EIE330 FPGA Based System Design

Report of Laboratory 3

FPGA Lab3: Vending Machine

Source Code of this report can be found at: EIE330-FPGA-Laboratory

by

PENGRUI K. TANG
Student No: 1220031811
from EIE330 D1,
Faculty of Innovation Engineering,
Universidade de Ciência e Tecnologia de Macau

XIAOYANG S. ZHEN
Student No: 1220034170
from EIE330 D1,
Faculty of Innovation Engineering,
Universidade de Ciência e Tecnologia de Macau

Oct, 2024
(Coloane, Macao SAR)

This report is a part of the result for EIE330 FPGA Based System Design in U.C.T.M

This page is intentionally left blank.

Conteúdo

Capítulo I

Abstract

I Result location

The Source Code of this report can be found in the git-hub repository: EIE330-FPGA-Laboratory.

II Contribution

All relevant output in this report is shown in the following directory.

- **FPGA Lab3**
  1. 2-0. Report [T][Z]
     (a) Text [T][Z]
  2. 3-2. Simple Vending Machine
     (a) Design [Z]
     (b) Test [Z]
  3. 3-3. Vending Machine
     (a) Design [T]
     (b) Test [T]

And the contribution of each member is labeled with their last name:

1. Pengrui Tang: [T]
2. Xiaoyang Zhen: [Z]

Capítulo II

Simple Vending Machine

I Introduction

I: i Background

- **State Machine**
  The state machine is a model used to design and represent systems switching from state to state based on inputs or conditions. We are mainly focused on two types of state machines in this lecture.
  They have different features, the outputs depend on the current state and the input, another one is the outputs depend only on the current state but not the input. The former is called the Mealy machine, and the latter is called the Moore machine.
- **Advantage**
  When managing complex systems and controlling sequential operations, state machine offers significant advantages over other methods Its features of modularity, reliability, efficiency, scalability, maintainability, and flexibility make it highly practical and adaptable, enabling its use across a wide range of industries.
- **Importance**
  State machines are essential to both software engineering and hardware engineering because they provide a structured and effective way to control complex systems.
- **Application**
  As described above, the state machine has a wide range of practical applications. From embedded systems to user interfaces, game development to operating systems, automation to digital circuits, et cetera. State machines are versatile tools adapted across industries to manage complex workflows, especially repetitive workflows.

II Purpose of Experiment

We are going to design a vending machine that sells beverages of price 2 patacas per cola. It contains the functions of accepting coins of 50 avos and 1 pataca, delivering cola, and output coins for change.

II: i Design and Key-results

The designed circuit should be able to accept 0.5 and 1 pataca coins and deliver cola when a total of 2 patacas is accepted, meanwhile, it also should refund money for change under the situation of over 2 patacas accepted.

We successfully designed the circuit and tested it in the Quartus II application simulation using Verilog code.

III Materials and Methods

III: i Designed Structure Block Diagram

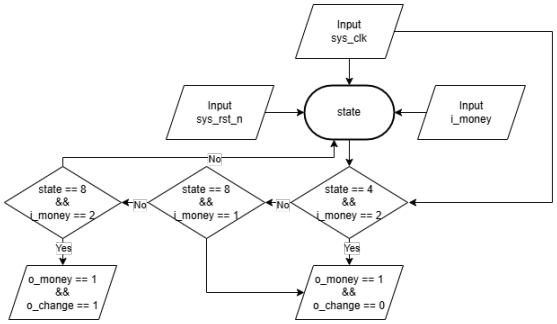

Figura 1: Logic Design of Vending Machine

III: ii Designed Signal Waveforms

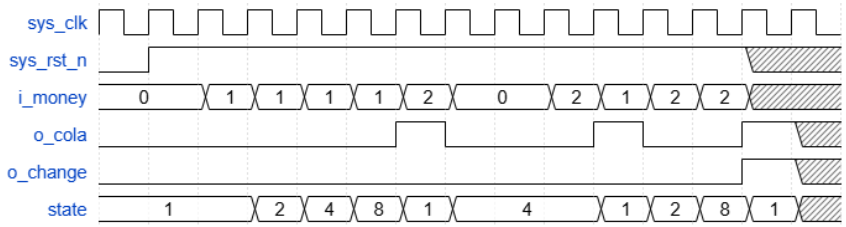

Figura 2: Design Wave of Vending Machine

III: iii Designed State Transfer Diagram

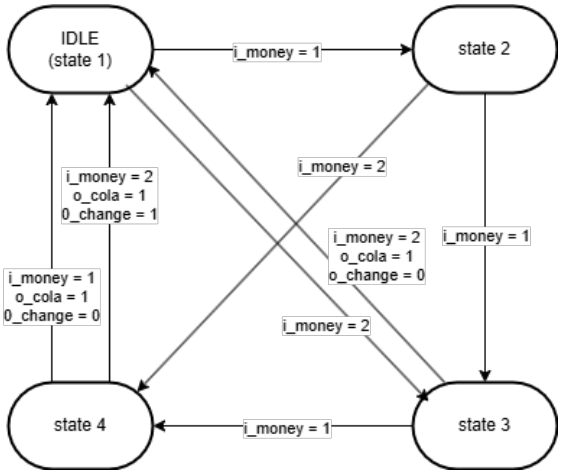

Figura 3: Designed State Transition Diagram

The vending machine contains one clock unit, one internal variable, two inputs, and two outputs.

As the design wave (figure 7) shows, at the time RESET (sys_rst_n) is pressed, the system is set to the initial state and the system starts running.

Then the structure block diagram (figure 1) and the state transition diagram (figure 3) represent how the state is affected by the i_money and transfer from state to state. Furthermore, they show how the signal of the outputs is decided when the state changes.

IV Results

IV: i Testbench

In this experiment, the testbench was built to accept two inputs: RESET (sys_rst_n) and Money (i_money). The value of inputs is assigned randomly, the range of the inputs' value is:

- sys_rst_n: 0 to 1.
- i_money: 00 to 10 in binary.

The input and output will be presented by the waveform.

IV: ii Simulated Result

As figure 4 shows, once the RESET (sys_rst_n) signal goes downward, the whole system is initiated and started. In this simulation, the signals are represented as binary-base:

- Money input (i_money):
  - 1 means 0.5 pataca is inserted.

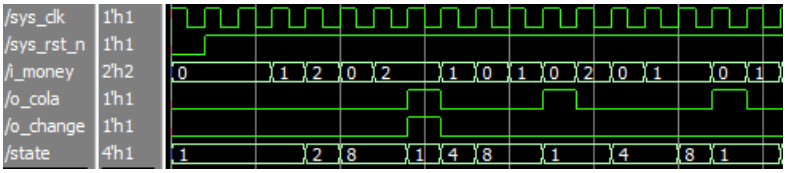

Figura 4: The simulation result of the Waveform

  - 2 means 1.0 pataca is inserted.
  - 0 means no money is inserted.
- State (*state*):
  - 1 means IDLE state, it is waiting for money input.
  - 2 means state 2, where 0.5 pataca was accepted.
  - 4 means state 3, where 1.0 pataca was accepted.
  - 8 means state 4, where 1.5 pataca was accepted.

Since the whole system starts running, inputs are generated with random values for testing.

When *i_money* is 1, the *state* turns from *state* 1 into *state* 2. Then the signal of *i_money* is set to 2, which represents 1.0 pataca, and the *state* turns to *state* 4. At this moment, the total money accepted is 1.5 pataca. After that, *i_money* is simulated as no money inserted, signal is 0. At last, the signal of *i_money* is generated as 2, which means there is another 1.0 pataca inserted, and the total money accepted is over 2.0 pataca. So the signal of change (*o_change*) is set to 1, which means there will be 0.5 pataca being refunded to the customers. And *o_cola* is also set to 1 simultaneously, the cola will be delivered.

When the cola is delivered, the whole system is back to its initial state, the *state* is set to 1 and waiting for money input.

The rest waveform shows it is still functioning after its first coin-deliver cycle.

V Discussions and Conclusion

V: i Discussions

Sequential or not

When writing code for output, we faced a choice between using a sequential circuit or not. After discussion, we notice this problem depends on the signal of the coin input.

If we use a combinational circuit here, the output of this block, (o_cola, o_change) will change if any of its inputs change, in this case, the *state* and *i_coin*. As *i_coin* is an external signal, we have no idea of its property, and it might be unstable or very short. So using a sequential to

update the output only at the clock pulse.

V: ii Conclusion

In this lab, we learned how to design a system via using a state machine.

VI Appendices

The code in this experiment referenced the following sources (IEEE Style):

Referências

[1] ZeppelinSCB, Nov, 2024, "EIE330-FPGA-LAB-Report/../3_VendingMachine_withChange," distributed on Github, https://github.com/ZeppelinSCB/EIE330-FPGA-LAB-Report/blob/main/Lab3/3_VendingMachine_withChange/Sim/tb_CVM.v

Capítulo III

VendingMachine

I Introduction

I: i Purpose

We aim to design a vending machine that sells beverages of price 2 patacas per cola. It contains the functions of accepting coins of 50 avos and 1 pataca, delivering cola, output coins for refund, and output coins for change.

I: ii Design and Key-results

In this section, the newly designed vending machine includes all the functionalities of the previous model.

This designed circuit will be able to accept 0.5 and 1 pataca coins and deliver cola when a total of 2 patacas are accepted. On the other hand, it also should refund money for change under the situation of over 2 patacas accepted.

Nevertheless, it also is designed to refund the coins when the signal of input refund is triggered. The amount of money refunded should be the same as the money inserted.

II Materials and Methods

II: i Structure Block Diagram

As shown in the figure, our vending machine design contains two logic blocks, three inputs, and two outputs.

II: ii State Transfer Diagram

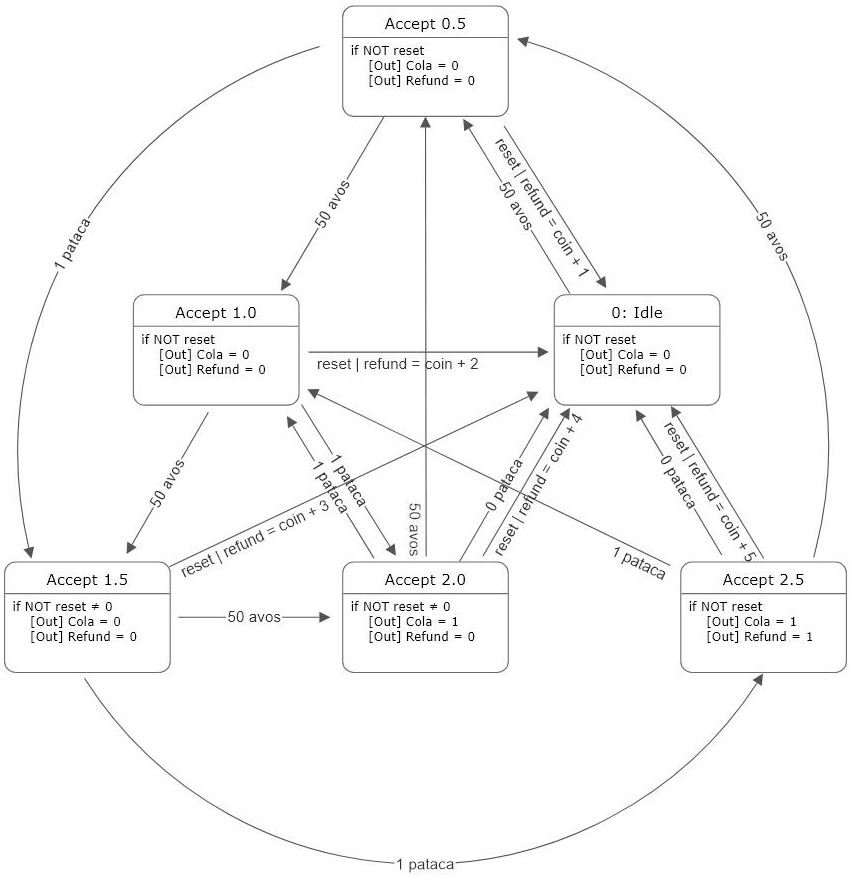

Figura 5: Designed State Transition Diagram

We design our FSM as a mixed type. When in normal operation, it works as a Moore-type state machine, while when the refund button is pressed, it will generate output as a Mealy-type state machine. Where it records the total coins it is accepted in convenience of implementing the refund function. However, we code the design to not stay in the state of accepted 2 Pataca or more money, so to save logic from

II: iii Designed Signal Waveforms

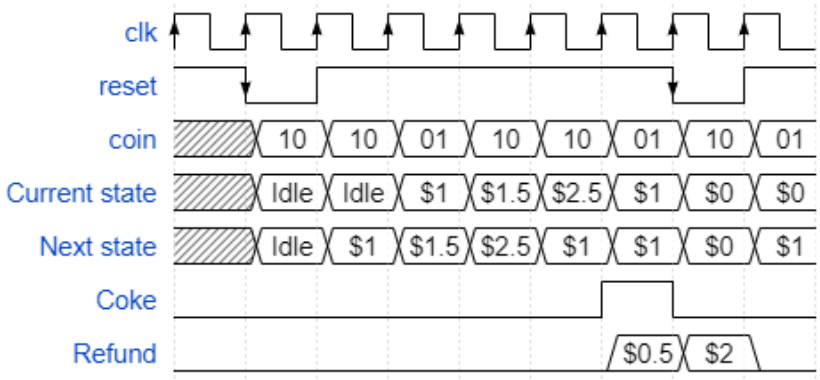

Figura 6: Design Wave of Vending Machine

III Results

III: i Test bench

Inconvenience of checking the result, we write the test bench the same as the design wave, so we can see whether it's the same as our expected easily.

III: ii Simulated Result

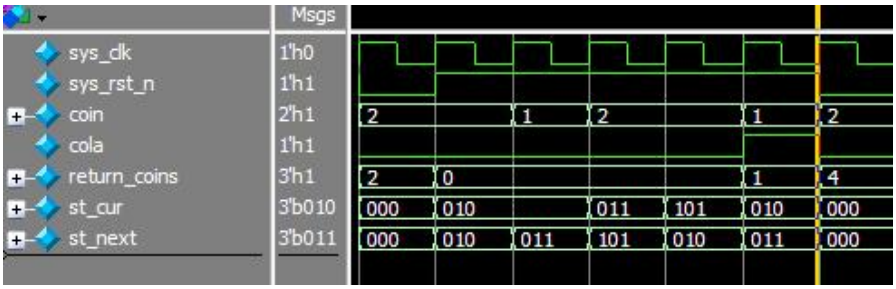

Figura 7: Simulated Result

From the figure, we can see as the reset button is pressed, the current and next state are reset to zero, and the previously inserted coins are refunded. In the following clock, the stat changes according to the inserted pataca. As it accumulates to bigger than 2 pataca, it will make changes, and release a can of coke.

IV Discussions and Conclusion

IV: i Discussions

In the design, I want to save resources by not allowing the state machine to jump to the state where

Moore too slow

Initially, used Moore as Mealy, but it raised Read-and-Write hazards, it caused problems like this. to fix, change back to pure Moore machine

having async reset work at the clock is the least ideal situation

IV: ii Conclusion

In this lab, we learned how to design a system via using a state machine.

V Appendices

The code in this experiment referenced the following sources (IEEE Style):

Referências

[1] pikipity, Aug, 2024, "FPGA-Laboratory/../1_Simple_FSM," distributed on Github, https://github.com/pikipity/FPGA-Laboratory/blob/main/Lab3/1_Simple_FSM/Sim/tb_simple_fsm.v