Lab#5: Sequential Counters in Verilog

Leonardo Fusser (1946995)

Purpose:

a) Extend the understanding in implementation and applications of counters.
b) Implement a synchronous system.
c) Understanding calculation of timing references in FPGA designs.
d) Using counter and flip-flops as 2N clock divider.

To hand in:

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

1. This sheet with answers copied in a word document.
2. Commented Verilog source files - module and top module for all parts.
3. Commented resource summary report with analysis for part2 only. Must be copied in a word document.
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. Part 2 block diagram using Visio or other application (your own block diagram, not an ISE generated diagram!).

Implementing a clock divider:

In Lab 4, we implemented clock division in a factor of 2^N

```
/* For a 50 MHz clk */
module cnter_presc(
          input clk,
          output div_out
    );

    reg [23:0] cnt;     // 24 bit register
    wire [23:0] cnt_next;  // Next data logic circuit

    /*Data Register*/
    always @(posedge clk )
          begin
                cnt<=cnt_next;
          end

    /* Next state data*/
    assign cnt_next<=cnt+1;


    /* Output logic */
    assign div_out=cnt[23];    // 50MHz / 2^(23+1) = 2.98Hz

    //assign div out=cnt[18];  // 50MHz / 2^(18+1) = 95.36 Hz
endmodule
```

In this lab, you will implement a clock divider that can control output frequency more precisely, in a factor of $2N$. Refer to theory.

Lab Work:

In your code, you must separate the system in five sections:

1. Internal connection wires declarations.
2. DFFs declaration.
3. Data register.
4. Next data logic.
5. Output logic and other logic.

All data register modules must be synchronized and as such they should have a shared mclk input.

Part 1: A one-second counter

1. Using the concept of pulse generator as explained in the theory part, you are required to design a pulse generator module `pulse_gen` that produces a 1 Hz pulse. The pulse width must be the period of the mclk.

```
module pulse_gen (
   input mclk,
   output Hz_1_pulse
   );
```

What would be the required N constant value for the pulse generator module to produce a 1Hz pulse? Provide details please:

- The required N constant value for the pulse generator module to produce a 1Hz pulse is 50'000'000. This is because 50'000'000 ticks is equivalent to 1 second, and the period for a 1Hz pulse is exactly 1 second (1/1Hz = 1 second). Therefore, in order to generate a 1Hz pulse, the N constant value must be set to 50'000'000.

2. Also, implement an 8-bit counter module: `counter_8_bits`

```
module counter_8_bits (
      input mclk,rst, pulse,
      output [7:0] reg count_reg
);
```

3. Create a new ISE top module named `lab5a`. Use the 1Hz pulse generated to drive an 8-bit counter and tie the outputs of this 8-bit counter to the 8 LEDs on Basys2 board.

   Both modules must be synchronized and as such they should have a shared mclk input.

   Your top module should contain two modules: `pulse_gen` and `counter_8_bits`.

4. Download to your Basys2 board for testing and verification. Demonstrate your working code to your instructor.

5. Modify the `pulse_gen` module to also generate a 96 Hz pulse:

```
module pulse_gen (
      input mclk,
      output Hz_1_pulse, Hz_96_pulse
);
```

6. Test the new pulse by downloading to your Basys2 board. Demonstrate your working code to your instructor.

Part 2: A Timer display on 7sd

In part 2, you are required to design a timer in the form of *mm.ss* (2-digit minutes, 2-digit seconds) that displays the counting on the 4 digits 7sd of Basys2 board. The timer will reset to 00.00 when it reaches 59.59.

You must reuse the 1 Hz pulse to count time and the 96 Hz pulse to multiplex the digits.

7. Using a CAD like Visio, draw a block diagram of the system. You must describe the details of all blocks inside the module – see the example from theory.

   Give a list of the requirements in terms of DFFs, counters, logic circuits and output logic circuits.

DFFs needed	Counters	Output and other logic circuits needed
79 DFFs	1S counter (for 1Hz pulse generator).	Pulse generator (for 1Hz and 200Hz pulses).
	5mS counter (for 200Hz pulse generator).	7 segment decoder (takes BCD value and convert it to a 7sd digit).
	Seconds-unit counter (for 1-hour timer/counter).	Multiplexer for 7sd anodes.
	Seconds-ten counter (for 1-hour timer/counter).	Multiplexer for BCD value (7sd digit).
	Minute-unit counter (for 1-hour timer/counter).	
	Minute-ten counter (for 1-hour timer/counter).	
	7sd multiplexer counter (for toggling between all four 7sd on Baysys2).	

8. Create/reuse all necessary Verilog modules. You might want to reuse the `bin2seg` module from a previous lab.

   Again, all data register modules must have a shared mclk input.

9. Create a new ISE top module named `top_timer_7s`:

```
module top_timer_7s(
      input mclk,
      input [0:0] btn,
      output [0:6] seg,
      output [3:0] an,
      output dp                  // dp separating min from sec
);

pulse_gen U1(
 . . .
);
count_mux U2(
 . . .
);

bin2seg U3(
 . . .
);
```

10. Generate an UCF file for this top module.

11. Compile and generate programming bit file. Download to your Basys2 board for testing and verification.

Demonstrate your working code to your instructor.

You must provide a commented resource summary report with analysis.

- Refer to "figure 1" below:

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	79	1,920	4%	
Number of 4 input LUTs	144	1,920	7%	
Number of occupied Slices	102	960	10%	
Number of Slices containing only related logic	102	102	100%	
Number of Slices containing unrelated logic	0	102	0%	
Total Number of 4 input LUTs	196	1,920	10%	
Number used as logic	144			
Number used as a route-thru	52			
Number of bonded IOBs	14	83	16%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	2.79			

Figure 1. Resource summary report for Part 2 of Lab 5 (1-hour timer/counter). 79 DFFs are used in total (54 in use in pulse_gen module, 18 in use in Mux module and 7 in use in bin2seg module). 14 IOBs are used in total (1 for MCLK, 1 for RESET, 7 for seg , 1 for dp and 4 for an). 196 LUTs are used to accommodate all the logic truth tables across the pulse_gen, Mux and bin2seg modules. 102 slices are use, each slice having 2 LUTs and 2 DFFs, which are plenty to satisfy the 196 LUTs and 79 DFFs demand.

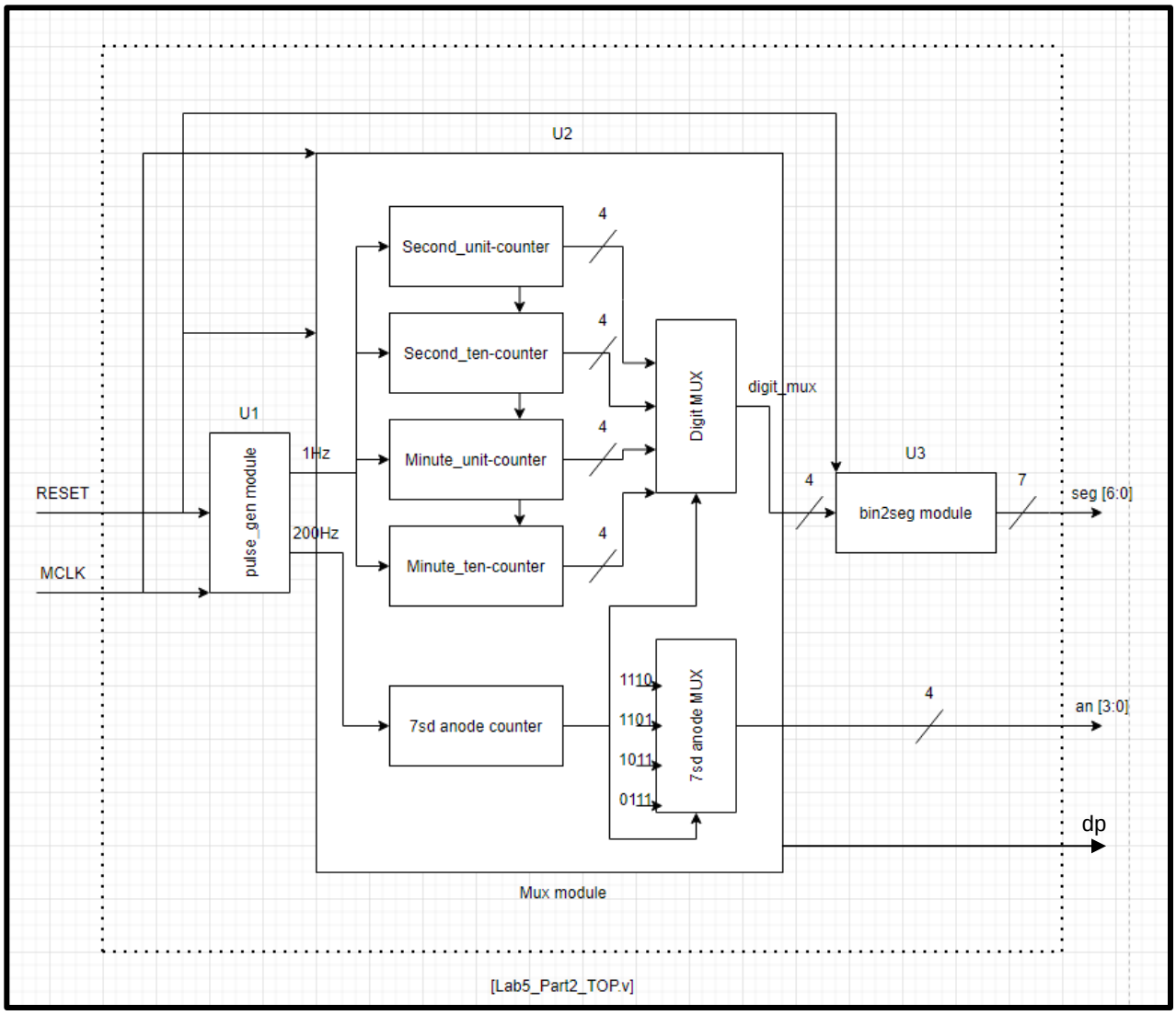

Figure 2. Block diagram for Part 2 of Lab 5 (1-hour timer/counter).

[Verilog code for Part 1]

Refer to code for pulse generator in “pulse_gen.txt” attached to this report submission.
Refer to code for 0-255 counter in “counter_8_bits.txt” attached to this report submission.
Refer to code for top module in “Lab5_Part1_TOP.txt” attached to this report submission.

[Verilog code for Part 2]

Refer to code for pulse generator in “pulse_gen.txt” attached to this report submission.
Refer to code for 1-hour counter/1-hour timer and select 7sd multiplexer with decimal point in “Mux.txt” attached to this report submission.
Refer to code for binary-to-segment decoder in “bin2seg.txt” attached to this report submission.
Refer to code for code for top module in “Lab5_Part2_TOP.txt” attached to this report submission.

[Constraints file for Part 1 and Part 2]

Refer to code for constraints file in “Lab5_constraints.txt” attached to this report submission.