Vending Machine

EE271 Final Project

Professor: Binh Le

San Jose State University

Members:

Anudeep Yejjala-012426557

Akshit Sushil Rohra-014538550

Suryanto Phienanda-011539554

December 08, 2019

Table of Contents

Abstract

In this project, we designed a vending machine using a Basys3 FPGA and a keypad. The vending machine consists of seven states and after each purchase, the vending machine will go back to the initial state. The vending machine is also capable of cancelling the item, dispensing change and notifying the consumer if the stock is unavailable.

Introduction

Nowadays, there have been so many types of vending machines and each machine behaves in a different way. Based on the complaints that customers have and the vending machines out there, we decide to design a vending machine that satisfy customer's needs. Some vending machines require customers to pay in the exact amount and do not give change. To overcome this issue, we design a change dispenser that will dispense change for customers once they have bought the product. In order to achieve our goal, we designed a state diagram with a number of 7 states, so that the vending machine will work properly, and we understand the flow of the vending machine. The vending machine is also limited to 5 products only and it will only accept: $1, $2, and $5. The seven-segment display is used to display the price of each item and the refund/change from his session. The vending machine algorithm was designed in Vivado, but it was simulated using VCS. In order to verify whether the vending machine works properly, we ran a testbench to see whether specific inputs will result in expected outputs. Our main goal from this project is to build a vending machine that overcomes people's complaints on other vending machines.

Working

There are many ways a vending machine can work. With this vending machine, we design the vending machine, so that, customers can check the price of each of the five products available first, then buy. Once customers insert their money, the seven-segment display will show how much money remaining should be inserted and which item the buyer selected.

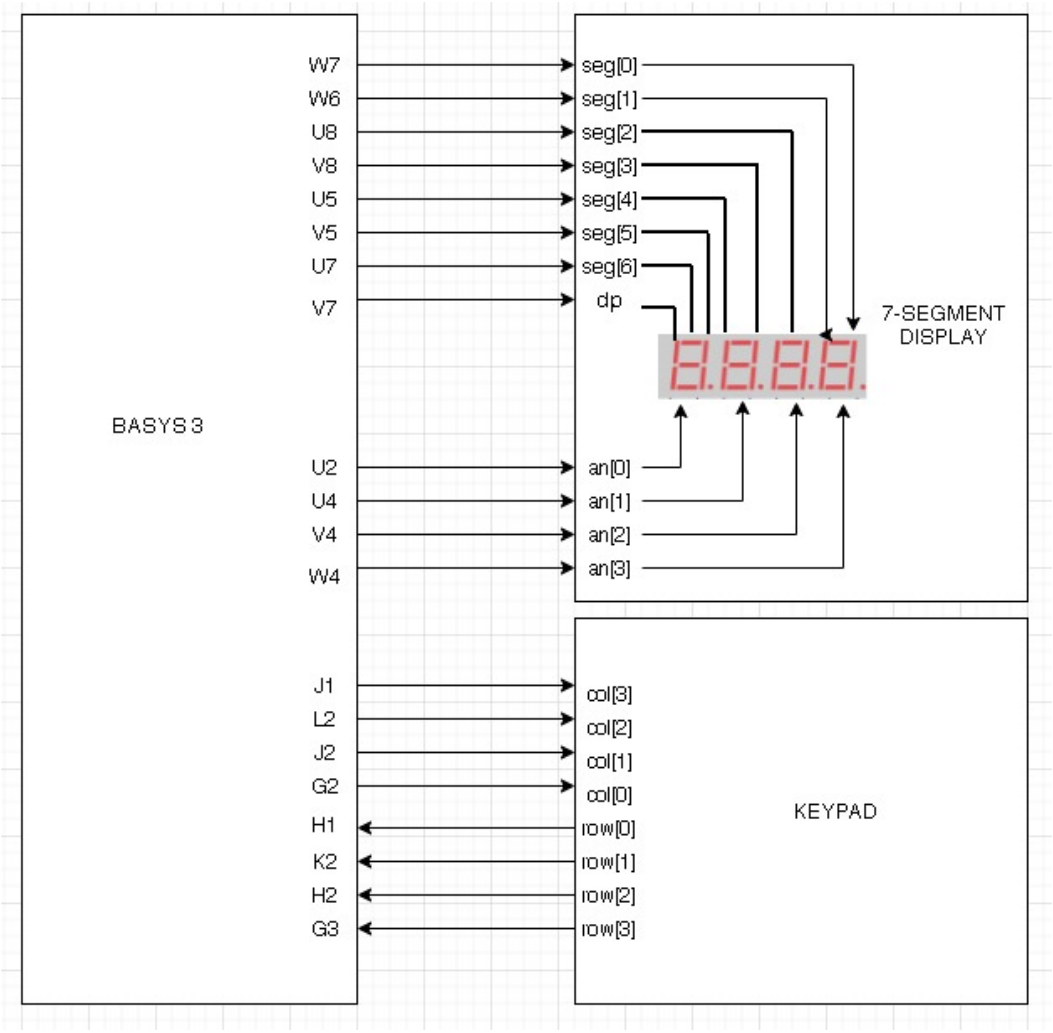

Fig 3.1 Vending Machine Block Diagram

If the selected item is still available, then it will show the remaining money that should be inserted. However, if the item is no longer available (out of stock), then it will go to the seventh state, “No Stock” state. Once the money inserted is enough, the item will be dispensed, and change will be given based on the amount inserted. If the money inserted is more than what is required, then change will be given. However, if the money is exact, then it will not give any change. There is also a “Refund” state, which is basically a cancel button. This state is only triggered if the customer has inserted money to buy the product, but then decides to cancel the item. If the required amount of money has not been met, the customer can always press the

cancel button to get the money back. Once the item has been dispensed, the vending machine must be reset in order to buy a new item.

Since the inputs come from the keypad, we needed to make a clock divider, so that the inputs will not be read incorrectly. We create a clock divider that reads the input every half second. This means that if a customer holds down an input button, it will take half a second to keep reading the input from the keypad. This clock divider is very important because without it, we will run the keypad using the clock provided by FPGA, which is 100 MHz. This means that it will read an input every 10 ns. It is almost impossible to read a single output at this period. As a result, the input read will be incorrect.

Schematic

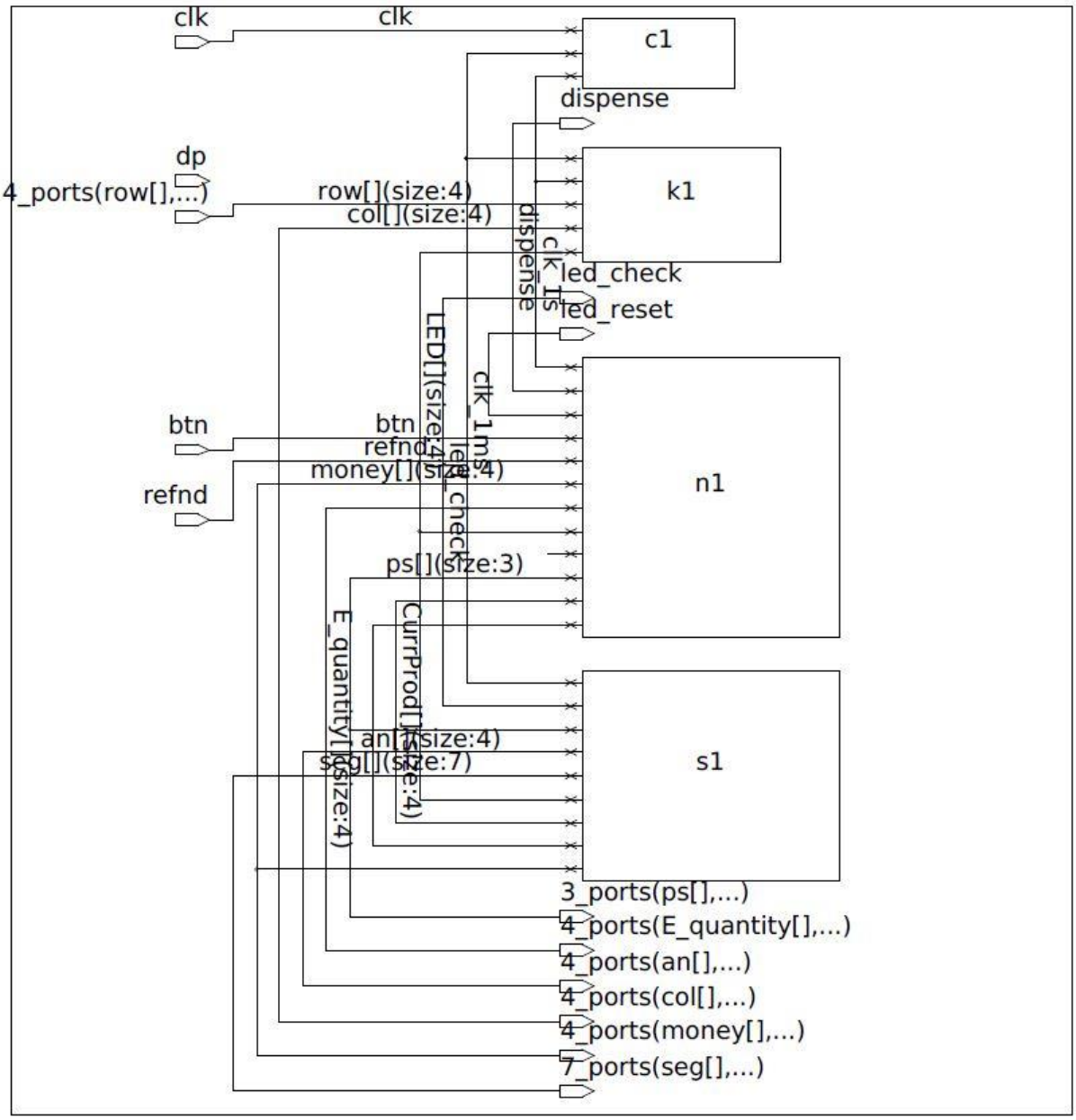

Fig 4.1 Vending Machine Design Schematic

The vending machine that we designed can be built using the modular schematic found in Fig 4.1.

State Diagram

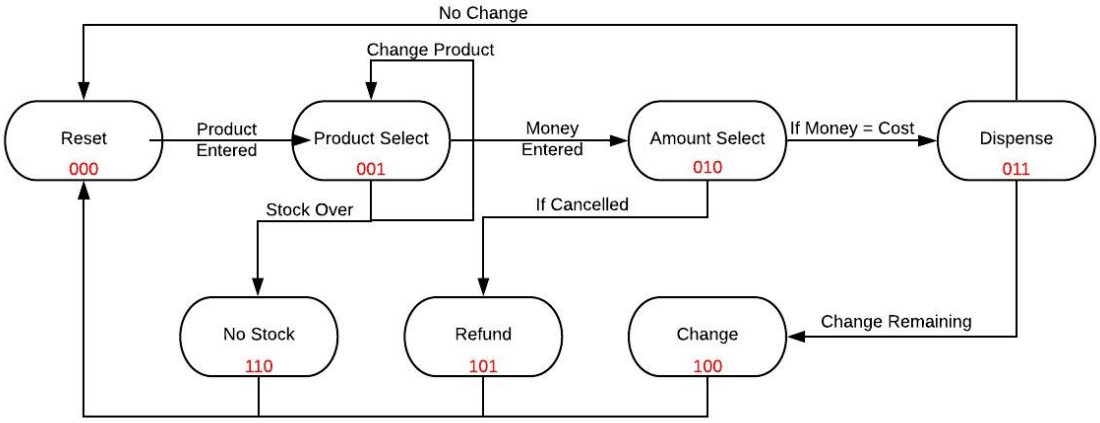

Fig 5.1 Vending Machine State Diagram

[1] Reset State
[2] Product Selection State
[3] Money Amount State
[4] Dispense Item State
[5] Change State
[6] Refund State
[7] Stock State

In the state diagram, we can see that in state 2, there are 3 possible outcomes. Transition from state 2 to state 7 will be triggered automatically if and only if the item is not available. Otherwise, state 2 will only go to state 3 or stay in state 2. Also, in state 4, it will only be triggered to state 5 if and only if the money inserted is more than the money required. Otherwise, we need to press the reset button to go back to the initial state. We can also see that state 6 can only be triggered by state 3. This is logical because state 6 is the refund state, which can only be triggered when customers insert money. For state 5, 6 and 7 to go back to reset state, the reset button needs to be pressed.

Simulation

```
[011539554@coe-ee-cad11 ~/project_271]$ ./simv
Chronologic VCS simulator copyright 1991-2014
Contains Synopsys proprietary information.
Compiler version I-2014.03-2; Runtime version I-2014.03-2;  Dec  6 04:39 2019
state: xxx input: x product: x price:  x money:  x cancel: 0 change:  x dispense: x time:                    0
state: 000 input: x product: x price:  x money:  x cancel: 0 change:  x dispense: x time:                30000
state: 000 input: b product: x price:  x money:  x cancel: 0 change:  x dispense: x time:                40000
state: 000 input: b product: b price:  x money:  x cancel: 0 change:  x dispense: x time:                50000
state: 000 input: 1 product: b price:  x money:  x cancel: 0 change:  x dispense: x time:                60000
state: 001 input: 1 product: b price:  z money:  0 cancel: 0 change:  0 dispense: 0 time:                70000
state: 001 input: 1 product: b price:  5 money:  1 cancel: 0 change:  0 dispense: 0 time:                90000
state: 001 input: 2 product: b price:  5 money:  1 cancel: 0 change:  0 dispense: 0 time:               100000
state: 010 input: 2 product: b price:  5 money:  3 cancel: 0 change:  0 dispense: 0 time:               110000
state: 010 input: 2 product: b price:  5 money:  5 cancel: 0 change:  0 dispense: 0 time:               130000
state: 010 input: 0 product: b price:  5 money:  5 cancel: 0 change:  0 dispense: 0 time:               140000
state: 011 input: 0 product: b price:  5 money:  5 cancel: 0 change:  0 dispense: 0 time:               170000
state: 011 input: 0 product: b price:  5 money:  5 cancel: 0 change:  0 dispense: 1 time:               190000
state: 011 input: 0 product: b price:  5 money:  5 cancel: 1 change:  0 dispense: 1 time:               200000
state: 000 input: 0 product: b price:  5 money:  5 cancel: 1 change:  0 dispense: 1 time:               230000
state: 000 input: 0 product: b price:  5 money:  5 cancel: 0 change:  0 dispense: 1 time:               240000
state: 000 input: b product: b price:  5 money:  5 cancel: 0 change:  0 dispense: 1 time:               250000
state: 001 input: 2 product: b price:  z money:  0 cancel: 0 change:  0 dispense: 0 time:               270000
state: 001 input: 2 product: b price:  5 money:  2 cancel: 0 change:  0 dispense: 0 time:               290000
state: 010 input: 2 product: b price:  5 money:  4 cancel: 0 change:  0 dispense: 0 time:               310000
state: 010 input: 5 product: b price:  5 money:  9 cancel: 0 change:  0 dispense: 0 time:               330000
state: 010 input: 0 product: b price:  5 money:  9 cancel: 0 change:  0 dispense: 0 time:               350000
state: 011 input: 0 product: b price:  5 money:  9 cancel: 0 change:  0 dispense: 0 time:               370000
state: 011 input: 0 product: b price:  5 money:  9 cancel: 0 change:  0 dispense: 1 time:               390000
state: 100 input: 0 product: b price:  5 money:  9 cancel: 0 change:  0 dispense: 1 time:               410000
state: 100 input: 0 product: b price:  5 money:  9 cancel: 0 change:  4 dispense: 1 time:               430000
state: 100 input: 0 product: b price:  5 money:  9 cancel: 1 change:  4 dispense: 1 time:               450000
state: 000 input: 0 product: b price:  5 money:  9 cancel: 1 change:  4 dispense: 1 time:               470000
$finish called from file "top_tb.v", line 58.
$finish at simulation time              490000
         V C S   S i m u l a t i o n   R e p o r t
```

Fig 6.1 Simulation (Testbench) Results

In Fig 6.1, we tried two cases. First case was a case where a customer inserts an exact amount of money and does not receive any change. The second case was when a customer inserts more money than required. As we can see, the customer received $4 change when buying $5 item with $9 money.

Waveform

These are some of the waveforms for various states:

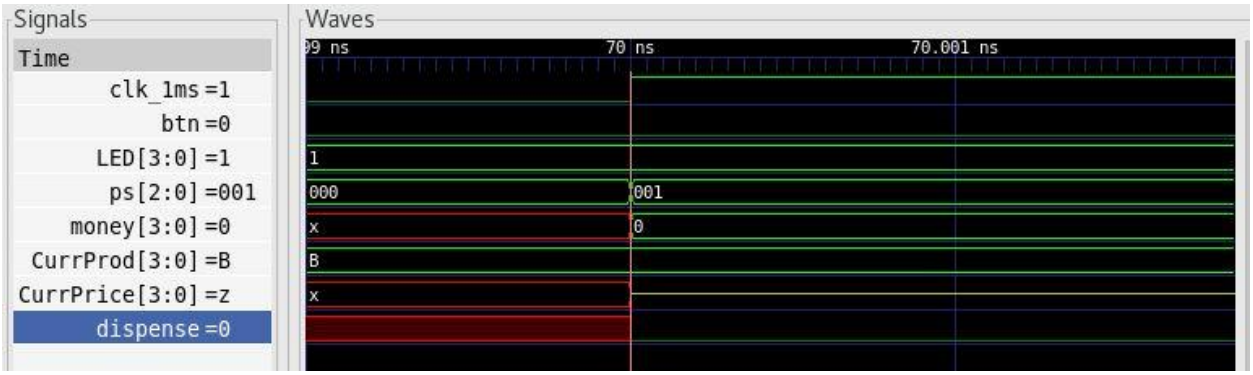

Fig 7.1 First Transition from State 0 to State 1

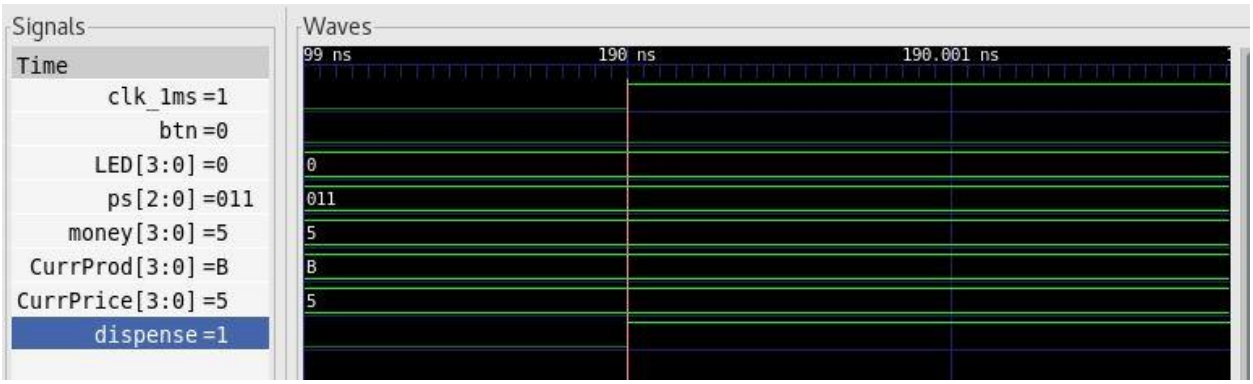

Fig 7.2 Dispensing Item as Money Meets Current Price

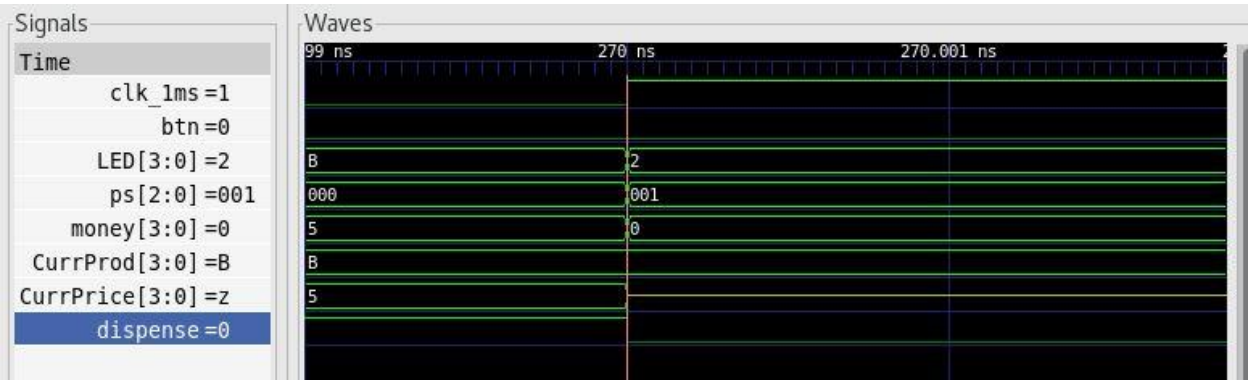

Fig 7.3 Second Case Transition from State 0 to State 1

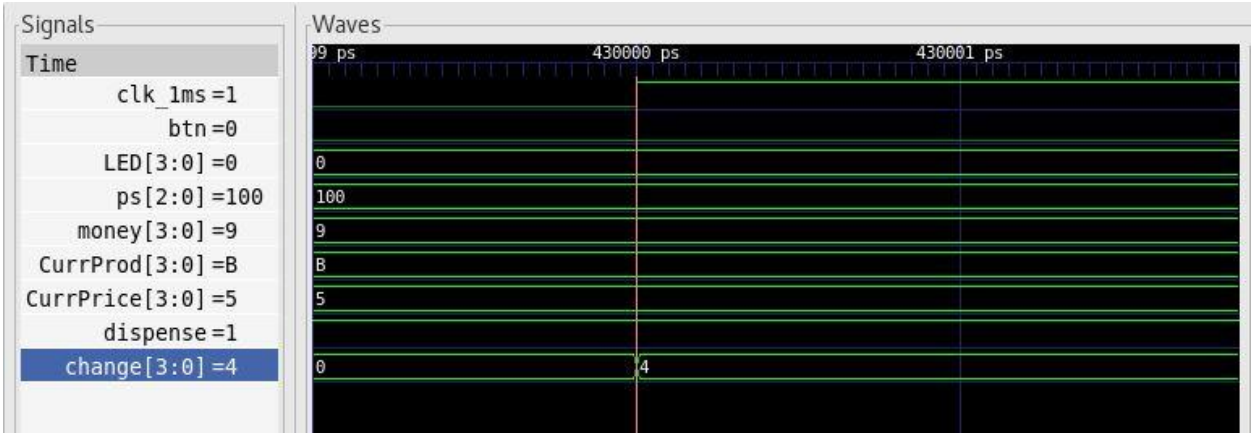

Fig 7.4 Dispensing Item as Money Meets Current Price and Dispense Change

Conclusion

In this project, we apply a state machine technique to design a vending machine. There were some issues that we encountered during working on the project, like managing speeds of input with the clock speed, and interfacing Keypad with FPGA. The main issue that we encountered was solving the bouncing input. We decided to build a different clock divider for the input instead of creating a de-bouncer. Overall, this project was well accomplished, and we were able to perform a vending machine that works properly, dispenses change, refunding money and notifying customers about the availability of the product.

Contribution

- *Akshit:* Works on the No-Stock State and Product Selection State
- *Anudeep:* Works on the Change State and Dispense State
- *Suryanto:* Works on the Money Amount State and Refund State

Many of the tasks, such as figuring out the FPGA and PMOD KYPD, we did together as well as the reset state. However, for the specific state that each of us worked on, we solved it first, then changed the variables in the code, so that our code could work together. Also, the testbench and the waveform was done in group.

References

[1] https://reference.digilentinc.com/basys3/refmanual (Basys3 Reference Manual)

[2] https://reference.digilentinc.com/reference/pmod/pmodkypd/reference-manual (PMOD KYPD Reference Manual)

[3] https://reference.digilentinc.com/_media/basys3:basys3_sch.pdf (Basys3 Schematics)

Annexure

Fig 11.1 Basys3 PMOD Interface

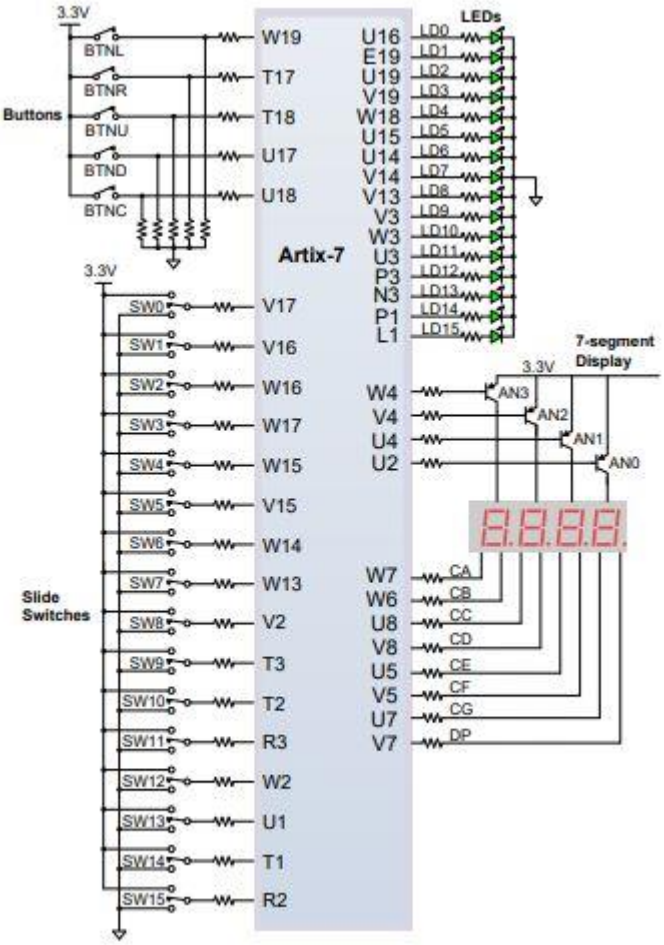

Fig 11.2 Basys 3 Schematic

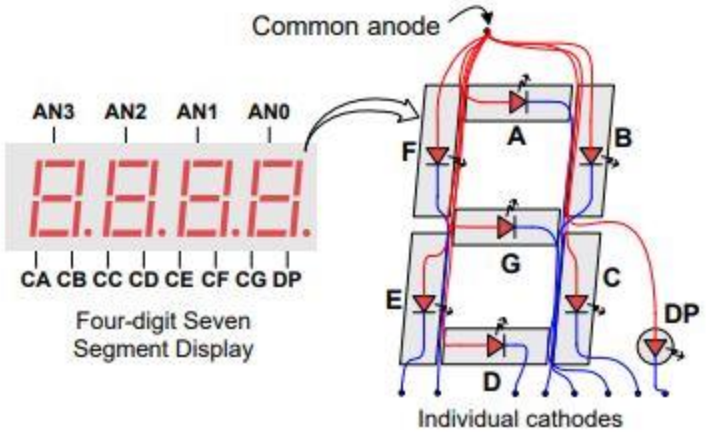

Fig 11.3 Seven-Segment Display

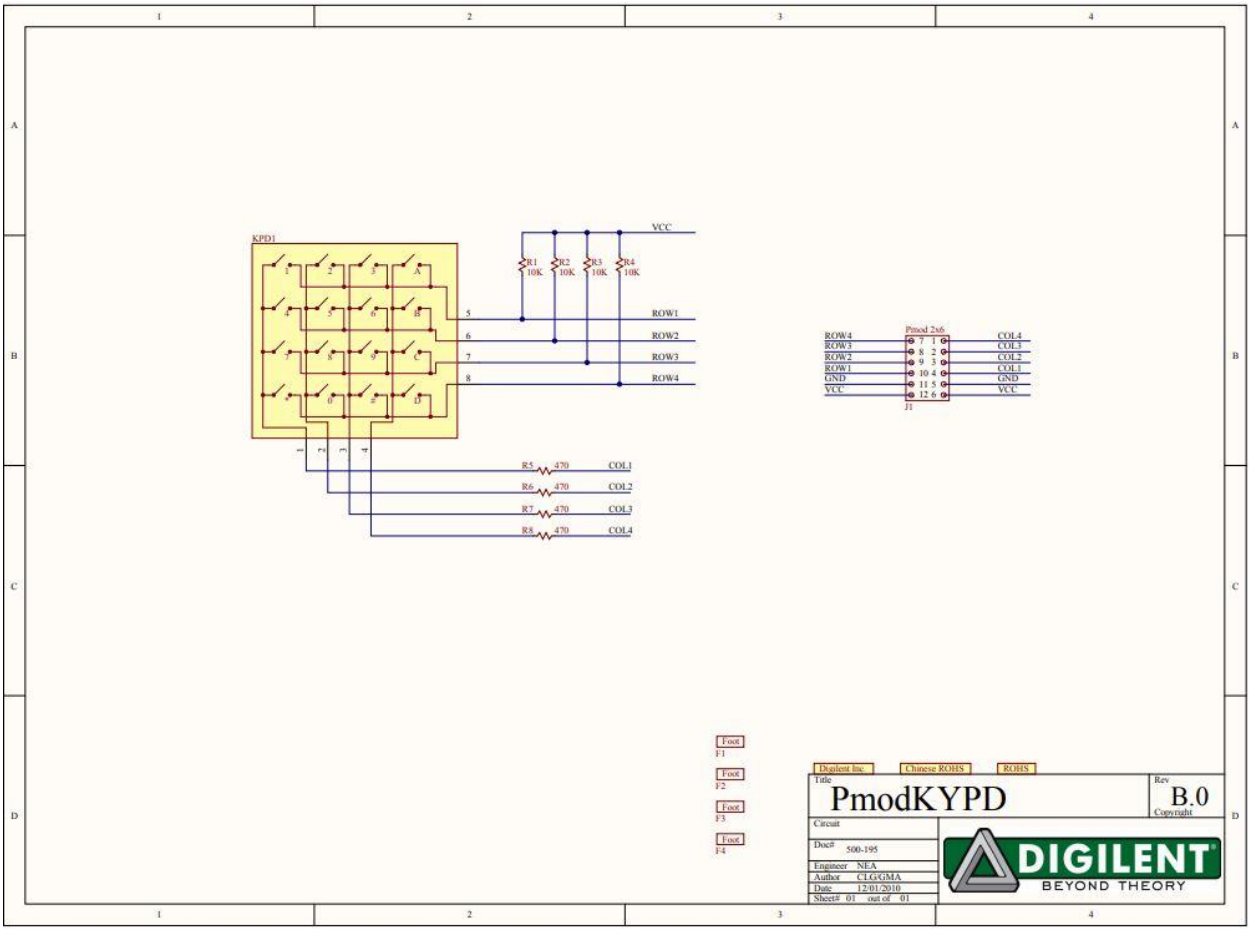

Fig 11.3 PMOD Keypad Schematics