Report: Lab2

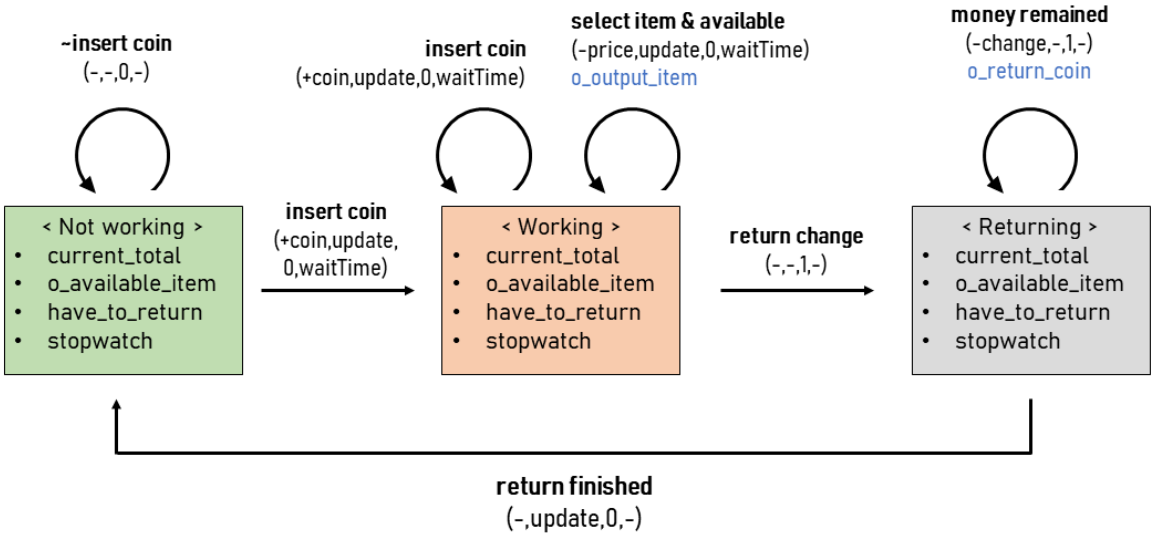
20190533 Hyemin Lee
20190235 Gangtae Park

1. Introduction

Lab 2는 verilog를 이용해서 입력 및 출력이 가능한 자판기(vending machine)을 제작하는 랩이다. 입력의 경우는 얼마의 값을 받았는지, 어떤 item을 선택했는지, 그리고 return을 하는지 이 세 가지를 input으로 받게 된다. 그리고 출력값으로는 돈을 입력 받았을 때 선택 가능한 item들을 표시해야하며 선택 가능한 item을 골랐을 때, 그 item을 출력할 수 있어야 한다. 또한, return을 했을 때 거스름돈을 output으로 출력해야 한다. 자판기의 복잡한 과정들을 이해하고 verilog 코드로 쉽게 구현하기 위해서는 각 state에 따라 state가 바뀌는 조건, input, output을 나타내는 FSM을 그려야 했다. 우리는 먼저 FSM을 그리고, 이를 토대로 코드를 작성하였다.

2. Design

우리는 크게 세 state로 나누었다. 그리고 고려할 변수로는 총액(current_total), available한 아이템의 갱신 여부(o_available_item), return의 여부(have_to_return), 그리고 stopwatch의 초기화 여부(stopwatch)이다. 첫 번째는 “Not Working” state로, 자판기에 돈을 투입하기 전의 상태를 나타낸다. 이 state에서 돈이 투입되지 않는다면 변수들에 변화가 없다. 돈이 투입되면 총액은 돈의 금액만큼 증가, 그 총액만큼 available한 아이템을 갱신하고, stopwatch를 waitTime으로 설정해야 한다. 두 번째는 “Working” state로, “Not Working” state에서 돈이 투입되면 “Working” state로 전환된다. 이 state에서는 waitTime을 넘겨서 시간이 초과되거나 return을 input으로 받게되면 “Returning” state로 전환된다. 이 외에, 돈이 투입되면 “Not Working” state일 때와 같은 작업을 수행하고, available한 아이템을 선택한다면 총액은 그 아이템의 금액만큼 감소하고 그 금액에 따라 available한 아이템을 갱신해야하며 stopwatch를 초기화해야한다. 그리고 선택한 아이템을 output으로 출력(o_output_item)해야한다. “Returning” state에서는 have_to_return 변수가 0이 될 때까지 output으로 거스름돈을 계속해서 출력(o_return_coin)해야 한다. FSM 디자인은 아래 그림과 같다.



3. Implementation

State에 사용되는 변수로는 current_total, o_available_item, have_to_return, stopwatch가 있고 위에서 설명했듯이 3가지의 state case가 존재하기 때문에 그에 맞게 코드를 짰다. 또한, 각각의 state case에서 가능한 input case들, 위 그림에서 화살표에 해당하는 statement를 코드로 구현하였다.

```
//State(current_total, o_available_item, have_to_return, stopwatch)
//case0: Not Working
if (current_total == 0)
	//input case 1: insert coin
	if (i_input_coin)   //(+coin, update, 0, waitTIme)
else
	//case1: Returning
	if (have_to_return) //(-change, -, 1, -)
                        output: o_return_coin
	//case2: Working
	else
		//input case 1: insert coin
		if (i_input_coin)   //(+coin, update, 0, waitTIme)
		//input case 2: select item
		if (i_select_item)  //(-price, update, 0, waitTime)
                            output: o_output_item
		//input case 3: trigger return or stopwatch end
		if (i_tregger_return || stopwatch == 0)//(-, -, 1, -)
```

4. Evaluation

ModelSim을 통해서 skeleton code로 주어진 vending_machine_TB.v의 testbench를 컴파일 및 시뮬레이션하기를 반복하여 vending_machine.v가 정확히 구현되었는지 확인하였다. 최종 vending_machine.v는 vending_machine_TB.v의 모든 testcase(10개)를 통과하였다.

5. Discussion

FSM을 그리는 과정에서 어려움이 있었다. 처음에는 state case를 두 가지, “Not Working” state와 “Working” state 두 가지로만 구현하려고 하였으나 어려움이 있었다. 거스름돈을 반환하는 과정에서 한 번에 모든 output을 출력하는 것이 아닌 단위가 가장 큰 돈 순서대로 하나씩 출력한다는 것을 알게 되었다. 따라서 state case가 한 가지 더 있으면 좋을 것 같다는 생각을 하게 되었고 “Returning” state를 추가하여 문제를 해결하였다.

6. Conclusion

이번 Lab 2는 verilog를 이용해서 vending machine(자판기)를 구현하는 Lab이었다. 이번 Lab을 통해 FSM을 디자인하고 잘 디자인 된 FSM를 토대로 verilog로 구현하는 방법을 알 수 있었다. 주어진 상황에 맞게 state를 선언하고 각 state별로 전환되는 input 조건이 무엇이며 선언된 변수들은 조건에 만족하도록 어떻게 바뀌어야 하는지 생각하면서 verilog를 다루는 능력을 기를 수 있었다.