# Lab No. 4: Sequential Logics in Verilog

Leonardo Fusser (1946995)

## Purpose:

a) Understand sequential logic applications.
b) Implement, simulate, and test sequential circuits in Verilog.
c) Use of the dataflow modeling.

## To hand in:

Listing of the following for each of your designs on Lea:

1. This sheet with answers and screenshots.
2. **Commented Verilog source files - module and top module for all parts.**
3. Commented resource summary report with analysis for part1 only. Must be copied in this word document (must be commented).
4. The top module must include a **prolog header:** Lab number and name, author,description, version number.
5. Each and every module should include a prolog header.
6. Verilog test bench and simulation waveform for part 1. Must be copied in this word document (must be commented).
7. View Technology Schematics for part 2 only – you must exclude the clock divider from part 1. Must be copied in this word document (must be commented).
8. In part 2 only, a block diagram of the top module (using Visio or other CAD tools). Must be copied in this word document (must be commented).

## Lab Work

In your code, you must separate the system in three parts:

- Next data logic code.
- Data register code.
- Output logic code.

# Part 1: Implement a clock divider to blink LEDs

1. Complete the following Verilog module for a clock divider that can produce four clock frequencies of 11.92Hz, 5.96Hz, 2.98Hz and 1.49Hz on a Basys2 board.

```
module counter(
        input clk, rst,
        output _ Hz11_92, Hz5_96, Hz2_98, Hz1_49
);
```

On Basys2 mclk can be set to 100Mhz , 50MHz or 25MHz.

You need to find out about the jumper setting.

Hint: look at the schematics.

- The default selected master clock frequency used on the Baysys2 board is 50MHz (with no jumper selected).

2. Write a Verilog testbench to test the clock divider.

3. Implement clock divider that you have designed.

4. Perform simulation on your clock divider module using Xilinx ISIM tool.

What should be the time delay for the clock to simulate the actual clock frequency on Basys2 board?

The appropriate time delay to simulate the actual master clock frequency used on the Baysys2 board would be 10nS. This is because the actual master clock frequency used on the Baysys2 board is 50MHz. Refer to calculation below for proof:

$$Simulated\ time\ delay = \frac{1}{MCLK} = \frac{1}{50MHz} = 10nS$$

Note: MCLK value depends on selected master clock frequency used on Baysys2 board. Default master clock frequency (no jumper selected) on Baysys2 board is 50MHz. Refer to Baysys2 schematics for more details.

What would be the minimum simulation length required to view at least a full cycle of the lowest frequency clock output from module?

Since there is an initial low pulse for around 336mS, the minimum simulation length required to view at least a full cycle of the lowest frequency clock output from module (which is 1.49Hz) would be around 1S (T = ~672mS, ~672mS + ~336mS = ~1S). Refer to 1.49Hz clock divide output in “Figure 1” screenshot below.

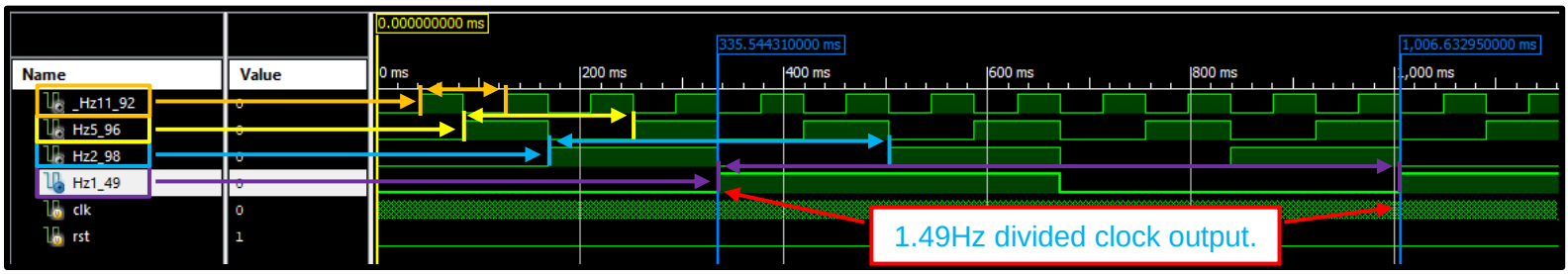



*Figure 1. Simulation result from Verilog "Counter_TB" file testing the logic from the Verilog "Counter" file. Figure above shows all four divided clock outputs (11.92Hz, 5.96Hz, 2.98Hz and 1.49Hz) from master clock (50MHz). Master clock is MCLK on Baysys2 board (simulated to produce the four divided clock outputs above).*

5. Create a new ISE top module named `lab4a`. Implemented your clock divider onto Basys2 board, where the 4 clock outputs will each be driving a LED.

6. Download to your Basys2 board for testing and verification. Demonstrate your working code to your instructor.
   - Demonstrated to Manijeh Khataie on October 25th, 2021.

## Part 2: Implement a shift register to drive LEDs

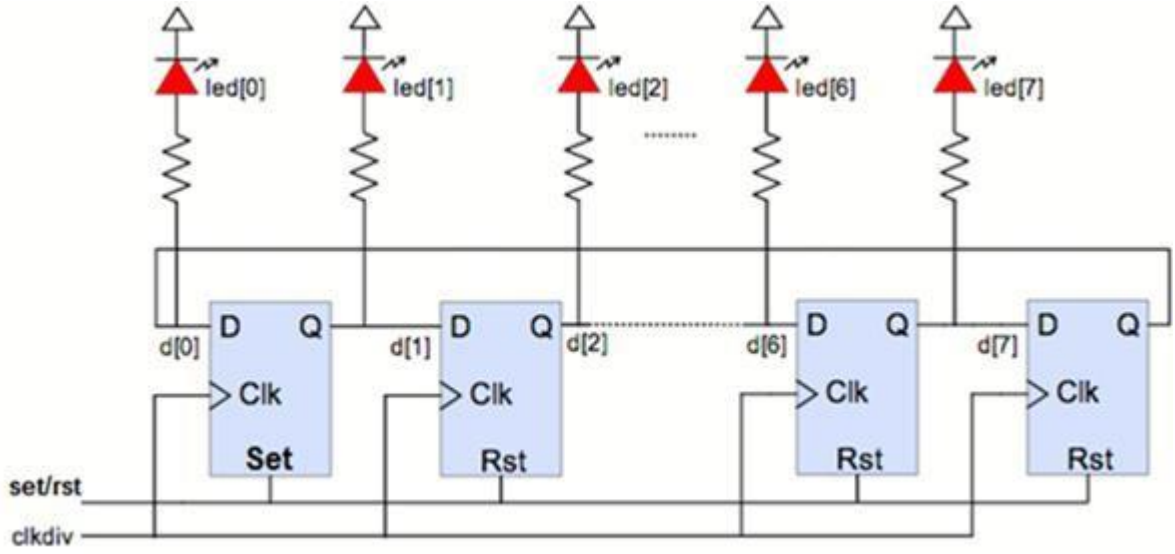
In part 2, you are required to implement a circuit with the following design requirements:

- When set/rst button is pressed, LED0 is on, while LED1 to LED7 are off.
- When set/rst button is released, the light keeps shifting from LED0 all the way to LED7, and back to LED 0.
- The speed of the LED shifts is controlled by 2 switches. Example if switch[1:0] is 0, LED shift frequency is 1.49Hz, while if switch[1:0] is 3, LED shift frequency is 11.92Hz. There are four possible speeds (see part 1).

Hints: implement the shift register as one module. Design a Mux module to select the clock. You may reuse the Verilog modules that you have created previously.

7. Using Visio (or other CAD tool) draw a block diagram of top module that connects all modules together.

8. Create a new ISE top module named `lab4b`.




9. Generate an UCF file for this top module.

10. Compile and generate programming bit file. Download to your Basys2 board for testing andverification.

    Demonstrate your working code to your instructor.

Code:

```
`timescale 1ns / 1ps
/* ************************************************************************************
*  Module name: Counter.v
*  Description: Code that allows Basys2 MCLK to be divided by certain values. These
*               divided clocks are used as input to the Mux module.
*
*  Author            Date                  Revision    Comments
**************************************************************************************
*  Leonardo Fusser   25 October 2021       v1.0.0      Created Counter.v file.
*  Leonardo Fusser   8 November 2021       v1.0.1      Completed Counter.v file.
*
**************************************************************************************/


/*Counter module*/
module Counter(
    input COUNTER_CLK, COUNTER_RST,          //CLK(MCLK) and RST inputs.
    output Hz11_92, Hz5_96, Hz2_98, Hz1_49   //CLK-divide outputs.
    );

    reg [24:0] Q;       //Register to store all the possible present-state CLK outputs.
    wire [24:0] Q_next; //Wire to store all the possible next-state CLK outputs.

    /*Data register*/
    always @(posedge COUNTER_CLK or posedge COUNTER_RST)
        begin
           if(COUNTER_RST == 1) Q <= 0;
           else Q <= Q_next;
        end

        /*Next state logic*/
        assign   Q_next = Q+1;

        /*Output logic*/
        assign   Hz11_92 = Q[21];  //clk/2^(22+1).
        assign   Hz5_96 = Q[22];   //clk/2^(23+1).
        assign   Hz2_98 = Q[23];   //clk/2^(24+1).
        assign   Hz1_49 = Q[24];   //clk/2^(25+1).

endmodule
```

*Figure 3. Verilog "Counter" module used in Part 1 and Part 2 to implement the logic to produce four divided clock outputs. Input is MCLK on Baysys2 board. The four divided clock output values are given in the above figure. Data register, next state logic and output logic sections are shown.*

```
`timescale 1ns / 1ps
/* ********************************************************************************
*  Module name: Mux.v
*  Description: Code that selects which divided MCLK should be used for ShiftRegister
*               module. Takes multiple inputs from Counter module.
*
*  Author            Date                  Revision    Comments
**********************************************************************************
*  Leonardo Fusser   25 October 2021       v1.0.0      Created Mux.v file.
*  Leonardo Fusser   8 November 2021       v1.0.1      Completed Mux.v file.
*
**********************************************************************************/


/*Mux module*/
module Mux(
    input A, B, C, D, MUX_CLK,    //CLK-divide (F1, F2, F3, F4) and CLK(MCLK) inputs.
    input [1:0] MUX_SEL,          //2 MUX select inputs.
    output reg Y                  //MUX output.
    );

always @(posedge MUX_CLK)
   begin
      case (MUX_SEL)
         0: Y <= A;         //If MUX select = 0, MUX output = F1 (connected to A).
         1: Y <= B;         //If MUX select = 1, MUX output = F2 (connected to B).
         2: Y <= C;         //If MUX select = 2, MUX output = F3 (connected to C).
         3: Y <= D;         //If MUX select = 3, MUX output = F4 (connected to D).
         default: Y <= A;   //Otherwise, MUX output = F1 (connected to A).
         endcase
   end
endmodule
```

*Figure 4. Verilog "Mux" module used in Part 2 to implement the logic to output one of the four divided clock outputs based on user input. Inputs are the four divided clock outputs from the Verilog "Counter" module used in Part 1 and two slide switches on Baysys2 board (which select which of the four divided clock outputs are used). A synchronized clock is also used to ensure no timing issues are generated during HDL synthesis. In other words, this is a 1-bit 4x1 MUX. Data register, next state logic and output logic sections are shown.*

```
`timescale 1ns / 1ps
/* ********************************************************************************
*  Module name: ShiftRegister.v
*  Description: Code that performs an 8-bit shifting operation one way. Input is taken
*               from output of Mux module (selected clock divider) and is used to
*               determine shifting speed.
*
*  Author            Date                  Revision    Comments
**********************************************************************************
*  Leonardo Fusser   25 October 2021       v1.0.0      Created ShiftRegister.v file.
*  Leonardo Fusser   8 November 2021       v1.0.1      Completed ShiftRegister.v file.
*
**********************************************************************************/


/*Shift-register module*/
module ShiftRegister(
    input SR_CLK, SR_CLR,  //Selected CLK-divide and RST inputs.
    output [7:0] Shift_OUT //Shifting output.
    );

   reg [7:0] Present_8bits; //Register to store all the possible present-state shifted outputs.
   wire [7:0] Next_8bits;   //Wire to store all the possible next-state shifted outputs.

   /*Data register*/
   always @(posedge SR_CLK or posedge SR_CLR)
      begin
         if (SR_CLR == 1)
            Present_8bits <= 8'b00000001;
         else
            Present_8bits <= Next_8bits;
      end

      /*Next state logic*/
      assign Next_8bits = {Present_8bits[6:0], Shift_OUT[7]};  //1-bit shift operation.

      /*Output logic*/
      assign Shift_OUT = Present_8bits;   //Resulting shifted output.

endmodule
```

*Figure 5. Verilog "ShiftRegister" module used in Part 2 to implement the logic of an 8-bit wide shift register with 1-bit shifting behavior. Inputs are the selected divided clock output from the Verilog "Mux" module (determines shifting speed) and a reset input (one of the slide switches on Baysys2 board). Output is the resulted shifted 8-bit operation (connected to LEDs on Baysys2 board). Data register, next state logic and output logic sections are shown.*

```
`timescale 1ns / 1ps
/* ********************************************************************************
*  Module name: Lab4-Part1_TOP.v
*  Description: Top module for Part 1 of Lab 4. All connections and instantiations
*               for all Verliog modules are performed here.
*
*  Author            Date                  Revision    Comments
**********************************************************************************
*  Leonardo Fusser   25 October 2021       v1.0.0      Created Lab4-Part1_TOP.v file.
*  Leonardo Fusser   8 November 2021       v1.0.1      Completed Lab4-Part1_TOP.v file.
*
**********************************************************************************/


/*Lab4 (Part1) TOP module*/
module Lab4Part1_TOP(
    input MCLK,RST,     //RST and MCLK inputs.
    output  F1,F2,F3,F4 //CLK-divide outputs.
    );

    /*Counter module instantiation*/
    Counter U1(
    .C_CLK(MCLK), //Connects to MCLK.
    .C_RST(RST),  //Connects to RST.
    .Hz11_92(F1), //Connects to F1.
    .Hz5_96(F2),  //Connects to F2.
    .Hz2_98(F3),  //Connects to F3.
    .Hz1_49(F4)   //Connects to F4.
    );

endmodule
```

*Figure 6. Verilog "Lab4-Part1_TOP" module used in Part 1 to map all the inputs and outputs in the Verilog "Counter" module to various input and output definitions in this top module. The final physical connections to the Baysys2 FPGA pins are implemented in the constraints file (shown later).*

```
`timescale 1ns / 1ps
/* ********************************************************************************************
*  Module name: Lab4-Part2_TOP.v
*  Description: Top module for Part 2 of Lab 4. All connections and instantiations
*               for all Verliog modules are performed here.
*
*  Author            Date                  Revision    Comments
**********************************************************************************************
*  Leonardo Fusser   25 October 2021       v1.0.0      Created Lab4-Part2_TOP.v file.
*  Leonardo Fusser   8 November 2021       v1.0.1      Completed Lab4-Part2_TOP.v file.
*
**********************************************************************************************/


/*Lab4 (Part2) TOP module*/
module Lab4Part2_TOP(
    input [1:0] SW,      //CLK-divide select input switches.
    input RST, MCLK,     //RST and MCLK inputs.
    output [7:0] LED     //Shifting LED outputs.
    );

   wire clk_sel, F1, F2, F3, F4; //Selected CLK-divide output wire and CLK-divide input wires.

   /*Counter module instantiation*/
   Counter U1(
      .COUNTER_CLK(MCLK), //Connects to MCLK.
      .COUNTER_RST(RST),  //Connects to RST.
      .Hz11_92(F1),       //Connects to F1.
      .Hz5_96(F2),        //Connects to F2.
      .Hz2_98(F3),        //Connects to F3.
      .Hz1_49(F4)         //Connects to F4.
   );

   /*MUX module instantiation*/
   Mux U2(
      .MUX_CLK(MCLK),     //Connects to MCLK.
      .MUX_SEL(SW),       //Connects to CLK-divide select input switches.
      .A(F1),             //Connects to F1.
      .B(F2),             //Connects to F2.
      .C(F3),             //Connects to F3.
      .D(F4),             //Connects to F4.
      .Y(clk_sel)         //Connects to clk_sel.
   );

   /*ShiftRegister module instantiation*/
   ShiftRegister U3(
      .SR_CLK(clk_sel),   //Connects to clk_sel.
      .SR_CLR(RST),       //Connects to RST.
      .Shift_OUT(LED)     //Connects to shifting LED outputs.
   );

endmodule
```

*Figure 6. Verilog "Lab4-Part2_TOP" module used in Part 2 to map all the inputs and outputs in the Verilog "Counter", "Mux" and "ShiftRegister" module to various input and output definitions in this top module ("connects" all the Verilog modules together in a unique way). The final physical connections to the Baysys2 FPGA pins are implemented in the constraints file (shown later).*

```
////////////////////////////////////////////////////////////////////////////////////
// Module name: Lab4_CONSTRAINTS.ucf
// Description: Constraints file for Lab 4. All mappings that need to be connected to the
//              Baysys2 FPGA are done here.
//
// Author            Date                  Revision    Comments
////////////////////////////////////////////////////////////////////////////////////
// Leonardo Fusser   25 October 2021       v1.0.0      Created Lab4_CONSTRAINTS.ucf file.
// Leonardo Fusser   8 November 2021       v1.0.1      Completed Lab4_CONSTRAINTS.ucf file.
//
////////////////////////////////////////////////////////////////////////////////////


//[For Part 1 and Part 2]
//Baysys2 clock and reset switch.
NET "MCLK" LOC = "B8";      //MCLK.
NET "RST" LOC = "P11";      //RST.

//[For Part 1]
//Clock divider output LEDs.
//NET "F4" LOC = "P6";      //F4.
//NET "F3" LOC = "P7";      //F3.
//NET "F2" LOC = "M11";     //F2.
//NET "F1" LOC = "M5";      //F1.

//[For Part 2]
//Shifting frequency select switches.
NET "SW<0>" LOC = "E2";     //SW[0].
NET "SW<1>" LOC = "N3";     //SW[1].

//[For Part 2]
//Shifting output LEDs.
NET "LED<0>" LOC = "M5";    //LED[0].
NET "LED<1>" LOC = "M11";   //LED[1].
NET "LED<2>" LOC = "P7";    //LED[2].
NET "LED<3>" LOC = "P6";    //LED[3].
NET "LED<4>" LOC = "N5";    //LED[4].
NET "LED<5>" LOC = "N4";    //LED[5].
NET "LED<6>" LOC = "P4";    //LED[6].
NET "LED<7>" LOC = "G1";    //LED[7].
```

*Figure 7. Verilog "Lab4_CONSTRAINTS" file used in Part 1 and Part 2 to connect all the various defined inputs and outputs in the Verilog "Lab4-Part1_TOP" and "Lab4-Part2_TOP" modules to the physical pins located on the Baysys2 FPGA.*

```
`timescale 1ns / 1ps
/* ********************************************************************************
*  Module name: Counter_TB.v
*  Description: Test bench file for Counter module in Lab 4. Used to test Counter
*               module logic.
*
*  Author             Date                    Revision    Comments
**********************************************************************************
*  Leonardo Fusser    25 October 2021         v1.0.0      Created Counter_TB.v file.
*  Leonardo Fusser    8 November 2021         v1.0.1      Completed Counter_TB.v file.
*
**********************************************************************************/


module Counter_TB;

   //Inputs.
   reg clk;
   reg rst;

   //Outputs.
   wire _Hz11_92;
   wire Hz5_96;
   wire Hz2_98;
   wire Hz1_49;

   //Instantiate the Unit Under Test (UUT).
   Counter uut (
      .clk(clk),
      .rst(rst),
      ._Hz11_92(_Hz11_92),
      .Hz5_96(Hz5_96),
      .Hz2_98(Hz2_98),
      .Hz1_49(Hz1_49)
   );

   initial begin

      //Initialize Inputs.
      clk = 0;
      rst = 1;

      #10;      //Wait for 10nS.

      rst = 0;

   end

      //Stimulus.
      always
         begin
         #10;              //Wait for 10nS.
         clk = ~clk;       //Toggle the clock.
         end

endmodule
```

*Figure 8. Verilog "Counter_TB" file used in Part 1 used to test and verify all the logic and operation of the Verilog "Counter" module using the Xilinx iSIM simulator. A simulated MCLK is used to simulate the real clock on the Baysys2 board (50MHz clock).*

Technology Schematic (for Part 2):

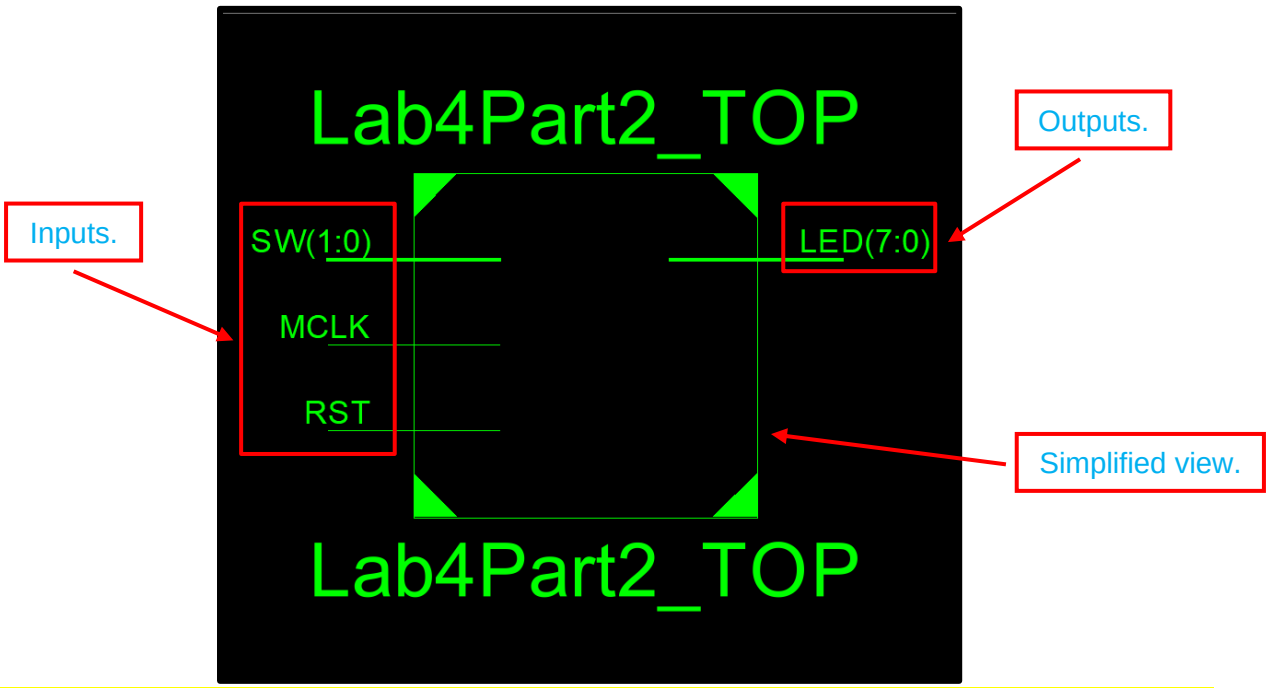



*Figure 9. Resulting Technology Schematic for the Verilog "Lab4-Part2_TOP" module. This view gives only a "top-level" view of the actual logic being implemented. Inputs and outputs shown above.*

RTL Schematic (for Part 2):

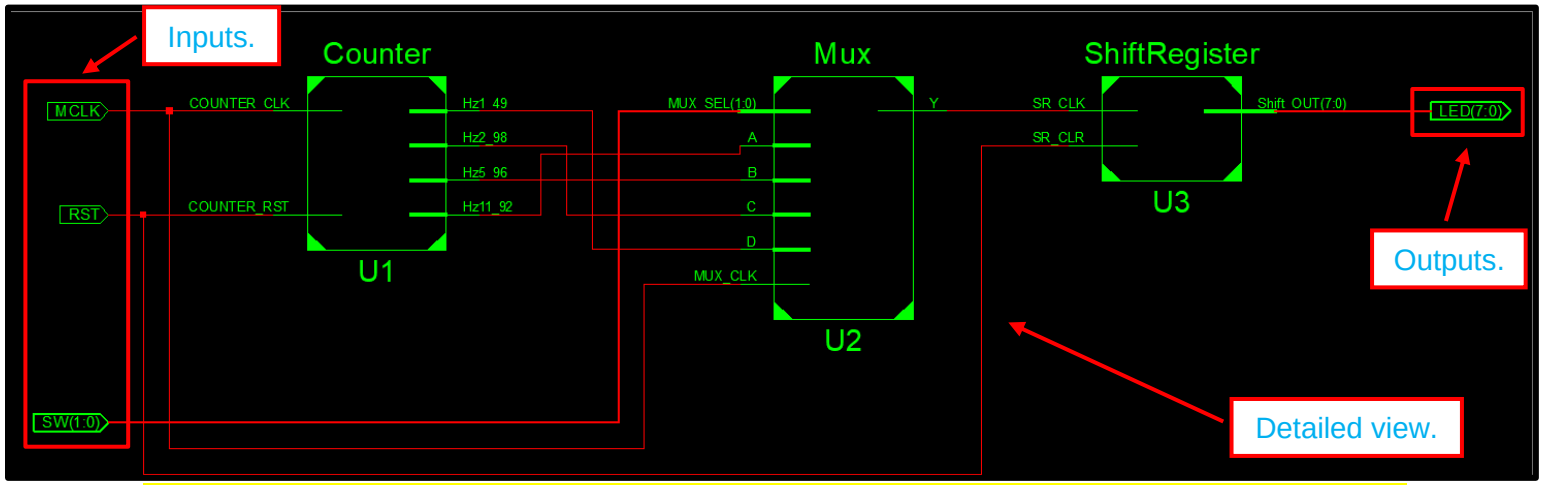



*Figure 10. Resulting RTL Schematic for the Verilog "Lab4-Part2_TOP" module. This view gives a more detailed "gate-level" view of the actual logic being implemented. Inputs and outputs shown above. Inside the modules are other logic devices such as LUTs (not shown above).*

Design Summary Report (for Part 1):

| Device Utilization Summary (estimated values) | | | [-] |
|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** |
| Number of Slices | 13 | 960 | 1% |
| Number of Slice Flip Flops | 25 | 1920 | 1% |
| Number of 4 input LUTs | 25 | 1920 | 1% |
| Number of bonded IOBs | 6 | 83 | 7% |
| Number of GCLKs | 1 | 24 | 4% |

*Figure 11. Resulting design summary report for the Verilog "Lab4-Part2_TOP" module. There are a total of 13 slices being used, since 25 D-FFs are needed, and each slice has 2 D-FFs, a total of 13 slices are needed. 25 D-FFs and 4-bit LUTs are needed since we are using registers that are 25-bits deep in the Verilog "Counter" module. 1 GLCK is only required since there is only one master clock that is being used (MCLK on Baysys2 board).*

Block Diagram (for Part 2 top module):

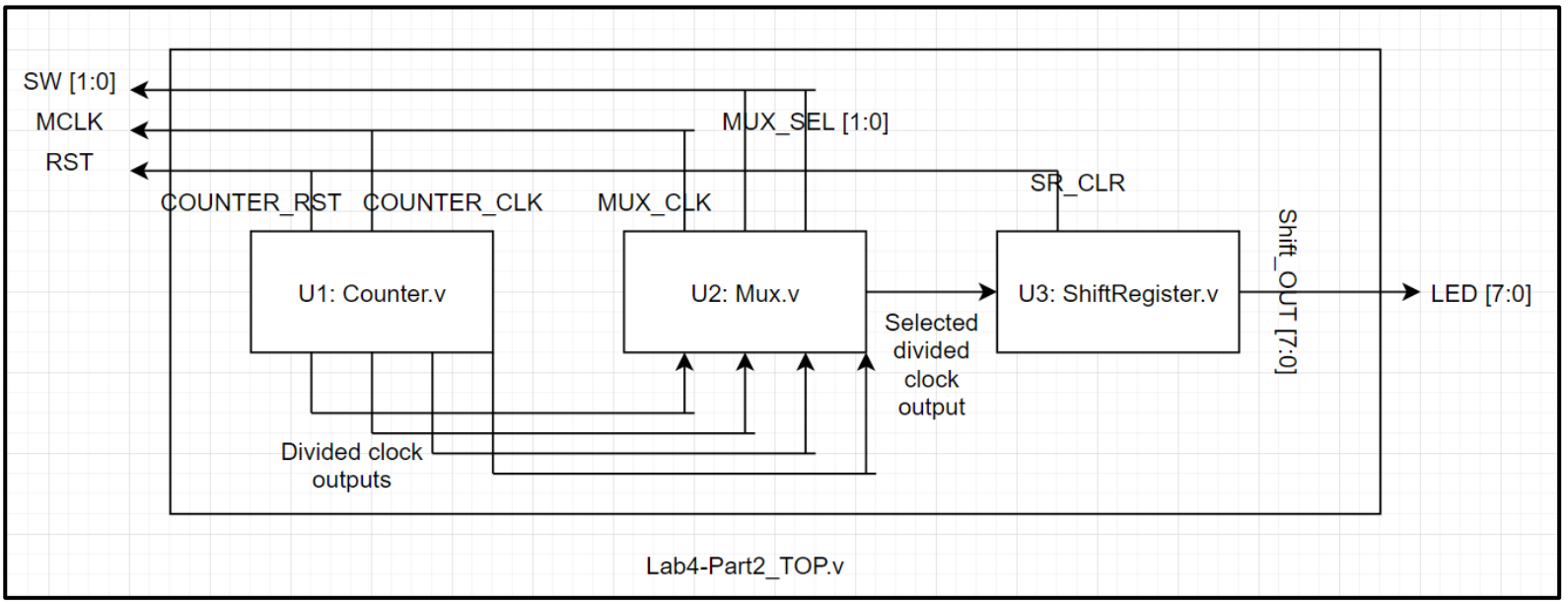



*Figure 12. Resulting block diagram for the Verilog "Lab4-Part2_TOP" module. The SW inputs are used as the select switches for the Verilog "Mux" module, which will control the shifting speed of the Verilog "ShiftRegister" module. MCLK input is used as main clock input for the Verilog "Counter" module so that it can be subdivided into four distinct clock outputs, and it is also used to avoid any timing warnings for the Verilog "Mux" module. RST input is used as a reset input for the Verilog "Counter" and "ShiftRegister" modules. The Verilog "ShiftRegister" module performs the shifting operation, and the speed of the shifting operation depends on the outputted selected divided clock from the Verilog "Mux" module. The LEDs are the resulting shifted outputs from the Verilog "ShiftRegister" module. The SW and RST inputs are connected to slide-switches on the Baysy2 board, MCLK connected to MCLK on Baysys2 board and LEDs connected to the eight LEDs on the Baysys2 board (done in .ucf file).*

After lab questions:

**Q1-** Improve the shift register by implementing an input `sh_ld` that loads a new value to the shift register when asserted low, otherwise shifts when asserted high. You need to implement 8 extra inputs named `data_in`. Your solution must provide a new Verilog module.

```
`timescale 1ns / 1ps

/*Shift-register module*/
module ShiftRegister(
      input SR_CLK, SR_CLR,  //Selected CLK-divide and RST inputs.
      input sh_ld,           //Load enable input.
      input [7:0] data_in,   //Shifting input.
      output [7:0] Shift_OUT //Shifting output.
    );

    //Register to store all the possible present-state shifted outputs below.
    reg [7:0] Present_8bits;
    //Wire to store all the possible next-state shifted outputs below.
    wire [7:0] Next_8bits;

     /*Data register*/
     always @(posedge SR_CLK or posedge SR_CLR or negedge sh_ld)
           begin
                 if (SR_CLR == 1)
                       Present_8bits <= 8'b00000001;

                 else if (sh_ld == 0)
                       Present_8bits <= data_in;

                 else
                       Present_8bits <= Next_8bits;
           end

           /*Next state logic*/
           //1-bit shift operation below.
           assign Next_8bits = {Present_8bits[6:0], Shift_OUT[7]};

           /*Output logic*/
           //Resulting shifted output below.
           assign Shift_OUT = Present_8bits;
endmodule
```