GRAND VALLEY STATE UNIVERSITY

SEYMOUR AND ESTHER PADNOS COLLEGE OF ENGINEERING AND COMPUTING

EGR680 High Level Implementation on FPGA

Laboratory 03

State Machine Design

Professor: Dr. C. Parikh

Student: Dimitri Häring

September 17, 2018

Contents

1 Introduction **3**

2 Design **3**
2.1 Vending machine specification 3
2.2 Vending machine design description 4
2.2.1 Display 4
2.2.2 De-bounce 4
2.2.3 Edge detector 5
2.2.4 3 s Delay 5
2.2.5 State Machine 5
2.3 Vending machine State Machine 6
2.4 Recommended improvements for future work 6

3 Simulation **7**

4 Conclusion **7**

5 Appendix **8**
5.1 Lesson Learned 8
5.1.1 Vivado Language Templates 8
5.1.2 Clock divider not working 8
5.1.3 Implementation Error [Place 30-574] Poor placement for routing between an I/O pin and BUFG 9

1 Introduction

The goal of the lab 3 is to familiarize the student with a finite state machine implementation in verilog.

2 Design

In this section the design and decisions that where made to achieve the laboratory are discussed. The code for lab 03 can be reviewed online under the flowing link http://github.com/haringd/EGR680_FPGA_Labs/tree/master/VENDMA

2.1 Vending machine specification

Verilog is used to describe a decoder that shall control a seven-segment display with two switches and one button. The given task is as followed:

The design should be implemented on t he PYNQ board using two Pmod 7-segment displays. VEND-MACH is a vending machine that accepts nickels, and dimes, and dispenses gum, apple, or yogurt. A gum pack costs 10¢, an apple is 15¢, and yogurt is 20¢. The machine is only allowed to accept up to 20¢. Any coins inserted that pushes the value beyond 20¢should be ignored.

- **NICKEL** â signal that becomes 1 when a nickel is deposited in the coin slot.
- **DIME** a signal that becomes 1 when a dime is deposited in the coin slot.
- **GUM** a signal that becomes 1 when the gum selection button is pressed.
- **APPLE** a signal that becomes 1 when the apple select ion button is pressed.
- **YOGURT** a signal that becomes 1 when the yogurt select ion button is pressed.

In addition ion to these “user” inputs, the machine has two control inputs:

- **CLOCK** a timing signal that sequences the state transit ions of the machine.
- **RST** an initialization signal that resets the machine to a suitable starting state.

The machine has three outputs:

- **MONEY ENTERED** The amount of money inserted into the machine should be displayed on one of the 7-segment Pmod displays. This should update every time a coin is inserted (i.e. 5¢should read 05). Upon startup or reset, the display should read VEND.
- **DISPENSED ITEM** The item that was just purchased should be displayed on the 7-segment: g for gum, A for apple, and y for yogurt, as indicated in figure below.
- **CHANGE** The amount of money returned in change. The machine returns change using only cents; the number of cents returned should be displayed as a binary number using the on-board LEDs.

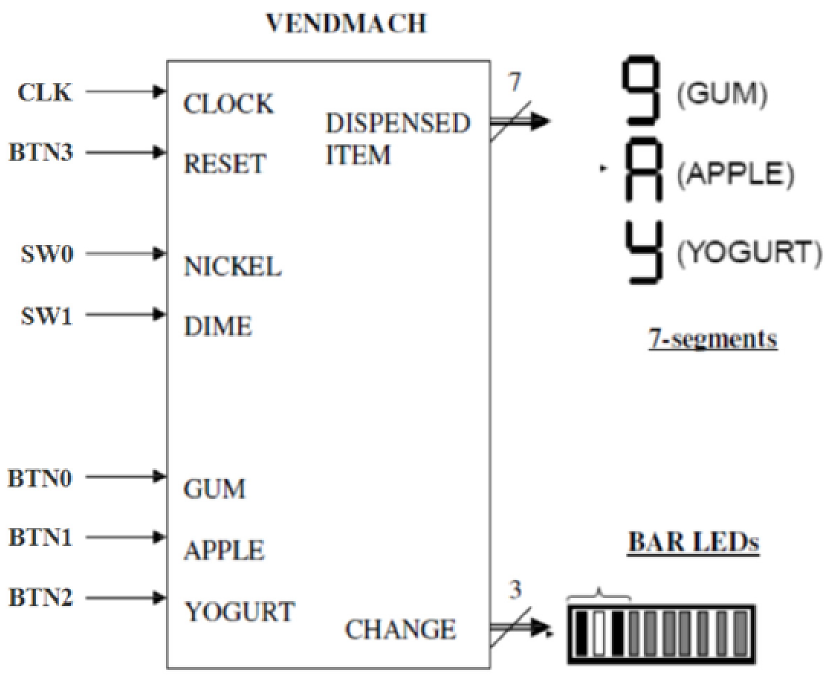


Figure 1: VENDMACH top level module.

2.2 Vending machine design description

The above specification leads to the following table truth tables which can be used as aid to implement the logic in HDL. Table 1 shows the logic relation between switch an coin. Table 2 shows the relation ship between button and product.

2.2.1 Display

The knowledge that different values at different displays have to be shown lead to the idea of using an ASCII to seven-segment converter (ascii2seg) as separate module so in the state machine the actual out put can be represented as ASCII character. The ascii2seg output is routed in to the decoder that is based on a four to two MUX. The decoder is synchronized with a slower clk signal otherwise the display would not be visible. The state machine module has four character outputs theretofore the ascii2seg converter is build as module and used four times between state machine module and decoder.

2.2.2 De-bounce

Input signals on the PYNQ-Z1 board are realized with switches and buttons that according to testing are not hardware de-bounced. This means that a button or switch has a ringing while the element switches from off to on state or from on state to off state. This can be easily encountered in software with a de-bouncer. The de-bouncer is realized with an two bit shift register and a state comparison between synchronized input and synchronized output that determines the output logic. To make it a denouncer the clock has to be slower then the ringing period of the switched element, so it uses a slow clock. The de-bouncer is implemented as module which allows the instantiation of the module for as many signals as have to be de-bounced. Notice, that because the de-bounce module uses a synchronous reset the reset button can not be de-bounced with it.

2.2.3 Edge detector

After an input signal is de-bounced usually an edge detector is switched in serious that is build as the same as the de-bouncer but operates on the system clock to account for the faster switching past of the state machine (SM). Due to the fact that the Edge detector has the same clock as the SM it was implemented in the state machine file as in depended process in a synchronous always statement.

2.2.4 3 s Delay

A 3 s delay is used to keep the state after the vending process was successful that the display does not switch immediately to idle state. Therefor a simple counter was used to count up for $375 * 10^6$ ticks for a 125 MHz system clock used in the state machine.

2.2.5 State Machine

The state machine is implemented in Moore Machine with an case statement synchronized on the the state value used for output logic. A second case statement is used to implement the logic flow depended on the given input states. The drawn a bit simplified state machine is shown in figure 2 therefore it can be easily used in the file as header which helps a lot by development on small screens.

Description	Switches	Displayed Pmod A
Nickel is 5 ¢	SW0	05
Dime is 10 ¢	SW1	10

Table 1: VENDMACH allowed coins.

Product	Price	Displayed Pmod B
Gum	10 ¢	g
Apple	15 ¢	A
Yogurt	20 ¢	y

Table 2: VENDMACH products and prices.

2.3 Vending machine State Machine

```
/**********************************************************************************
** FSM State machine                                                            **
***********************************************************************************
                   000
+---------------------+               +----------------------+
| Idle                |               |                      |
| Display "VENT"      |<--------------| Reset                |<------+
|                     |               | BTN3                 |       |
+---------------------+               +----------------------+-      |
    |                                                                |
    |                                                                |
    |              002                                               |       001
+---------------------+ Add to coin_val                   +---------------------+
|                     |---------------+                   | Delay               |
| Coin entered show   |               |                   | 3s                  |
| coin_val on Pmod A  |<--------------+                   |                     |
+---------------------+                                   +---------------------+
    |                                                               / \
    |                                                              / | \
    |              003                                               |       006
+---------------------+                                   +---------------------+
| IF value > Gum      |                                   | Vend                |
| & coin_val > 10     |---------------------------------->|                     |
| Display 'g' by BTN0 |                                   | Disp returned cents |
+---------------------+                                   +---------------------+
   |                                                                / \
   |                                                               / | \
   |             004                                                 |
+---------------------+                                              |
| IF value > Apple    |                                              |
| & coin_val > 15     |------------------->-------------------->-----+
| Display 'A' by BTN1 |                                              |
+---------------------+                                              |
    |                                                                |
    |                                                                |
    |              005                                               |
+---------------------+                                              |
| IF value > Yogurt   |                                              |
| & coin_val > 20     |------------------->-------------------->-----+
| Display 'y' by BTN2 |
+---------------------+
```

Figure 2: VNEDMACH state machine.

2.4 Recommended improvements for future work

1. It is highly recommended for the development process that simulation and original hardware is used simultaneously if possible. A lot of time was lost due to a simple bug that most likely would have been found easily if in early design states would have used the hardware as well instead of simulation. Furthermore, the module initialization should be done with labels so that the order of the module

input signals does not matter.

2. The Edge detector should be implemented as independent module.
3. Decide if a signal shall be set with blocking or non blocking statement, but do not mix those among another.

3 Simulation

Describes the result of the behavioral simulation based on the synthesized hardware description language. The most important module to simulate is the state machine because it includes the logic applied out of the state diagram. One problem by simulating out of the top level are the clock divider which would need way more simulation time then 1000 ns therefore it is good practice to simulate each module once and proof its function by writing individual test bench files for each module. Figure 3 shows the simulated state machine with the state gum. By using radix for the char01 to char 12 signals an easy readable simulated result can be achieved. For an extended report more simulations would be added and explained in more detail.

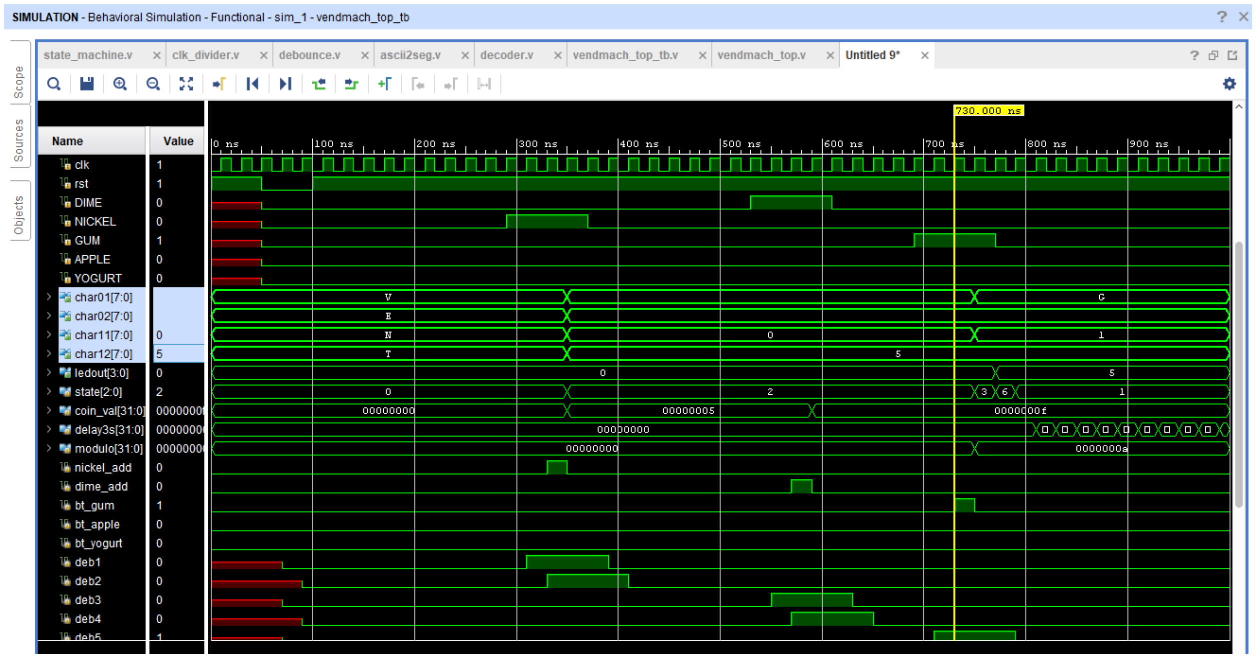


Figure 3: Vivado behavioral simulation of the state machine module.

4 Conclusion

The lab demonstrates the use of state machines implemented as a Moor machine. Important concepts of HDL and logic implementation could be learned trough to the warriors problems encountered during implementation.

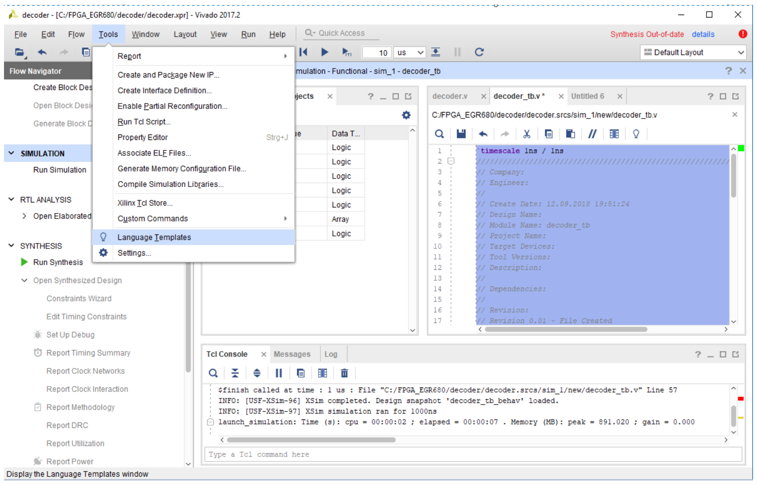
5 Appendix

The appendix contains code listening and other large information parts that contain partial or complete relevance to the reports topic.

5.1 Lesson Learned

5.1.1 Vivado Language Templates

Figure 4 shows a handy tool build into Vivado software package that is called Language Templates and it seems to contain most of the verilog syntax.



C:/FPGA_EGR680/decoder/decoder.srcs/sim_1/new/decoder_tb.v

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 12.09.2018 19:51:24
// Design Name:
// Module Name: decoder_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
```

```
$finish called at time : 1 us : File "C:/FPGA_EGR680/decoder/decoder.srcs/sim_1/new/decoder_tb.v" Line 57
INFO: [USF-XSim-96] XSim completed. Design snapshot 'decoder_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:02 ; elapsed = 00:00:07 . Memory (MB): peak = 891.020 ; gain = 0.000
```

Type a Tcl command here

Figure 4: Vivado Language Templates.

5.1.2 Clock divider not working

Listening 1 shows a code snipe out of the clock divider that caused a lot of trouble by preventing the clock divider preventing dividing the clock. It seems that the last statement $clk_out <= clk_out$; overwrote the statement $clk_out <= clk_out$; which results either in an unknown state or an zero on the divided clock instead of the desired toggling clock behavior.

Listing 1: Clock divider issue.

```
else
begin

clk_temp <= clk_temp + 1;
if (clk_temp >= 500000)
//if (clk_temp >= 2) // Used for testbench
begin
clk_out <= ~clk_out; // no clue why this line does not toggle the clk_out
//    if (clk_out == 0)
//    begin
//    clk_out=1;
//    end
//    else
//    begin
//    clk_out=0;
//    end
```

```
clk_temp <= 0;
end
end // else rst
//clk_out <= clk_out; // could this lane of HDL be causong the problem of the not work
```

5.1.3 Implementation Error [Place 30-574] Poor placement for routing between an I/O pin and BUFG

Poor placement for routing between an I/O pin and BUFG is a state where a clk signal is routed to anon clock net or a non clock signal is routed to clock net like a posedge or negedge clk signal. Now it seems this clock placement error that occurs by implementing the project is somewhat related to the decoder. As the decoder module is commented out of the top level block the error is not there any more. Still the source or what pin shall causing the issue could not be located at first. The issue was that the in the decoder initialization the clk and rst pin where switched.