**Vending Machine**

DLD\_4301

Final project

BASU\_UNIVERSITY

Author : Mehrdad Varmaziyar

Content :

* Introduction……………………………………………………………………….3
* Design……………………………………………………………………………………….4
  + Finite state machine………………………………………………………..4
  + Module functionality………………………………………………………..5
* Testing report……………………………………………………………………….6
* User Guide……………………………………………………………………………7
  + Files and Components…………………………………………….…..7
  + Instructions and signals……………………………………..…….7
    - Coin\_input\_controller…………………………………………………….7
    - Dispense\_Item…………………………………………………………………..9
    - Item\_Selector…………………………………………………………………….11
    - Vending\_Machine……………………………………………………………..12

Introduction

In this project we have designed a Vending Machine using VHDL .

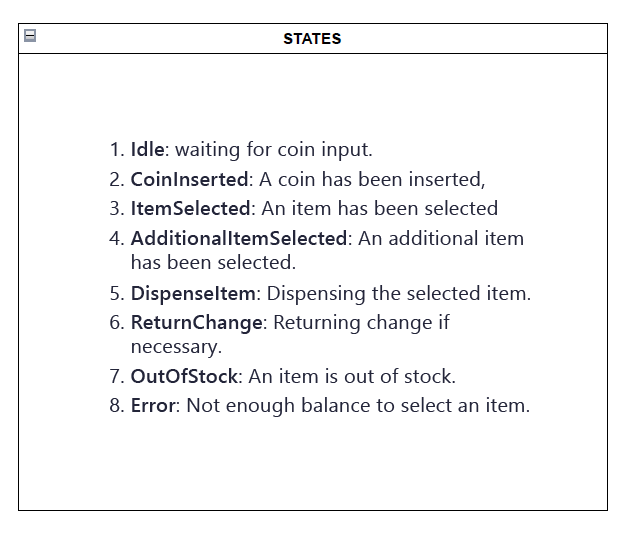
In the following we will discuss state machine and digital design of this hard were and then we will discuss the code and test cases and at the final part you can see the user guide of the project and we have explained components and main code to help you understand it comprehensively .

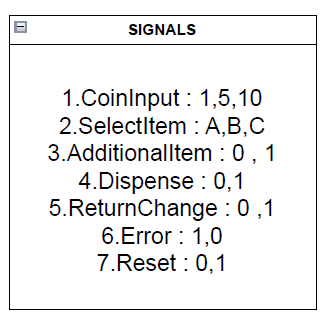
Design

In this part we will check and analyze the design of circuit using FSM and then functionality of module will be explained .

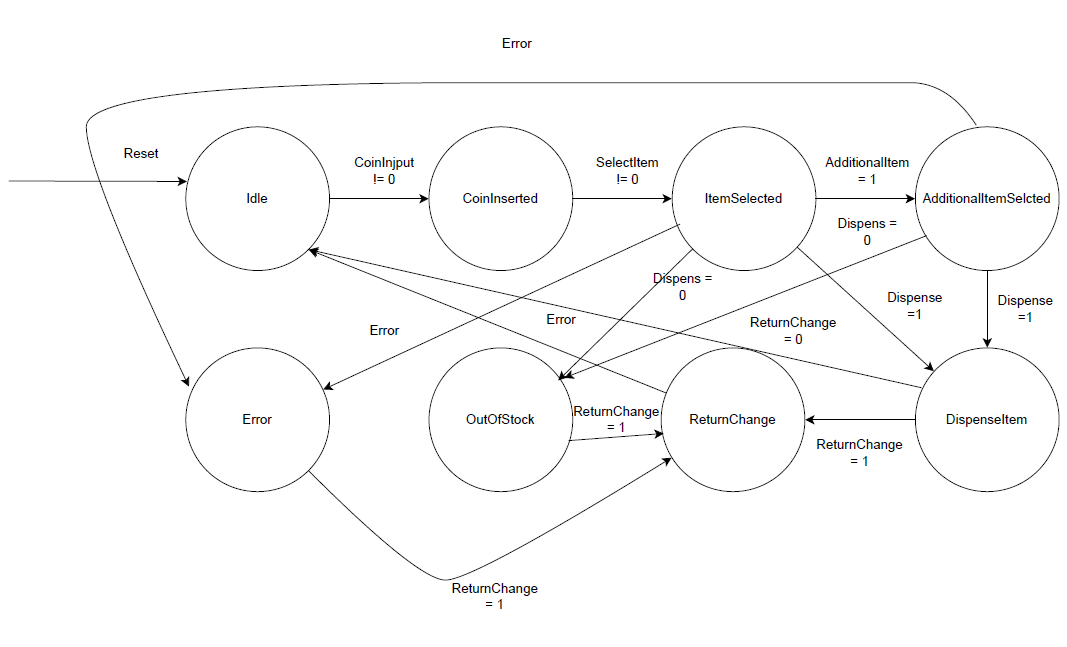
Finite state machine

First of all we should define states for our Vending machine then we will connect states to each other based on different inputs and signals.





Module functionality

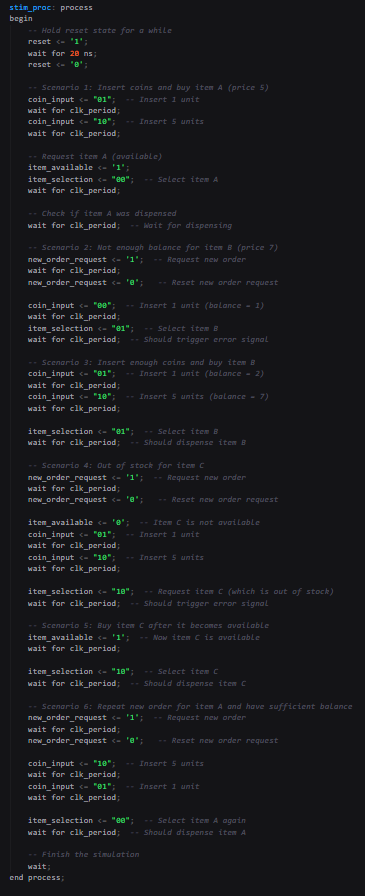


Module Functionality

Our first state is idle and it means the Machine is expecting a coin (signal) to go to our next state Coin Inserted then after inserting coin the machine expect a signal for selecting item and if an item is selected it will go to next state but here we face a dilemma choosing an extra item or dispensing the item and also another dilemma error or out of stock and they will happen based on the signal that you can see on their edges and if we go to out of stock or error or dispense Item we will go to return change and and after that we will get back to idle .

Reset is connected to idle so we somehow reset the system by getting back to Idle.

Testing report

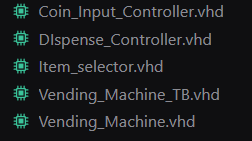


All actions are obvious if you take a look at the comments so I prefer not to explain them again one by one but as a summery when we enter a coin our balance will be increased and then we select item now we have several scenarios consist of error for not having enough balance or out of stock for not having an item or user may choose an extra item and each of this scenarios has their own signal and pass you can see step by step explanation on comments of the code .

User Guide

In this part we will analyze the code and explain usage of each component and explain their inputs , outputs , entities and architectures.

Files and Components

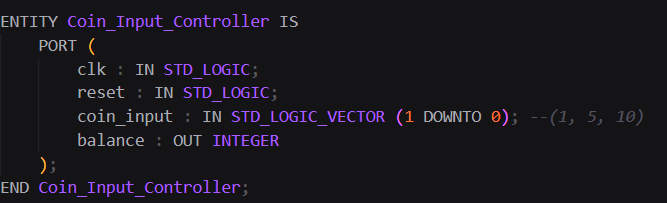


As you can see we have 3 components and main file for vending machine which is made of other 3 components and we have test \_bench file for our project.

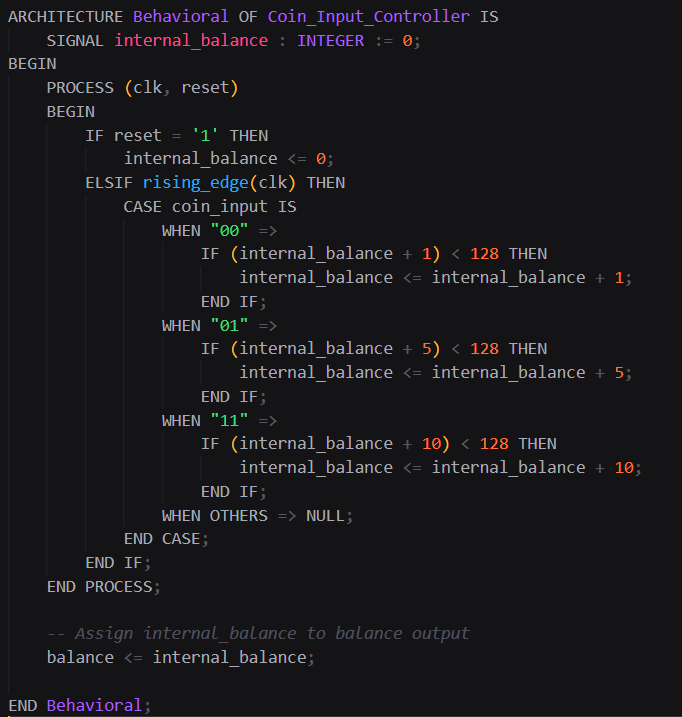
Instruments

Let’s check them all one by one :

* Coin\_Input \_Controller : this component’s duty is to get coin as input and give us a balance as output as you can see on the photo of its entity. (all components have clk and reset as their input so I would not kepp mentioning them for all sections.



Now take a look at its architecture :



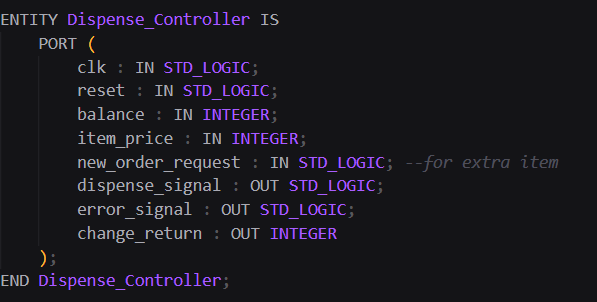
Here we got a behavioral architecture for our component so let’s start to explain it:

We have an internal signal here name internal\_balance and here I will brief the reason of using internal signals and in other part I won’t repeat it again :

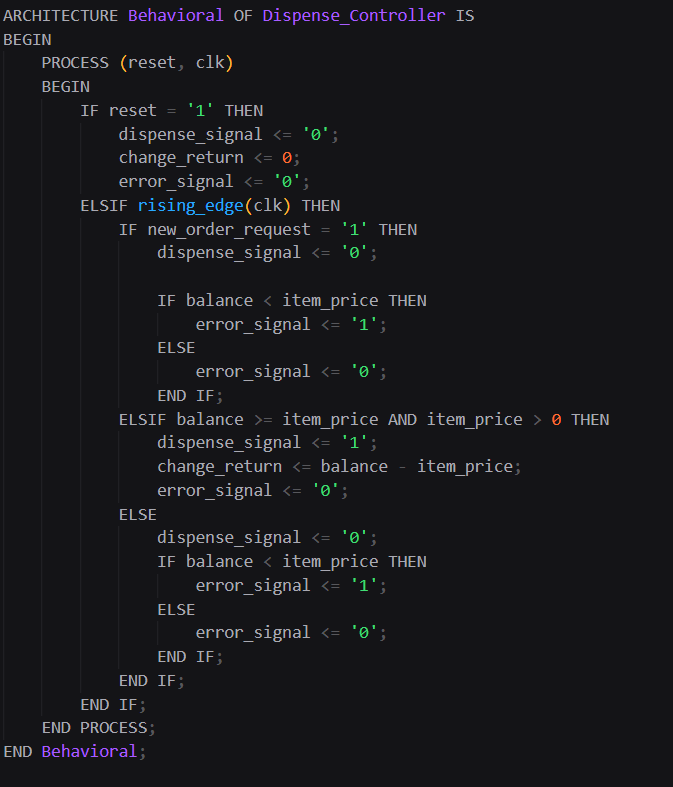
1. it’s the main difference between combinational logic and sequential logic if we assign the value in clk edges to balance directly it would lead to combinational logic.
2. Storage : signals in vhdl acts like a storage element and in digital design it corresponds to using Flip\_Flops.
3. Using an internal signal allows you to control when the balance updates happen. In our design, the balance is only updated when there is a rising edge on the clock. If we were to use the balance output directly, we would have a situation where it could change on every input signal change, which could lead to unpredictable behavior in a synchronous circuit.

The other part in this component is clear and easy to understand we just have a progress which increase the balance based on the coin input and you may notice the condition that controls balance not to exceed 128 , we use it to have an limitation for our system and avoid potential errors from wrong inputs but we could’ve not to use it .

* Dispense\_Controller : so here we get balance , item price and extra item request as inputs and dispense signal and change (of balance ) and error as output.

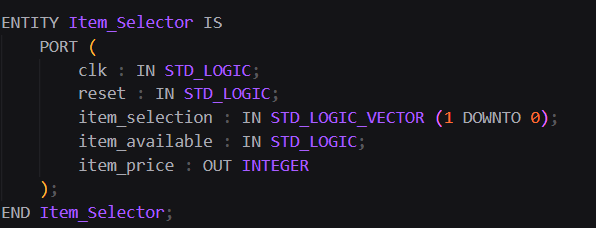


Take a look at its architecture :

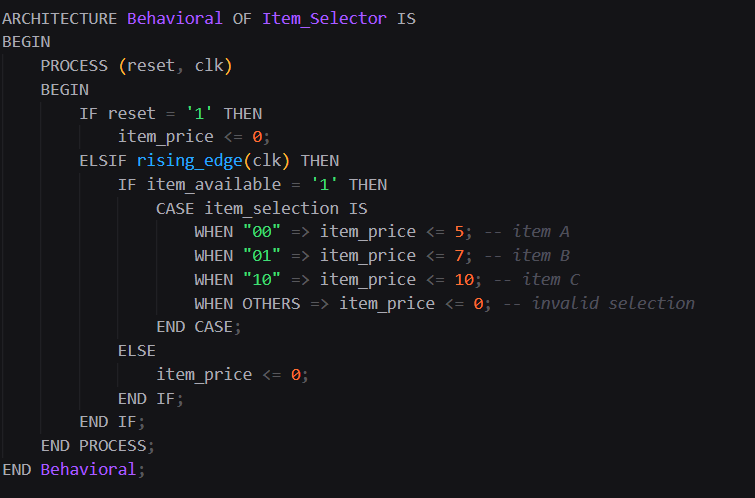


It is only an obvious progress I forgot to talk about reset in previous section here I tell you if reset is 1 everything will be 0 so now about other parts in rising edge of clk if user has an extra request dispense signal would be 0 it means we won’t dispense item until user choose next item but now for dispensing we’re gonna have an error if user has not enough balance and if he has we will dispense item and return change to him from balance.

* Item\_Selector : item available and item selection are our inputs and item price is our output.

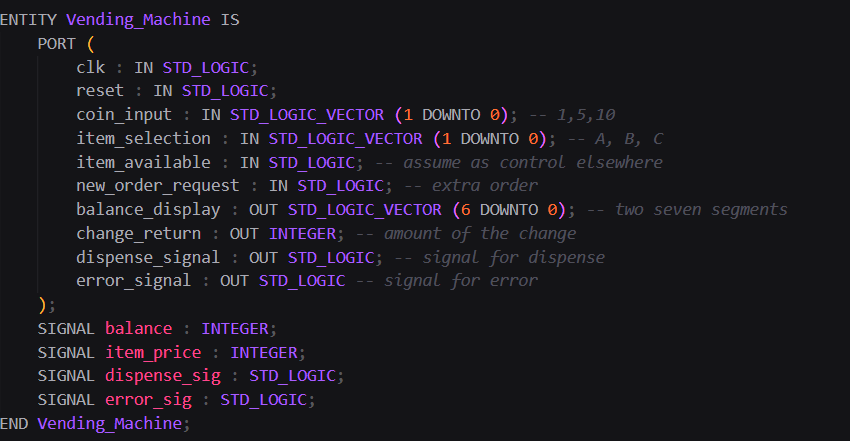


take a look at its architecture :



So simple we just assign item price based the item that is selected and we get it from input and also if it’s available otherwise nothing will happen .

* Vending\_Machine : so now we explain our main part , the Vending Machine here we have used structural style of coding and we have the other tree elements as components here so all inputs and output are clear just take a look at the entity here :



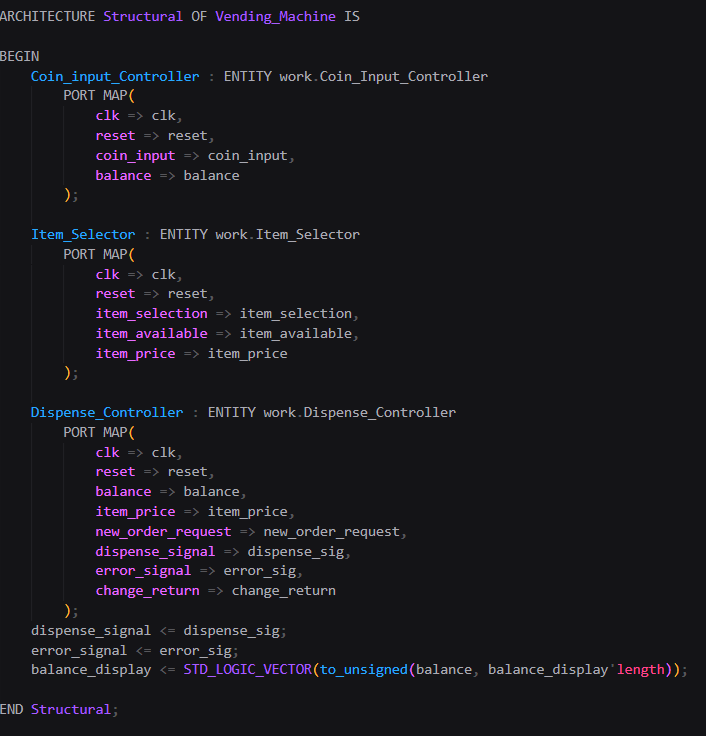
And we have 4 internal signals here to make our system sequential (they are storage elements )

So let’s analyze the architecture :

We got 3 component of the mentioned ones on previous pages and we have connected the inputs and outputs of our machine to them and o won’t bother to explain all the connections one by one because you can see them and they also have same names .

And at the end we assign the internal values to output signals .

That’s it no need to explain anymore.



The end.