HW5: Test All

Summer 2021

Thiha Myint

Demo

https://youtu.be/xBi6JQqSB-0

Description

This lab tests all the instructions with several tests from the file given in lab manual. The final simulation is expected to show the value from 0 to 25 in consecutive order on the seven segment Basys board. After play around program counter, the value are shown on board and restart when the seven segment has reached 24.

Waveform

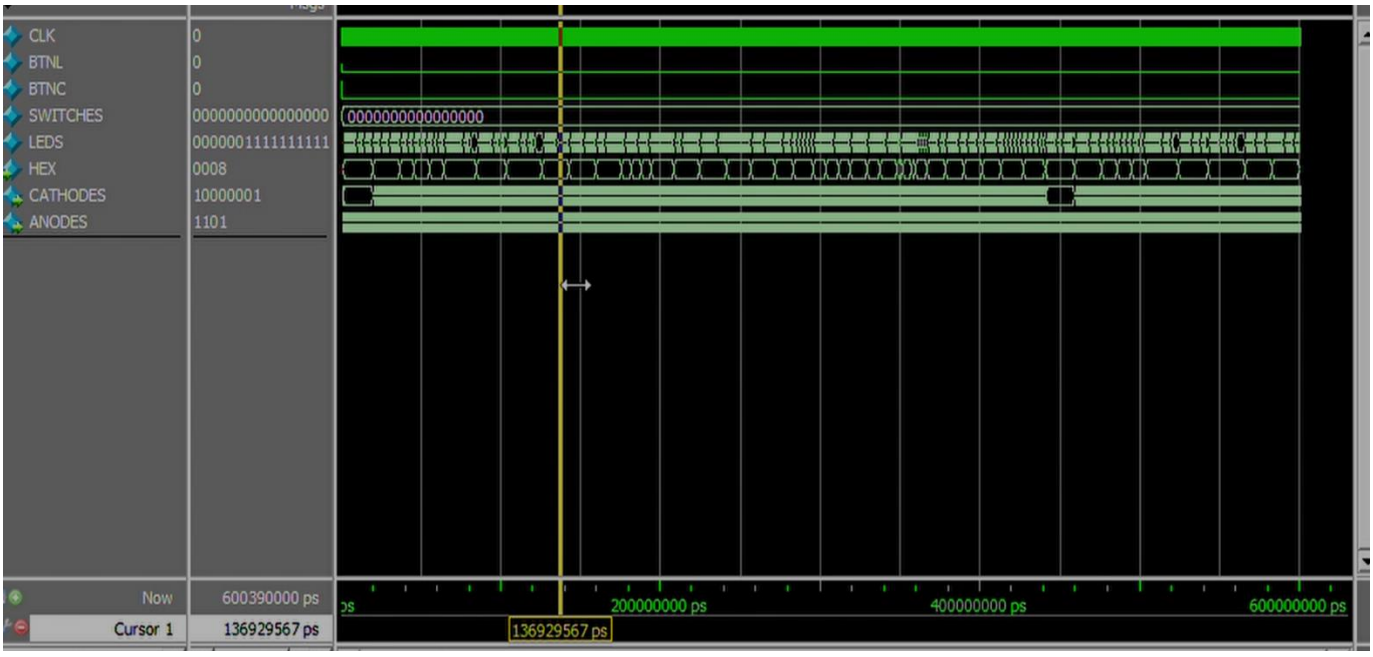

Fig – Waveform consecutively showing the value from 0 to 24 when sliding through

Code

```
`timescale 1ns / 1ps

module OTTER_TB;
    reg CLK;
    reg BTNL;
    reg BTNC;
    reg [15:0] SWITCHES;
    wire [15:0] LEDS;
    wire [7:0] CATHODES;
    wire [3:0] ANODES;

    OTTER_Wrapper DUT(.*);

    initial begin CLK = 1'b0; forever #10 CLK=~CLK; end

    initial begin
        BTNC = 1'b1;
        BTNL = 1'b0;
        SWITCHES = 16'd0;
            repeat(20)@(posedge CLK);
            BTNC = 1'b0;
            repeat(30000)@(posedge CLK);
            $finish;
    end
endmodule
```

```
module SevSegDisp(
    input CLK,
    input MODE,
    input [15:0] DATA_IN,
    output [7:0] CATHODES,
    output [3:0] ANODES
    );

    logic [15:0] BCD_Val;
    logic [15:0] Hex_Val;

    BCD BCDMod (.HEX(DATA_IN), .THOUSANDS(BCD_Val[15:12]),
                .HUNDREDS(BCD_Val[11:8]), .TENS(BCD_Val[7:4]),
                .ONES(BCD_Val[3:0]));

    CathodeDriver CathMod (.HEX(Hex_Val), .CLK(CLK), .CATHODES(CATHODES),
                           .ANODES(ANODES));

    // MUX to switch between HEX and BCD input
    always_comb begin
        if (MODE == 1'b1)
```

```
            Hex_Val = BCD_Val;
        else
            Hex_Val = DATA_IN;
    end


endmodule
```

```
module OTTER_Wrapper(
   input CLK,
   input BTNL,
   input BTNC,
   input [15:0] SWITCHES,
   output logic [15:0] LEDS,
   output [7:0] CATHODES,
   output [3:0] ANODES
   );


    // INPUT PORT IDS
////////////////////////////////////////////////////
    // Right now, the only possible inputs are the switches
    // In future labs you can add more MMIO, and you'll have
    // to add constants here for the mux below
    localparam SWITCHES_AD = 32'h11000000;

    // OUTPUT PORT IDS
////////////////////////////////////////////////////
    // In future labs you can add more MMIO
    localparam LEDS_AD      = 32'h11080000;
    localparam SSEG_AD     = 32'h110C0000;


   // Signals for connecting OTTER_MCU to OTTER_wrapper
//////////////////////////
   logic s_interrupt, btn_int;
   logic s_reset,s_load;
   logic sclk;// = 1'b0;
   reg sevenSegCLK;
   reg [2:0] sevenSegCounter;

   logic [15:0]  r_SSEG;// = 16'h0000;

   logic [31:0] IOBUS_out,IOBUS_in,IOBUS_addr;
   logic IOBUS_wr;

   assign s_interrupt = btn_int;


    // Declare OTTER_CPU
////////////////////////////////////////////////////
   OTTER_MCU MCU (.RST(s_reset),.INTR(s_interrupt), .CLK(sclk),

.IOBUS_OUT(IOBUS_out),.IOBUS_IN(IOBUS_in),.IOBUS_ADDR(IOBUS_addr),.IOBUS_WR(I
OBUS_wr));
```

```
   // Declare Seven Segment Display ////////////////////////////////////////
   SevSegDisp SSG_DISP (.DATA_IN(r_SSEG), .CLK(CLK), .MODE(1'b0),
                        .CATHODES(CATHODES), .ANODES(ANODES));

   // Declare Debouncer One Shot  ////////////////////////////////////////
   debounce_one_shot DB(.CLK(sclk), .BTN(BTNL), .DB_BTN(btn_int));


   clk_div clkDIv(CLK, sclk);

   assign s_reset = BTNC;

     // Connect Board peripherals (Memory Mapped IO devices) to IOBUS
////////////////////////////////////////
    always_ff @ (posedge sclk)
    begin

        if(IOBUS_wr)
            case(IOBUS_addr)
                LEDS_AD: LEDS <= IOBUS_out;
                SSEG_AD: r_SSEG <= IOBUS_out[15:0];

            endcase

    end

    always_comb
    begin
        IOBUS_in=32'b0;
        case(IOBUS_addr)
            SWITCHES_AD: IOBUS_in[15:0] = SWITCHES;

            default: IOBUS_in=32'b0;
        endcase
    end

    initial begin
        sevenSegCLK = 1'b0;
        sevenSegCounter = 3'b0;
    end
    always@(posedge CLK) begin
        if(IOBUS_addr == SSEG_AD)
            sevenSegCLK <= 1'b1;

        if(sevenSegCLK)
            sevenSegCounter <= sevenSegCounter +1;

        if(sevenSegCounter == 3'b111) begin
            sevenSegCLK <= 1'b0;
            sevenSegCounter <= 3'b0;
        end
    end
   endmodule
```