CS M152A
DIGITAL DESIGN LAB

Lab 5 Traffic Light Controller

In this lab, you will be designing a traffic light controller.

CS M152A DIGITAL DESIGN LAB

TRAFFIC LIGHT CONTROLLER

Introduction

In this project, you will be implementing a traffic light controller that controls a main street, a side street, and walk lamps. You will be using a finite state machine to implement this controller. You will find a Finite State Machine Tutorial on CourseWeb. This tutorial will guide you to design a FSM using VHDL. You will use VHDL **only** for designing FSM, and **use schematics for the rest of the logic designs**.

Prelab

Due Next Session:

1. There exist two different models to construct a Finite State Machine (FSM): (i) Mealy model and (ii) Moore model. Please discuss the differences between the two models and discuss which model is more suitable for our design.
2. Sketch your initial design for the FSM block in Figure 3 below.

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. Moreover, 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 gate. For this reason, you may assume that the controller only needs a single input called *Walk-Request*.

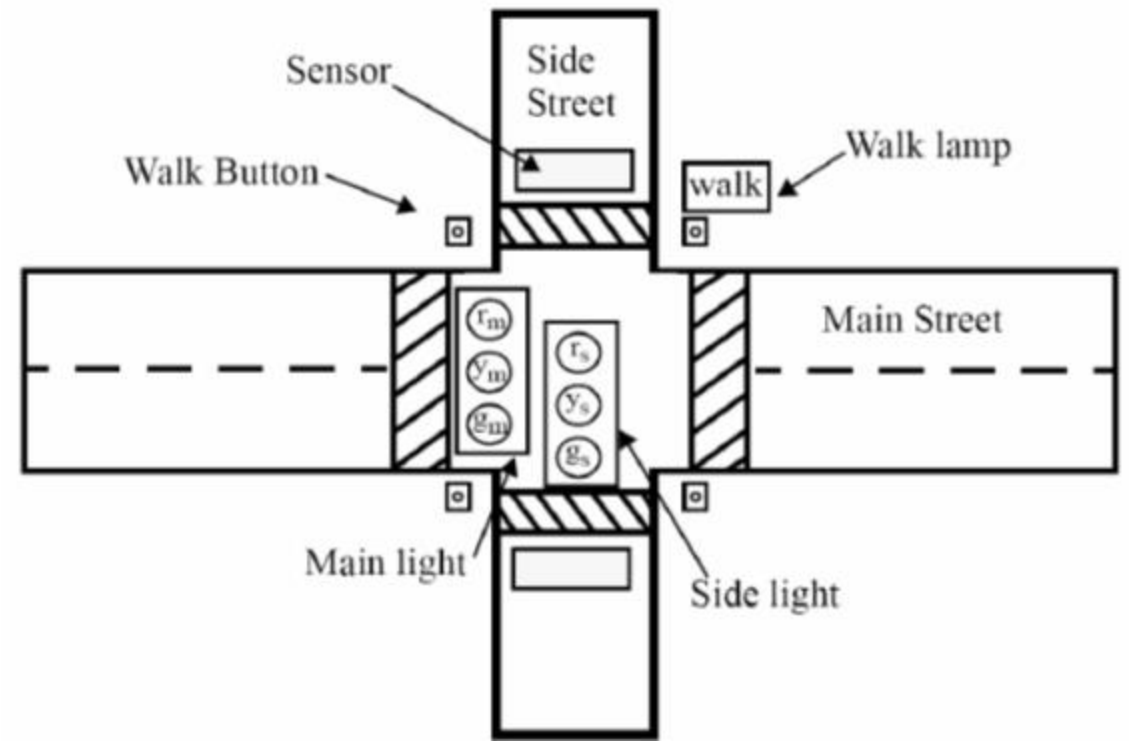

FIGURE 1 - DIAGRAM FOR INTERSECTION, WITH CORRESPONDING LIGHTS

Interval Name	Symbol	Parameter Number	Default Time (s)	Time Value
Base Interval	Tbase	00	6	0110
Extended Interval	Text	01	3	0011
Yellow Interval	Tyel	10	2	0010

FIGURE 2 - DEFAULT TIMING PARAMETERS

The side street sensor is placed near the intersection to tell the controller when there are cars passing over the sensor. You may assume that the sensor will remain 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 (Tbase), the extended interval (Text), and the yellow light interval (Tyel). The default values listed in Figure 2 are to be hard coded into the FPGA.

The operating sequence of this intersection begins with the Main Street having a green light for 2 lengths of *Tbase* seconds. Next, the Main lights turn to yellow for *Tyel* and then turn red while simultaneously turning on the Side Street green light. The Side Street is green for *Tbase*, and its yellow is held for *Tyel*. 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 street lights to red for *Text* seconds. During the Walk Request service, you must create a buzz sound (similar to a real world crosswalk that makes sound for blind pedestrians) using a speaker on the ProtoBoard that you used in Lab 1 and Lab 2. In order to do so, you must use one of the output pins from Nexys3 Board and hardwire it to the speaker on the ProtoBoard. Note that no specific information will be provided to students about which output pin to use and the TAs will NOT help you regarding this. Students are expected to read "Nexys3ReferenceManual" and figure this out. After a walk of *Text* seconds, the traffic lights should return to their usual routine by turning the Side Street green. The Walk Register should be cleared at the end of a walk cycle.

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

Block Descriptions/Implementation

You should implement this lab by programming each block individually and then instantiating and connecting the modules in your top level schematic module. Then you should compile your implementation using the Xilinx tools, download it to your kit's FPGA, and demonstrate its operation by using the LEDs. **Use your board's push buttons for the walk request and reset button and a switch for sensor input**. Dedicate one LED to your clock. A proposed high-level block diagram of the design is shown in Figure 3.

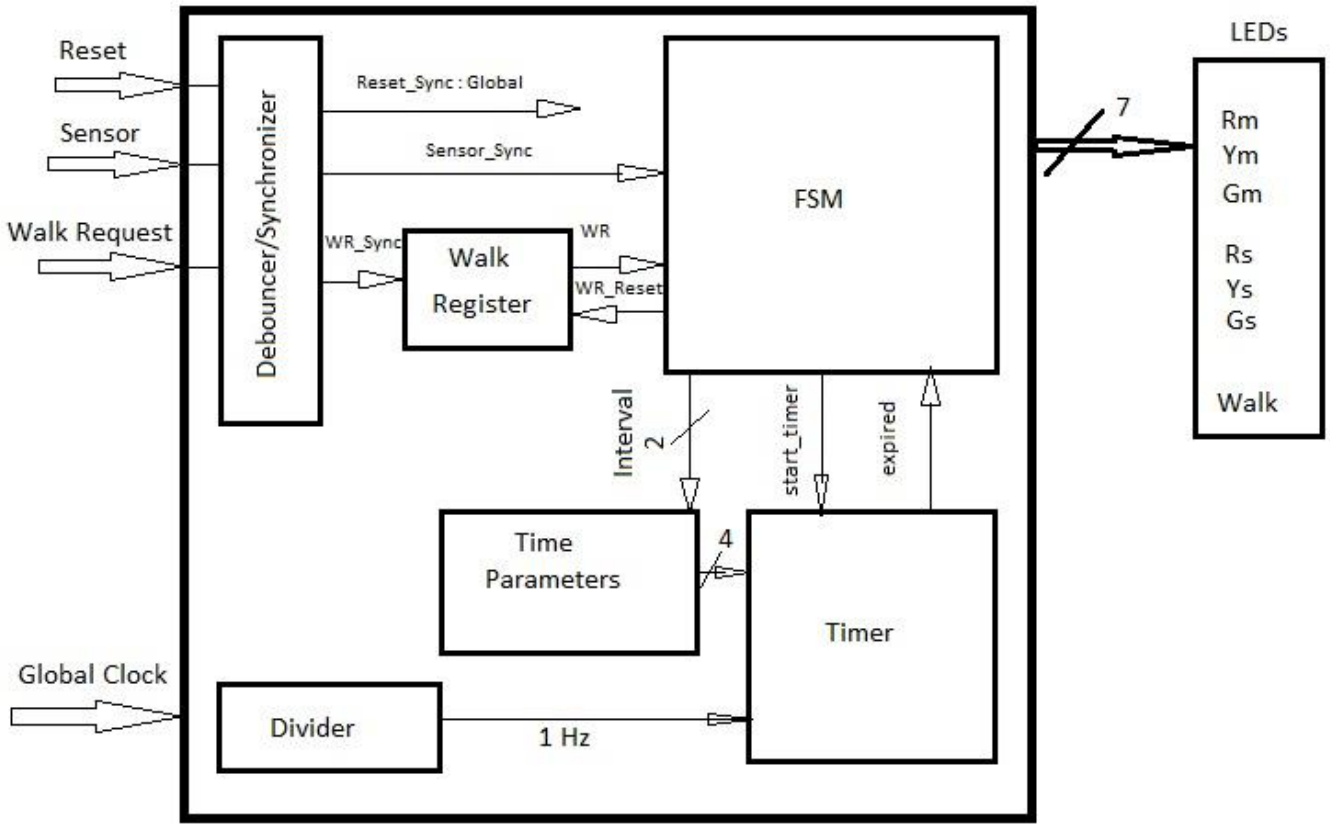

FIGURE 3 - BLOCK DIAGRAM OF TRAFFIC LIGHT CONTROLLER

Debouncer/Synchronizer

On the block diagram, you will 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.

Walk Register

The Walk Register allows pedestrians to set a walk request at any time. There is also a signal controlled by the finite state machine that will be able to reset the register at the end of the actual walk cycle.

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, this module 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 1 second.

Synchronization and Debouncing

Your clocked state machine is controlled by asynchronous inputs that might be changed by the user at any time, potentially creating a problem with meta-stability 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: 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.vhd** is a VHDL implementation provided for you of a digital re-triggerable one-shot that requires that an input transition be stable for small fraction of a second before reporting a transition on its output. This file is a VHDL file and you will need to convert it to a schematic file. Check the Modelsim Schem Tutorial to see how this can be done.

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. Though you will be provided with this file, we would like you to know that it may not have been designed correctly, so be sure to test its functionality using your newly acquired Xilinx skills before continuing.

Report

Please write a maximum 3 pages report describing your design of traffic light controller. You must have an overall description of your design as well as detail descriptions for each sub-component of your design. Please include one or more diagram in order to support your explanation. Strictly follow the report template, otherwise you will lose points.

In this project, you must submit both hard and softcopy of your project. Please do not include ANY vhdl codes or schematic diagrams in the hard copy of report that you must submit in the class. All software codes must be compressed in a "zip" file and submitted to the CourseWeb under Assignments section. Please organize your "zip" file to contain:

1. README.txt
    a. Names and student IDs for all group members
    b. Comments that you want to leave to help your TA evaluating your design
2. Project File (director)
    a. Please include all project files

Please also name your submission file as "LAB5_(Partner Name 1)_(Partner Name 2).zip".