**EE260 Lab 6 Sign-Off Sheet**

Your Name:

Lab Partner: Date Performed:

## Demonstrated correctly:

•Pre-lab Complete (1) (2) (10 pts)

* + 10 Hz clock flashing an LED (10 pts)
  + Stand alone Mod 10, 10th of second counter (15 pts)
  + Single seconds and 0.1 sec display (15 pts)
* Four digit stopwatch (15 pts)
* TA Questions: (1) (2) (5 pts )

Each student graded individually

* Report (one per team) (30 pts)

(including Verilog code, Verilog test benches and simulation screen shots,

constraint files, pre-labs and grading rubric)



**EE260: Introduction to Digital Circuit Design**

**Lab 6 – Using Sequential Logic Circuits**

**A Digital Stop Watch**

# Objective:

In this lab you will create a digital stop watch capable of counting and displaying time from 0 to 10 minutes with 0.1 second accuracy. This lab will use the 4 digit 7-segment display that you created in your last lab. You will implement this lab completely in Verilog and will again need to use some sequential Verilog constructs.

# Pre-lab Assignment:

This pre-lab assignment is to be completed before your lab session and must be

signed-off by the TA during your lab session. Pre-labs help you to become oriented to the problem before you enter lab, help complete your design in advance and prevent wasting time in lab.

1. READ the whole lab assignment!
2. This lab will require a 10 Hz clock. First, explain how a 10 Hz clock will allow us to measure 0.1 sec.
3. Using clkDiv10K.v as a model, create a clkDiv1K.v module that divides down the fequency of the input clock by 1000. Verify that if we cascade the *clkDiv10K.v* followed by the new *clkDiv1K.v* this will result in the desired 10 Hz clock.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 100MHz  System | clkDiv |  | clkDiv |  |
| Clock | 10K |  | 1k | 10 Hz Clk |

3) This lab also requires a 4 bit binary counter. As we saw last week, a "Binary Counter" is a sequential circuit that generates a repeating binary sequence on its outputs. The value on the output of a binary counter advances by 1 count with each rising clock edge. Thus this circuit keeps a "count" of the clock periods that have passed. If the clock period is set to a convenient unit of time, like 0.1 seconds, we can create a human readable clocks displaying seconds, minutes, hours, years, etc all by using different sized binary counters or by combining multiple smaller binary counters. In this lab we will use four 4-bit binary counters. When enabled a 4-bit counter will generate the sequence of binary outputs from 0000, 0001, 0010 to 1111. Using the Verilog code below as a model, write the Verilog code to implement a 4-bit counter. Your module should have 3 single bit inputs *clk, CE* and *Reset* and a 4-bit output called *Q*. The function table for a 4-bit counter is shown below.

clk

4-bit Binary Counter

Q(3:0)

Reset

CE

## Function Table for a 4-bit Binary Counter

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Inputs*** | | | ***Outputs*** | | | |
| ***Reset*** | ***CE*** | ***CLK*** | ***Q*3** | ***Q*2** | ***Q*1** | ***Q*0** |
| 0 | 0 | x | Last ***Q*3** | Last ***Q*2** | Last ***Q*1** | Last ***Q*0** |
| 1 | x | ⇧ | 0 | 0 | 0 | 0 |
| 0 | 1 | ⇧ | 0 | 0 | 0 | 1 |
| 0 | 1 | ⇧ | 0 | 0 | 1 | 0 |
| 0 | 1 | ⇧ | 0 | 0 | 1 | 1 |
| 0 | 1 | ⇧ | 0 | 1 | 0 | 0 |
| 0 | 1 | ⇧ | 0 | 1 | 0 | 1 |
| 0 | 1 | ⇧ | 0 | 1 | 1 | 0 |
| 0 | 1 | ⇧ | 0 | 1 | 1 | 1 |
| 0 | 1 | ⇧ | 1 | 0 | 0 | 0 |
| 0 | 1 | ⇧ | 1 | 0 | 0 | 1 |
| 0 | 1 | ⇧ | 1 | 0 | 1 | 0 |
| 0 | 1 | ⇧ | 1 | 0 | 1 | 1 |
| 0 | 1 | ⇧ | 1 | 1 | 0 | 0 |
| 0 | 1 | ⇧ | 1 | 1 | 0 | 1 |
| 0 | 1 | ⇧ | 1 | 1 | 1 | 0 |
| 0 | 1 | ⇧ | 1 | 1 | 1 | 1 |
| 0 | 1 | ⇧ | 0 | 0 | 0 | 0 |
| 0 | 1 | ⇧ | ... | ... | ... | ... |

module bin\_cnt4( input clk, input Reset, input CE, output [3:0] Q

);

reg [3:0] cnt;

// This module implements a 4-bit binary counter with a

// count enable (CE) and synchronous reset (

always @ (posedge clk) if (Reset == 1)

begin

cnt = 4'b0000; end

else begin if (CE)

begin

if (cnt == 4'b0000) begin

cnt = 4'b0001; end

else if (cnt == 4'b0001) begin

cnt = 4'b0010; end

. . .

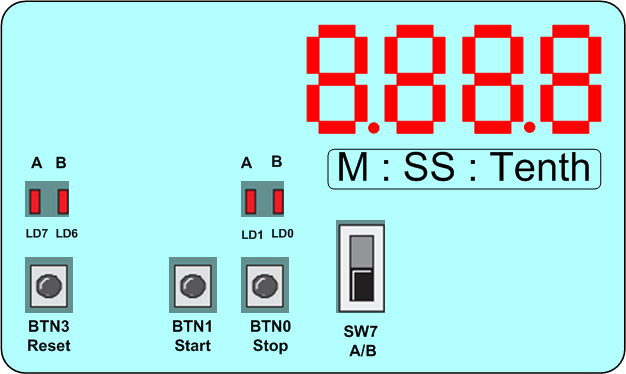
else if (cnt == 4'b1111) begin

cnt = 4'b0000; end

end // if (CE) end // else begin

assign Q = cnt; endmodule

# Lab Assignment:

Figure 1 shows the overall picture of the digital stop watch you will implement using the Basys 3 board. The most significant digit displays minutes, the two middle digits represent tens of seconds and seconds respectively; and the least significant digit displays tenths of a second (e.g. 4.45.2 = 4 min 45.2 sec). You will also use a switch to start, stop and a button to reset the stop watch as follows:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | |  |
|  |
| **BTNR** Start/Stop  **Reset** | | |

***Start***: Activating this switch enables the stopwatch to start or resume counting. If the stopwatch was originally idle, it starts counting from 0. If the stopwatch was stopped, it resumes counting from the last time.

***Stop***: If the stopwatch is idle, this switch does not have any effect but if the stopwatch is counting sliding this switch to 0 stops the counter at its current count.

***Reset***: This button resets the count to 0.00.0 regardless of whether the stopwatch is currently counting.

If you examine the function table for the 4-bit binary counter above you will see that the desired behavior for Start and Stop can be achieved by appropriately using the count enable *CE* input while the Reset behavior can be achieved direclty from the *Reset* input. To implement this 4 digit stopwatch we will use four 4-bit binary counters cascaded together with additional combinational logic to control the *CE* and *Reset* of each digit.

1. Create a new project in Vivado and then add add the *clkDiv10K.v* module from the class website to your project.
2. Add a new Verilog module called *clkDiv1K.v* and enter you clock divide by 1,000 code from pre-lab. Like *clkDiv10K.v* this module should have one single bit input called *clk\_in* and a single output called *div\_clk*. Save your project.
3. Add a new Verilog module toyour Sources for Design called *tenHzFlash.v* and make it the Top Module. This module should have a single input, *CLK* and a single output c*lk10Hz.* Cascade instantiations of *clkDiv10K* and *clkDiv1K* to create a 10 Hz clock.
4. Add a constraint file to your project that connects *CLK* to the 100 MHz system clock on W5 and connects output *clk10Hz* to an LED. Blinky lights are always cool. Save your project and generate a bit file. ***Show your 10 Hz clock to the TA for sign-off.***
5. Add a new Verilog module to your project under Design and call it *bin\_cnt4.v.* Imple- ment your 4-bit binary counter from pre-lab.
6. Add a Verilog test bench to your project and associate it with *bin\_cnt4.v*. Simulate at least 24 to 32 clock periods by alternately setting *clk* = 0 and *clk* = 1 for 1000 ns. *Reset* should equal 1 for at least 4 different rising clock edges and CE should equal 0 for at least 4 rising clock edges. Run the simulation of your 4-bit counter. Verify that your counter has a *Synchronous Reset*, that is the output count goes to 0000 with the rising edge following *Reset*=1 and not immediately. Save your project. ***Be sure to include a screen capture of your counter test bench results and anwer the Reset question in your report****.*
7. The output of 4-bit counter advances by 1 with each rising clock edge and will count from 0000, 0001, 0010, etc. upto 1111 or equivantly from 0 to F hex. In our stopwatch we want to count 10th of a second, seconds, 10s of second and minutes and we want to do this in a base 10 positional display. For example, the least significant digit of the stopwatch should count from .0 to .9 and then "roll over" to .0 again. We need to add logic to the output of the 4-bit count to make it reset after reaching a count of 9. The resulting circuit is called a *Modulo 10 binary counter.*

*clk10Hz Q0*

*4-bit binary cntr*

*Q1*

*Reset = R\_out*

*Q2*

*Q3*

*CE=1 R\_out = Q3 Q2 Q1 Q0 + R*

## Modulo 10 Bibary counter

Add a new Verilog module to your project called *mod10cnt.v.* and set it as Top. The module should have three single bit inputs CLK, *R,* and *CE* and one 4-bit output *Q* and one single bit output *R\_out*. Instantiate your 4-bit binary counter as shown below then add the R\_out logic given above to the module and save your project.

bin\_cnt4 U1(CLK,R\_out,CE,Q); assign R\_out =

1. Add your *hex2bin.v* with the decimal point and sign from last lab to your project. Then add a new Verilog module called *tenthsDisplay.v* and make it the top module. This module should have 3 inputs *CLK, R,* and *CE* and 3 outputs 7-bit *segs*, single bit *DP* and 4-bit *anodes*. Complete the following instantiations.

wire clk\_10K, clk\_10Hz, R\_out; wire [3:0] Q;

// Divide the system clock to 10KHz (see lab 4)

. . .

// Now divide the 10KHz clock to 10Hz using your new clkDiv1K

. . .

// modulo 10 counter counting 10th of seconds mod10cnt U3(clk\_10Hz,R,CE,Q,R\_out);

// Display output of counter without decimal point hex2seg U4(Q,0,0,segs,DP);

assign anodes = 4'b1110';

1. Remove but don't delete the old constraints file and add a new *.xdc* file for *tenthsDisplay.v*. Input *R* is tied to BTNR and *CE* to SW0. You will need to slide switch 0 high to enable counting. CLK should be connected to the system clock (W5). Outputs *segs*, *DP* and *anodes s*hould be connected to CA thru CG, DP and AN0-AN3 as you have done before.
2. Save your project then make a .bit file. Test you counter then

***Show your 10th digit***

## counter with Reset and CE to the TA for sign-off.

1. To create a positional multidigit display we will need to cascade one 4-bit timer for each digit. However, we need to be careful how we clock and enable the counters to achieve a properly incrementing multidigit display. Basically, the enable and reset for the more significant digits will depend on the reset of the less significant digits. For example the seconds digit should only be enabled to count when the 10ths digit has reached its full count (i.e. .9). Likewise the 10s of seconds digit should only be enable when the seconds and tenth are at full count (i.e. 9.9). We shall start by cascading counters for the seconds and 10th of seconds digits. Add a new Verilog module called *secondsDisplay.v* and make it the top module. The module should have 3 inputs *CLK, R,* and *CE* and 3 outputs 7-bit *segs*, single bit *DP* and 4-bit *anodes*.
2. Add your *fourDigitDisplay.v* from last lab (***be sure*** all the sub-modules are added, too!) and save your project.
3. Complete the following instantiations in to *secondsDisplay.v.*

wire clk\_10K, clk\_10Hz, R\_out1, R\_out2; wire [3:0] Qth,Qsec;

// Divide the system clock to 10KHz (as you did above)

. . .

// Divide the 10 KHz clock to 10Hz (as you did above)

. . .

// modulo 10 counter – for 10ths digit mod10cnt U3(clk\_10Hz,R,CE,Qth,R\_out1);

// modulo 10 counter – for seconds digit mod10cnt U4(clk\_10Hz,R,R\_out1,Qsec,R\_out2);

// Display output of 2 digit counter with decimal point

**// Check your fourDigitDisplay code for the order of inputs**

// Use the system CLK as the input clock and turn on the

// decimal point on the seconds digit only fourDigitDisplay U5(4'b0000,4'b0000,Qsec,Qths, ... ,CLK,

segs,DP,anodes);

1. Associate the constraints file for *tenthsDisplay.v* with *secondsDisplay.v*. You may have to remove it then add it back in. Make sure *secondsDisplay* is Top Module. Again *R* is tied to BTNR, *CLK* is W5 and *CE* is SW0. The outputs *segs, DP* and *anodes* are all connected as per usual.
2. Make a .bit file and save your project. ***Show your 2-digit counter with Reset to the TA for sign-off.***
3. Conceptually we can add the other 2 digits in the same fashion. Unfortunately, time is measured with 60 seconds per minute instead of 100. We need to have the 10s of seconds digit count from 0 to 5 instead on 0 to 9. This is modulo 6 behavior. Add a new module to your project called *mod6cnt.v.* Using *mod10cnt.v* as a guide make this module a binary counter that resets after reaching a count of 5 = 4'b0101. Save your project.
4. Add a new Verilog module called *stopWatch.v* and make it the top module. The module should have 3 inputs *CLK, R,* and *CE* and 3 outputs 7-bit *segs*, single bit *DP* and 4-bit *anodes*. Using *secondsDisplay.v* as a guide make the full 4 digit stopwatch. Associate the contraints file from above with this new top module (delete and re-add if necessary).

wire clk\_10K, clk\_10Hz, R\_out1, R\_out2, R\_out3, R\_out4; wire [3:0] Qth, Qsec, Qsec10, Qmin;

// Divide the system clock to 10KHz (as you did above)

. . .

// Divide the 10 KHz clock to 10Hz (as you did above)

. . .

// modulo 10 counter – for 10ths digit mod10cnt U3(clk\_10Hz,R,CE,Qth,R\_out1);

// modulo 10 counter – for seconds digit mod10cnt U4(clk\_10Hz,R,R\_out1,Qsec,R\_out2);

// Add a modulo 6 counter – for 10s of seconds digit

// Be careful about the logic for CE input (see class notes)

. . .

// Add a modulo 10 counter for minutes (0-9)

// Be careful about the logic for CE input (see class notes)

. . .

// Display output of 4 digit counter with decimal point

**// Check your fourDigitDisplay code for the order of inputs**

// Use the system CLK as the input clock and turn on the

// decimal points on for min and sec digits

fourDigitDisplay U7(Qmin,Qsec10,Qsec,Qths, ... ,CLK, ...

segs,DP,anodes);

1. Make a .bit file and save your project. ***Show your Stopwatch with Reset and Stop/Start to the TA for sign-off.***