`include "FSM1.v"

module testbench();

reg clock, reset, x;

wire z;

FSM1 FSM(clock,reset,x,z);

initial begin

$monitor("%4d: z = %b", $time, z);

clock = 0;

reset = 1;

x = 0;

#10 reset = 0;

end

always

begin #5clock = ~clock;

end

initial begin

#10 x = 1; $display("%4d: x = %b", $time, x);

#10 x = 0; $display("%4d: x = %b", $time, x);

#10 x = 0; $display("%4d: x = %b", $time, x);

#10 x = 1; $display("%4d: x = %b", $time, x);

#10 x = 1; $display("%4d: x = %b", $time, x);

#10 x = 0; $display("%4d: x = %b", $time, x);

#10 x = 0; $display("%4d: x = %b", $time, x);

#10 x = 1; $display("%4d: x = %b", $time, x);

#10 x = 0; $display("%4d: x = %b", $time, x);

#10 x = 1; $display("%4d: x = %b", $time, x);

#10 x = 0; $display("%4d: x = %b", $time, x);

#10 x = 0; $display("%4d: x = %b", $time, x);

#10 x = 1; $display("%4d: x = %b", $time, x);

#10 x = 0; $display("%4d: x = %b", $time, x);

#10 x = 0; $display("%4d: x = %b", $time, x);

#10 x = 1; $display("%4d: x = %b", $time, x);

#10 x = 0; $display("%4d: x = %b", $time, x);

#10 x = 0; $display("%4d: x = %b", $time, x);

#10 $finish;

end

endmodule

//TESTBENCH

module NSG(

input q1,q0,x,z,

output d1, d0

);

assign d1 = (q1 & ~q0 & ~x) | (~q1 & q0 & ~x);

assign d0 = x | (q1 & ~q0);

endmodule

//NSG

module OG(

input q1,q0,x,

output z

);

assign z = q1 & q0 & x;

endmodule

//OG

module fsm2(

input clock, reset, x,

output reg z

);

parameter A = 2'b00,

B = 2'b01,

C = 2'b10,

D = 2'b11;

reg [1:0] current\_state, next\_state;

always@(\*)

begin

casex(current\_state)

A: if (x == 1) begin

next\_state = B;

z = 0;

end

else begin

next\_state = A;

z = 0;

end

B: if (x == 1) begin

next\_state = B;

z = 0;

end

else begin

next\_state = C;

z = 0;

end

C: if(x == 1) begin

next\_state = B;

z = 0;

end

else begin

next\_state = D;

z = 0;

end

D: if(x == 1) begin

next\_state = B;

z = 1;

end

else begin

next\_state = A;

z = 0;

end

default: begin

next\_state = 2'bxx;

end

endcase

end

always@(posedge clock, posedge reset)

begin

if(reset == 1)

current\_state <= A;

else

current\_state <= next\_state;

end

endmodule //FSM

module FF(

input clock,reset,x

output reg z);

always@(posedge clock, posedge reset)

begin

if (reset == 1)

current\_state<= 0;

else

current\_state<=next\_state;

end

endmodule

//FLIP FLOP