _____ Roll No.= _____

Verilog file:

Test bench

Name: Harsh Motwani

Roll Number: 180101028

Moore code

```
module counter(
   input a,
   input clk,
    output o

   );

    reg currstate;
    reg nextstate;
    reg outputreg;

    assign o = outputreg;

    initial
```

```
begin
  currstate = 1'b0;
  nextstate = 1'b0;
  outputreg=0;

end

always @(posedge clk)
begin

  currstate=nextstate;

end

always @(a, currstate)
begin


  nextstate=(~currstate)&a;

end

always @(currstate)
begin
  outputreg=currstate;
end
```

```
endmodule
```

Moore testbench

```
module counter_test;

   // Inputs
   reg a;
   reg clk;

   // Outputs
   wire o;

   // Instantiate the Unit Under Test (UUT)
   counter uut (
      .a(a),
      .clk(clk),
      .o(o)
   );

   integer i;

   initial begin
      // Initialize Inputs
      a = 0;
      clk = 0;

      // Wait 100 ns for global reset to finish
      #100;

       forever
```

```
        begin

          a=~a;
           for(i=0; i<100; i=i+1)
           begin
           #5
           clk=~clk;
           end


        end




       // Add stimulus here

    end

endmodule
```

Mealy code

```
module parity_checker_mealy(
    input a,
    input clk,
    output o
    );

     reg currentstate;
```

```
    reg nextstate;
    reg outputreg;
    assign o=outputreg;

    initial
    begin
      currentstate=1'b0;
      nextstate=1'b0;

    end

    always @(posedge clk)
    begin
      currentstate=nextstate;
    end

    always @(a, currentstate)
    begin

      outputreg <= (~currentstate)&a;
      nextstate <= (~currentstate)&a;

    end



endmodule
```

Mealy testbench

```
module mealy_test;
```

```
// Inputs
reg a;
reg clk;

// Outputs
wire o;

// Instantiate the Unit Under Test (UUT)
parity_checker_mealy uut (
   .a(a),
   .clk(clk),
   .o(o)
);

integer i;

initial begin
   // Initialize Inputs
   a = 0;
   clk = 0;

   // Wait 100 ns for global reset to finish
   #100;

   forever
    begin

      #10
```

```
        a=~a;
         for(i=0; i<100; i=i+1)
         begin
         #5
         clk=~clk;
         end


      end

      // Add stimulus here

   end

endmodule
```

NAME= S PHANI SAI PRANAV

ROLL NO= 180101070

Verilog File:

Moore Code:

```
module moore_parity(input clk, input reset, input x, output reg parity, output reg next_state
   );
reg cur_state;
parameter s0=0,s1=1;
always @(posedge clk or posedge reset)
if(reset)
   cur_state<=s0;
else
   cur_state <=next_state;
always @ (cur_state)
begin
   if(cur_state == s0)
      parity=0;
   else
      parity=1;
end
always @(cur_state or x)
begin
   next_state=s0;
   if(cur_state==s0 && x)
      next_state=s1;
   if(cur_state==s1 && !x)
      next_state=s1;
end
endmodule
```

Moore Testbench:

```
module moore_parity_test;
```

```
// Inputs
reg clk;
reg reset;
reg x;

// Outputs
wire parity;
wire next_state;

// Instantiate the Unit Under Test (UUT)
moore_parity uut (
   .clk(clk),
   .reset(reset),
   .x(x),
   .parity(parity),
   .next_state(next_state)
);

initial begin
clk=0;
forever #10 clk=~clk;
end
initial begin
   // Initialize Inputs
   reset = 1;
   x = 0;

   // Wait 100 ns for global reset to finish
```

```
        #20;
        reset=0;
        x=1;
       #20;
        // Add stimulus here
        reset=0;
        x=0;
       #20;
        reset=0;
        x=1;
       #20;
        reset=0;
        x=0;
        #20;
    end
endmodule
```

Mealy Parity:

```
module mealy_parity(
    input clk,
    input reset,
    input x,
    output reg parity,
     output reg next_state
    );
reg cur_state;
parameter s0=0,s1=1;
always @(posedge clk or posedge reset)
if(reset)
    cur_state<=s0;
```

```
else
   cur_state<=next_state;
always @(cur_state or x)
begin
   parity=0;
   if(cur_state==s0)
   begin
      if(x)
      begin
         next_state=s1;
         parity=1;
      end
      else
         next_state=s0;
   end
   else
   begin
      if(x)
         next_state=s0;
      else
      begin
         next_state=s1;
         parity=1;


      end
   end
end
Endmodule
```

Mealy test bench:

```
module mealy_parity_test;

   // Inputs
   reg clk;
   reg reset;
   reg x;

   // Outputs
   wire parity;
   wire next_state;

   // Instantiate the Unit Under Test (UUT)
   mealy_parity uut (
      .clk(clk),
      .reset(reset),
      .x(x),
      .parity(parity),
      .next_state(next_state)
   );


   initial begin
   clk=0;
   forever #10 clk=~clk;
   end
   initial begin
      // Initialize Inputs
      reset = 1;
      x = 0;
```

```
        // Wait 100 ns for global reset to finish
        #20;
        reset=0;
        x=1;
      #20;
        // Add stimulus here
        reset=0;
        x=0;
      #20;
        reset=0;
        x=1;
      #20;
        reset=0;
        x=0;
        #20;
    end

endmodule
```

Name: Kousik Rajesh

Roll no: 180101094

Moore

```
module moore_parity(input clk, input reset, input x, output reg parity, output reg next_state
   );
reg cur_state;
parameter s0=0,s1=1;
always @(posedge clk or posedge reset)
if(reset)
   cur_state<=s0;
else
   cur_state <=next_state;
always @ (cur_state)
begin
   if(cur_state == s0)
      parity=0;
   else
      parity=1;
end
always @(cur_state or x)
begin
   next_state=s0;
   if(cur_state==s0 && x)
```

```
      next_state=s1;
   if(cur_state==s1 && !x)
      next_state=s1;
end
endmodule
```

Mealy

```
module paritychecker(
 input bitv,
 input clk,
 output reg ans
);
 reg next,state;
 always @(posedge clk)
   begin
    state=next;
   end
 always @( bitv or state)
   begin
    next=0;
    if(bitv==1 && state==0) next=1;
    if(bitv==0 && state==1) next=1;

    case(next)
     0:ans=0;
     1:ans=1;
    endcase
   End
```

Testbench

```
module paritycheckerTb;
```

```
 wire ans;
 reg bitv;
 reg clk;
 paritychecker parch(bitv, clk, ans);
 //enabling the wave dump
 initial begin
  $dumpfile("dump.vcd"); $dumpvars;
 end
  initial begin
    clk=0;
    bitv=0;
    end
  initial
 begin
  bitv = 0; #10;

  $display("Parity\t is %b with %b", ans,bitv);
  bitv = 0; #10;
  $display("Parity\t is %b with %b", ans,bitv);
  bitv = 1; #10;
  $display("Parity\t is %b with %b", ans,bitv);
  bitv = 0; #10;
  $display("Parity\t is %b with %b", ans,bitv);
  bitv = 0; #10;
  $display("Parity\t is %b with %b", ans,bitv);
  bitv = 1; #10;
  $display("Parity\t is %b with %b", ans,bitv);
  bitv = 1; #10;
  $display("Parity\t is %b with %b", ans,bitv);
  bitv = 0; #10;
  $finish;

 end
 always

   #5  clk =  ! clk;


endmodule
```

Student Name = Pranav Gupta Roll No. = 180101058

Verilog file: Moore

```
module moore_3processes(input clk, input reset, input x, output reg parity);
reg state, nextstate;
parameter S0=0, S1=1;
always @(posedge clk or posedge reset)
if (reset)
state <= S0;
else
state <= nextstate;
always @(state)
begin
case(state)
S0: parity = 0;
S1: parity = 1;
endcase
end
always @(state or x)
begin
nextstate = S0;
case(state)
S0: if(x)
nextstate = S1;
S1: if(!x)
nextstate = S1;
endcase
end
        endmodule
```

Verilog file: Mealy

```
module mealy_2processes(input clk, input reset, input x, output reg parity);
reg state, nextstate;
parameter S0=0, S1=1;
always @(posedge clk or posedge reset)
if (reset)
state <= S0;
else
state <= nextstate;
always @(posedge clk or posedge reset)
begin
parity = 1'b0;
case(state)
S0: if(x)
begin
parity = 1; nextstate = S1;
end
else
nextstate = S0;
S1: if(x)
nextstate = S0;
else
begin
parity = 1; nextstate = S1;
end
default:
nextstate = S0;
endcase
end
```

```
endmodule
```

Test bench : Moore

```
`timescale 1s/100ms
`include "ass6_moore.v"

module ass6_tb();
   reg clk;
   reg reset;
   reg x;
   wire parity;

moore_3processes iass6(clk, reset, x, parity);



initial begin
   clk=0;
forever begin
   #20 clk = ~clk;
end
end

initial
begin
   $monitor("reset=%b, x=%b, parity=%b", reset, x, parity);
   #100 reset = 0; x = 1;
   #100 reset = 0; x = 0;
   #100 reset = 0; x = 0;
   #100 reset = 0; x = 0;
```

```
end
        endmodule
```

Test bench : Mealy

```
`timescale 1s/100ms
`include "ass6_mealy.v"

module ass6_tb();
   reg clk;
   reg reset;
   reg x;
   wire parity;

mealy_2processes iass6(clk, reset, x, parity);



initial begin
   clk=0;
forever begin
   #20 clk = ~clk;
end
end

initial
begin
   $monitor("reset=%b, x=%b, parity=%b", reset, x, parity);
   #50 reset = 0; x = 1;
   #50 reset = 0; x = 1;
```

```
    #50 reset = 0; x = 0;
    #50 reset = 0; x = 0;

end
endmodule
```

Student Name= Bhasker Goel Roll No.= 180101015

Moore Verilog file:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:         IIT Guwahati
// Engineer:        Bhasker Goel
//
// Create Date:    13:37:48 04/05/2020
// Design Name:
// Module Name:    ParityChecker
// Project Name:   Session 6
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
```

```
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module ParityChecker(sequence_in, clock, reset, detector_out);
   input sequence_in;
   input clock;
   input reset;
   output reg detector_out;
   parameter even=1'b0;
   parameter odd=1'b1;
   reg current_state, next_state;

   always @(posedge clock, posedge reset)
   begin
      if(reset == 1'b1)
         current_state <= even;
      else
         current_state <= next_state;
   end

   always @(current_state, sequence_in)
   begin
      case(current_state)
         even:begin
            if(sequence_in == 1'b1)
               next_state <= odd;
            else
               next_state <= even;
```

```
        end
        odd:begin
          if(sequence_in == 1'b0)
            next_state <= odd;
          else
            next_state <= even;
        end
        default
          next_state <= even;
      endcase
    end

    always @(current_state)
    begin
      case(current_state)
        even:begin
          detector_out <= even;
        end
        odd:begin
          detector_out <= odd;
        end
        default:detector_out <= even;
      endcase

    end

endmodule
```

Moore Test bench

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   14:07:13 04/05/2020
// Design Name:   ParityChecker
// Module Name:   /home/bhasker/Lab6/tb_parity_checker.v
// Project Name:  Lab6
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: ParityChecker
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module tb_parity_checker;

   // Inputs
   reg sequence_in;
   reg clock;
```

```
reg reset;

// Outputs
wire detector_out;

// Instantiate the Unit Under Test (UUT)
ParityChecker uut (
    .sequence_in(sequence_in),
    .clock(clock),
    .reset(reset),
    .detector_out(detector_out)
);
initial begin
    clock = 0;
    forever #10 clock = ~clock;
end

initial begin
    // Initialize Inputs
    sequence_in = 0;
    reset = 0;

    // Wait 20 ns for global reset to finish
    #20;
  sequence_in = 1;

    // Wait 20 ns for global reset to finish
    #20;
    sequence_in = 0;
```

```
        // Wait 20 ns for global reset to finish
        #20;
        sequence_in = 1;

        // Wait 20 ns for global reset to finish
        #20;
        sequence_in = 1;

        // Wait 20 ns for global reset to finish
        #20;
        sequence_in = 0;

        // Wait 20 ns for global reset to finish
        #20;
        sequence_in = 0;

        // Wait 20 ns for global reset to finish
        #20;
        sequence_in = 1;
    end

Endmodule
```

Verilog file: Mealy

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
```

```
//
// Create Date:    14:28:55 04/05/2020
// Design Name:
// Module Name:    Parity_check_Mealy
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module Parity_check_Mealy(sequence_in, clock, reset, detector_out, current_state, next_state);
   input sequence_in;
   input clock;
   input reset;
   output reg detector_out;
   parameter state_1 = 1'b0;
   parameter state_2 = 1'b1;
   output reg current_state, next_state;

   always @(posedge clock, posedge reset)
   begin
      if(reset == 1'b1) begin
         current_state <= state_1;
```

```
        detector_out <= state_1;
    end else begin
        current_state <= next_state;
        detector_out <= next_state;
    end
end

always @(sequence_in or current_state)
begin
    case(current_state)
        state_1:begin
            if(sequence_in == 1'b0) begin
                next_state <= state_1;
            end else begin
                next_state <= state_2;
            end
        end
        state_2:begin
            if(sequence_in == 1'b0) begin
                next_state <= state_2;
            end else begin
                next_state <= state_1;
            end
        end
        default:begin
            next_state <= state_1;
        end
    endcase
end
```

endmodule

Testbench: Mealy

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   15:07:27 04/05/2020
// Design Name:   Parity_check_Mealy
// Module Name:   /home/bhasker/Lab6/tb_parity_checker_mealy.v
// Project Name:  Lab6
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: Parity_check_Mealy
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module tb_parity_checker_mealy;
```

```
// Inputs
reg sequence_in;
reg clock;
reg reset;

// Outputs
wire detector_out;
wire current_state;
wire next_state;

// Instantiate the Unit Under Test (UUT)
Parity_check_Mealy uut (
   .sequence_in(sequence_in),
   .clock(clock),
   .reset(reset),
   .detector_out(detector_out),
   .current_state(current_state),
   .next_state(next_state)
);

initial begin
   clock = 10;
   forever #10 clock = ~clock;
end

initial begin
   // Initialize Inputs
   sequence_in = 0;
```

```
    // Wait 20 ns for global reset to finish
    #20;
   sequence_in = 1;
    reset = 0;

    // Wait 20 ns for global reset to finish
    #20;
    sequence_in = 0;

    // Wait 20 ns for global reset to finish
    #20;
    sequence_in = 1;

    // Wait 20 ns for global reset to finish
    #20;
    sequence_in = 1;

    // Wait 20 ns for global reset to finish
    #20;
    sequence_in = 1;

    // Wait 20 ns for global reset to finish
    #20;
    sequence_in = 1;

    // Wait 20 ns for global reset to finish
    #20;
    sequence_in = 0;
end
```

endmodule

Student Name = Pranav Gupta Roll No. = 180101058

Verilog file: Moore

```
module moore_parity(input clk, input reset, input x, output reg parity);
reg curstate, nextstate;
parameter S0=0, S1=1;
always @(posedge clk or posedge reset)
if (reset)
curstate <= S0;
else
curstate <= nextstate;
always @(curstate)
begin
case(curstate)
S0: parity = 0;
S1: parity = 1;
endcase
end
always @(curstate or x)
begin
 case(curstate)
S0: if(x) nextstate = S1;
else nextstate = S0;
 S1: if(!x) nextstate = S1;
```

```
else nextstate = S0;
 endcase
end
endmodule
```

Verilog file: Mealy

```
module mealy_parity(input clk, input reset, input x, output reg parity);
reg curstate, nextstate;
parameter S0=0, S1=1;
always @(posedge clk or posedge reset)
if (reset)
curstate <= S0;
else
curstate <= nextstate;
always @(posedge clk or posedge reset)
begin
case(curstate)
S0: if(x)
begin
parity = 1; nextstate = S1;
end
else
begin
parity = 0; nextstate = S0;
end
S1: if(x)
begin
```

```
parity = 0;
nextstate = S0;
end
else
begin
parity = 1; nextstate = S1;
end
default:
begin
nextstate = S0;
parity=0;
end
endcase
end
endmodule
```

Test bench : Moore

```
`timescale 1s/100ms
`include "ass6_moore.v"

module ass6_tb();
   reg clk;
   reg reset;
   reg x;
   wire parity;

moore_parity iass6(clk, reset, x, parity);
initial begin
```

```
   clk=0;
forever begin
   #20 clk = ~clk;
end
end

initial
begin
   $monitor("reset=%b, x=%b, parity=%b", reset, x, parity);

   #100 reset = 1; x = 1;
   #100 reset = 0; x = 1;
   #100 reset = 0; x = 0;
   #100 reset = 0; x = 0;

 end

endmodule
```

Test bench : Mealy

```
`timescale 1s/100ms

`include "ass6_mealy.v"
 module ass6_tb();

   reg clk;
```

```
    reg reset;

    reg x;
     wire parity;
 mealy_parity iass6(clk, reset, x, parity);

initial begin

    clk=0;

forever begin

    #20 clk = ~clk;

end
 end
 initial

begin

    $monitor("reset=%b, x=%b, parity=%b", reset, x, parity);

    #50 reset = 1; x = 1;

    #50 reset = 0; x = 1;

    #50 reset = 0; x = 0;
```

```
   #50 reset = 0; x = 0;
end

endmodule
```

Student Name : Drishti Chouhan **Roll Number : 180101021**

Verilog File : Moore

```
module moore(input clk,input reset,input inp, output reg parity);
 reg curr_state , next_state;
 parameter S0,S1;
 always@(posedge clk or posedge reset)
   if(reset)
   curr_state <= S0;
   else
    curr_state <= next_state;
 always@(curr_state)
   begin
    case(curr_state)
      S0:parity = 0;
      S1:parity = 1;
    endcase
   end;
 always@(curr_state or inp)
 begin
   next_state = S0;
   case(curr_state)
    S0:if(inp)
      nextstate = S1;
    S1:if(!inp)
```

```
      nextstate = S0;
   endcase
  end
endmodule
```

Test Bench : Moore

```
module moore;
 // Inputs

   reg clk;
   reg reset;
   reg inp;




   // Outputs

   wire parity;
   wire next_state;




   // Instantiate the Unit Under Test (UUT)

   moore_parity uut (

      .clk(clk),
      .reset(reset),
      .inp(inp),
```

```
    .parity(parity),
    .next_state(next_state)

);



initial begin
clk=0;
forever #10 clk=~clk;
end

initial begin

    // Initialize Inputs

    reset = 1;
    inp = 0;

    // Wait 100 ns for global reset to finish

    #20;
    reset=0;
    inp=0;

  #20;

    // Add stimulus here
```

```
      reset=0;
      inp=1;

    #20;

      reset=1;
      inp=0;

    #20;

      reset=0;
      inp=0;

    #20;

   end

endmodule
```

Verilog File : Mealy

```
// Code your design here
module mealy(input clk,input reset,input inp, output reg parity , output reg next_state);
  reg curr_state ;
  parameter S0=0 ,S1=1;

  always@(posedge clk or posedge reset)
    if(reset)
      curr_state <= S0;
    else
```

```
   curr_state <= next_state;


always@(curr_state or inp)
 begin
  parity = 1'b0;
  case(curr_state)
   S0: if(inp)
       begin
        parity=1;
        next_state = S1;
       end
       else
       begin
        next_state = S0;
       end
   S1: if(inp)
       begin
        //parity=0;
        next_state = S0;
       end;
       else
       begin
        next_state = S1;
        parity = 1;
       end
   default:
    next_state = S0;
  endcase
 end
```

```
endmodule
```

Test Bench : Mealy

```
module mealy;
 // Inputs

    reg clk;
    reg reset;
     reg inp;
// Outputs

    wire parity;
     wire next_state;
 // Instantiate the Unit Under Test (UUT)

    mealy_parity uut (

        .clk(clk),
       .reset(reset),
      .inp(inp),
      .parity(parity),
     .next_state(next_state)
 );
 initial begin
 Clk=0;
```

```
forever #10 clk=~clk;
end
 initial begin

  // Initialize Inputs

       reset = 1;
      inp = 0;
     // Wait 100 ns for global reset to finish

       #20;

       reset=0;
      inp=0;

     #20;

       // Add stimulus here

       reset=0;
       inp=1;

     #20;

       reset=0;
      inp=0;

     #20;
```

```
        reset=1;
        inp=0;

        #20;

    end
Endmodule
```

Name: Kartikeya Saxena

Roll Number: 180101034

Moore Code

```
`timescale 1ns / 1ns
module moore_parity_checker(clk, rst, in, out, state);
input clk, rst, in;
output reg out, state;


parameter zero = 1'b0,
one = 1'b1;
reg next_state;
always@(posedge clk or posedge rst)
begin
if(rst)
state <= zero;
else
begin
state <= next_state;
end
end


always@(in or state)
begin
if(state == one)
begin
if(in == 1)
```

```
next_state <= zero;
else
next_state <= one;
end
else
begin
if(in == 1)
next_state <= one;
else
next_state <= zero;
end
end

always@(state)
begin
out = state;
end
endmodule

`timescale 1ns / 1ns
module moore_parity_checker_test_bench();
reg clk, in, rst;
wire out, state;

moore_parity_checker m(.clk(clk), .in(in), .out(out), .state(state), .rst(rst));
```

```
initial
begin
$dumpfile("output.vcd");
$dumpvars(0, m);


clk = 1'b0;
forever
#10 clk = ~clk;
end


initial
begin
$monitor("time = %3d, state = %d, in = %d, out = %d\n", $time, state, in, out);
end


initial
Begin`
rst = 1;
in = 0;
#30 rst = 0;
#10 in = 1;
#11 in = 0;
#10 in = 1;
#10 in = 1;
#10 in = 0;
#10 $finish;
end
```

```
endmodule
```

Mealy Code

```
`timescale 1ns / 1ns
module mealy_parity_checker(clk, rst, in, out, state);
input clk, rst, in;
output reg out, state;


parameter zero = 1'b0,
one = 1'b1;
reg next_state;
always@(posedge clk or posedge rst)
begin
if(rst)
state <= zero;
else
begin
state <= next_state;
end
end


always@(in or state)
begin
if(state == one)
begin
if(in == 1)
begin
next_state <= zero;
out <= 0;
```

```
end
else
begin
next_state <= one;
out <= 1;
end
end
else
begin
if(in == 1)
begin
next_state <= one;
out <= 1;
end
else
begin
next_state <= zero;
out <= 0;
end
end
end
endmodule
`timescale 1ns / 1ns
module mealy_parity_checker_test_bench();
reg clk, in, rst;
wire out, state;

mealy_parity_checker m(.clk(clk), .in(in), .out(out), .state(state), .rst(rst));
```

```
initial
begin
$dumpfile("output.vcd");
$dumpvars(0, m);

clk = 1'b0;
forever
#10 clk = ~clk;
end

initial
begin
$monitor("time = %3d, state = %d, in = %d, out = %d\n", $time, state, in, out);
end

initial
begin
rst = 1;
in = 0;
#30 rst = 0;
#10 in = 1;
#11 in = 0;
#10 in = 1;
#10 in = 1;
#10 in = 0;
```

```
#10 $finish;
end
endmodule
```