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

Laboratory 2 – Finite State Machines

INTRODUCTION

This laboratory exercise concerns the design and implementation of a traffic light controller[1] for an intersection. Your implementation of this system is to be by a synchronous finite state machine (FSM).

This lab is designed to give you a methodology for designing and building a system and creating procedures for testing completeness.

We are going to make a traffic light controller similar to those used in some countries in Europe. Operation of the traffic light is somewhat similar to that here: you may go through an intersection on green and must stop on red. What you do on yellow is up to your own conscience.

This traffic light controller also has provision for a walk light and for a vehicle sensor in one of the streets. For convenience we will refer to one of the streets as "main" and the other as "side". The traffic sensor is in the side street. Normally the side street has a shorter 'green' interval than the main street, but if there is traffic in the side street when the controller is about to cycle to turn that green light off, it will extend the green light by the shorter (side street) green interval, but it will do this only once and then return to the regular interval.

The walk light comes on only after the main street green interval, and then only if the walk light request button has been pushed. Now, we have a problem here since the guys that buy apparatus for the lab are kind of cheap and we don't have a separate walk light. So we will simply use the old Massachusetts convention and turn on both red and yellow lights in both directions to note a walk interval.

Your traffic light controller FSM is also given the task of loading static RAM locations with timing parameters and of displaying these parameters by reading the RAM locations. You should implement and test the functions of depositing and examining RAM locations before you go ahead and test your traffic light controller.

Traffic Light Controller

The intersection to be controlled is between a busy (Main) street and a somewhat less busy (Side) street, (see Figure 1). Both streets have ordinary (Red, Yellow, Green) signal lights. The intersection is fitted with a sensor for side-street traffic and with a walk request button.

There are four timing parameters, described in Table 1. These are the base interval (TBASE), the extended interval (TEXT), the time for a yellow light (TYEL) and a blink interval (TBLINK). These timing parameters are expressed in number of "tics" of a basic one second "long clock" which you must synthesize.

The side street sensor is a level indicating that there is traffic waiting. This signal should be provided by a switch on your kit. It is not latched, but should be synchronized. The Walk Request is provided by a pushbutton, and must be latched.

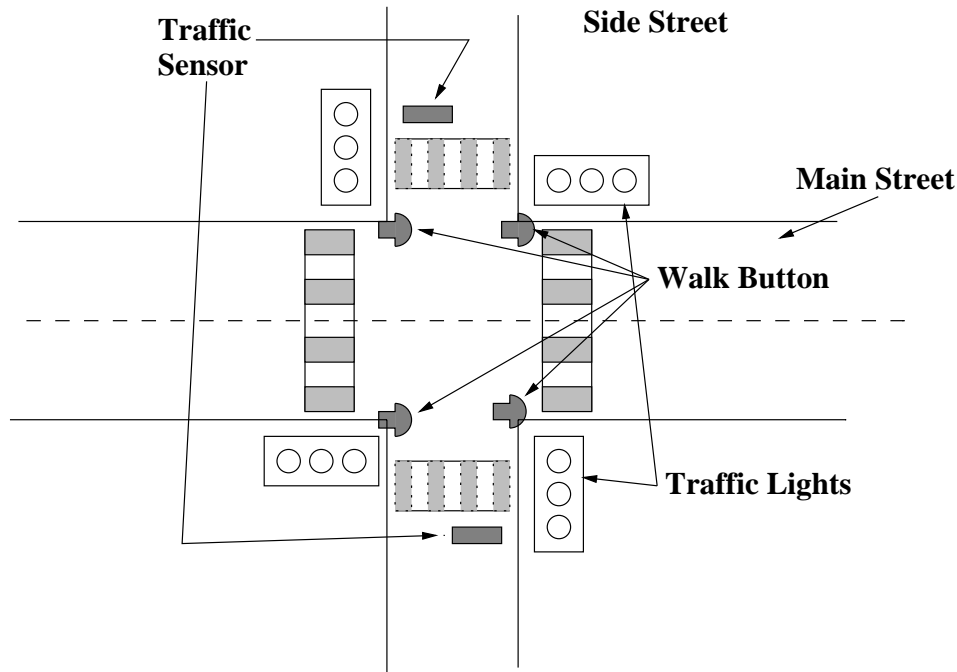[1] Adapted from a laboratory problem used by Professor Randy Katz at U. C. Berkeley.

Figure 1: Intersection to be controlled

The operating sequence is that the Main street has a green light for a period of time equal to TBASE+TEXT, then the system cycles through the normal yellow/red combination to the side street having a green light for TBASE, and the system cycles through yellow/red back to the Main street having a green, and the cycle is repeated.

The period of time during which the Main street has a green light should be regarded as two time periods, TBASE and TEXT. If, at the end of this interval there is a WALK request pending, the system goes to the Main yellow/Side red for TYEL and then to WALK (all red and yellow lights on) for TEXT which we will take to be the length of the WALK interval. At the end of this the system goes to Side green.

Normally, the side street stays green for TBASE. If at the end of a green interval the traffic sensor shows there is still traffic, the green light stays on for an additional TBASE.

Note that the WALK request is handled only after the Main street has been green. If there is a pending WALK request at the end of the Main yellow/Side red interval, it should be handled then.

Note also that the WALK request must be explicitly UNlatched by your controller, at the time the WALK signal is serviced. The WALK light should stay on for only ONE period of TEXT at a time, and should ignore any WALK requests made while the WALK light is on.

Finally, as happens often in Massachusetts, late at night or when something in the system is not working, the light must go into a "blinking" pattern. This should show the Yellow lights on the Main street and red lights on the side street, blinking ON and OFF with the timing interval TBLINK for both ON and OFF periods.

Specifications

A simple block diagram of the traffic light controller is shown Figure 2. A more detailed functional block
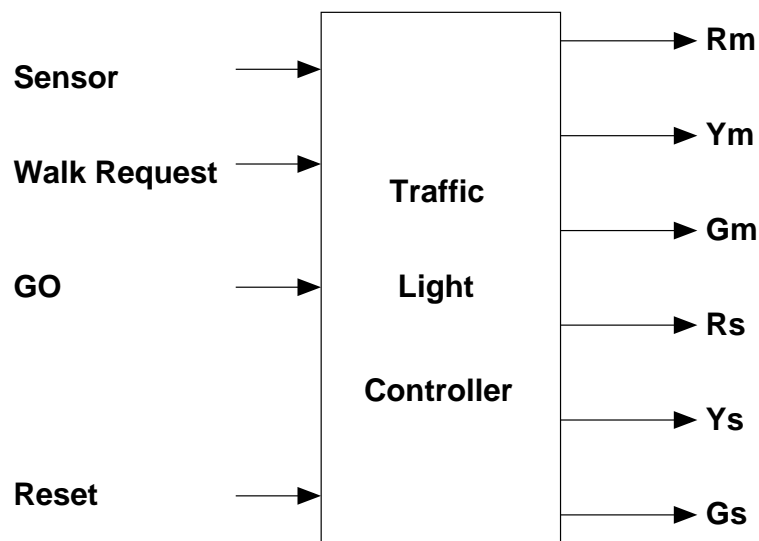

Figure 2: Traffic Light Block Diagram

diagram is shown in Figure 3. Note that you may fit as much of this functionality into a CPLD as you wish and can. You MUST use a CPLD for at least the finite state machine (FSM). The synchronizer consists simply of D flip-flops.

The DIVIDER is a series of counters driven by your crystal oscillator and produces the FSM clock /CLK, and a much slower clock, 1/SEC, which is used to drive the TIMER. The TIMER is a counter unit which counts for a number of 1-second intervals which are specified by data stored in the static RAM.

The input and output signals for the FSM are listed and described in Table 1. You may use any polarity you like, e.g., /WE or WE as you choose. The four functions specified by the two function switches are also listed, and the meaning of the four RAM locations is given in the table. The values stored in the RAM represent time durations in seconds. Remember to wire unused RAM address lines to GND.

Traffic Sensors

Traffic sensors buried beneath the side street indicate the presence of a vehicle over the sensor. The sensor has an added feature in that the sensor output stays asserted for a short time after a vehicle has gone past the sensor. This is important when a continuous bunch of vehicles goes over the sensor. Without the delay, the sensor output would pulse once per vehicle. With the delay the sensor signal is asserted at the beginning of the bunch and stays asserted until a short time after the last vehicle in the bunch has gone over the sensor.

Please remember that vehicles do not have any way of knowing the precise timing details of your finite state machine system clock. That is, the sensors' signals should be considered to be ASYNCHRONOUS to your system clock.

The Walk Request button is pushed once and must be latched to form the WR signal, which is to be cleared as soon as the WALK interval (RED + YELLOW) begins.

While it is possible to effect this synchronization by being clever and absorbing the synchronizing function within your FSM, it is strongly suggested that you explicitly synchronize the sensor signals (or stretched

FSM Input Signal Definitions

RESET	(from a switch)
GOSYNC	(from SYNCHRONIZER)
F1 and F0	Determine one of four different functions (from switches)
L1 and L0	Specify a location in the SRAM (from switches)
AUX	Auxiliary control switch (Synchronized but not latched)
WR	Walk Request (From Re-settable Latch fed by pushbutton)
EXPIRED	Signals when a pre-specified time has elapsed (from TIMER)

FSM Output Signal Definitions

A1 and A0	Specify an address in the SRAM (to SRAM address lines)
WE	Drives value from switches on to bus, writes into SRAM
STARTTIMER	Resets 1-second clock and 1-second increment counter
Gm, Ym, Rm,	
Gs, Ys, Rs	Traffic light control signals

Table of Functions

F1	F0	
0	0	Examine memory location specified by address switches
0	1	Store new value in memory location of address switches
1	0	Run traffic light
1	1	Light Blinks

Values Stored in SRAM

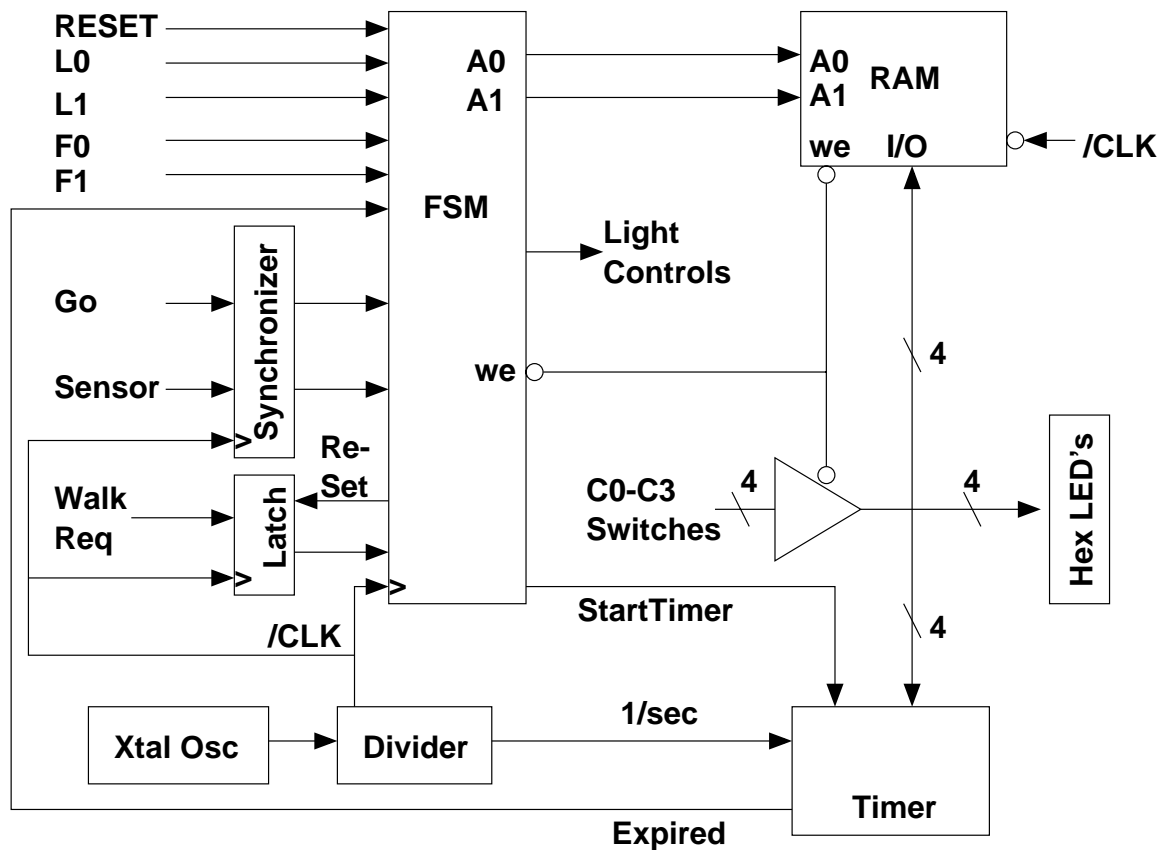| A1	A0		Nominal Value	
0	0	TYEL	3	Time for yellow light
0	1	TBASE	6	BASE (Green) interval
1	0	TEXT	6	extended interval
1	1	TBLINK	1	Time light stays on (and off) while blinking

Table 1

Figure 3: Controller Block Diagram

sensor signals) with D flip-flops. These D flip-flops can be part of a CPLD if you choose.

Your system clock is to be derived from a counter which is driven by a crystal oscillator such as used in Laboratory 1. Timing intervals should be derived by a programmable counter which is clocked by an appropriate frequency and which is initialized by signals derived from your FSM. Basically, the time intervals are to be determined by loading the programmable counter with a number and detecting when the carry out signal is asserted. Remember to reset your TIMER clock when starting the TIMER.

A partially completed VHDL source file is located in the 6.111 locker.

Copy it to your locker by executing

```
cp /mit/6.111-nfs/vhdl/lab2.f02/stoplight.vhd
chmod 600 stoplight.vhd
```

The VHDL source file provided is not complete enough to create a CPLD file yet. For example, it does not include the complete FSM specification.

Procedures and Requirements

To provide the possibility for demonstrating your controller on a "real" traffic light, you should provide a space for us to plug in a DIP cable to your kit. The signals that should be present are shown in Figure 4. Do NOT wire anything to the right hand side of this space: the dip cable will have the signals shown in parentheses so that it will work even if plugged in upside down.

1. Before proceeding with the details of the FSM design, you should design the circuitry needed to synchronize the GO signal to the system clock.

   Since you want the function specified by F1 - F0 to be performed only once per assertion of the GO signal, it will be convenient to have the synchronized GO signal asserted for exactly one period of the system clock.

2. Provide timing diagrams which completely demonstrate the operation of each function of your FSM.

3. Provide a complete logic diagram.

4. Use VHDL to generate all combinational logic equations for all of the control signals required by the FSM and the data paths, as well as the D inputs of your state variables. You should discuss your design with a member of the teaching staff before programming your CPLD.

5. Demonstrate your entire system and all of its functions to a member of the teaching staff. Have all of your timing diagrams, state diagrams, VHDL file, and logic diagrams available for this demonstration.

```
+5 V
 Green  Street 1
 Yellow Street 1
 Red    Street 1
 Green  Street 2
 Yellow Street 2
 Red    Street 2
 Ground
```

Figure 4: Traffic Signal Light Connections

Laboratory Report

You are to provide a laboratory report which meets the requirements specified in the "Report Guide" handout. Your report should include the following: data paths, an FSM, VHDL source file and the corresponding state file, one logic diagram, and all timing diagrams. You should also include some text describing your design and methods of implementing it. The report should flow, be well organized, and, most importantly, be complete. Verbosity is not a requirement.

Design Notes

Data sheets for the 6264 SRAM are attached. PLEASE read the data sheet carefully as this chip is easily damaged by incorrect use (wiring). ASK QUESTIONS IF YOU ARE NOT SURE!

The 6264 has a tristate Input/Output (I/O) bus. Reread the handout "Gates, Symbols, and Busses" which pertains to bussing. The I/O bus of the 6264 MUST be driven by a tristate buffer; use the 74LS244 included in your kit.

Tristate bus contention occurs when two (or more) drivers are active at the same time. The 6264 tristate output is enabled when the /OE input is asserted low, the /CS is asserted low, and the /WE line is high. While it is true that many logic designers allow tristate bus contention to occur for short times (due to chip delays), it is not a good idea. For this laboratory exercise you are to ensure that NO tristate bus contention can occur.

The actual write pulse is the AND of both the /CS and the /WE asserted low. It is essential that the address lines to the SRAM not change when the write pulse is active. Otherwise you may write to multiple locations!
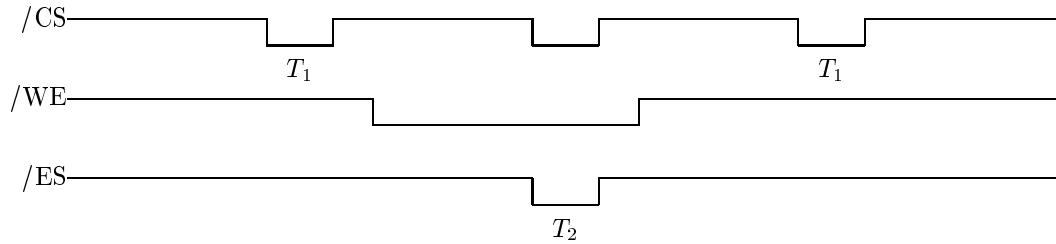
Figure 5: Example Timing Diagram for SRAM I/O

While the 6264 is advertised as a static RAM, a memory cycle is actually initiated whenever ANY address line changes. Thus, the address lines may NOT be tristated whenever the /CS is asserted, as the internal timing circuitry is actuated by noise on the HI-Z address lines.

One way to ensure both that tristate bus contention does not occur and that the address lines do not change when the write pulse is active is to connect the system clock, /CLK, to the chip select pin; see Figure 5. The address lines do not change until after the rising edge of /CLK. The /WE line can then be provided by your FSM. As long as the /WE line is low prior to (or concurrent with) the chip select being asserted, then the SRAM will not drive the I/O pins. The control line to the tristate gate connected to the switches can also be an output of your FSM, but it should also be gated with the system clock.

During T1 data from the SRAM will appear at the I/O pins, and during T2 the data from the switches will appear at the I/O pins. (/ES is the tristate enable for the switches.)

You should not use monostables (74LS123) to generate the /CS or /WE inputs to the SRAM.