Isai Mercado Oliveros

Lab 8

Nov 11, 2014

MOD4 Verilog code (5)

```
module MOD4(inc,rst,clk,curS);
   input inc, rst, clk;
   output[1:0] curS;
   wire[1:0] nxtS;

   // next state logic
   //assign nxtS = (curS == 2'd3)? 0 : curS + 2'd1;
   assign nxtS = (inc == 1'd1)? curS + 2'd1 : curS;

   FF flip1(curS[1],clk,rst,nxtS[1]);
   FF flip0(curS[0],clk,rst,nxtS[0]);


endmodule



module FF(q, clk, clr, d);
    input clk, clr, d;
    output reg q;

    always @(posedge clk)
        if (clr) q <= 0;
        else q <= d;
endmodule
```

MOD4 TCL file (4)

```
#add all signals to the waveform viewer
wave add / -radix hex
#q, clk, d, clr,din,write

#define how the data input signals will behave when you run the simulation
#the command "isim force add (signal) 0 -time 0 -value 1 -time 20ns -repeat 40ns" means that (signal)
#will have a value of 0000 from time 0, then change to 1 at time 20ns, and then repeat that cycle every 40ns
isim force add clk 0 -time 0 -value 1 -time 10ns -repeat 20ns
isim force add rst 1 -time 0 -value 0 -time 15ns
isim force add inc 0 -time 0 -value 1 -time 25ns -value 0 -time 35ns -repeat 40ns
#Nothing will change in the waveform viewer until you run the simulation for some period of time.
run 320ns
```

MOD4 simulation waveform (4)

Programmable timer Verilog code (for testing timer) (5)

```
module prog_timer (clk, reset, clken, load_number, counter, zero, tp);
  input clk, reset, clken;
  input [23:0] load_number;
  output reg [23:0] counter;
  output reg zero, tp;

  wire cnt0;

  assign cnt0 = ~(|counter);                          // is current count = 0?

  always @(posedge clk or posedge reset)
    if (reset == 1'b1)                                 // on reset
      begin
         counter = load_number-1;                     // initialize counter with preload
         zero = 1'b0;                                  // clear ceo output
         tp = 1'b0;                                // clear tp output
      end
   else if (cnt0 & clken)                             // if count is 0
      begin
         counter = load_number-1;                     // initialize counter with preload
         zero = 1'b1;                                 // set the ceo output
         tp = ~tp;                                 // toggle the tp output
      end
   else if (clken)
      begin
            counter = counter - 24'h000001;              // decrement the counter
         zero = 1'b0;                                 // clear the ceo output
         tp = tp;                               // maintain the tp output
      end

endmodule
```

```
prog_timer timer (clk_in,1'b0,1'b1,24'd250000, ,zero_out,tp_out); //prog_timer (clk, reset, clken, load_number, counter, zero, tp);
MOD4 mod (zero_out,reset_in,clk_in,curS_out); //MOD4(inc,rst,clk,curS);
```

Programmable timer UCF file (4)

```
## FX2 connector
NET zero_out LOC = "B4";  # Bank = 0, Pin name = IO_L24N_0, Type = I/O, Sch name = R-IO1
NET tp_out LOC = "A4";  # Bank = 0, Pin name = IO_L24P_0, Type = I/O, Sch name = R-IO2
#NET "PIO<2>"  LOC = "C3";  # Bank = 0, Pin name = IO_L25P_0, Type = I/O, Sch name = R-IO3
#NET "PIO<3>"  LOC = "C4";  # Bank = 0, Pin name = IO, Type = I/O, Sch name = R-IO4
#NET "PIO<4>"  LOC = "B6";  # Bank = 0, Pin name = IO_L20P_0, Type = I/O, Sch name = R-IO5
#NET "PIO<5>"  LOC = "D5";  # Bank = 0, Pin name = IO_L23N_0/VREF_0, Type = VREF, Sch name = R-IO6
```

4x7 Segment Controller Verilog code (5)

```
////////////////////////////////////////////////////////////////////////////
module SEG7dec(num_in, numCode_out);
    input[3:0] num_in;
    output[6:0] numCode_out;

    assign numCode_out = (num_in == 4'd0)?  7'b0000001 :
                         (num_in == 4'd1)?  7'b1001111 :
                         (num_in == 4'd2)?  7'b0010010 :
                         (num_in == 4'd3)?  7'b0000110 :
                         (num_in == 4'd4)?  7'b1001100 :
                         (num_in == 4'd5)?  7'b0100100 :
                         (num_in == 4'd6)?  7'b0100000 :
                         (num_in == 4'd7)?  7'b0001111 :
                         (num_in == 4'd8)?  7'b0000000 :
                         (num_in == 4'd9)?  7'b0000100 :
                         (num_in == 4'd10)? 7'b0001000 :
                         (num_in == 4'd11)? 7'b1100000 :
                         (num_in == 4'd12)? 7'b0110001 :
                         (num_in == 4'd13)? 7'b1000010 :
                         (num_in == 4'd14)? 7'b0110000 : 0111000;
endmodule
```

TestBench Verilog code (5)

```
////////////////////////////////////////////////////////////////////////////////
module testBench( dp0_in , dp1_in , dp2_in , dp3_in , clk_in , reset_in , seg_out , an_out , zero_out , tp_out , dp_out);
    input dp0_in,dp1_in,dp2_in,dp3_in,clk_in,reset_in;
    output[6:0] seg_out;
    output[3:0] an_out;
    output zero_out,tp_out,dp_out;
    wire[1:0] curS_out;
    wire[3:0] mux4bits_out,an_out_wire;
    wire dp_out_wire;


    prog_timer timer (clk_in,1'b0,1'b1,24'd250000, ,zero_out,tp_out); //prog_timer (clk, reset, clken, load_number, counter,
    MOD4 mod (zero_out,reset_in,clk_in,curS_out); //MOD4(inc,rst,clk,curS);

    DEC2to4 dec (curS_out,an_out_wire); //DEC2to4(in,out);
    MUX4to1_1bit smallMux (dp0_in,dp1_in,dp2_in,dp3_in,dp_out_wire,curS_out); //MUX4to1_1bit(input0,input1,input2,input3,out
    assign dp_out = ~dp_out_wire;
    assign an_out = ~an_out_wire;
    MUX4to1_4bits bigMux ( 4'b0001 , 4'b1010 , 4'b1011 , 4'b1000 ,mux4bits_out,curS_out); //MUX4to1_1bit(input0,input1,input
    SEG7dec seg (mux4bits_out,seg_out); //SEG7dec(num_in, numCode_out);

endmodule
```

TestBench UCF file (4)

lab8.ucf - Notepad

File Edit Format View Help

```
# This file is a general .ucf for Nexys2 rev A board
# To use it in a project:
# - remove or comment the lines corresponding to unused pins
# - rename the used signals according to the project


## clock pin for Nexys 2 Board
NET clk_in   LOC = "B8"; # Bank = 0, Pin name = IP_L13P_0/GCLK8, Type = GCLK, Sch name = GCLK0

## Switches
NET dp3_in LOC = "G18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW0
NET dp2_in LOC = "H18"; # Bank = 1, Pin name = IP/VREF_1, Type = VREF, Sch name = SW1
NET dp1_in LOC = "K18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW2
NET dp0_in LOC = "K17"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW3

## Buttons
NET reset_in LOC = "B18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = BTN0

## 7 segment display
NET seg_out[6] LOC = "L18"; # Bank = 1, Pin name = IO_L10P_1, Type = I/O, Sch name = CA
NET seg_out[5] LOC = "F18"; # Bank = 1, Pin name = IO_L19P_1, Type = I/O, Sch name = CB
NET seg_out[4] LOC = "D17"; # Bank = 1, Pin name = IO_L23P_1/HDC, Type = DUAL, Sch name = CC
NET seg_out[3] LOC = "D16"; # Bank = 1, Pin name = IO_L23N_1/LDC0, Type = DUAL, Sch name = CD
NET seg_out[2] LOC = "G14"; # Bank = 1, Pin name = IO_L20P_1, Type = I/O, Sch name = CE
NET seg_out[1] LOC = "J17"; # Bank = 1, Pin name = IO_L13P_1/A6/RHCLK4/IRDY1, Type = RHCLK/DUAL, Sch name = CF
NET seg_out[0] LOC = "H14"; # Bank = 1, Pin name = IO_L17P_1, Type = I/O, Sch name = CG
NET dp_out     LOC = "C17"; # Bank = 1, Pin name = IO_L24N_1/LDC2, Type = DUAL, Sch name = DP

NET an_out[0] LOC = "F17"; # Bank = 1, Pin name = IO_L19N_1, Type = I/O, Sch name = AN0
NET an_out[1] LOC = "H17"; # Bank = 1, Pin name = IO_L16N_1/A0, Type = DUAL, Sch name = AN1
NET an_out[2] LOC = "C18"; # Bank = 1, Pin name = IO_L24P_1/LDC1, Type = DUAL, Sch name = AN2
NET an_out[3] LOC = "F15"; # Bank = 1, Pin name = IO_L21P_1, Type = I/O, Sch name = AN3
```

Anomalies

It hard to debug!!!!