Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.111 – Introductory Digital Systems Laboratory

Problem Set 2

Problem Set Issued: Feb. 18, 2004
Problem Set Due: Feb. 27, 2004

Comments: This problem set is intended to prepare you for Laboratory 2 which is a Traffic Light Controller and Memory Tester. By working through this problem set you should be able to design simple counter and memory circuits in both hardware and software, design finite state machines, and design and use a testbench to simulate your own Verilog code in ModelSim.

Verilog coding and its simulation in ModelSim are major components of this lab. A tutorial for using ModelSim is provided in the Software Tools section of the course webpage.

We expect this problem set to take significantly more time than Problem Set 1 so please start early. The upside to the extra work put into this problem set is that you will be able to reuse some of the Verilog modules created here for your Lab 2 (namely the one second timer and the memory tester).

Problem 1: Counters

Part (a)
Using 74LS163 counters and a 1.8432 MHz clock design a circuit that divides the clock by counting to 128 thus producing a 14.4 KHz clock. To verify your design you may want to build this simple circuit using your lab kit although this is not a requirement. The 74LS163 is a four-bit counter whose data sheet can be found on the course webpage under Lab1.

Hint: 128 factors into 16 and 8.

Part (b)
How many LS163s would it take to produce a 1 Hz clock?

Hint: What is the binary representation of 1843200?

Part (c)
What is the difference between an LS393 and an LS163?

Part (d)
In this part of the problem you will be creating a Verilog module which will be used later in this problem set and in Lab 2.

Create a Verilog module that takes as input a 1.8432 MHz clock and outputs a 1 hz enable signal that pulses high for only one cycle of the 1.8432 MHz clock and is low otherwise. That is to say that the output is NOT a 50% duty cycle signal but a pulse every 1 second. Don’t forget a reset signal which will reset the count back to zero when asserted.

Submit your code for this part.

Problem 2: Finite State Machines

You are an engineer working for NASA. They want you to design a FSM to test their newest rover Stata (named after the new Course VI building) around the MIT campus. **You should assume the rover starts at Killian court.**

Stata will be required to go to the following locations (their 3-bit binary representations are listed as well): Killian (000) Kresge (001), Tang (010), Z-Center (011), Student Center (100), Building 34 (101), LCS (110), and appropriately the Stata Center (111).

From any state the robot can only travel to one of two other states. The information which tells the robot which location to visit next is wirelessly transmitted to the robot's FSM by NASA. The output of the FSM is the current state of the robot.

The movements from location to location are as follows.

Killian:	If 0 stay at Killian.	If 1 go to Kresge.
Kresge:	If 0 go to Stata Center.	If 1 go to Tang.
Tang:	If 0 stay at Tang.	If 1 go to the Student Center
Z-Center:	If 0 go to the Student Center.	If 1 stay at the Z-Center
Student Center:	If 0 go to the Z-Center.	If 1 go to LCS.
Building 34:	If 0 go to LCS	If 1 go to Stata Center.
LCS:	If 0 stay at LCS.	If 1 go to Building 34.
Stata Center:	If 0 go to Tang.	If 1 go to Killian

Part (a)
Draw the state transition diagram for this FSM.

Part (b)
If the rover is given the sequence 10001 write down the path (MIT locations) the rover visits.

Part (c)
If the rover is forever given a sequence of ones what location will it never visit?

Part (d)
Design a module in Verilog for the rover's FSM (fsm.v). Submit your code for this part.

Problem 3: Verilog Testbench

In this question you are asked to link some of the Verilog modules you have created so far in this problem set and verify that they are working properly by using a testbench that you will generate.

First you should create a file called top.v. In this file instantiate your counter module from Problem 1.d and your FSM from Problem 2.d. The output of this module should be the current state of your FSM.

The one hz enable signal which is an output from your counter module should be the input enable for your FSM. You can specify the passing of signals from one module to another by declaring this signal as a wire in the top level file. Feel free to use a enable signal faster than one hz for simulation purposes. That is to say that for the simulation you could run off say a 230.4 khz enable signal (1.8432MHz/8).

Next create a file called testbench.v with the timescale command:

```
`timescale 1ns/10ps
```

This file should instantiate your top level module (top.v) and verify that it is functioning properly.

This instantiation might look something like:

```
top top1
(
.clk(clk),
.reset(reset),
.fsm_input(fsm_input),
//insert other inputs if necessary
.fsm_state(fsm_state)  //this is the output from the fsm module and is the current state
);
```

To generate an approximate 1.8432 MHz clock in your testbench you can use the command:

```
always #542 clk = ~clk
```

which says that every 542ns the clock signal invets itself. We know that the time units are in nanoseconds as we have already specified this in the timescale statement above.

In your testbench have the rover take the path outlined in 2(b).

You can do this by using a series of commands that might look something like

```
initial begin
    clk = 0;
    reset = 0;
    foo_input = 0;
    //initialize all other inputs to zero
    #2000
    reset = 1;
    fsm_input = 1;
    #2000
    reset = 0;
    // continue to take the rover through path of 2(b)...
    // ...
  end
```

Verify that your rover is transitioning properly from location to location by viewing the Wave Window.

We ask that you submit a printout of your code (top.v and testbench.v) as well as a screen capture of the wave window demonstrating your rover properly transitioning from state to state.

Problem 4: Memory Tester

Now that your have helped NASA successfully navigate the MIT campus they have asked for your help in reading and writing to the memory on the rover. This problem asks you to design a memory tester that can write to memory and then read these values to validate their accuracy. For this problem you will be using the HM6264 SRAM whose datasheet can be found on the course webpage under Lab2.

You have been told that the rover memory uses four address-bits (assume that the higher order address bits are grounded) and four I/O pins (ignore the higher order bits). The memory tester for this problem will write to each address the binary value of that address. For example if the address is 4’b0010 then the value stored in memory should be 4’b0010.

Create a success output signal and a failure output signal from your memory tester

The following approach should be implemented for the memory tester:

- Write all 16 locations starting with address 0. Location 0 gets the value of 4’b0000, location 1 gets the value 4’b0001, 2 gets the value 4’b0010, etc…
- Read the locations from 0x0 to 0xF and verify that the correct values were written. If the correct values were written your success output should go high otherwise failure is high.

For your memory tester create a testbench that demonstrates the functionality described in the bulleted points above.

Submit your code and a simulation screen capture for this part. While the memory tester for Lab 2 is somewhat more complicated than the one implemented in this problem you should be able to reuse a large portion of your code here during Lab 2.