Lab 5: The Fancy Mask Vending Machine

Submission Due Dates:

Source Code: `2020/11/10 18:30`

Report: `2020/11/15 23:59`

Objective

1 Getting familiar with modeling finite state machines with Verilog.
2 Getting familiar with the FPGA design flow and the demo-board I/Os.

Description

Due to the COVID-19 tsunami, Wakanda government decides to set up mask vending machines everywhere. Now, you are chosen to be the warrior to help them design the Fancy Mask Vending Machine, which behaves as follows:

- The basic concept of how this mask vending machine works:
  1 Initially, the machine will show “0000” on the 7-segment display. All LEDs turn off.
  2 When the customer presses the money button, the corresponding coin will be deposited into the machine. The two **rightmost** 7-segment digits will show the balance (the amount of deposited money); the two **leftmost** 7-segment digits will show the number of masks that the customer can buy with the current balance.
  3 After finishing depositing coins, the customer can adjust the number of masks to purchase.
  4 When the customer presses the **cancel** button, the machine will return the deposited money.
  5 When the customer presses the **check** button after depositing coins and adjusting the purchase amount, masks will come out. While releasing the masks, the 7-segment display will show “MASK” (or customized alphabetic messages), and all LEDs will be flashing to emulate the masks coming out.
  6 After the masks are released, the change will be returned. The 7-segment display will show the decreasing balance to indicate that the change is returning coin by coin.

- The detailed spec:
  1 INITIAL state: (5%)
  After being reset, the machine goes to the INITIAL state. The 7-segment display will show “0000”. All the LEDs will be off. Then, the machine goes to the DEPOSIT state.

  2 DEPOSIT state: (25%)
  a. First, the customer may deposit $5 coin by pressing the *money_5* button or $10

coin by pressing the *money_10* button. The two **rightmost** 7-segment digits will show the total amount of money that has been deposited so far (*balance*). Note that the *balance* cannot be larger than 50.

b. The price of a piece of the mask is $5. The two **leftmost** 7-segment digits will show the number of masks that the customer can buy (*max*) with the current *balance*. For example, if the present *balance* is 35, the *max* will be 7; the two **leftmost** 7-segment digits will show “07”. Note that a customer can buy only 9 pieces at most, i.e., the *max* cannot be over 9 (even if the *balance* is 50).

c. The machine will enter the CHANGE state to return all the deposited money if the customer presses the *cancel* button and the *balance* > 0.

d. If the customer presses the *check* button and the *balance* > 0, the machine will enter the AMOUNT state.

3 AMOUNT state: (20%)

a. Initially, the 7-segment display will remain the same as the end of DEPOSIT state, i.e., if the *balance* is 35 and the *max* is 7, then the 7-segment display will remain “0735”.

b. Then, the customer can press the *count_down* button to adjust the number of masks that the customer wants to buy according to the *max*. E.g., if the *max* is 7, then the *amount* can only be in 1~7 ([1, *max*]). The two **leftmost** 7-segment digits will behave as a 1~7 counter that only counts between 1 and 7.

c. For the *count_down* button’s detailed behaviors, assume the 7-segment display shows “0735” at first (*amount* = *max* = 7). If the customer presses the *count_down* button, the *amount* will be decreased by 1. The display will then become “0635”. If the customer presses the *count_down* button when the *amount* is 1, the *amount* will return to 7 (*max*).

d. The machine will enter the CHANGE state to return all the deposited money if the customer presses the *cancel* button.

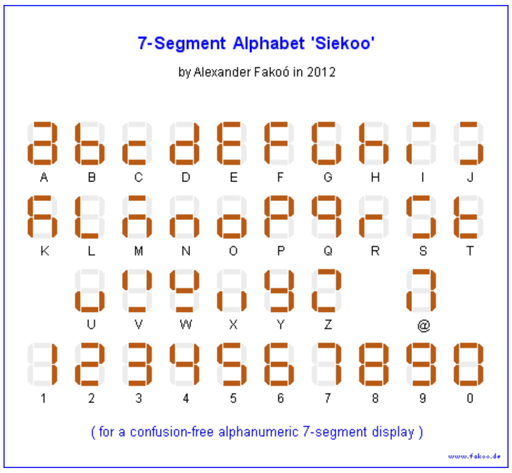
e. If the customer presses the *check* button, the machine will enter the RELEASE state.

4 RELEASE state: (20%)

a. All LEDs will turn on and off repeatedly. To be specific, all LEDs will turn on for 1 second and off for another 1 second, and so on.

b. The 7-segment display will show “MASK” or other customized alphabetic messages (not numbers) for about 5 seconds **(You may use one cycle of clk/(2^27) as 1 second).** You can refer to the 7-segment Alphabet Chart to design your own message instead of “MASK”.

c. After about 5 seconds, the machine will enter the CHANGE state, and all LEDs will turn off.


7-Segment Alphabet 'Siekoo'
by Alexander Fakoó in 2012
A B C D E F G H I J
K L M N O P Q R S T
U V W X Y Z @
1 2 3 4 5 6 7 8 9 0
(for a confusion-free alphanumeric 7-segment display)
www.fakoo.de

5 CHANGE state: (30%)

a. The two **leftmost** 7-segment digits will show “00”, and the two **rightmost** 7-segment digits will show the change to be returned to the customer ($balance$ – ($amount$ * 5)).

b. If there are \$10 or more to return, the change will be decreased by \$10 (to simulate dropping a \$10 coin) **with the frequency of clk/(2^27)**. When there is less than \$10 to return, the change will be decreased by \$5 (to simulate dropping a \$5 coin).

c. The two **rightmost** 7-segment digits will show the decreasing change. E.g., if the total change to be returned is \$15, then the display sequence will be: “15” -> “05” -> “00” sequentially.

d. After all the change being returned, the machine will return to the INITIAL state.

6 *Bonus feature: (5%)

If the customer doesn’t take any action for about 5 seconds (refer to Hint c) in the AMOUNT state, the machine will enter the CHANGE state to return all the deposited money.

I/O signal specification

- ***clk***: clock signal with the frequency of 100MHz (connected to pin **W5**).
- ***rst***: active-high reset (connected to **SW0**).

✧ Each signal from the pushbutton in the following should be processed by debouncing and one-pulse converters properly.

- ***money_5***: deposit a \$5 coin (connected to **BTND**).
- ***money_10***: deposit a \$10 coin (connected to **BTNU**).
- ***cancel***: cancel the purchase and enter the CHANGE state. (connected to **BTNC**).
- ***check***: confirm the operation (DEPOSIT and AMOUNT) and enter the next state (connected to **BTNR**).
- ***count_down***: adjust the number of masks of the purchase (connected to

BTNL).

- ***LED[15:0]***: signals to control LEDs (connected to **LD15 ~ LD0**).
- ***DIGIT[3:0]***: signals to enable one of the 7-segment digits.
- ***DISPLAY[6:0]***: signals to show the digits on the 7-segment display.

Note

1 The operating frequency of each debouncing or one-pulse converter may be clk/(2^16).

2 The operating frequency of the 7-segment display controller may be clk/(2^13).

3 Demo video:
https://youtu.be/F1P1whgtG1w

Hint

1 You must design at least one finite state machine (FSM).

   a. There should be at least five states. More states or multiple FSMs are acceptable. But remember to explain your design in the report.

   b. The proper operating frequency for each state may be:

      ✓ INITIAL: clk/2^16
      ✓ DEPOSIT: clk/2^16
      ✓ AMOUNT: clk/2^16
      ✓ RELEASE: clk/2^27
      ✓ CHANGE: clk/2^27

   You may operate at a higher clock rate and use a counter to slow down the display rate. (e.g., the FSM can operate at the clock rate of clk/2^16. But in the RELEASE or CHANGE state, it may count for 2^11 cycles for each state transition to behave as the clock rate of clk/2^27).

   Or you may use a clock selector with multiple clock sources as a quick fix. Usually, it also works.

   c. The period of clk/2^29 is about 5 seconds. You may design a counter to measure this period for the bonus.

2 You can use the following template for your design:

```
module lab5(clk, rst, money_5, money_10, cancel, check,
count_down, LED, DIGIT, DISPLAY);
  input clk;
  input rst;
  input money_5;
  input money_10;
```

```
    input cancel;
    input check;
    input count_down;
    output [15:0] LED;
    output [3:0] DIGIT;
    output [6:0] DISPLAY;
    // you can declare the output data type by yourself:)
    // e.g. output (reg/wire) [15:0] LED;
    // add you design here
endmodule
```

3 It would help if you drew the state diagram(s) before coding. You should put your state diagram(s) in the report.

Attention

- ✓ You should hand in only one Verilog file, lab5.v. If you have several modules in your design, integrate them in lab5.v. (From this lab, you don't need to put debounce and one-pulse in the file you hand in. Please do not integrate them in lab5.v)
- ✓ You should also hand in your report as lab5_report_StudentID.pdf (e.g., lab5_report_108062666.pdf).
- ✓ Please do not hand in any compressed files, which will be considered as an incorrect format.
- ✓ You should be able to answer the questions of this lab from TA during the demo.
- ✓ You need to generate the bitstream before the demo.
- ✓ If you have any questions about the spec, feel free to ask on the iLMS forum.