Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.111 - Introductory Digital Systems Laboratory (Spring 2006)
Laboratory 2 (Traffic Light Controller) Check Off Sheet

Student Name:

TA Signature/Date:

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.

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

Laboratory 2 - Traffic Light Controller (Spring 2006)

Issued: February 22, 2006
Checkoff Due: March 6, 2006
Report Due: March 8, 2006

Introduction

In this lab you will implement a traffic light controller that operates 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 a member of the teaching staff to review your design. This will help catch any major mistakes early in the process.

The next phase is to implement the lab using the FPGA. After you verify the traffic light controller's functionality, you can get checked off. Be ready to demonstrate 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*.

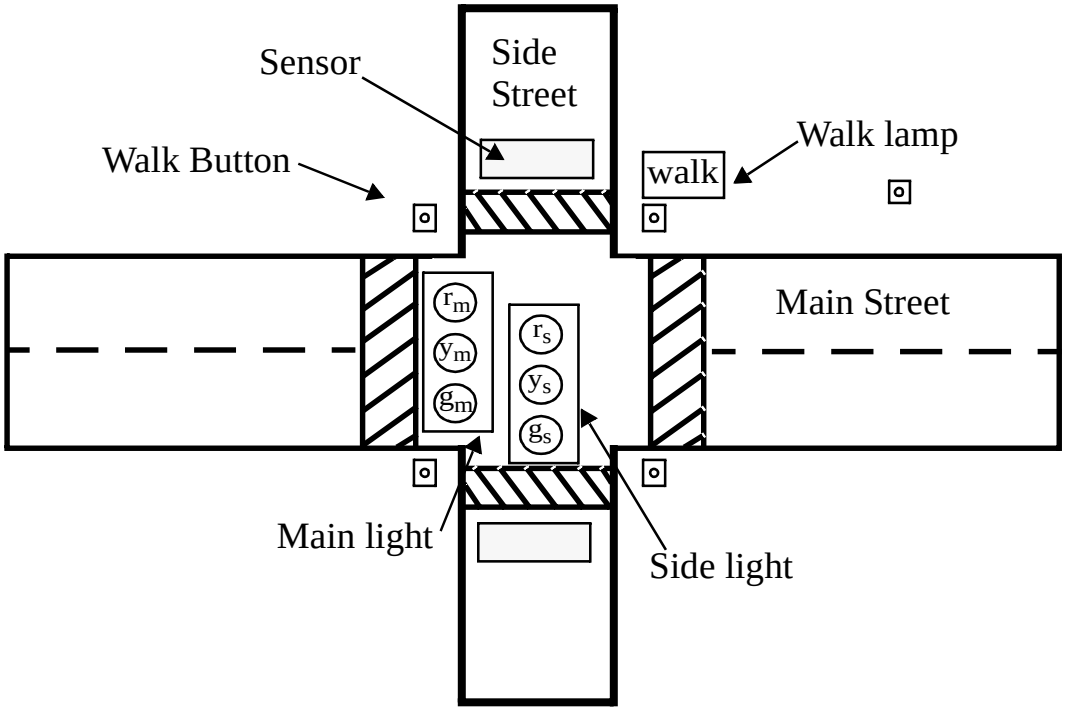

Figure 1: Diagram for intersection with corresponding lights.

Table 1: Default Timing Parameters.

Interval Name	Symbol	Parameter Number	Default Time (sec)	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 registers in 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; there-

fore, 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. The internal Walk Register should be 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 by programming each block individually and then instantiating and connecting the Verilog modules together in the top level lab2_labkit.v module.

Synchronizer

On the block diagram, you see that all input 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 (27MHz).

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.

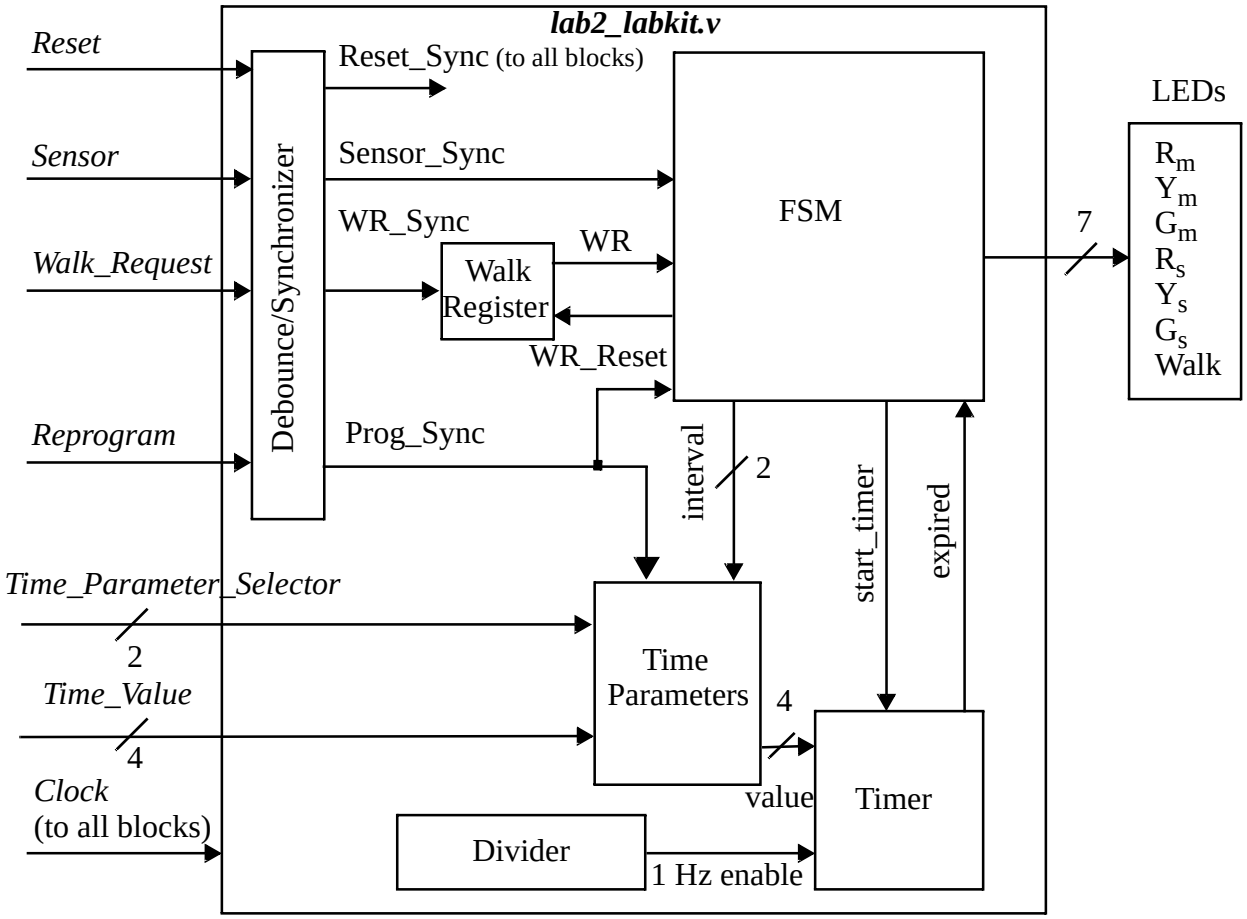

Figure 2: Diagram for intersection with corresponding lights.

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. Whenever a parameter is reprogrammed, the FSM should be reset to its starting state.

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 (27Mhz) 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 Parameters *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.

lab2_labkit.v

The labkit.v template top-level file can be downloaded from labs section. Basically lab2_labkit.v allows a user to utilize all of the labkit features including buttons, switches, leds, logic analyzer pins, user input/ output pins, audio video capabilities, etc. We suggest that you review "Getting Started with ISE", "Programming the Labkit" and "Simulating with ModelSim" tutorials.
After working through the tutorials you will be ready to program your traffic light controller onto the labkit FPGA.

You will need to instantiate all submodules in this top level file as well as create wires to connect the submodules. For example, to connect and instantiate your divider and timer blocks you could add something similar to the following lines in the lab2_labkit.v file:

```
//declare wires as shown in Figure 2 to connect the submodules:
wire reset_sync;
wire one_hz_enable;
wire [3:0] value;
wire expired, start_timer;

//instantiate the submodules and wire their inputs and outputs accordingly
//(use the labkit's clock_27mhz as the clock to all blocks)
divider divider1 (clock_27mhz, reset_sync, one_hz_enable);
timer timer1(clock_27mhz, reset_sync, value, expired, start_timer);
```

In addition, you will need to wire the input buttons and switches that you use to the appropriate submodules. Use the push buttons for the *reset*, *walk request*, and *reprogram* signals (pushing the button produces a logic zero). Use six of the slide switches to select the *time parameter selector* and *time value* to be used during reprogramming. Also implement the *sensor* input using the slide switch. Currently the labkit's eight output LEDs: output [7:0] led; are all defaulted to their off state. If you choose to use them as your traffic lights, then you need to reassign these LEDs (assigning an LED to zero turns it on). In addition, you must assign the LEDs in the following order (MSB to LSB): Main-Red, Main-Yellow, Main-Green, Side-Red, Side-Yellow, Side-Green, OFF, Walk. Alternatively you could wire more colorful leds to the breadboard and control these leds through assigning the user input/output pins: inout [31:0] user1, user2, user3, user4.

Synchronization and debouncing

Your clocked state machine is controlled by several asynchronous inputs that might be changed by the user at any time, potentially creating a problem with metastability in the state registers if one of the inputs changes too near a rising clock edge. In general asynchronous inputs need to be

synchronized to the internal clock before they can be used by the internal logic. You should feed all asynchronous inputs through an instance of the synchronize module and use the output signal in the design of your system.

A second problem arises from the mechanical “bounce” inherent in switches and buttons: as a metal contact opens and closes it may bounce a couple of times, creating a sequence of on/off transitions in rapid succession. So you need to use debouncing circuitry to filter out these unwanted transitions. debounce.v is a Verilog implementation of a digital retriggerable one-shot that requires that an input transition be stable for 0.01sec before reporting a transition on its output. debounce.v can be downloaded from the labs section.

This module happens to produce a synchronous output, so a separate synchronizer is not required. You should use an instance of the debounce module to debounce any switch inputs you use in your design.

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 and the check-off sheet. 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:
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.

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.

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 ModelSim simulation (e.g., test bench), and the FPGA implementation. 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.