Verilog Code: lab2_example.v

```
`timescale 1ns / 1ps

module lab2_example(
      input   wire    [4:0] okUH,
      output  wire    [2:0] okHU,
      inout   wire    [31:0] okUHU,
      inout   wire    okAA,
      input   wire    sys_clkn,
      input   wire    sys_clkp,
      input   wire    reset,
      // Your signals go here
      input [3:0] button,
      output [7:0] led
   );

   wire okClk;            //These are FrontPanel wires needed to IO communication
   wire [112:0]    okHE;  //These are FrontPanel wires needed to IO communication
   wire [64:0]     okEH;  //These are FrontPanel wires needed to IO communication

   //Declare your registers or wires to send or recieve data
   wire [31:0] clkdivFreq;      //signals that are outputs from a module must be wires
   wire counter_reset;
   //This is the OK host that allows data to be sent or recived
   okHost hostIF (
      .okUH(okUH),
      .okHU(okHU),
      .okUHU(okUHU),
      .okClk(okClk),
      .okAA(okAA),
      .okHE(okHE),
      .okEH(okEH)
   );

   //Depending on the number of outgoing endpoints, adjust endPt_count accordingly.
   //In this example, we have 2 output endpoints, hence endPt_count = 2.
   localparam  endPt_count = 2;
   wire [endPt_count*65-1:0] okEHx;
   okWireOR # (.N(endPt_count)) wireOR (okEH, okEHx);

   // Clock
   wire clk;
   reg [31:0] clkdiv;
   reg slow_clk;
   reg [7:0] counter;

   IBUFGDS osc_clk(
        .O(clk),
        .I(sys_clkp),
        .IB(sys_clkn)
    );

    initial begin
        clkdiv = 0;
        slow_clk = 0;

    end

    //  variable_1 is a wire that contains data sent from the PC to FPGA.
    //  The data is communicated via memeory location 0x00
    okWireIn wire10 (   .okHE(okHE),
                        .ep_addr(8'h00),
                        .ep_dataout(clkdivFreq));

    okWireIn wire11 (   .okHE(okHE),
                        .ep_addr(8'h01),
                        .ep_dataout(counter_reset));

    // This code creates a slow clock from the high speed Clk signal
    // You will use the slow clock to run your finite state machine
    // The slow clock is derived from the fast 200 MHz clock by dividing it 10,000,000 time and
another 2x
    // Hence, the slow clock will run at 10 Hz
    always @(posedge clk) begin
        clkdiv <= clkdiv + 1'b1;
        if (clkdiv == clkdivFreq) begin
            slow_clk <= ~slow_clk;
            clkdiv <= 0;
        end
    end

    assign led = ~counter;
    //The main code will run fr0m the slow clock.  The rest of the code will be in this section.
    //The counter will increment when button 0 is pressed and on the rising edge of the slow clk
    //The counter will decrement when button 0 is pressed and on the rising edge of the slow clk
    always @(posedge slow_clk) begin
            if ((button [0] == 1'b0) && (button[1]==1'b0 || button[2]==1'b0 || button[3]==1'b0)) begin
                counter <= counter + 1'b0;
            end
            else if ((button [1] == 1'b0)  && (button[0]==1'b0 || button[2]==1'b0 || button[3]==1'b0))
begin
                counter <= counter + 1'b0;
            end
            else if ((button [2] == 1'b0)  && (button[1]==1'b0 || button[0]==1'b0 || button[3]==1'b0))
begin
               counter <= counter + 1'b0;
            end
            else if ((button [3] == 1'b0)  && (button[1]==1'b0 || button[2]==1'b0 || button[0]==1'b0))
begin
               counter <= counter + 1'b0;
            end
           else if (button [0] == 1'b0) begin
               counter <= 8'hFF;
           end
           else if (button [1] == 1'b0) begin
               counter <= 8'h00;
           end
           else if (button [2] == 1'b0) begin

                if (counter > 8'hFD) begin
                   counter <= 8'h00;
                end
                else
                   counter <= counter + 2'b10;
           end
           else if (button [3] == 1'b0) begin
                counter <= counter - 2'b10;
                if (counter < 8'h02) begin
                   counter <= 8'h00;
                end
                else
                   counter <= counter - 2'b10;
           end

           if (counter_reset == 1) begin
               counter = 8'h00;
           end

    end

    // result_wire is transmited to the PC via address 0x20
    okWireOut wire20 (  .okHE(okHE),
                       .okEH(okEHx[ 0*65 +: 65 ]),
                       .ep_addr(8'h20),
                       .ep_datain(counter));
endmodule
```