UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Tarea 2

Diseño y Verificación de Máquinas de Estado Finitos

Sistemas Digitales Programables

Máster Universitario en Ingeniería de Sistemas Electrónicos

2020/2021

Irene Garcia do Amaral

Diseño de un controlador para un motor paso a paso

Implementación en Verilog:

```
module control_motor
(
    input CLK, RESET, UP_DOWN, HALF_FULL, ENABLE,
    output A, B, C, D, INH1, INH2
);

    reg [5:0] Estado_Actual, Estado_Siguiente;
    parameter [5:0] S1=6'b010111, S2=6'b000101, S3=6'b100111, S4=6'b100010, S5=6'b101011, S6=6'b001001, S7=6'b011011, S8=6'b010010;

    assign {A, B, C, D, INH1, INH2} = Estado_Actual;

    always @(posedge CLK, negedge RESET)
    begin
       if(!RESET)
          Estado_Actual <= S1;
       else
          Estado_Actual <= Estado_Siguiente;
    end

    always @(Estado_Actual, ENABLE, UP_DOWN, HALF_FULL)
    begin
       case (Estado_Actual)

          S1:
             if(ENABLE)
                if(HALF_FULL)  //modo FULL
                   if(UP_DOWN)
                      Estado_Siguiente = S2;
                   else
                      Estado_Siguiente = S8;

                else           //modo HALF
                   if(UP_DOWN)
                      Estado_Siguiente = S3;
                   else
                      Estado_Siguiente = S7;

             else
                Estado_Siguiente = S1;


          S2:
             if(ENABLE)
                if(HALF_FULL)  //modo FULL
                   if(UP_DOWN)
                      Estado_Siguiente = S3;
                   else
                      Estado_Siguiente = S1;

                else           //modo HALF
                   if(UP_DOWN)
                      Estado_Siguiente = S4;
                   else
                      Estado_Siguiente = S8;

             else
                Estado_Siguiente = S2;

          S3:
             if(ENABLE)
                if(HALF_FULL)  //modo FULL
                   if(UP_DOWN)
                      Estado_Siguiente = S4;
                   else
                      Estado_Siguiente = S2;

                else           //modo HALF
                   if(UP_DOWN)
                      Estado_Siguiente = S5;
                   else
                      Estado_Siguiente = S1;

             else
                Estado_Siguiente = S3;

          S4:
             if(ENABLE)
                if(HALF_FULL)  //modo FULL
                   if(UP_DOWN)
                      Estado_Siguiente = S5;
                   else
                      Estado_Siguiente = S3;

                else           //modo HALF
                   if(UP_DOWN)
                      Estado_Siguiente = S6;
                   else
                      Estado_Siguiente = S2;

             else
                Estado_Siguiente = S4;
```

```
        S5:
            if(ENABLE)
                if(HALF_FULL)  //modo FULL
                    if(UP_DOWN)
                        Estado_Siguiente = S6;
                    else
                        Estado_Siguiente = S4;

                else           //modo HALF
                    if(UP_DOWN)
                        Estado_Siguiente = S7;
                    else
                        Estado_Siguiente = S3;

            else
                Estado_Siguiente = S5;

        S6:
            if(ENABLE)
                if(HALF_FULL)  //modo FULL
                    if(UP_DOWN)
                        Estado_Siguiente = S7;
                    else
                        Estado_Siguiente = S5;

                else           //modo HALF
                    if(UP_DOWN)
                        Estado_Siguiente = S8;
                    else
                        Estado_Siguiente = S4;

            else
                Estado_Siguiente = S6;

        S7:
            if(ENABLE)
                if(HALF_FULL)  //modo FULL
                    if(UP_DOWN)
                        Estado_Siguiente = S8;
                    else
                        Estado_Siguiente = S6;

                else           //modo HALF
                    if(UP_DOWN)
                        Estado_Siguiente = S1;
                    else
                        Estado_Siguiente = S5;

            else
                Estado_Siguiente = S7;
```

```
        S8:
            if(ENABLE)
                if(HALF_FULL)  //modo FULL
                    if(UP_DOWN)
                        Estado_Siguiente = S1;
                    else
                        Estado_Siguiente = S7;

                else           //modo HALF
                    if(UP_DOWN)
                        Estado_Siguiente = S2;
                    else
                        Estado_Siguiente = S6;

            else
                Estado_Siguiente = S8;

        default: Estado_Siguiente = S1;
    endcase
end

endmodule
```

RTL Viewer:

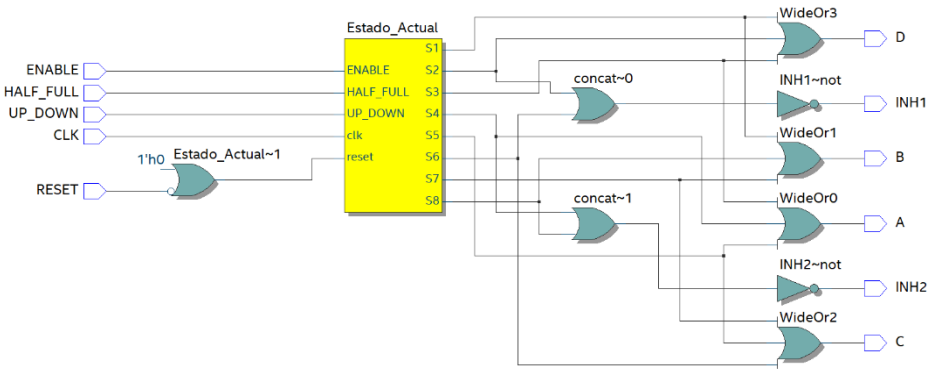

Figura 1: RTL Viewer del controlador para un motor paso a paso

Implementación del Testbench:

Utilizando un testbench han sido realizadas 8 tasks para simulación:

- Test de RESET
- Test en modo HALF, UP e IMPAR
- Test en modo HALF, UP y PAR
- Test en modo HALF, DOWN e IMPAR
- Test en modo HALF, DOWN y PAR
- Test de DISABLE

```
`timescale 1ns/1ps

module tb_control_motor();

localparam T = 20;

reg CLK, RESET, UP_DOWN, HALF_FULL, ENABLE;

wire A, B, C, D, INH1, INH2;

lixo control_motor_inst
(
   .CLK(CLK) , // input  CLK_sig
   .RESET(RESET) ,   // input  RESET_sig
   .UP_DOWN(UP_DOWN) ,  // input  UP_DOWN_sig
   .HALF_FULL(HALF_FULL) , // input  HALF_FULL_sig
   .ENABLE(ENABLE) , // input  ENABLE_sig
   .A(A) ,  // output  A_sig
   .B(B) ,  // output  B_sig
   .C(C) ,  // output  C_sig
   .D(D) ,  // output  D_sig
   .INH1(INH1) ,  // output  INH1_sig
   .INH2(INH2)    // output  INH2_sig
);

defparam control_motor_inst.S1 = 'b010111;
defparam control_motor_inst.S2 = 'b000101;
defparam control_motor_inst.S3 = 'b100111;
defparam control_motor_inst.S4 = 'b100010;
defparam control_motor_inst.S5 = 'b101011;
defparam control_motor_inst.S6 = 'b001001;
defparam control_motor_inst.S7 = 'b011011;
defparam control_motor_inst.S8 = 'b010010;

// defparam [5:0] Estado_Actual, Estado_Siguiente; --------????

initial
begin

   RESET = 1;
   UP_DOWN = 1;       //UP
   HALF_FULL = 0;     //HALF=0, FULL=1
   ENABLE = 0;

   $display("COMIENZA LA SIMULACIÓN!");
   #(T*1);
```

```
    reset(5);
    ModoHalfUpImpar(10);
    ModoHalfUpPar(10);
    ModoHalfDownImpar(10);
    ModoHalfDownPar(10);
    ModoFullUp(10);
    ModoFullDown(10);
    Disable(5);

    $display("FIN DE SIMULACIÓN!");
    $stop;

end

always
   begin
      CLK = 0;
      #(T/2) CLK <= ~CLK;
   end

/* ----------- ZONA DE TASK --------- */

task reset(input integer ciclos_reset); //TEST DE RESET
begin
    $display("Caso RESET (%d ciclos)", ciclos_reset);

    RESET = 0;
    #(T*2);
    RESET = 1;

end
endtask

task ModoHalfUpImpar(input integer ciclos_halfupimpar);  //TEST EN MODO HALF, UP, IMPAR and ENABLE
begin
    $display("Caso 1: TEST EN MODO HALF UP IMPAR (%d ciclos)", ciclos_halfupimpar);

    HALF_FULL = 1'b0;
    UP_DOWN = 1'b1;
    #(T*2) RESET = 1'b1;
    @(negedge CLK) ENABLE = 1'b1;
    #(T*12)
    @(negedge CLK) ENABLE = 1'b0;
    #(T*5);

end
endtask
```

```
task ModoHalfUpPar(input integer ciclos_halfuppar);   //TEST EN MODO HALF, UP, PAR and ENABLE
begin
    $display("Caso 1: TEST EN MODO HALF UP PAR (%d ciclos)", ciclos_halfuppar);

    HALF_FULL = 1'b1;
    UP_DOWN = 1'b0;
    #(T*2) RESET = 1'b1;
    @(negedge CLK) ENABLE = 1'b1;
    #(T*3/4)HALF_FULL = 1'b0;
    #(T*12)
    @(negedge CLK) ENABLE = 1'b0;
    #(T*5);

end
endtask

task ModoHalfDownImpar(input integer ciclos_halfdownimpar); //TEST EN MODO HALF, DOWN, IMPAR and ENABLE
begin
    $display("Caso 1: TEST EN MODO HALF UP IMPAR (%d ciclos)", ciclos_halfdownimpar);

    HALF_FULL = 1'b1;
    UP_DOWN = 1'b0;
    #(T*2) RESET = 1'b1;
    @(negedge CLK) ENABLE = 1'b1;
    #(T*3/4)HALF_FULL = 1'b0;
    #(T*12)
    @(negedge CLK) ENABLE = 1'b0;
    #(T*5);
end
endtask

task ModoHalfDownPar(input integer ciclos_halfdownpar);  //TEST EN MODO HALF, DOWN, PAR and ENABLE
begin
    $display("Caso 1: TEST EN MODO HALF UP PAR (%d ciclos)", ciclos_halfdownpar);

    HALF_FULL = 1'b1;
    UP_DOWN = 1'b0;
    #(T*2) RESET = 1'b1;
    @(negedge CLK) ENABLE = 1'b1;
    #(T*3/4)HALF_FULL = 1'b0;
    #(T*12)
    @(negedge CLK) ENABLE = 1'b0;
    #(T*5);
end
endtask
```

```
task ModoFullUp(input integer ciclos_fullup);   //TEST EN MODO FULL, UP and ENABLE
begin
    $display("Caso 1: TEST EN MODO FULL UP(%d ciclos)", ciclos_fullup);

    HALF_FULL = 1'b1;
    UP_DOWN = 1'b1;
    #(T*2) RESET = 1'b1;
    @(negedge CLK) ENABLE = 1'b1;
    #(T*12)
    @(negedge CLK) ENABLE = 1'b0;
    #(T*5);

end
endtask

task ModoFullDown(input integer ciclos_fulldown);  //TEST EN MODO FULL, DOWN and ENABLE
begin
    $display("Caso 1: TEST EN MODO FULL DOWN (%d ciclos)", ciclos_fulldown);

    HALF_FULL = 1'b1;
    UP_DOWN = 1'b0;
    #(T*2) RESET = 1'b1;
    @(negedge CLK) ENABLE = 1'b1;
    #(T*12)
    @(negedge CLK) ENABLE = 1'b0;
    #(T*5);
end
endtask

task Disable(input integer ciclos_dis); //TEST DISABLE
begin
    $display("Caso DISABLE (%0t ps)", $time);
    RESET = 1'b1;
    ENABLE = 1'b0;
    #(T*ciclos_dis);
end
endtask

endmodule
```

RTL Simulation:

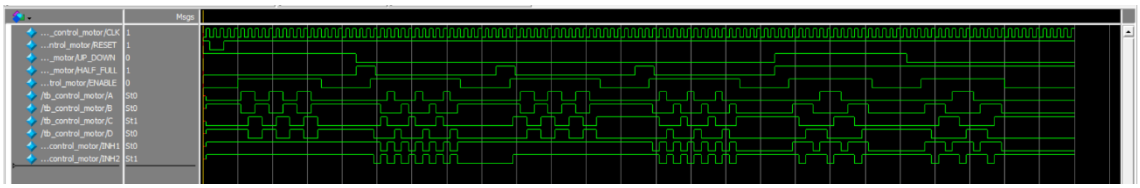

Figura 2: RTL Simulation del controlador para un motor paso a paso con todas las tasks mencionadas arriba

Diseño de una máquina de estados para el juego de luces para el coche fantástico

Implementación en Verilog:

La implementación de *cartop_2* está dividida en 3 bloques:

- Bloco1: contiene el *counter2* y el *FSM_luces_kit_medvedev*
- Bloco2: contiene el *FSM_speed_mealy* y el *contador_up_down*
- Bloco3: *contador_variable*

FSM_luces_kit_medvedev:

```
module FSM_luces_Kit_medvedev (

    input CLK, RESET, ENABLE,
    output [7:0] LEDG
);

    reg [8:0] state;

    parameter [8:0]
        s1 = 9'b010000000,
        s2 = 9'b001000000,
        s3 = 9'b000100000,
        s4 = 9'b000010000,
        s5 = 9'b000001000,
        s6 = 9'b000000100,
        s7 = 9'b000000010,
        s8 = 9'b000000001,
        s9 = 9'b100000010,
        s10 =9'b100000100,
        s11 =9'b100001000,
        s12 =9'b100010000,
        s13 =9'b100100000,
        s14 =9'b101000000;

    always @(posedge CLK or negedge RESET)

    if(!RESET)
        state <= s1;
    else if (ENABLE)
        case (state)
            s1: state<=s2;
            s2: state<=s3;
            s3: state<=s4;
            s4: state<=s5;
            s5: state<=s6;
            s6: state<=s7;
            s7: state<=s8;
            s8: state<=s9;
            s9: state<=s10;
            s10: state<=s11;
            s11: state<=s12;
            s12: state<=s13;
            s13: state<=s14;
            s14: state<=s1;

            default: state<=s1;
        endcase

    assign LEDG = state [7:0];

endmodule
```

Cartop_2:

```
module cartop_2(

    input CLOCK_50,
    input [1:0] SW,
    input [2:0] KEY,

    output [7:0] LEDG,
    output [7:0] LEDR
);

wire counter2_variable;
wire fsm_variable;


bloco1 bloco1_inst
(
    .SW(SW[0]) ,   // input [0:0] SW_sig
    .CLOCK_50(CLOCK_50) ,   // input  CLOCK_50_sig
    .KEY(KEY[0]) , // input [0:0] KEY_sig
    .enable(fsm_variable),
    .LEDG(LEDG),   // output [7:0] LEDG_sig
    .counter2_variable(counter2_variable)
);

bloco2 bloco2_inst
(
    .CLOCK_50(CLOCK_50) ,   // input  CLOCK_50_sig
    .KEY(KEY) , // input [2:0] KEY_sig
    .LEDR(LEDR[3:0])  // output [3:0] LEDR_sig
);

bloco3 bloco3_inst
(
    .clock(CLOCK_50) ,   // input  clock_sig
    .reset(KEY[0]) ,  // input  reset_sig
    .entrada(LEDR[3:0]) ,   // input [width_counter-1:0] entrada_sig
    .enable(counter2_variable) ,  // input  enable_sig
    .modo(SW[1]) , // input  modo_sig
    .cuenta(LEDR[7:4]) , // output [width_counter-1:0] cuenta_sig
    .fin_cuenta(fsm_variable)  // output  fin_cuenta_sig
);

defparam bloco3_inst.width_counter = 4;

endmodule
```

RTL Viewer:

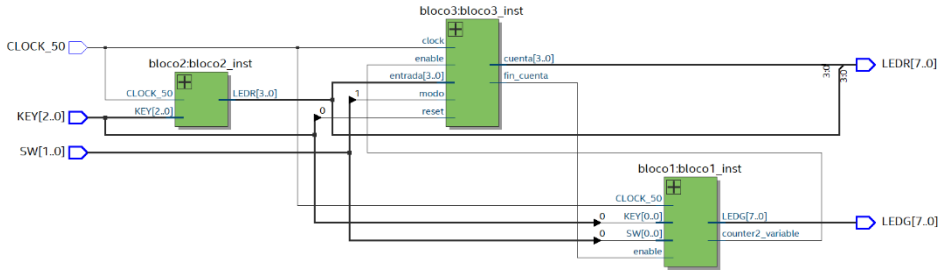

Figura 3: RTL Viewer del coche fantástico