EGR680 High Level Implementation on FPGA

Laboratory 03

State Machine Design

Professor: Dr. C. Parikh

Student: Dimitri Häring

September 17, 2018

Contents

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.

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. VENDMACH 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** a 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.

The above specification leads to the following table.

Description	Displayed Pmod A
Nickel is 5 ¢	05
Dime is 10 ¢	10

Table 1: μC pin occupation

As software package to implement the decoder in Verilog Vivado 2017.2 is used. Further improvements that would increase the code quality and re-usability of the modules would be to change the clk_divider module that the divider ratio is given as input to the clock divider.

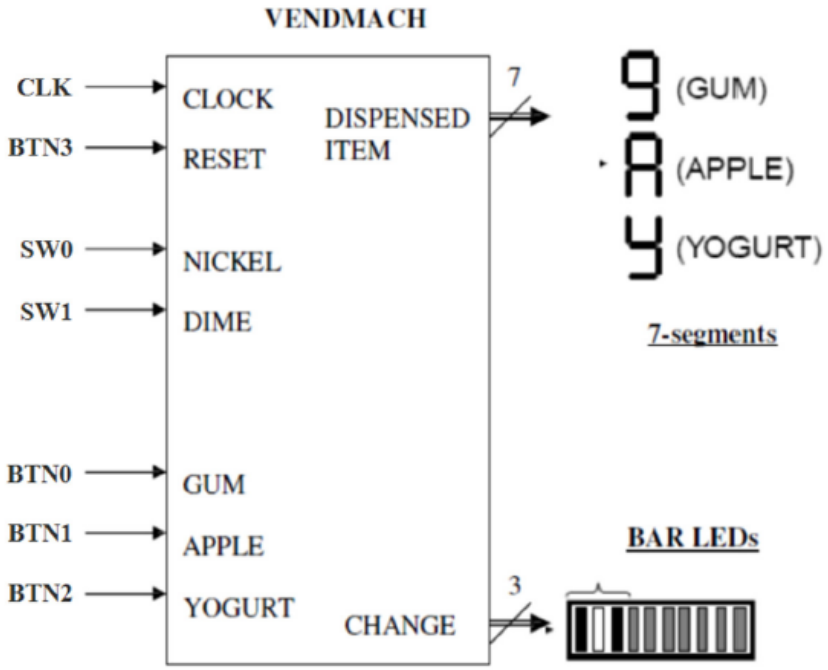

Figure 1: VENDMACH top level module.

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

Table 2: μC pin occupation

3 Simulation

Describes the result of the behavioral simulation based on the synthesized hardware description language.

4 Conclusion

The lab demonstrates in three parts how to build up in simple steps an top level hierarchy with multiple models by using simulations to proof behavior of a module based on set parameters in a test bench. Furthermore, the synthesized model was used to generate a bit stream and loaded on to the PYNQ development board for demonstration purpose.

5 Appendix

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

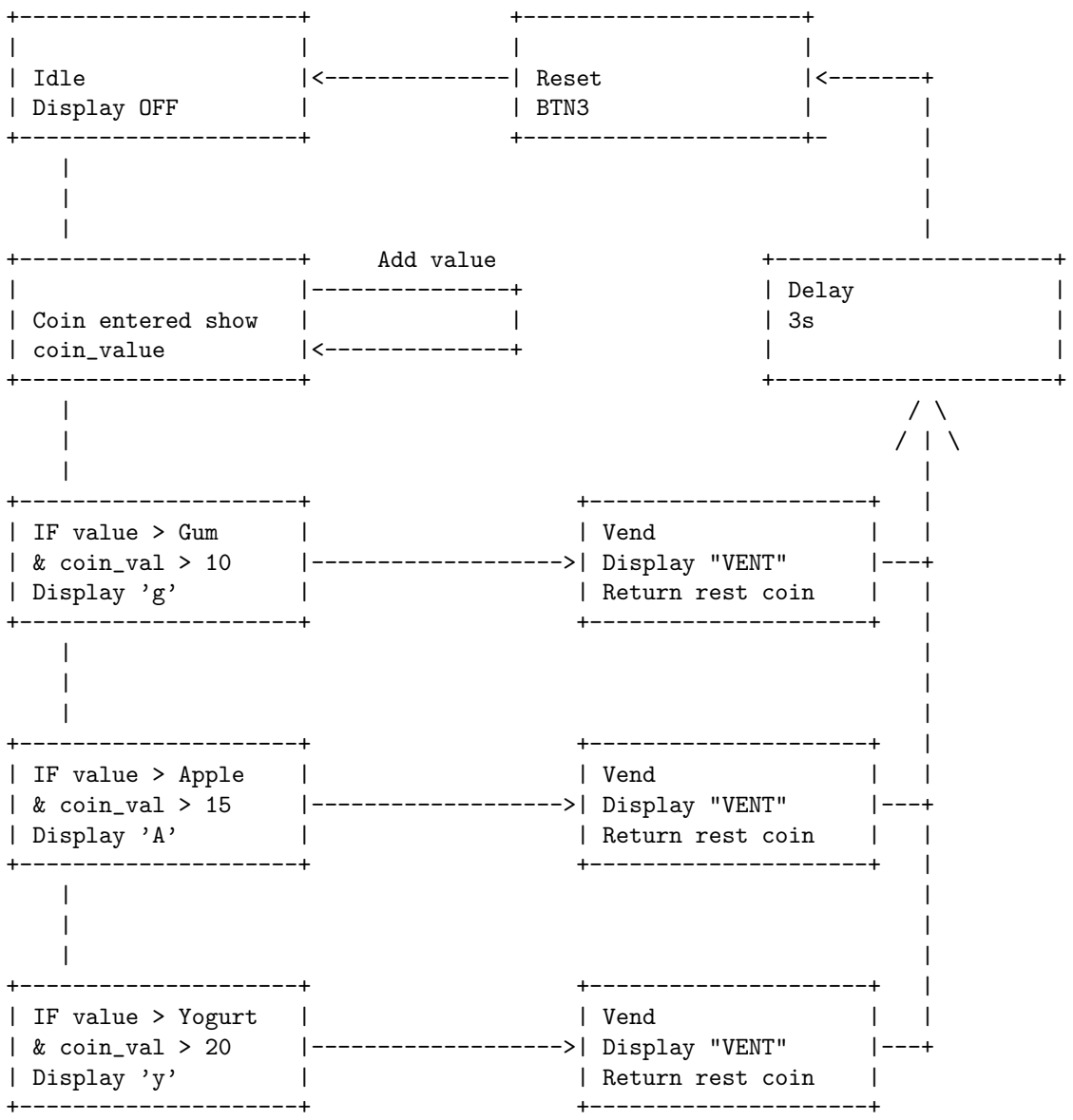

Figure 2: VNEDMACH state machine.

5.1 Lesson Learned

5.1.1 Vivado Language Templates

Figure 3 shows a handy tool build into Vivado software package that is called Language Templates and it seems to contain most of the verilog syntax.

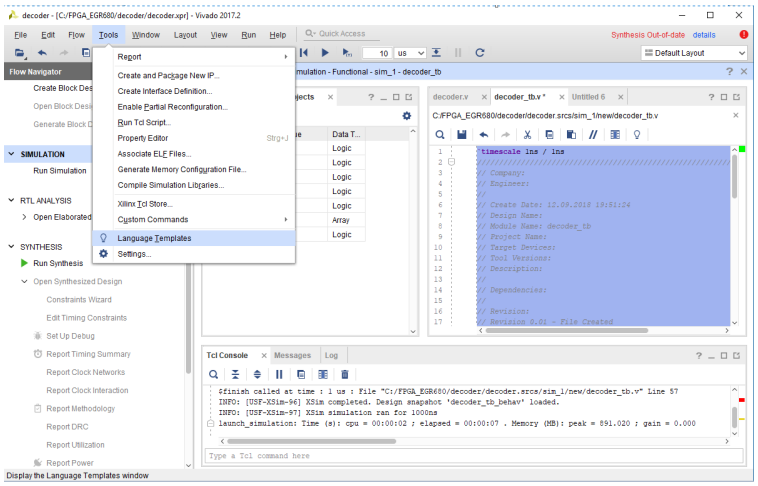

Figure 3: Vivado Language Templates.

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
```