Assignment 2 – Vending Machine Controller

Design Description

This main tool for implementing a vending machine controller was a state machine. I used seven states to cover all steps that the vending machine controller (VMC) needed to handle.

Description of states

- NEED_LETTER: the idle state for VMC, this is were the VMC sits waiting for input, the state machine stays here until a letter is pressed
- DUMP_EVERYTHING: intended to be a state were everything is emptied out, this state is entered with a the coin return is pressed on the machine, the need_letter state is always entered after this state is finished.
- RESET_STUFF: only signaled for entry when the reset signal is sent to the VMC, need_letter is always entered
- NEED_NUMBER: the intermediary state for input, the state machine stays here until a number is pressed
- SEL_DONE: state signaling that selection of a product is done, the state machine will stay in this state until enough money for the desired product is entered
- NEED_CHANGE: this state is always entered after the required amount of money is entered, it's immediately left if there's no change, and repeatedly entered until all the needed change has been dispensed
- SEL_OK: the final state, change has been given, and the product is dispensed, the state machine always returns to need_letter from this state

Accumulation of currency

I made a separate process to accumulate currency, it runs in the background and is clocked, so whenever a coin is entered it's registered in the CURRENT_PAYMENT register. The register is only used when the user makes a selection. Because of this design decision there is no state in which the VMC is waiting for money, it can receive money in any of the states. However, it is possible for the VMC to seem likes it's waiting for money in the SEL_DONE state, because it will not change out of this state until it has received sufficient money to dispense the currently selected product.

Knowing about the User's Input

Another concurrent process, not time Dependant, what knowing what the user had pressed at any given moment. The DECODE_SELECTION process handled figuring out whether what the user had pressed was a letter or a number and what price that selection had. I used a huge if/else structure to implement this, the if/else structure figured out what type of input and it's value for later conversion into an index for the price array.

Testing procedures

Considering real life user input

As discussed in class, we can make certain assumptions that the VMC doesn't have to deal with a great deal of concurrency problems, since there's few cases where the thing providing input (the user) can cause a problem. As such tests were structured with that in mind, I tried to simulate every realistic situation that the VMC could encounter.

Difficulties in Testing

Synchronization was the main difficulty in constructing a test bench that would output a set of tests that covered all the possible situations. I didn't want to manually change the test bench for each situation, that would be easier in the short range, but in the long range if I needed to change something in the VMC then I'd need to redo my whole manual testing procedure.