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

Problem Set 2

Problem Set Issued: Feb. 14, 2005 **Problem Set Due:** Feb. 25, 2005

NOTE: This problem set is intended to prepare you for the two parts of Laboratory 2, Traffic Light Controller and Memory Tester. You should be able to reuse or slightly modify modules created here for your Lab 2 designs. After completing this problem set, you should be able to:

- ✓ Design simple counters in software and hardware.
- ✓ Design finite state machines (FSM) in software.
- ✓ Create a testbench to simulate and verify your designs in ModelSim or Max+ Plus II.
- ✓ Understand how a 6264 SRAM works.

This problem set is longer than the first one, so please start early! Tutorials for ModeslSim and Max+ Plus II are available at the course website under 'Software Tools'.

Problem 1: Counters

a) Describe the differences between the **`74LS393`** and **`74LS163`** 4-bit counters.

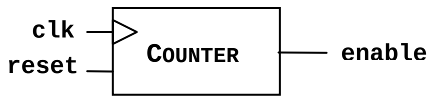
b) Using **`74LS163`** 4-bit counters, design a circuit that divides a 1.8432 MHz clock by counting up to 64, thus producing a 28.8 kHz clock.

Hint: 64 factors into 16 and 4.
Note: The **`74LS163`** datasheet can be found on the course website under 'Labs'. To verify your design, you may want to build this simple circuit using your lab kit, although this is not a requirement.

c) Using a 1.8432MHz clock, how many **`74LS163`**'s are required to produce a 1Hz clock?

Hint: What is the binary representation of 1843200?

d) Design a module, **`counter`**, in Verilog that takes two inputs, the 1.8432 MHz **`clock`** and **`reset`**, and outputs a 1Hz **`enable`** signal. The **`enable`** signal should pulse high only for one period of the 1.8432 MHz clock and stay low otherwise. The **`enable`** signal is NOT a 50% duty cycle signal. The **`reset`** signal resets the count back to 0 when high. **Submit your code for this part.**



Note: This will be used in Lab 2.

Problem 2: Finite State Machines (FSM)

You are an engineer working for NASA. They want you to design a FSM that will test their newest rover Stata (named after the new Course VI building) on the MIT campus. NASA wirelessly transmits the travel plans to Stata, and then Stata moves according to that information.

To design your FSM, you first select the following locations around the MIT campus and assign each location with a state in 3-bit binary representation: Killian`[000]`, Kresge`[001]`, Z-Center`[010]`, Syd-Pac`[011]`, Student Center`[100]`, Building 34`[101]`, 6.111 Lab`[110]`, and appropriately the Stata Center`[111]`.

To simplify your test, you inform NASA to send Stata's FSM a binary sequence for travel plans (e.g.'1-0-0-0-1' to cause Stata to move five times). In other words, Stata receives either '0' or '1' for each move and travels to the next destination as specified below. Stata starts off at Killian Court for each test run, and your FSM should output Stata's current location.

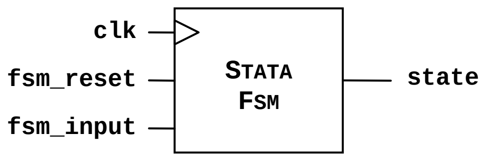
Killian:	If 0, stay at Killian.	If 1, go to Kresge.
Kresge:	If 0, go to Z-Center.	If 1, go to Student Center.
Z-Center:	If 0, go to Syd-Pac.	If 1, go to Student Center.
Syd-Pac:	If 0, stay at Syd-Pac.	If 1, go to Killian Court.
Student Center:	If 0, go to Stata Center.	If 1, go to Building 34.
Building 34:	If 0, go to Syd-Pac.	If 1, go to 6.111 Lab.
6.111 Lab:	If 0, go to Stata Center.	If 1, stay at 6.111 Lab.
Stata Center:	If 0, go to Kresge.	If 1, go to Building 34.

a) Draw the state transition diagram for this FSM.

b) If Stata is forever given a sequence of ones (i.e. 11111…), where will it eventually end up?

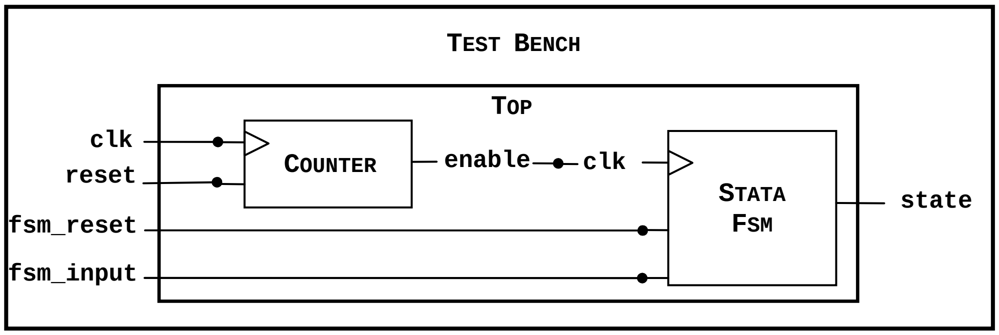
c) If Stata is forever given a sequence of 01s (i.e. 010101…), which location will it never visit?

d) Design a module in Verilog for this FSM. **Submit your code for this part.**



Problem: Verilog Testbench

In this problem, you will link two Verilog modules you have created and create a testbench to verify that the two modules work correctly as shown below:



First create a file called **`top.v`**, in which you instantiate the **`counter`** module from Problem 1d) and the **`Stata_FSM`** module from Problem 2d). Next create a file called **`testbench.v`**, in which you instantiate the **`top`** module and specify the timescale as below such that time units are in nanoseconds.

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

In your testbench, Stata should take the path outlined in Problem 2b). When it stays in one location more than three times, it should return to Killian using **`fsm_reset`** and take the path outlined in Problem 2c). Verify that Stata is transitioning properly from location to location by viewing the Wave Window and verify your answers to 2b) and 2c). **Please submit a printout of your file, `top.v` and `testbench.v`, and a screen capture of the wave window demonstrating Stata properly transitioning from state to state.**

Note1: To check that Stata stays in one location more than three times, just look at your output (state) waveform, change the fsm_reset signal in the testbench at the appropriate time, and apply the input sequence from 2c), as opposed to trying to automatically detect if Stata stays in one location more than three times.

Note2: For simulation purposes, feel free to use an **`enable`** signal faster than 1 Hz. For example, you could use a 230.4 kHz enable signal (1.8432MHz/8) for a faster simulation.

Hint1: Instantiation of the **`top`** module might look something like:

```
Top T(.clk(clk),.reset(reset),.fsm_input(f_input),.fsm_state(f_state));
```

Hint2: Generating an approximate 1.8432 MHz clock in the **`testbench`** might look something like:

```
always #271 clk = ~clk;
```

This command makes the **`clock`** signal toggle every 271ns.

Hint3: An example of testbench code can be found in slide 27 from lecture 3.

Problem 4: Memory Tester

a) Understand how the **`6264 SRAM`** works by reading the datasheet, which can be found on the course website under 'Labs'. Then, fill in the following truth table:

$\overline{E1}$	E2	$\overline{G}$	$\overline{W}$	**Mode**	**Output**	**Cycle**
H	X	X	X	N/S		-
X	L	X	X	N/S	High-Z	-
L	H	H	H		High-Z	-
L	H	L	H			
L	H	L*	L	Write		Write cycle
L	H	X	L		High-Z	

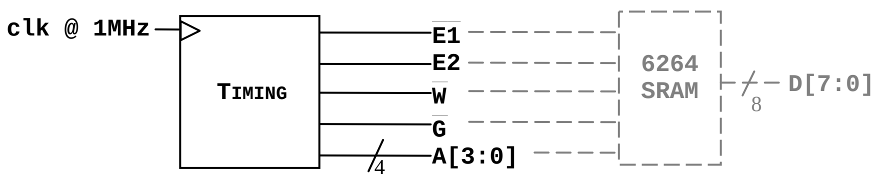
* Mode is **`write`** and thus assume **`~G`** goes low coincident with or after **`~W`** goes low.

b) Design a module, **`timing`**, that generates the input signals (**`~E1,E2,~W,~G,A`**) for the 6264 SRAM which accomplish one cycle of the following operations.

1. Write the 4-bit binary value of the address to the first 16 memory locations, starting with address 0. (i.e. write **`4'b0000`** to location 0, **`4'b0001`** to location 1, **`4'b0010`** to location 2, etc …)

2. Then, read the values from locations **`4'h0`** to **`4'hF.`**

Note: For signal **`A`**, generate only the first four LSB signals, A[3:0], ignoring A[12:4]. **Submit your code and a screen capture of the waveform window for this part.**


clk @ 1MHz
TIMING
$\overline{E1}$
E2
$\overline{W}$
$\overline{G}$
A[3:0]
4
6264
SRAM
8
D[7:0]