Term Project

Due:	ASM Chart (hardcopy)	07/12/2015, 19:30
	Simulation Deadline	14/12/2015, 19:30
	Simulation Demo	Week of 14-18/12/2015
	Final Deadline	21/12/2015, 19:30
	Implementation Demo	Between 21/12/2015-30/12/2015

Queue Management System for Bank (Version 1)

You will design a simple queue machine and implement it on the FPGA device. You will be asked to demonstrate your implementation on the BASYS FPGA board.

Inputs

There will be 7 (seven) inputs in your circuitry:

- *Teller0* will be used to apply the operations of the first teller (teller is a person who deals with customers over the counter). This input should be provided from BTN0.
- *Teller1* will be used to apply the operations of the second teller. This input should be provided from BTN1.
- *Customer* will be used to get numbers for customers. This input should be provided from BTN2.
- *reset* will set your circuitry to its initial state. This input should be provided from BTN3.
- *Operation switches* are sliding switches on the FPGA board used to define teller's operations. This inputs are represented by SW[2:0]
    - *Counter* will be used to open or close the counter. This input should be provided from SW[0]
    - *ChangePassword* will be used to change teller's password. This input should be provided from SW[1]
    - *Bank* will be used to open or close the bank. This input should be provided from SW[2]
- *Password*, the remaining sliding switches, will be used to change teller's password. This input should be provided from SW[7:4].

Outputs

There will be two different types of outputs in your circuitry: the LED outputs and the SSD outputs.

LED outputs:

- LED[0] indicates that *Teller*0 is available or not (his/her counter open or closed).
- LED[1] indicates that *Teller*1 is available or not (his/her counter open or closed).
- LED[2] indicates that Bank is open or closed.
- The other 5 (five) LEDs will show the current state of the circuit.

7-segment displays (SSDs): SSD requirements will be explained in the next section.

Operation Steps

There are two different modes, in which your design operates: *BankClosed* and *BankOpen.* The circuitry will start in the IDLE state, in which all counters are closed and the initial passwords for both tellers are "0000".

Basic Version(77%)

In the basic version, the design operates only in *BankOpen* mode and there is no need for *ChangePassword* and *Bank* switches since *ChangePasword* and *Bank* operations will not be implemented.

BankOpen Mode

Initially, the counters are closed and SSDs display '0000'. Mainly, the rightmost and the left most two digits of the SSDs belong to *Teller*0 and *Teller*1, respectively. The tellers can *open/close their counters* and *call the next customer*. The customer has only one operation, which is *obtaining a queue number*.

Obtaining queue numbers: The customer can get a number by pressing the *Customer* button. When the button is pushed, a new number will be displayed on SSDs for approximately 1 second. During that period, the leftmost two digits will be blank and the right most two digits will display the customer number in hexadecimal. You can assume that the maximum number of customer is 0xFF in hexadecimal. There is no requirement for an overflow check for the number of customers.

Opening/closing counters: In order to open/close his/her counter, a teller will enter his/her *password* from the *Password* switches, set the *Counter* switch to 1, and finally push the teller button (*Teller*0 or *Teller*1). While the counter is open, the corresponding LED will be on (i.e. LED[0] , LED[1] are on for *Teller*0 and *Teller*1, respectively).

Call next customer: This operation can be executed only if the counter is open. In order to do that, all operation switches will be set to 0 and the teller will push his teller button (*Teller*0 or *Teller*1). When the operation is done, the SSDs corresponding to teller's counter should display the number of the next customer.

Full Version(110%)

In addition to the operations in the basic version, in the full version, you have to implement the *BankClosed* mode in addition to *BankOpen* mode. In the full version, the system is initially in the *BankClosed* mode.

BankClosed Mode

In the *BankClosed* mode, the SSDs display 'CLSd' and there is only one operation, which is opening the bank. In order to open the bank both tellers should enter their passwords one after the other. For example, the first teller enters his password on the *Password* switches and sets the *Bank* switch to 1, resets the remaining switches to 0, and pushes the *Teller*0 button. The second teller will repeat the same operation. The order these operations are performed is not important (i.e., the second teller may start first). If the operations are successful, the bank will be in the *BankOpen* mode.

BankOpen Mode

When the bank is in the *BankOpen* mode, the tellers can perform two additional operations besides the operations in the basic version: *closing the bank* and *changing the password*.

Closing bank: Closing the bank operation can be performed if the counters are closed. Similar to the bank opening operation, it has two steps. The steps are the same as the steps in the bank opening; namely the tellers follow the same steps that are used to open the bank. After one teller initiates the bank closing operation, no other operation can be performed, except for completing the closing.

Changing password: This operation will be performed to change the password of a teller. This operation can be performed if the teller's counter is open. There are two steps to change password. In the first step, the teller, who wants to change his password, will enter his/her password using the *Password* switches and *set ChangePassword* switch to 1, reset the other switches to 0 and pushes the teller button. Then he will set his new password and push the teller button again.

Note that there are three operation switches (i.e., SW[2:0]) and you can assume that only one of them is set to 1 at a time. The circuit will do nothing if two or more operation switches are set to 1.

Bonus Parts (+5.5% and +11%)

Bonus parts can be done if you have the full version of the design. You can do one of the two bonus operations described below:

1. *Calling next customer* operation has an extra requirement. In addition to requirements in the basic version, while displaying the next number, the corresponding SSDs blink 2(two) or 3(three) times with a frequency of 1 Hz (one blink/s).
2. *The Closing bank* operation can be performed if all desks are closed and there is no customer in queue. If customer queue is not empty, remaining number of customers is displayd at SSD in the format of "rcXX" for 1 second. (XX represents the remaining number of customers.)

Appendix A: An Example Template

```
module fsm (
   input clk,
   input rst,                          // BTN3
   input teller0,                      // BTN0
   input teller1,                      // BTN1
   input customer,                     // BTN2
   output reg[27:0] ssd,
   output reg[7:0] led,
   input [7:0] sw                      // Counter SW[0], ChangePassword SW[1],
                                       // Bank SW[2], Password SW[7:4]
);

        reg [7:0] current_state;
        reg [7:0] next_state;
        reg [7:0] cust_count;
        reg [7:0] teller_count;
        …                                   // additional registers

        // sequential part – state transitions
        always @ (posedge clk or posedge rst)
        begin
                …               // your code goes here
        end

        // combinational part – next state definitions
        always @ (*)
        begin
                …               // your code goes here
        end

        // sequential part – control registers
        always @ (posedge clk or posedge rst)
        begin
                …               // your code goes here
        end

        // sequential part – outputs
        always @ (posedge clk or posedge rst)
        begin
                …               // your code goes here
        end

        …               // additional always statements

endmodule
```

Please note that this template is only an example. You can add more registers, change the bit-size of the given registers, initialize the values for the given registers, define parameters and extend the number of "always" statements as your design requires.

Appendix B: Details on Simulation Demo and Final Demo

During the simulation demos (see "Project Plan and Deadlines" section for the demos and their schedules), you are only expected to show the functionality (i.e., the correct transitions between the states) of your Finite State Machine (FSM). You do not need to add any other parts to your design during the simulation demo.

Appendix C: Extra Modules Needed for the Implementation on FPGA

In the final implementation of your circuit, you will need some extra modules which we provide you under "**Resources→Term Project**". These are "clock divider", "debouncer" and "seven segment driver" modules. There are comments in the modules about the units the modules implement. You can use these files just as verilog or you can create a symbol from the verilog files and add them to a schematic file; you are free to choose any option.

- Clock divider module generates a clock signal with a period of 50 ms, from a 25 MHz input clock.
- Debouncer circuit gets the input from a push button and generates a one clock pulse output.
- Seven segment driver module, drives the segments.

In *ssd.v*, the given module uses the **UN**-divided clock, while in *debouncer.v*, the given module uses the divided clock. Besides, your cicrcuit should also use the divided clock.

A demo schedule will be organized before the demo dates and will be announced through *suCourse*. If you want to demonstrate your design before the deadlines, you can do so by scheduling an appointment with your TA. Besides, if you want to **check your work**, you can come to the lab on the days that will be announced.

Please do not forget these facts:

- Each group will have 20 minutes to present their project and answer the questions we will be asking. Be prepared for any question we might ask about your project (design, coding, etc.). Every group member should be able to answer these questions.
- You must bring a **hard copy (i.e. printed copy)** of your **latest ASM chart**.
- Any group member not attending the demo without proper excuse (e.g. doctor's report, existing final exam) will cause in the decrease of the grade of the whole group.
- For the final (implementation) demo, you **must submit your work** and **your report** through the assignment "**Final Project**" before its deadline. We will be checking the correctness of the submitted design.
- Each group must be present at the door of the lab **at least 10 minutes prior** to their scheduled time.
- No group should enter the door until they are invited by one of the TAs.
- If you are not able to demonstrate your work on time, **you will not be given extra time.**

Appendix D: Simulation Scenario

1. If you implemented Basic Version
    a. Customer obtains a queue number
    b. Teller0 opens his/her counter
    c. Teller0 calls next customer
    d. Teller1 opens his/her counter
    e. Customer obtains a queue number
    f. Teller1 calls next customer
    g. Teller0 closes his/her counter
    h. Teller1 closes his/her counter

2. If you implemented Full Version
    a. Reset the circuit
    b. Teller1 performs corresponding operation to open the bank
    c. Teller0 performs corresponding operation to open the bank
    d. Customer obtains a queue number
    e. Teller0 opens his/her counter
    f. Teller0 calls next customer
    g. Teller1 opens his/her counter
    h. Customer obtains a queue number
    i. Teller0 changes his password. New password is “4’b1000”
    j. Teller1 calls next customer
    k. Teller0 closes his/her counter
    l. Teller1 closes his/her counter
    m. Teller1 performs corresponding operation to close the bank
    n. Teller0 performs corresponding operation to close the bank

3. If you implemented any of Bonus Parts
    a. Simulate the scenario that is given for Full Version

Notes:

1. The given scenario should be submitted with your project and ISim should work properly. You are **not allowed** to change your codes during your demo slot.
2. For simulation, you must not use clock divider, debouncer and ssd modules.
3. Check the other rules from Appendix B

Appendix E: Project Plan and Deadlines

You must follow the project plan given below and demonstrate that you meet a specific deadline by submitting your work. Each work item in the project plan will be graded separately. The project plan and grade of each work item is as follows:

1. REQUIREMENTS: Project requirements are given to the students.
   Nov 27, 2015

2. ASM_CHART: ASM Chart is submitted in *hardcopy* to Atıl Utku Ay.
   (You can learn the comments made on your design and take your ASM charts back, during the office hour, in the next Monday)
   Dec 7, 2015 19:30

3. SIMULATION: ISim simulation is demonstrated to the lab assistants.
   Dec 14-21, 2015 (Note the Due Dates for submissions above)

4. REPORT: A project report is submitted via *suCourse* as described in the document named *Submission Guidance for the Lab Assignments.pdf.*
   Dec 21, 2015 19:30

5. DEMO: The circuit is realized on FPGA and a demonstration must be made.
   Dec 21-30, 2015
6. **Note that these deadlines are hard, and there will be no additional time.**
7. **Notes:**

   - You can work with your lab partner in this project.
   - You can use Verilog language. (Recommended)
   - The first two groups that finish the project earliest will be awarded by five (5) bonus points in the final exam.

Some Tips: Before writing your project on Verilog, you can try to write some small examples to warm up. This will help you to complete your task quicker and painless. Also there some sites that you can find some examples to study Verilog. One of them is: http://www.asic-world.com/verilog/veritut.html. These sites might help you to understand combinational, sequential circuits and state diagrams.