Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.111 - Introductory Digital Systems Laboratory
Laboratory 2 Check Off Sheet

Student Name:

TA Signature/Date:

Part 1: Traffic Light Controller
Must Show to TA at beginning of Chekoff

- FSM State Transition Diagram ☐
- Verilog Code Printout ☐

Be Able to Demonstrate Your Working Lab

- You will be first asked to demonstrate regular operation with default values ☐
- You will be asked to reprogram your time values and continue operation ☐
- You will be asked to demonstrate functionality of Walk Request Register ☐
- You will be asked to demonstrate functionality of the side sensor ☐

Be Able to Respond to any of the Following Questions (and possibly others). You will likely be asked two questions from the following by a TA ☐

- What could happen if an input were not synchronized to the clock?
- Describe your synchronizer module and why it is important.
- Describe your walk request Register.
- Describe your divider module.
- What is the difference between a Moore and a Mealy machine?
- Describe the design flow for your Traffic Light Controller.

Student Name:
TA Signature/Date:

Part 2: Memory Tester
Must Show to TA at beginning of Chekoff

- Block diagram of memory tester ☐
- Verilog code printout ☐

Be Able to Demonstrate Your Working Lab

- From reset, demonstrate writing and reading from different locations ☐
- Demonstrate operation with the MSB of the DATA pin disconnected (D[3]) ☐

Be Able to Respond to any of the Following Questions (and possibly others). You will likely be asked two questions from the following by a TA ☐

- What are possible problems with the address or data glitching when the write on the memory is enabled?
- Describe your timing for the memory interface.
- Can you think of a faulty circuit connection between the memory and FPGA that will pass the test?
- What are the limitations of the proposed simple memory test?

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

Laboratory 2

Issued: February 18, 2004
Checkoff Due: March 5, 2004
Report Due in 38-107 by 11AM: March 8, 2004

Part 1: Traffic Light Controller

Introduction

Part 1 of this lab is a traffic light controller that controls a main street, side street and walk lamps. You will be using a finite state machine to implement this controller. This lab provides you with a design methodology that will be useful in future labs and final projects. This involves planning your design, coding, wiring, and debugging your design.

Procedure

There are two major phases. The first is the design phase, which consists of reading through the lab, planning, and coming up with a design. Although not required, it is suggested that you schedule a conference with your TA to review your design. This will help catch any major mistakes early in the process.

The next phase is to implement the first part of the lab using the FPGA. After you verify the traffic light controller's functionality, you can get checked off for part 1 of the lab. Be ready to demonstrate part 1 of the lab, and be ready to present solutions for the problems asked in the checklist.

You will be required to write a detailed report (see guidelines for lab 2 report at the end).

Traffic Light Controller Description

The traffic light controller is for an intersection between a Main Street and a Side Street. Both streets have a red, yellow, and green signal light. Pedestrians have the option of pressing a walk button to turn all the traffic lights red and cause a single walk light to illuminate. Lastly, there is a sensor on the Side Street which tells the controller if there are cars still on the Side Street. This is summarized in Figure 1.

You may assume that the 4 walk buttons placed at each street corner are hooked into the traffic light controller using a wired-OR. For this reason, you may assume that the controller only needs a single input called *Walk-Request*.

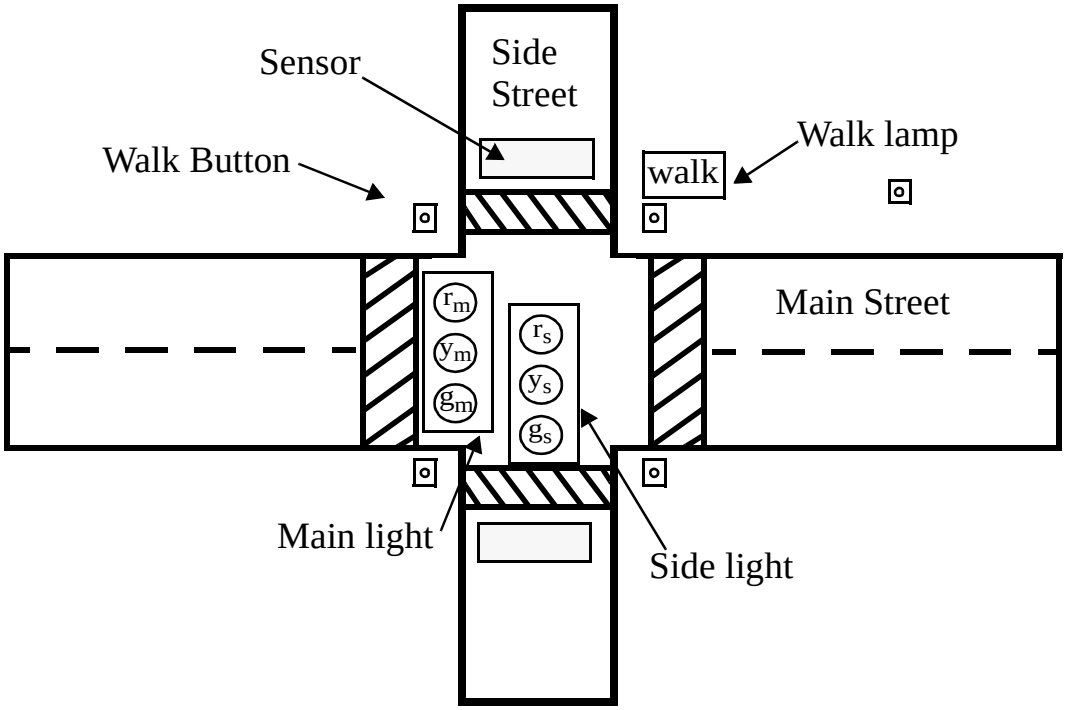

Figure 1: Diagram for intersection with corresponding lights.

Table 1: Default Timing Parameters.

Interval Name	Symbol	Parameter Number	Default Time (s)	Time Value
Base Interval	t_{BASE}	00	6	0110
Extended Interval	t_{EXT}	01	3	0011
Yellow Interval	t_{YEL}	10	2	0010

The side street sensor is placed near the intersection to tell the controller when there are cars passing over the sensor. You may assume the sensor remains constantly high if several cars pass over the sensor, rather than quick pulses, provided the cars are close enough together. You do not need to implement this specific functionality. This input is named *Sensor.*

The traffic lights are timed on three parameters (in seconds), the base interval (t_{BASE}), the extended interval (t_{EXT}), and the yellow light interval (t_{YEL}). The default values listed in Table 1 are to be loaded into the FPGA on reset, and may be reprogrammed on demand using switches and buttons on your kit with the *Time_Parameter_Selector*, *Time_Value*, and *Reprogram* signals. *Time_Parameter_Selector* uses the Parameter Number code to select the interval during programming. *Time_Value* is a 4-bit value representing the value to be programmed; therefore, it has a

duration of seconds between 0 and 15. The *Reprogram* button tells the system to set the currently selected interval to *Time_Value*.

The operating sequence of this intersection begins with the Main Street having a green light for 2 lengths of t_{BASE} seconds. Next, the Main lights turn to yellow for t_{YEL}, and switches to the Side Street green light. The Side street is green for t_{BASE}, and its yellow is held for t_{YEL}. Whenever a stoplight is green or yellow, the other street's stoplight is red. Under normal circumstances, this cycle repeats continuously.

There are two ways the controller can deviate from the typical loop. First, a walk button allows pedestrians to submit a walk request. This signal should set on a button press and the controller should service the request after the Main street yellow light by turning all lights to red, and the walk light to on. After a walk of t_{EXT} seconds, the traffic lights should return to its usual routine by turning the Side Street green. The walk button should be ignored during the walk service.

The second deviation is the traffic sensor. If the traffic sensor is high at the end of the first t_{BASE} length of the Main street green, the light should remain green only for an additional t_{EXT} seconds, rather than the full t_{BASE}. Additionally, if the traffic sensor is high during the end of the Side Street green, it should remain green for an additional t_{EXT} seconds.

Block Descriptions/Implementation

You will be implementing this lab on an FPGA by doing some exercises, programming each block individually, and connecting the Verilog modules together.

Synchronizer

On the block diagram, you see that all signals pass through the synchronizer before going to other blocks. The purpose of the synchronizer is to ensure that the inputs are synchronized to the system clock.

Walk Register

The Walk Register allows pedestrians to set a walk request at any time except for the walk service duration. There is also a signal controlled by the finite state machine that will be able to reset the Register during the actual walk service.

Time Parameters

The time parameters module stores the three different time parameter values, namely t_{BASE}, t_{EXT}, and t_{YEL} on the FPGA. The module acts like a (small) memory from the FSM and Timer blocks, where the FSM addresses the three parameters and the timer reads the data. From the user's perspective, the three time parameter values can be modified.

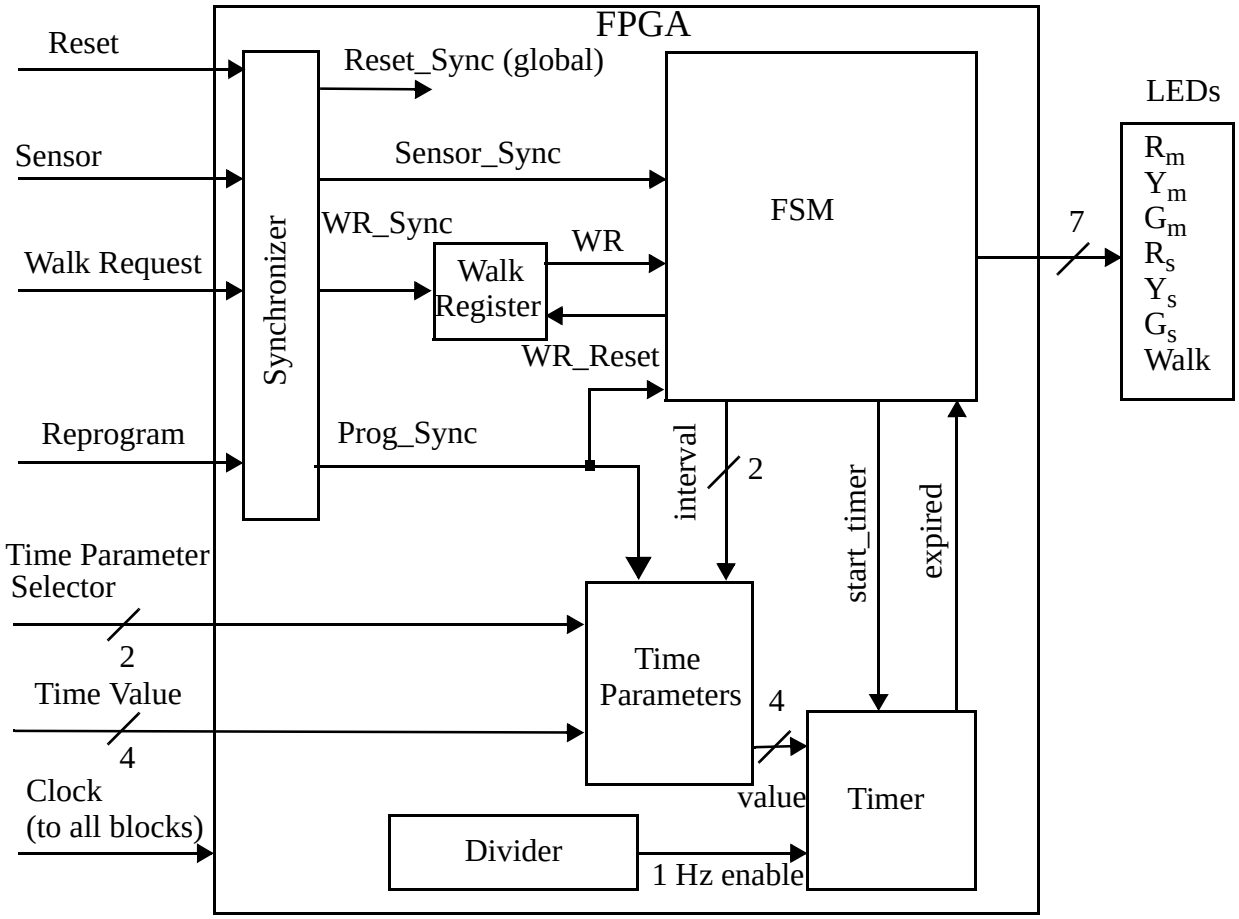

Figure 2: Diagram for intersection with corresponding lights.

On a reset, the three parameters should be respectively set to 6, 3, and 2 seconds. However, at any time, the user may modify any of the values by manipulating *Time_Parameter_Selector*, *Time_Value*, and *Reprogram*. Each of these values are 4 bits, and is selected using a 2 bit address.

Divider

The divider is necessary for the timer to properly time the number of seconds for any particular traffic light state. Using only the clock as input, it generates a 1 Hz enable, which is sent to the timer. The signal generated is a pulse that is high for one clock cycle every 1sec.

Timer

The timer is responsible for taking the start_timer, 1Hz enable, and Time Parameter value to properly time the traffic light controller. When done counting a particular state, the expired signal will go high to signal to the FSM that it should change states.

Finite State Machine

The finite state machine controls the ordering for the traffic light. As previously described, it changes states based on the Walk Register and sensor signals, and with the expired signal.

Part 2: Memory Tester

(Parts 1 and 2 are not related. You can get checked off for each individual part in any order. We recommend completing part 1 first)

Introduction

In second part of the lab, you will be designing a memory tester that will test a 6264 Static RAM. You will be designing a finite state machine to help test the memory.

Procedure

This part of the lab consists of two parts. The first is the design phase, which consists of reading through the lab, planning, and coming up with a design. The next part is to implement the lab using the FPGA and the external SRAM. After you verify the memory tester's functionality, you can get checked off for this part of the lab. Be ready to demonstrate the lab, and be ready to present solutions for the problems asked in this part of the lab.

Memory Tester Description

Digital systems often use memories to store information for processing. Because of this, memory testers are needed to ensure the functionality of these memories and the system setup. Ideally, a memory tester should check each bit location of the 6264 SRAM to ensure that it is accessible and interfaces correctly. To keep things simple, we will test only 16 locations; i.e., use the bottom 4 bits of the address space by grounding the most significant address bits of the 6264. We will be testing the bottom 4-bits of each location (the higher order bits of the RAM can be left floating or ideally connected to pullup resistors).

Devise a memory tester built in the FPGA that ensures that the memory chip is functional (at least the 16 locations in our case) and that the interconnection between the FPGA and Memory is functional. For our purpose, we can say that the chip and interface is functional if we can write and read a 0 or 1 from each individual bit location. The test should perform the write to all locations and then read, rather than a write/read cycle for each individual location. The reason for this is that the data busses have memory (the state can be stored on wires for a brief period of time).

The following approach should be implemented for the memory tester:

- Write all 16 locations, alternating 0x3 and 0xC starting with address 0. Location 0 gets the value of 0x3, location 1 gets the value of 0xC, 2 gets 0x3, etc.
- Read the locations from 0x0 to 0xF and verify that the correct values were written.
- Repeat the process by writing alternate 0xC and 0x3 to the memory. That is, location 0 gets the value of 0xC, location 1 gets the value of 0x3, 2 gets 0xC, etc.
- Read the locations from 0x0 to 0xF and verify that the correct values were written.

For your memory test use a 1sec enable for each read or write operation so the address and data written or read can be observed on the hex LEDs. Include the following LEDs:

- *write* LED - to indicate memory is currently being written
- *read* LED - to indicate that the memory is currently being read
- *failure* LED - to indicate that the memory test failed
- *success* LED - to indicate that the memory test passed

The tester should be initiated from an external reset signal and sequence through the locations and eventually assert the *success LED* or *failure LED*.

If any of those tests fail, then the system should stop at the failed address and assert the *failure LED* light. However, if all the memory write/read tests are successful, then the *success LED* light should be asserted.

In designing the finite state machine, be careful because the address can change at the same time as the write enable if you do the write in one state! So the solution may be to use multiple states. The same condition exists after the write as well; the address can change before the write finishes. Either case can cause write issues.

If a test fails, then the read address should stop at the failed address (so if the TA leaves one of the data bits open, then the read should fail on that address location).

What are the limitations of the proposed simple memory test? Propose (but do not implement) a more comprehensive testing approach for memories in your report.

Laboratory 2 - Guidelines for Lab 2 Report

6.111 will count as a Communications Intensive Major (CI-M) subject. This document specifies our technical expectations for the formal write-up of Laboratory 2 and outlines the quality of writing we hope to see in your report. Laboratory 2 will eventually count for 20% of your final grade with ten percent being assigned primarily on the technical merits of your lab work and the content of your lab report. This portion of your grade will be assigned by the 6.111 staff. The other ten percent will be assigned by the writing department. The writing department will provide feedback on your writing for the first version of your lab 2 report. You are required to make a revised version which will then be graded by the writing department. Your grade is based on the quality of writing found in your revised formal lab report. **Please turn in two copies of your Lab 2 report. One version (to be evaluated by the staff) should include the required appendix. Clearly mark the version going to the writing department on the cover page (Submitted as a part of the CI-M requirement)**

Cover Page/Abstract

Your report should contain a cover page with a title, your name, the name of your TA, the course name, and the date. Your cover page should also contain a paragraph abstract.

Table of Contents/Figures

You should have a Table of Contents and a List of Figures for your lab report.

Technical Portion (Part 1):
Part 1: Overview
The introduction should present a summary of your traffic light controller's functionality and how you went about implementing this project. We would like for you to describe how a user interfaces with your traffic light controller through I/O ports such as push buttons, switches, and LEDs. Be sure to present a table of timing parameters. Feel free to use Figure 1 (modified to fit your approach) from the lab handout for this part.

Part1: Module Description/Implementation

Describe the various modules in your design. You should tell us what each piece of the system does and explain how these various pieces work together. Be sure to discuss the synchronizer, Walk Register, FSM, memory, and divider modules. This section should make up the majority of your paper. We expect a thorough explanation of the various aspects of your design.

When explaining your FSM, make sure to present a detailed discussion of this module complete with a description of its inputs and outputs. You description should mention the various FSM states in your design and what happens in each of these state. A state transition diagram of your FSM is required. Additionally, we ask that you submit screen captures from wave windows in ModelSim for your synchronizer, Walk Register, and FSM.

Technical Portion (Part 2):

Part 2: Overview

Describe the overall approach to testing the memory and motivate why such a test is useful in digital system design.

Part2: Module Description/Implementation

Describe your memory tester implementation. What are the limitations of the simple test proposed (describe specific scenarios where the test will pass, but there is problem with the hardware). Include detailed block diagrams and appropriate schematics to highlight your design.

Testing and Debugging

In this section we would like you to talk about your experiences testing and debugging Laboratory 2. We would like for you to discuss how you tested your digital system and mention ways in which this could have been done better. Provide us with a description of the design methodology you used in the creation of your traffic light controller by discussing MaxII+, ModelSim, and the FPGA. Be sure to include specific details in this section.

Conclusion

In closing you paper we ask that you write about the objectives of this lab and in what ways you believe you have achieved them. We are particularly interested in hearing about what you learned from the completion of this lab and what you think are the important concepts to take away from the design of this digital system. Don't forget to mention where you might have done something differently if asked to implement this project again.

Appendix

For the technical submission of your lab report we ask that you include printouts of your Verilog code in the Appendix. This portion is not required for your submission to the writing department.

Writing Considerations

While the 6.111 staff will be grading your lab report for technical content, we will also be taking the quality of your writing into consideration when assigning grades. As such we ask that all figures and tables be created using a computer (i.e. no handwritten submissions). Because of the fact that ten percent of your grade in 6.111 is based on the writing quality found in your formal report we expect you to dedicate a significant amount of your time in writing this report to polishing and editing. On the course webpage we have made available to you a number of writing resources that we hope you will find useful. If you have any questions or concerns relating to your formal write-up please contact the 6.111 staff and we will do our best to help you.