20170699 최진성 20170080 김동규 Lab Report

Lab 2. Vending Machine Lab

1. Introduction

Finite State Machine (FSM)의 예시인 Vending Machine을 실제로 구현한다. Finite State Machine는 상태 기계라고도 불리며, 특정 시간마다 유한개의 상태 중 하나의 상태를 가지게 된다. 현재 상태와 외부 input에 의해 다음 상태가 정의되며, 상태가 바뀌는 것을 전이 (transition)라고 한다. 이러한 특성을 가지는 Machine들은 일상 속에서도 어렵지 않게 찾아볼 수 있고, 그 대표적인 예시 중 하나가 바로 Vending Machine이다. 이 Vending Machine을 구현하는 것이 이번 과제의 목표이다. FSM에는 output이 오직 현재의 state에 의해 결정되는 Moore Machine과 output이 현재의 state와 input에 의해 결정되는 Mealy Machine이 존재하는데, 그 중 Mealy Machine을 사용하고자 한다[1]. Register Transfer Level (RTL)이란 Synchronous circuit을 하드웨어 레지스터 사이의 데이터 흐름과 논리로 구현하는 레벨을 뜻한다[2]. Verilog와 같은 HDL (Hardware Description Languages)에서 주로 구현 가능하다. 해당 과제를 수행하기 위해 FSM, RTL에 대한 기본적인 이해와 더불어, clk마다 state가 업데이트되는 synchronous circuit의 combination Logic, sequential logic들을 구현하기 위한 이해 및 사전 설계가 필요하다. 이번 과제를 통해 FSM, RTL, verilog 코딩에 대한 이해도를 향상한다.

2. Design

Lab 2에서 구현할 Vending Machine은 3개의 input (동전 입력, 아이템 선택, 반환)을 입력 받고, 3개의 output (아이템 반환, 가능 아이템, 동전 반환)을 지속적으로 반복한다.

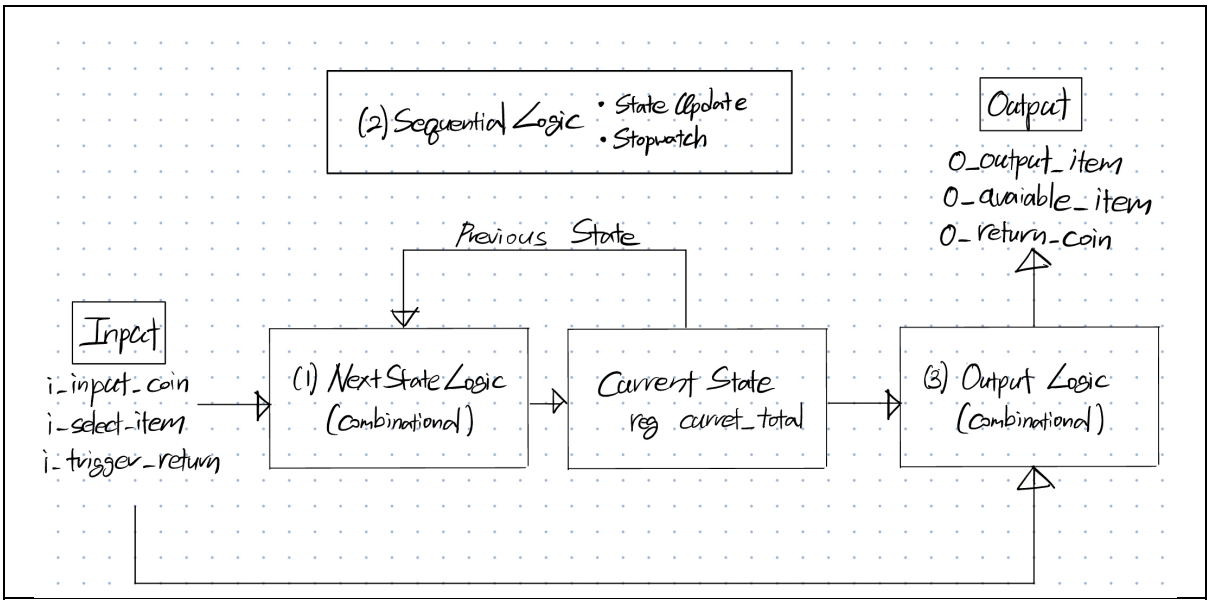

Figure 1. Lab 2에서 구현하는 Vending Machine의 Diagram (Mealy Machine)

[1] https://en.wikipedia.org/wiki/Finite-state_machine
[2] http://www.ktword.co.kr/test/view/view.php?m_temp1=4894

이를 주어진 모든 시간에 대하여 언제나 하나의 state로 존재하는 Finite State Machine 형태이다. 우리는 Output이 Input과의 Combinational Logic에 의해 결정되도록 하는 Mealy Machine 형태로 구현하였고 Vending Machine 구현을 위한 State 모식도는 Figure 1과 같다. (1) Input과 현재 State를 기반으로 다음 state를 계산하는 파트, (2) 현재 state를 업데이트하는 Sequential Logic, (3) 현재 State와 Input으로 Output을 계산하는 파트로 크게 나누어서 구현된다.

3. Implementation

Figure 1의 Mealy Machine 구현을 위해 (1) **다음 state를 계산하는 Combinational Logic**, (2) **현재 state을 업데이트하는 Sequential Logic**, (3) **Output을 계산하는 Combinational Logic**를 각각 구현하였다. State에는 현재 vending machine에서 유저가 사용할 수 있는 돈의 액수 정보 (current total)를 가지고 있도록 구현하였다.

(1) 다음 State 계산 논리 구현

유저가 Return 버튼을 눌렀을 경우, 유저가 item을 골랐을 경우, 유저가 coin을 투입했을 경우, 세 가지로 나누어서 구현하였다.

- 유저가 return 버튼을 눌렀을 경우: 현재 액수에서 반환 가능한 가장 큰 지폐의 액수를 뺀 나머지 액수를 다음 state로 넘긴다. 남은 액수가 0이 아닐 경우, 계속 return을 해줘야 함으로, 현재 return 중이라는 정보를 유지한다.
- 유저가 coin을 투입했을 경우: 해당 coin만큼의 액수를 현재까지의 액수에 더한 값을 다음 state로 넘기고, stopwatch를 초기화해준다.
- 유저가 item을 골랐을 경우: 해당 아이템이 현재까지의 액수보다 작은지 확인한다. 만약 작다면 현재 액수에서 해당 아이템만큼의 액수를 빼서 다음 state로 넘기고 stopwatch를 초기화해준다. 만약 아이템 가격이 현재 액수보다 크다면 현재 state 그대로 넘긴다.

(2) Output을 계산하는 Combinational Logic

- 구입 가능한 아이템 (o_available_item): 현재 state에서의 투입한 액수정보와 각 아이템의 가격을 비교하여 현재 액수보다 작은 아이템들을 출력한다.
- 아이템 구입 (o_output_item): *유저의 아이템 입력신호를 받아서*, 해당 아이템이 구입 가능하면 출력한다.
- 잔돈 반환 (o_output_coin): *유저가 return 버튼을 눌렀을 경우*, 현재 액수에서 반환 가능한 가장 큰 액수를 출력한다. (해당 파트는 구현상 편의를 위하여 (1) 다음 State 계산 논리 구현 파트과 함께 구현하였음)

(3) **현재 state을 업데이트하는 Sequential Logic**

모든 값을 초기화하는 reset, state 업데이트, Stopwatch 설정을 구현하였다.

- Reset: 설정한 모든 변수값을 초기화한다.
- State 업데이트: (1)에서 계산한 다음 state값을 현재 state값으로 설정해준다.
- Stopwatch 설정: Stopwatch가 0이 되면 (1)에서 return 버튼이 눌린 것과 동일하게 되도록 설정해준다. 이외의 경우 Stopwatch 값을 하나씩 줄여준다.

4. Evaluation

완성된 코드를 컴파일한 이후 주어진 vending_meachine_TB.v testbench 파일을 사용하여 시뮬레이션 결과를 확인하였다. 그 결과 모든 10 개 테스트에 대해서 다음과 같이 10 개의 정상적인 결과값이 나온다는 사실을 확인하였다.

```
# TEST                 Select4thItemTest :
# PASSED
# TEST                    WaitReturnTest :
# PASSED
# TEST                 TriggerReturnTest :
# PASSED
# Passed = 10, Failed = 0
# 1
# Break in Module vending_machine_tb at C:/modelsim_project/lab2/vending_meachine_TB.v line 118
```

Figure 2. 주어진 Testbench 파일로 실행한 시뮬레이션 결과

결과값 뿐만 아니라, 잔돈 반환 등의 Process 에서 중간값들이 정확하게 이루어지는지 확인하기 위하여 wave 를 확인하였고 모든 경우에 대해서 정상적인 결과가 나타난다는 것을 확인하였다.

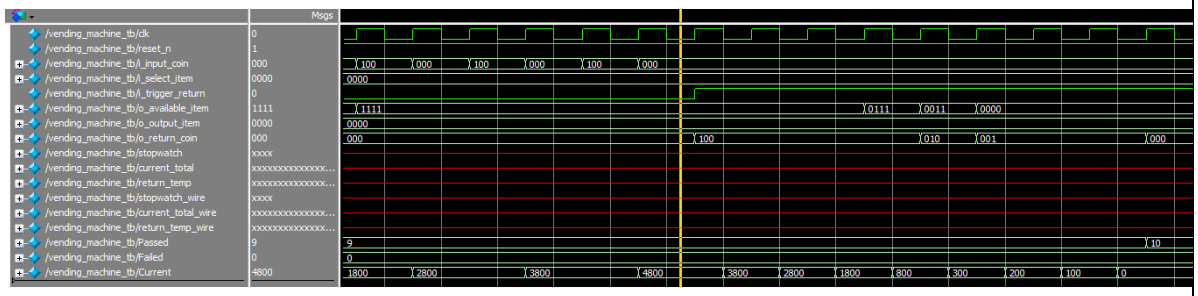

Figure 3.시뮬레이션 결과로 나온 Wave 결과

5. Discussion

이미 지정되있는 수많은 변수를 파악하고, 구현하고자 하는 Machine의 각 파트를 해당 변수들로 어떻게 구현할 수 있을지 고민하는데 많은 시간을 소비하였다. 또한 verilog의 경우 작성한 코드를 디버깅하는 것이 익숙하지 않아 많은 시간을 할애하였다. 그러나 Verilog를 사용하는 두번째 랩인만큼 첫번째 랩보다 문법과 코딩, ModelSim 시뮬레이션 등을 비교적 수월하게 진행할 수 있었다. 첫 번째 랩 이전에 참고자료로 주어졌었던 Verilog 설명 자료, 인터넷을 많이 참고하였고, Classum을 통한 조교님들과의 원활한 소통과 다른 팀들의 질문/답변이 굉장히 큰 도움이 되었다.

6. Conclusion

Vending Machine을 Verilog를 활용한 Finite State Machine (FSM)형태로 구현하면서 이전 디지털 시스템에서 배웠던 FSM과 함께, 더 많은 Verilog language의 특성과 문법을 이해할 수 있었다. 여러 State에 따른 여러 조건문을 고려해야 하는 과제였던 만큼 철저한 사전 설계의 중요성을 다시 한번 느끼게 되었다. 무엇보다 Lab1에는 없었던 clock이 포함된 synchronous digital circuit을 구현하면서 클럭 신호에 따라 동작하는 회로의 verilog 구현 형태를 이해하였고 RTL의 기본 내용과 전반적인 컨셉에 대해서 알게 되었다. Lab1를 구현할 때보다 길고 복잡한 구현을 하면서 추후 과제에서 활용하게 될 Verilog 코딩에 한층 더 익숙해졌다.