UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Tarea 1

Diseño de Sistemas Secuenciales

Sistemas Digitales Programables

Máster Universitario en Ingeniería de Sistemas Electrónicos

2020/2021

Irene Garcia do Amaral

Diseño de un decodificador de hexadecimal a 7-segmentos

Implementación en Verilog:

```
module decod (iHEX, o7SEG);

   input [3:0] iHEX;

   output reg [6:0] o7SEG;

   always @(iHEX)
   begin

      case (iHEX)
                           // GFEDCBA
         4'h0 : o7SEG = 7'b1000000;
         4'h1 : o7SEG = 7'b1111001;
         4'h2 : o7SEG = 7'b0100100;
         4'h3 : o7SEG = 7'b0110000;
         4'h4 : o7SEG = 7'b0011001;
         4'h5 : o7SEG = 7'b0010010;
         4'h6 : o7SEG = 7'b0000010;
         4'h7 : o7SEG = 7'b1111000;
         4'h8 : o7SEG = 7'b0000000;
         4'h9 : o7SEG = 7'b0011000;
         4'ha : o7SEG = 7'b0001000;
         4'hb : o7SEG = 7'b0000011;
         4'hc : o7SEG = 7'b1000110;
         4'hd : o7SEG = 7'b0100001;
         4'he : o7SEG = 7'b0000110;
         4'hf : o7SEG = 7'b0001110;
         default : o7SEG = 7'b1111111;
      endcase

   end
endmodule
```

RTL Viewer:

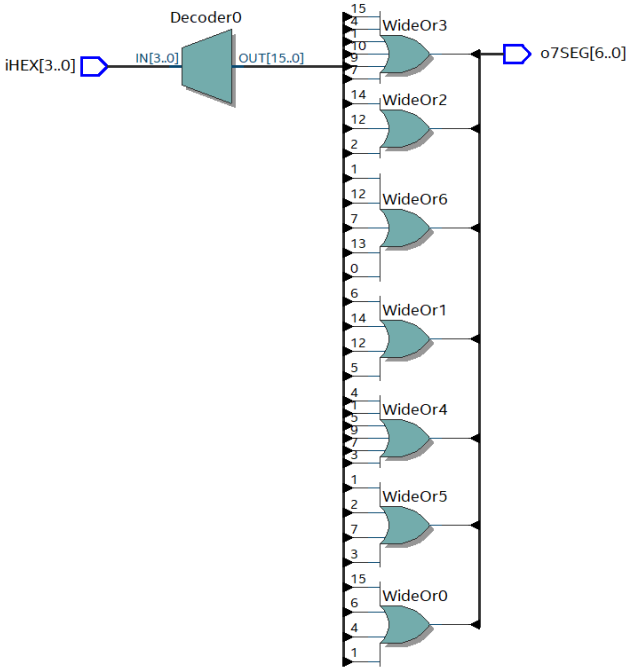

Figura 1: RTL Viewer del decodificador de hexadecimal a 7-segmientos

Diseño de un contador parametrizable

Implementación en Verilog:

```
module contador(iCLK, iRST_n, iENABLE, iUP_DOWN, oCOUNT, oTC);

   parameter fin_cuenta = 20;

   `include "MathFun.vh"
   parameter n = CLogB2(fin_cuenta-1);

   input iCLK, iRST_n, iENABLE, iUP_DOWN;
   output reg [n-1:0] oCOUNT;
   output oTC;

   always @(posedge iCLK or negedge iRST_n)
   begin
   if (!iRST_n)
      oCOUNT <= 0;

   else
   begin
      if (iENABLE == 1'b1)

         if(iUP_DOWN) //modo UP
            if(oCOUNT == fin_cuenta-1)
               oCOUNT <= 0;
            else
               oCOUNT <= oCOUNT +1;

         else //modo DOWN
            if(oCOUNT == 0)
               oCOUNT <= fin_cuenta -1;
            else
               oCOUNT <= oCOUNT -1;

      //else oCOUNT <= oCOUNT; //HOLD si no hay ENABLE (linha desnecesaria)
      end
   end

   assign oTC = (iUP_DOWN == 1 && oCOUNT == fin_cuenta-1) ? 1'b1 : (iUP_DOWN == 0 && oCOUNT == 0) ? 1'b1 : 1'b0;

endmodule
```

RTL Viewer:

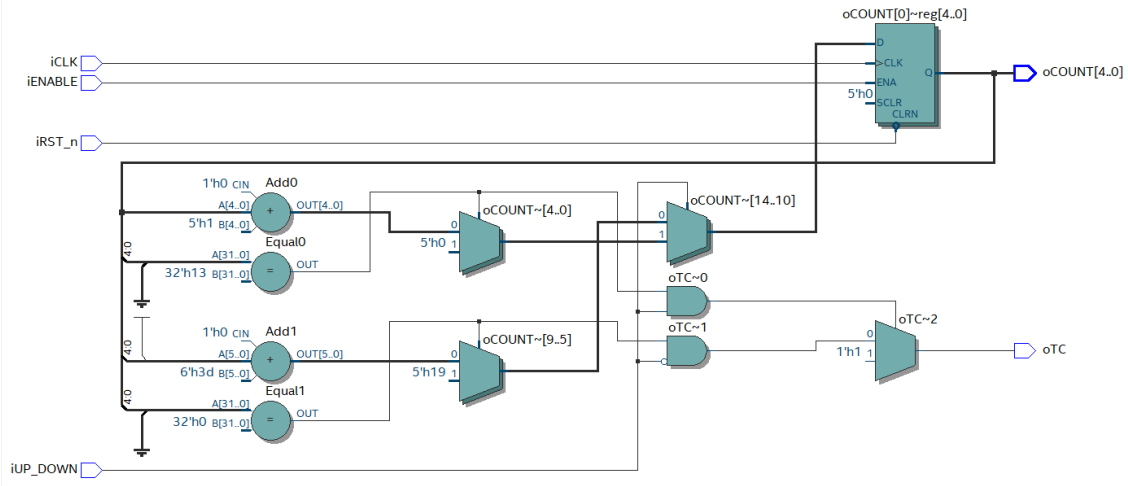

Figura 2: RTL Viewer del contador parametrizable

Waveform:

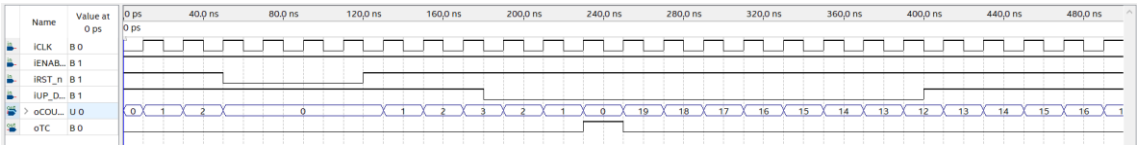

Figura 3: Waveform del contador parametrizable

Implementación del Testbench:

Utilizando un testbench han sido realizadas 6 tasks para simulación:

- Test de reset
- Test de contador en modo UP
- Test del enable e reset en modo UP
- Test de contador en modo DOWN
- Test del enable e reset en modo DOWN
- Test de disable

```
`timescale 1ns/1ps

module tb_contador ();

localparam T = 20;

reg               CLK, RST_n;
reg               ENABLE, UP_DOWN;

wire    [ 4:0]    COUNT;
wire              TC;

// Instanciación del DUT (Device Under Test)
contador #(.fin_cuenta(20)) DUT (
    .iCLK(CLK),
    .iRST_n(RST_n),
    .iENABLE(ENABLE),
    .iUP_DOWN(UP_DOWN),
    .oCOUNT(COUNT),
    .oTC(TC)
);

initial
 begin
        RST_n = 1'b0;
        CLK = 1'b0;
        ENABLE = 1'b0;
        UP_DOWN = 1'b1;

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

        caso1(5);
        caso2(6);
        reset(3);
        caso3(5);
        Disable(4);
        caso4(6);
        reset(4);

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

    end
```

```
always
    begin
        #(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);
    UP_DOWN = 1'b1;
    ENABLE = 1'b1;
    RST_n = 1'b0;
    #(T*ciclos_reset);
    RST_n = 1'b1;
end
endtask

task caso1(input integer ciclos1);  //TEST EN MODO UP
begin
    $display("Caso 1: TEST EN MODO UP (%d ciclos)", ciclos1);
    UP_DOWN = 1'b1;
    #(T*2) RST_n = 1'b1;
    @(negedge CLK) ENABLE = 1'b1;
    #(T*22)
    @(negedge CLK) ENABLE = 1'b0;
    #(T*ciclos1);
end
endtask

task caso2(input integer ciclos2); //TEST COM ENABLE E RESET EN MODO UP
begin
    $display("Caso 2: TEST COM ENABLE E RESET ACTIVOS EN MODO UP (%d ciclos)", ciclos2);
    fork
    UP_DOWN = 1'b1;
    RST_n = 1'b1;
    ENABLE = 1'b1;
    #(T*2)ENABLE = 1'b0;
    #(T*4)ENABLE = 1'b1;
    #(T*ciclos2);
    join
end
endtask
```

```
task caso3(input integer ciclos3);  //TEST EN MODO DOWN
begin
    $display("Caso 3: TEST EN MODO DOWN (%d ciclos)", ciclos3);
    UP_DOWN = 1'b0;
    #(T*2) RST_n = 1'b1;
    @(negedge CLK) ENABLE = 1'b1;
    #(T*22)
    @(negedge CLK) ENABLE = 1'b0;
    #(T*ciclos3);
end
endtask

task caso4(input integer ciclos4); //TEST COM ENABLE E RESET EN MODO DOWN
begin
    $display("Caso 4: TEST COM ENABLE E RESET ACTIVOS EN MODO DOWN (%d ciclos)", ciclos4);
    fork
    UP_DOWN = 1'b0;
    RST_n = 1'b1;
    ENABLE = 1'b1;
    #(T*2)ENABLE = 1'b0;
    #(T*4)ENABLE = 1'b1;
    #(T*ciclos4);
    join
end
endtask

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

endmodule
```

RTL Simulation:

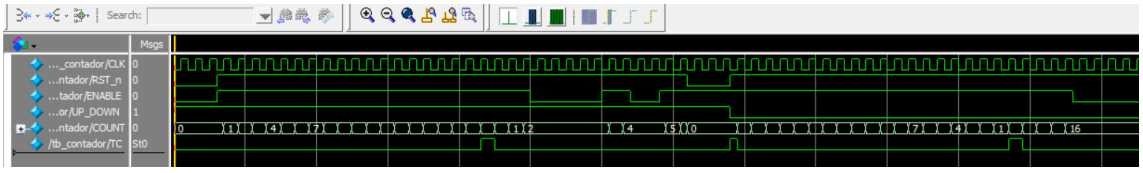

Figura 4: RTL Simulation del contador parametrizable con todas las tasks mencionadas arriba

Diseño de un registro de desplazamiento

Implementación en Verilog:

```
module reg7shift(iCLK, iRST_n, iENABLE, iSR, oPR, oSR);

   input iCLK, iRST_n, iENABLE, iSR;
   output reg [6:0] oPR;
   output oSR;

   always @(posedge iCLK)
   begin
      if (!iRST_n)
         oPR <= 0;

      else if (iENABLE)       //nao ha carga
         oPR[6:0] <= {iSR, oPR[6:1]};

      //else HOLD
   end

   assign oSR = oPR[0];

endmodule
```

RTL Viewer:

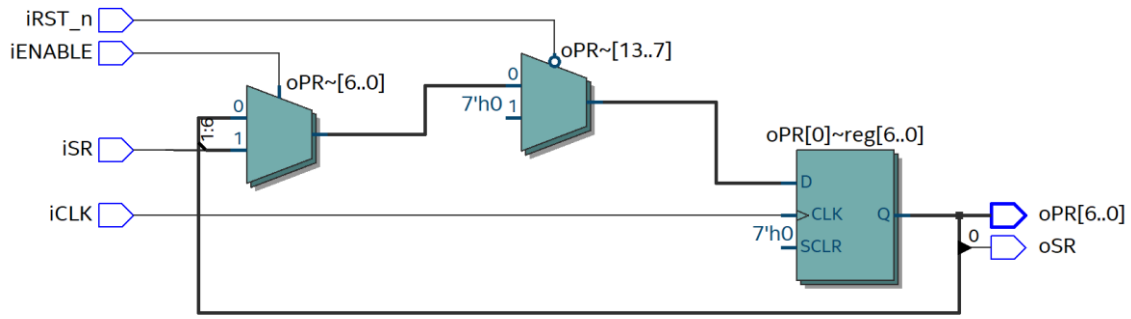

Figura 5: RTL Viewer del registro de desplazamiento

Waveform:

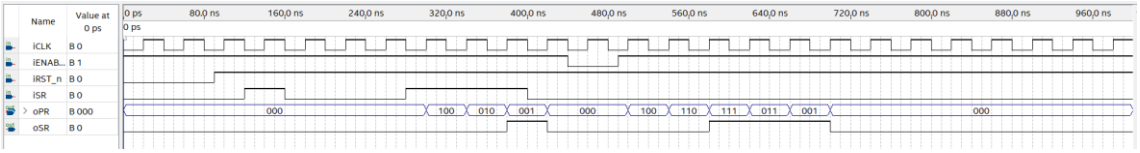

Figura 6: Waveform del registro de desplazamiento

Implementación del Testbench:

Utilizando un testbench han sido realizadas 3 tasks para simulación:

- Test de reset
- Test de enable
- Test de funcionamiento del bit de desplazamiento (0 e 1).

```
`timescale 1ns/100ps

module tb_reg7shift ();

   localparam T = 20;

   reg iCLK, iRST_n, iENABLE, iSR;

   wire [6:0] oPR;
   wire oSR;

   reg7shift DUT
   (
      .iCLK(iCLK) ,  // input  iCLK_sig
      .iRST_n(iRST_n) , // input  iRST_n_sig
      .iENABLE(iENABLE) ,  // input  iENABLE_sig
      .iSR(iSR) , // input  iSR_sig
      .oPR(oPR) , // output [N-1:0] oPR_sig
      .oSR(oSR)   // output  oSR_sig
   );

   //defparam reg7shift_inst.N = ;

   initial
   begin
      iCLK = 0;
      forever #(T/2) iCLK=~iCLK;
   end

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

      iENABLE = 1'b1;
      iSR = 1'b0;
      caso1 (5);
      Disable(2);
      reset(3);

      $display("FIN DE SIMULACIÓN!");

      $stop;
   end
```

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

   task reset(input integer ciclos_res);
   begin
      $display("Caso reset (%d ciclos)", ciclos_res);

      iRST_n = 1'b0;
      repeat (ciclos_res) @(negedge iCLK);
      iRST_n = 1'b1;
      #(T*ciclos_res);
   end
   endtask

   task Disable(input integer ciclos_en);
   begin
      $display("Caso Disable (%d ciclos)", ciclos_en);

      iRST_n = 1'b1;
      iSR = 1'b1;

      @(negedge iCLK) iENABLE = 1'b0;
      #(T*2)
      @(negedge iCLK) iENABLE = 1'b1;
      #(T*ciclos_en);
   end
   endtask

   task caso1 (input integer ciclos1);
   begin
      $display("TEST DE FUNCIONAMENTO DEL BIT DE DESPLAZAMIENTO (%d ciclos)", ciclos1);

      iRST_n = 1'b1;
      iENABLE = 1'b1;

      iSR = 1'b1;
      @(negedge iCLK);

      #(T*7) iSR = 1'b0;

      #(T*ciclos1);
   end
   endtask

endmodule
```

RTL Simulation:

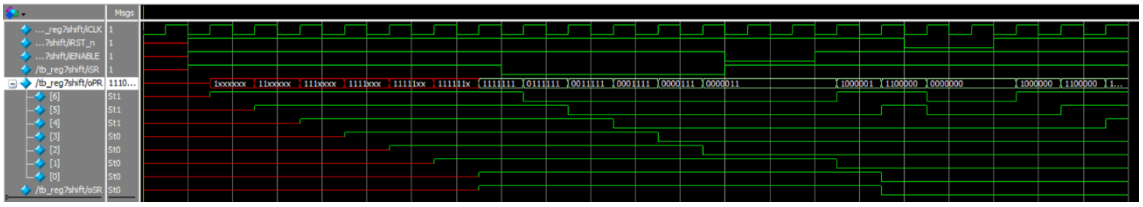

Figura 7: RTL Simulation del registro de desplazamiento con todas las tasks mencionadas arriba

Diseño del juego de luces del coche fantástico

Implementación en Verilog:

```
module cartop
(
input CLOCK_50,
input [1:0] KEY, SW,

output reg [7:0] LEDG
);

wire cable, cable2;
wire [6:0] salida_reg;

contador contador_inst
(
    .iCLK(CLOCK_50) , // input  iCLK_sig
    .iRST_n(KEY[0]) , // input  iRST_n_sig
    .iENABLE(SW[0]) , // input  iENABLE_sig
    .iUP_DOWN(1'b1) , // input  iUP_DOWN_sig
    .oCOUNT(oCOUNT_sig) ,   // output [n-1:0] oCOUNT_sig     (nao liga a nada)
    .oTC(cable)     // output  oTC_sig
);

defparam contador_inst.fin_cuenta = 12500000;
//defparam contador_inst.n = ;


reg7shift reg7shift_inst
(
    .iCLK(CLOCK_50) , // input  iCLK_sig
    .iRST_n(KEY[0]) , // input  iRST_n_sig
    .iENABLE(cable) , // input  iENABLE_sig
    .iSR(~cable2) ,   // input  iSR_sig
    .oPR(salida_reg) ,   // output [N-1:0] oPR_sig
    .oSR(cable2)   // output  oSR_sig
);
```

```
always @(salida_reg)
   case (salida_reg)
       7'b0000000: LEDG=8'b10000000;
       7'b1000000: LEDG=8'b01000000;
       7'b1100000: LEDG=8'b00100000;
       7'b1110000: LEDG=8'b00010000;
       7'b1111000: LEDG=8'b00001000;
       7'b1111100: LEDG=8'b00000100;
       7'b1111110: LEDG=8'b00000010;
       7'b1111111: LEDG=8'b00000001;
       7'b0111111: LEDG=8'b00000010;
       7'b0011111: LEDG=8'b00000100;
       7'b0001111: LEDG=8'b00001000;
       7'b0000111: LEDG=8'b00010000;
       7'b0000011: LEDG=8'b00100000;
       7'b0000001: LEDG=8'b01000000;

       default: LEDG=8'b10000000;

   endcase

endmodule
```

RTL Viewer:

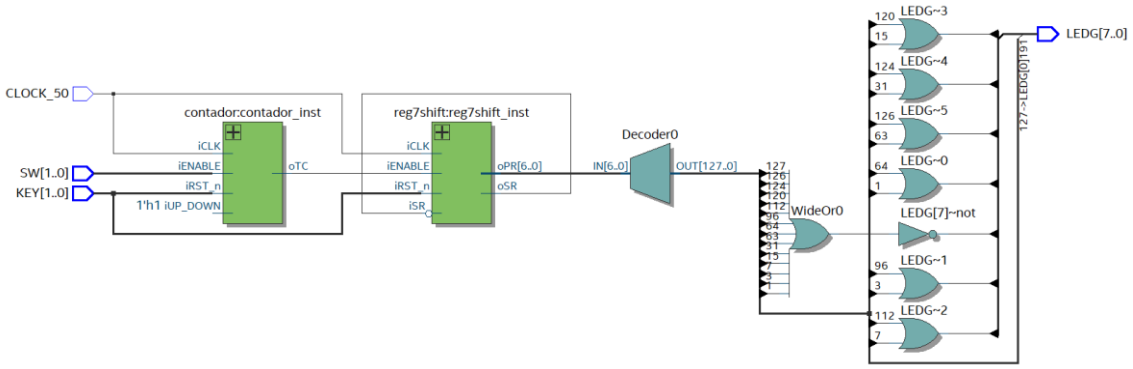

Figura 8: RTL Viewer del coche fantástico

Waveform:

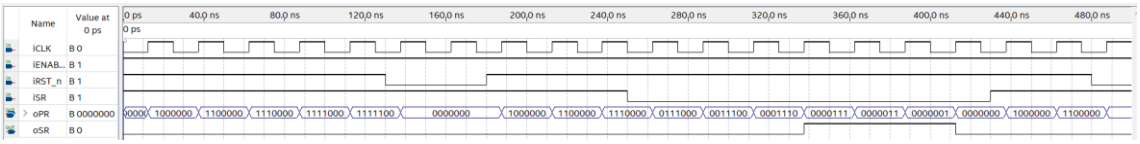

Figura 9: Waveform del coche fantástico

Implementación del Testbench:

```
`timescale 1ns/100ps

module tb_cartop ();

   localparam T = 20;

   reg CLOCK_50;
   reg [1:0] KEY, SW;

   wire [7:0] LEDG;

cartop cartop_inst
(
   .CLOCK_50(CLOCK_50) ,   // input  CLOCK_50_sig
   .KEY(KEY) , // input [1:0] KEY_sig                  --> RESET
   .SW(SW) ,   // input [1:0] SW_sig                   --> ENABLE
   .LEDG(LEDG)    // output [7:0] LEDG_sig
);

   initial
   begin
      CLOCK_50 = 0;
      forever #(T/2) CLOCK_50=~CLOCK_50;
   end

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

      SW = 1'b1;

      caso1(10);
      Disable(2);
      reset(3);

      $display("FIN DE SIMULACIÓN!");

      $stop;
   end
```

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

   task caso1 (input integer ciclos1);
   begin
      $display("TEST DE FUNCIONAMENTO (%d ciclos)", ciclos1);

      @(negedge CLOCK_50);
      KEY = 1'b0;
      SW = 1'b1;

      #(T*ciclos1);
   end
   endtask

   task reset(input integer ciclos_res);
   begin
      $display("Caso reset (%d ciclos)", ciclos_res);

      KEY = 1'b0;
      repeat (ciclos_res) @(negedge CLOCK_50);
      KEY = 1'b1;
      #(T*ciclos_res);
   end
   endtask

   task Disable(input integer ciclos_en);
   begin
      $display("Caso Disable (%d ciclos)", ciclos_en);

      KEY = 1'b1;

      @(negedge CLOCK_50) SW = 1'b0;
      #(T*2)
      @(negedge CLOCK_50) SW = 1'b1;
      #(T*ciclos_en);
   end
   endtask

endmodule
```

RTL Simulation:

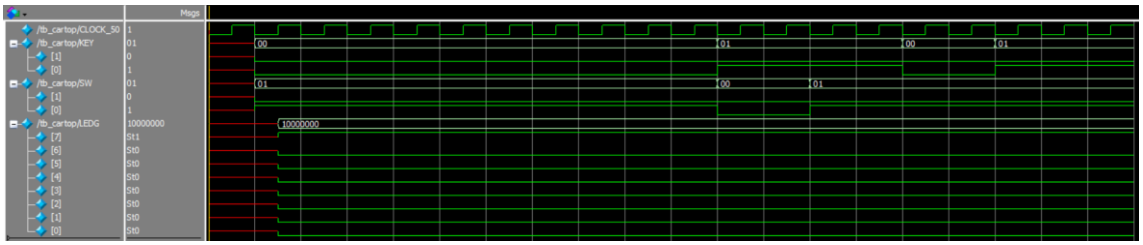

Figura 10: RTL Simulation del coche fantástico