Term Project

Due:	ASM Chart (hardcopy)	9/12/2019, 11:00 (11:00 a.m.)
	Simulation Deadline	16/12/2019, 11:00 (11:00 a.m.)
	Simulation Demo	Week of 16-20/12/2019
	Final Deadline	22/12/2019, 23:55 (11:55 p.m.)
	Implementation Demo	Week of 23-27/12/2019

A Simple Automated Teller Machine (ATM)

You will design a simple ATM and implement it on the FPGA device. You will be asked to demonstrate your implementation on the BASYS FPGA board (**we will not use BASYS2**). In your design, there will be login operation where the user first should insert its debit card and then enter his/her password. After having entered the password correctly, user may do: 1) deposit/withdraw money to/from his/her account, 2) change the password of the account or 3) log out from the ATM.

Inputs

There will be 5 (five) inputs in your circuitry:

- *rst* will set your circuitry to its initial state. This input should be provided from BTN0 of the BASYS board.
- *BTN3* will be used for different operations in different states. For example, in the IDLE state, it is used to insert the debit card while it is used for entering the password in another state. Details will be provided below. This input should be provided from BTN3 of the BASYS board.
- *BTN2* will be used for changing your password and withdrawing money from your account. Details will be provided below. This input should be provided from BTN2 of the BASYS board.
- *BTN1* will be used to log out from ATM and navigating in ATM menu. Details will be provided below. This input should be provided from BTN1 of the BASYS board.
- *SW* is a 4-bit input and it will be used to set password or the amount of money to deposit/withdraw to/from your account. Details will be provided below. This input should be provided from SW[3:0] of the BASYS board, where SW[0] will be the least significant bit of its value.

Outputs

There will be 2 (two) different types of outputs in your circuitry:

- LED outputs: You are required to show in which state your system is by using LEDs. Details will be provided below.
- 7-segment displays (SSDs): SSDs will be used to show messages and balance information. Details will be provided below.

Operation Steps

The circuitry will start in the IDLE state, in which it displays ‘CArd’ in SSDs with only LED0 is being ON. In IDLE state, the user can insert its debit card by simply pressing *BTN3*. Then, the circuitry will go to password entry state where ‘PE-3’ will be displayed in SSDs with only LED7 is being ON. The initial password of the user is ‘0000’ and the user should enter the password by setting the *SW* input and pressing *BTN3*.

The user has 3 rights to enter password correctly at the password entry state. If the user enters the password wrong for the first time, LED6 should also be ON along with LED7 and ‘PE-2’ should be displayed on SSD. If the user enters the password wrong for the second time, LED5 should also be ON along with LED6 and LED7, and ‘PE-1’ should be displayed on SSDs. If the user enters password wrong for the third time in a row, the ATM should be locked (no button works except for reset button) for 5 seconds where all LEDs should be ON and ‘FAIL’ should be displayed on SSDs. After 5 seconds, the circuitry should go to the IDLE state. If the user presses *BTN1* at any point of password entry stage, it should be logged out from the system and go to the IDLE state.

If the user enters the password correctly, it goes to the ATM menu state where only LED4 is ON and ‘OPEn’ is displayed on SSDs. In this state, the user has 3 options: 1) press *BTN3* and perform money deposit/withdraw operations, 2) press *BTN2* and change the password, 3) press *BTN1* and log out from the ATM (go to IDLE state).

1) If the user presses *BTN3* in the ATM menu state, the circuitry should go to a state where the user can see the account balance and deposit/withdraw money to/from the account. In this state, only LED3 is ON and the current balance of the user should be displayed on SSDs in 4-digit hexadecimal format. In this state (money operation state), the user can deposit money to the account by setting the *SW* input and pressing *BTN3*. After money is deposited into the account, the updated account balance should be displayed on SSDs. In this state, the user can withdraw money from the account by setting the *SW* input and pressing *BTN2*. After money is withdrawn from the account, the updated account balance should be displayed on SSDs. If the user tries to withdraw an amount of money that is higher than the money in the account, the circuitry should not perform the operation and it should be locked for 2.5 seconds (no button works except for reset button) and ‘-NA-’ should be displayed on SSDs with all LEDs are ON. Then, the circuitry should go back to the money operation state. In this state, if the user presses *BTN1*, the circuitry will go back to the ATM menu state.

2) If the user presses *BTN2* in the ATM menu state, the circuitry should go to a state where the user can change the password. In this state (password change state), only LED2 should be ON and ‘PC-3’ should be displayed on SSDs. In this state, the user first should enter the current password by setting the *SW* and pressing *BTN3*. The user has 3 rights to enter the current password correctly. If the user enters the password wrong for the first time, LED1 should also be ON along with LED2 and ‘PC-2’ should

be displayed on SSD. If the user enters the password wrong for the second time, LED0 should also be ON along with LED1 and LED2, and 'PC-1' should be displayed on SSDs. If the user enters the password wrong for the third time, the circuitry should be locked (no button works except for reset button) for 5 seconds, all LEDs should be ON and 'FAIL' should be displayed on SSDs. After 5 seconds, the circuitry should go to the IDLE state. If the current password is entered correctly, only LED1 should be ON and 'PASS' should be displayed on SSDs. Then, the user should enter the new password by setting the *SW* and pressing *BTN3*. Then, the circuitry should go back to the ATM menu state. In password change state, if the user presses *BTN1*, the circuitry will go back to the ATM menu state.

3) If the user presses *BTN1* in the ATM menu state, the user should be logged out and the circuitry should go to the IDLE state.

Unless the user resets the circuitry, user's current balance and password should not be reset. For example, the user enters the system, changes the password, deposit money into the account and log out from the ATM. If the user wants to re-enter the ATM, he/she should use the new password (and the user's account balance should be equal to the amount of money deposited.).

Appendix A: An Example Template

```
module atm (input clk,
            input rst,
            input BTN3, BTN2, BTN1,
            input [3:0] SW,
            output reg [7:0] LED,                          // LED[7] is the left most-LED
            output reg [6:0] digit4, digit3, digit2, digit1  // digit4 is the left-most SSD
           );

     reg [3:0] password;
     reg [15:0] balance;
     reg [7:0] current_state;
     reg [7:0] next_state;
     …                                          // 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: Extra Modules Needed for the Implementation on FPGA

In the final implementation of your circuit, you will need some extra modules, with which we provide you under "**Resources→Term Project→Modules**". These are "clock divider", "debouncer" and "seven segment driver" modules. There are comments in the modules about the units the modules implement.

> The clock divider module (**clk_divider.v**) generates a clock signal with a period of 50 ms, from a 25 MHz input clock (Please note that the BASYS boards can provide 3 different clock frequency: 25 MHz, 50 MHz, 100 MHz. These are set using the jumpers on the BASYS boards. We set the clock of all BASYS boards to 25 MHz. Please do not play with jumpers on the BASYS boards!).
> The debouncer (**debouncer.v**) circuit gets the input from a push button and generates a one clock pulse output.
> The seven segment driver (**ssd.v**) 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. Also, your circuit should also use the divided clock.

An SSD and its control signals 'abcdefg' are shown below. Using 7-bit 'abcdefg' control signals, you can display different digits and signs on an SSD. For example, 'abcdefg' should be '0000001' in order to display 0 and '0000110' in order to display 3. There are four SSDs on the BASYS board. Your Verilog module should have 7-bit output for each SSD; digit4, digit3, digit2 and digit1 are used to display letters or digits on the first (left-most), second, third and fourth (right-most) SSDs, respectively.

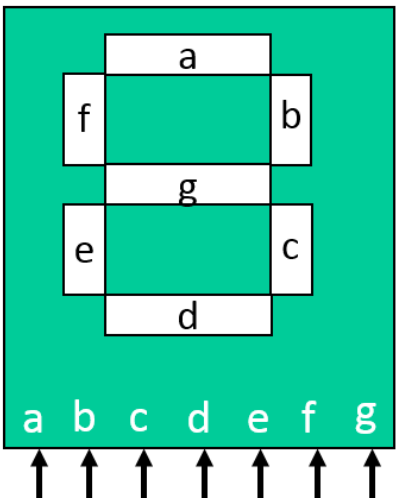

Fig. SSD control signals

Appendix C: Details on Simulation Demo and Final Demo

During the simulation demos (see "Project Plan and Deadlines" section for the demo dates), 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. During the final demos (see "Project Plan and Deadlines" section for the demo dates), you are expected to show your implementation working on the BASYS board.

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. Also, if you want to **check your work**, you can come to the FENS 1033 during the office hours.

Appendix D: Simulation Scenario

In the simulation demo, you need to show the correct transitions between the states for the simulation scenario below. This scenario is written as a Verilog testbench (**atm_test.v**) and provided to you under "**Resources→Term_Project→Test**". Please note that you may need to modify this testbench based on your variable/module names (read comments in atm_test.v).

In order to observe your state transition, you should be able to see your state signals. Please read "**Verilog Testbench**" document under **Resources→Lab_Materials→Test** carefully for learning how to debug (observe internal signals of your design) your Verilog code.

Simulation scenario:

a. Reset the circuit
b. Inset the debit card by pressing *BTN3*
c. Enter the password correctly and go to the ATM menu state
d. Go to the money operation state by pressing *BTN3*
e. Deposit 5 into your account (by setting the *SW* and pressing *BTN3*)
f. Go back to the ATM menu state by pressing *BTN1*
g. Go to password change state by pressing *BTN2*
h. Enter the current password correctly (by setting the *SW* and pressing *BTN3*)
i. Enter the new password (by setting the *SW* and pressing *BTN3*)
j. Log out from the ATM by pressing *BTN1*
k. Inset the debit card by pressing *BTN3*
l. Enter the password wrong for 3 times (by setting the *SW* and pressing *BTN3*)
m. Inset the debit card by pressing *BTN3*
n. Enter the password correctly and go to the ATM menu state (by setting the *SW* and pressing *BTN3*)
o. Go to the money operation state by pressing *BTN3*
p. Withdraw 4 from your account (by setting the *SW* and pressing *BTN2*)
q. Withdraw 2 from your account (by setting the *SW* and pressing *BTN2*)
r. Go back to the ATM menu state by pressing *BTN1*
s. Go to the password change state by pressing *BTN2*
t. Enter the current password wrong for 3 times (by setting the *SW* and pressing *BTN3* -- you should be logged out)

Notes:

1. Your project and ISim should work properly with the given scenario (testbench). You are not allowed to change your codes during your demo.
2. For simulation, you should 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 **on time**. 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.
   Dec 2, 2019

2. ASM_CHART: The ASM Chart is submitted in *hardcopy* to the boxes in front of FENS1033.
   Dec 9, 2019 11:00

3. SIMULATION: You should zip your project and submit it to SUCourse.
   Dec 16, 2019 11:00

4. SIMULATION DEMO: ISim simulation is demonstrated to the lab assistants.
   Dec 16, 2019 – Dec 20, 2019

5. FINAL: You should zip your project and project report, and submit them to SUCourse.
   Dec 22, 2019 23:55

6. DEMO: The circuit is realized on FPGA and a demonstration must be made.
   Dec 23, 2019 – Dec 27, 2019

Note that these deadlines are hard, and there will be no additional time.

Notes:

- **The first two groups that finish the project earliest will be awarded by ten (10) bonus points in the term project.**

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 websites 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.