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

Problem Set 5

Issued: Lecture 11 Day
Due: Lecture 12 Day

In this completely fictitious story, you are taking a class called 6.111 - Digital Death Laboratory. There are LAs who spend a lot of time in lab. Here is your chance to implement a system to reward these fellow LAs.

Functionality

Your 6.111 class has three TAs and four LAs. We would like to reward the LAs who work hard. Every week, the TAs and LAs fill out the hours they work. We propose the following system:

1. Take the minimum of all the hours put in by the TA's. Call this the weekly minimum. Call the maximum hours put in by the TA's that week the weekly maximum.
2. If any LA works fewer hours than the minimum, they get a warning point.
3. If a LA works more hours than the maximum, they are receive a reward point.
4. Each LA starts with 0 points. For each reward point they get, their score goes up by one. For each warning point, their score goes down by one, but never below 0.
5. If a LA's score reaches 3, they get a little bit of timeoff (denoted by the timeoff output to do whatever undergrads do in their spare time). Their score then resets to 0.

Implementation

You will be implementing this system using major and minor FSMs. Specs for the SRAM and for the entities are provided below. You must use the `la_rewarder` entity. The `maxmin` entity is suggested, but not required.

SRAM Block

The SRAM Block contains the standard 6264 I/O, and an additional status output. This output is called `SRAMbusy`.

This block is your interface to the EECS's payroll computer. They update the TA/LA hours weekly automatically to the RAM. During this update, `SRAMbusy` is asserted. Each staff member's hours for the week are accessible from 7 different SRAM locations. Here is the addressing scheme for the SRAM:

A[2:0]	Function	Name
000	TA	Neira Hajro
001	TA	Jennifer Maurer
010	TA	James Oey
011	LA	Levete Jakab
100	LA	Craig Mielcarz
101	LA	Colin Weltin-Wu
110	LA	Brian Wong

Major FSM Entity

```
entity la_rewarder is
port (
    clk       : in  std_logic;
    go        : in  std_logic;
    rdy       : out std_logic;
    timeoff   : out std_logic_vector (3 downto 0);
end la_rewarder;
```

Description of entity `la_rewarder`:

- `go` states when the system should calculate the reward output. Since the SRAM communicates with a different system to update the hours data, the computation may not start right away.
- `rdy` is an output telling when the reward data is valid.
- timeoff is a 4-bit output denoting who should be rewarded with time off. Each bit represents an LA.
  bit 0: Levente
  bit 1: Craig
  bit 2: Colin
  bit 3: Brian

Minor FSM Entity

```
entity maxmin is
port (
    clk       : in  std_logic;
    go        : in  std_logic;
    max       : in  std_logic;
    rdy       : out std_logic;
    data      : out std_logic_vector (7 downto 0));
end maxmin;
```

Description of entity `maxmin`:

- `go` tells the system to start the computation.
- `max` is the maximum and minimum function selector.
- `rdy` is the state machine done flag.
- `data` provides the maximim or minimum result.

Approach

1. Add to the `la_rewarder` entity to include the interface to the SRAM block described above. Submit a description of additional inputs/outputs. Design your entity for your minor FSM, and describe the inputs/outputs. The entities will be submitted with your VHDL code in step 3.
2. Draw the state machines for your major and minor FSMs. Be sure to follow the standard conventions for drawing state machines. Submit state diagrams for both FSMs.
3. Write the VHDL files that implements both FSMs. Be sure to instantiate the minor FSM inside your major FSM. Submit your VHDL code.