module ff ( input data, input c, input r, output q);

reg q;

always @(posedge c or negedge r)

begin

if(r==1'b0)

q <= 1'b0;

else

q <= data;

end

endmodule //End

// ---- FSM alto nÃ­vel com Case

module statem(input clk, input reset, input [1:0] a, output [2:0] saida);

reg [2:0] state;

parameter zero=3'd0, um=3'd1, dois=3'd2, tres1=3'd3, quatro=3'd4, cinco =3'd5, seis=3'd6, tresdois=3'd7;

assign saida = (state == zero)? 3'd0:

(state == um)? 3'd1:

(state == dois)? 3'd2:

(state == tres1)? 3'd3:

(state == quatro)? 3'd4:

(state == cinco)? 3'd5:

(state == seis)? 3'd6:3'd3;

always @(posedge clk or negedge reset)

begin

if (reset==0)

state = zero;

else

case (state)

zero: if ( a == 2'd0) state = um;

else if ( a == 2'd1 ) state = tresdois;

else if ( a == 2'd2 ) state = um;

else state = cinco;

um: if ( a == 2'd0 ) state = dois;

else if ( a == 2'd1 ) state = tresdois;

else if ( a == 2'b10 ) state = tresdois;

else state = cinco;

dois: if ( (a == 2'd0) ) state = zero;

else if (a == 2'd1 ) state = quatro;

else if (a ==2'd2) state = tres1;

else state = cinco;

tres1: if ((a[1] == 1'b0) && (a[0] == 1'b0) ) state = zero;

else if ((a[1] == 1'b0) && (a[0] == 1'b1)) state = dois;

else if ((a[1] == 1'b1) && (a[0] == 1'b0)) state = um;

else state = cinco;

quatro: if ( a == 2'd0 ) state = zero;

else if ( a == 2'd1 ) state = tresdois;

else if ( a == 2'd2 ) state = um;

else state = cinco;

cinco: if ( a == 2'd0 ) state = zero;

else if ( a == 2'd1 ) state = tres1;

else if ( a == 2'd2 ) state = um;

else state = seis;

seis: if ( a == 2'd0 ) state = zero;

else if ( a == 2'd1 ) state = tres1;

else if ( a == 2'd2 ) state = um;

else state = tres1;

tresdois: if ( a == 2'd0 ) state = zero;

else if ( a == 2'd1 ) state = dois;

else if ( a == 2'd2 ) state = dois;

else state = cinco;

endcase

end

endmodule

// FSM com portas logicas

module statePorta(input clk, input res, input [1:0] a, output [2:0] s);

wire [2:0] e;

wire [2:0] p;

assign s[0] = e[0];

assign s[1] = e[1];

assign s[2] = e[2]&~e[1] | e[2]&~e[0];

assign p[0] = ~e[2]&a[1] | ~e[0]&a[1] | ~e[2]&~e[1]&~e[0] | ~e[1]&~a[1]&a[0] | ~e[1]&a[1]&~a[0] | e[1]&a[1]&a[0] | e[2]&~e[0]&a[0];

assign p[1] = ~e[1]&~a[1]&a[0] | e[0]&~a[1]&a[0] | ~e[2]&~e[1]&e[0]&~a[0] | e[2]&~e[1]&e[0]&a[0] | e[2]&e[1]&~e[0]&a[0] | ~e[2]&e[1]&~e[0]&a[1]&~a[0] | e[2]&e[1]&e[0]&a[1]&~a[0];

assign p[2] = ~e[2]&~e[1]&a[0] | ~e[2]&~e[0]&a[0] | ~e[1]&~e[0]&a[0] | e[0]&a[1]&a[0] | ~e[2]&~e[1]&e[0]&a[1];

ff e0(p[0],clk,res,e[0]);

ff e1(p[1],clk,res,e[1]);

ff e2(p[2],clk,res,e[2]);

endmodule

module stateMem(input clk,input res, input [1:0] a, output [2:0] saida);

reg [5:0] StateMachine [0:31]; // 32 linhas e 6 bits de largura

initial

begin

StateMachine[0] = 6'b001000;

StateMachine[1] = 6'b111000;

StateMachine[2] = 6'b001000;

StateMachine[3] = 6'b101000;

StateMachine[4] = 6'b010001;

StateMachine[5] = 6'b111001;

StateMachine[6] = 6'b111001;

StateMachine[7] = 6'b101001;

StateMachine[8] = 6'b000010;

StateMachine[9] = 6'b100010;

StateMachine[10] = 6'b011010;

StateMachine[11] = 6'b101010;

StateMachine[12] = 6'b000011;

StateMachine[13] = 6'b010011;

StateMachine[14] = 6'b001011;

StateMachine[15] = 6'b101011;

StateMachine[16] = 6'b000100;

StateMachine[17] = 6'b111100;

StateMachine[18] = 6'b001100;

StateMachine[19] = 6'b101100;

StateMachine[20] = 6'b000101;

StateMachine[21] = 6'b011101;

StateMachine[22] = 6'b001101;

StateMachine[23] = 6'b110101;

StateMachine[24] = 6'b000110;

StateMachine[25] = 6'b011110;

StateMachine[26] = 6'b001110;

StateMachine[27] = 6'b011110;

StateMachine[28] = 6'b000011;

StateMachine[29] = 6'b010011;

StateMachine[30] = 6'b010011;

StateMachine[31] = 6'b101011;

end

wire [4:0] address; // 32 linhas = 5 bits de endereco

wire [5:0] dout; // 6 bits de largura 3+3 = proximo estado + saida

assign address[1] = a[1];

assign address[0] = a[0];

assign dout = StateMachine[address];

assign saida = dout[2:0];

ff st0(dout[3],clk,res,address[2]);

ff st1(dout[4],clk,res,address[3]);

ff st2(dout[5],clk,res,address[4]);

endmodule

module main;

reg c,res,a[1:0];

wire [2:0] saida;

wire [2:0] saida1;

wire [2:0] saida2;

statem FSM(c,res,{a[1], a[0]},saida);

//statePorta FSM(c,res,g,r,y);

statePorta FSM3(c,res,{a[1], a[0]},saida2);

stateMem FSM1(c,res,{a[1], a[0]},saida1);

initial

c = 1'b0;

always

c= #(1) ~c;

// visualizar formas de onda usar gtkwave out.vcd

initial begin

$dumpfile ("out.vcd");

$dumpvars;

end

initial

begin

//matricula:78473 bin : 1 00 11 00 10 10 00 10 10

$monitor($time," c %b res %b a1 %b a0 %b scase %d sMEM %d sPORTA %d",c,res,a[1],a[0],saida,saida1,saida2);

#1 res=0; a[1]=0; a[0]=0;

#1 res=1;

#10 a[1]=1; a[0]=0; //10 10

#5 a[1]=0; a[0]=0; //00

#10 a[1]=1; a[0]=0; //10 10

#5 a[1]=0; a[0]=0; //00

#5 a[1]=1; a[0]=1; //11

#5 a[1]=0; a[0]=0; //00

#5 a[1]=0; a[0]=1; //1

#4;

$finish ;

end

endmodule