**CSED311 Lab 1 Report**

Team number 3: 20210643 김현준, 20210273 하태혁

**1. Vending machine**

1. Introduction

Vending machine RTL을 Vivado 환경에서 Verilog로 구현하여, 간단한 알고리즘을 통해 자판기의 기능을 하도록 설계하는 것이 Lab1의 내용이다. 이 machine의 input으로는 동전이 insert되었다는 신호를 주는 i\_input\_coin, 아이템 선택에 관한 i\_select\_item과 리턴 신호인 i\_return\_trigger, 클락 신호인 clk와 reset\_n이 주어진다. 이 machine의 output으로는 아이템 출력의 o\_output\_item, 살 수 있는 아이템을 나타내는 o\_available\_item, 그리고 반환되는 잔돈을 나타내는 o\_return\_coin이 주어진다. 따라서 이러한 input, ouput을 기준으로 주어진 조건 하에 FSM을 구현하면 된다. 아래는 이 machine의 알고리즘이다.

1. Coin을 넣으면 100원, 500원, 1000원의 type에 따라서 i\_input\_coin의 값이 설정되며, waiting time이 100으로 초기화된다.
2. 본 Vending machine은 현재 보유한 금액 보다 가격이 낮은 모든 item, 즉 available item을 보여주며 waiting time을 1씩 감소시킨다.

3-a. Waiting time(2번) 도중에 coin을 insert하면 다시 1의 상태로 돌아가게 된다.

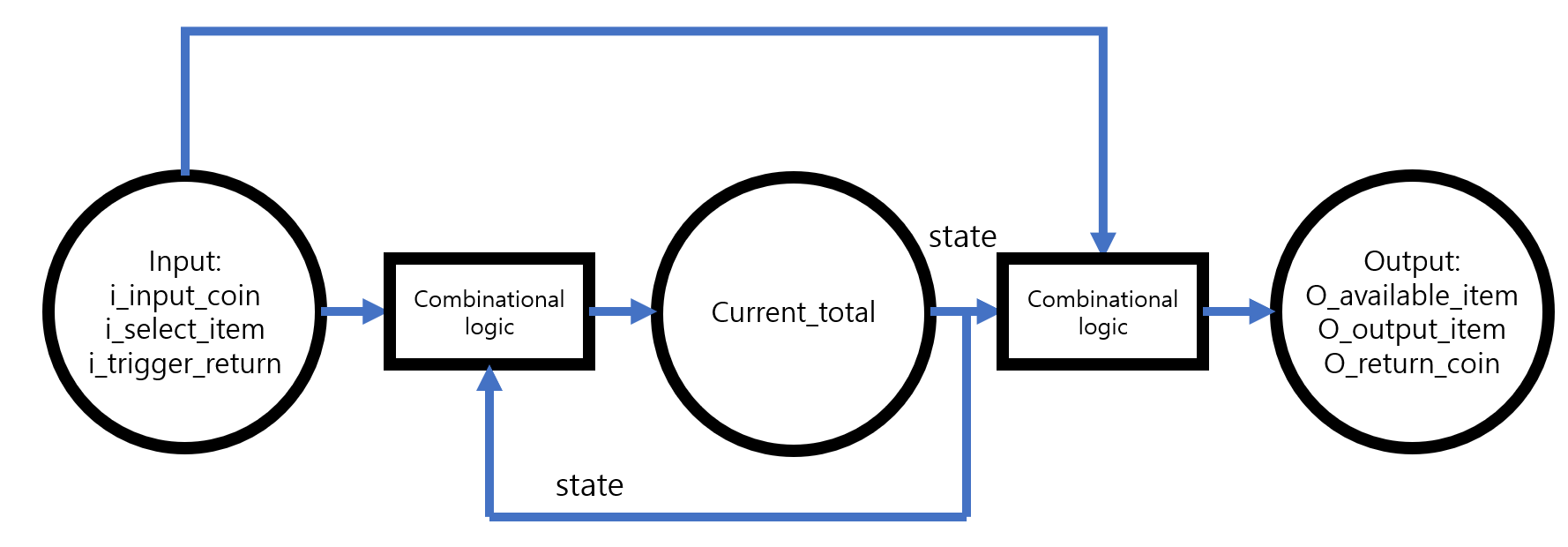
3-b-1. Waiting time(2번) 도중에 item을 선택하면, item이 available한 경우 item이 나오고 2로 돌아가며 waiting time이 100으로 초기화된다.

3-b-2. Waiting time(2번) 도중에 item을 선택하면, item이 available하지 않은 경우 아무 것도 변하지 않는다.

3-c. Waiting time(2번) 도중에 아무 것도 input이 없으면 잔돈을 반환한다.

* Return button이 눌릴 경우 잔돈이 반환되고 1의 상태로 간다.

1. Design



Moure machine과 Mealy machine 중 후자를 선택하였다. 현재 상태(current\_total)와 input(i\_input\_coin, i\_select\_item, i\_trigger\_return)을 이용하여 다음 상태를 계산할 수 있어야 하기 때문에 Mealy machine을 선택하였다. Design은 위의 그림과 같다.

Vending\_machine module을 구성하는 Submodule들은 아래와 같이 design하였다.

1. **Check\_time\_and\_coin**

자판기에서 동전을 반환하기 위해 동전 반환 버튼이 눌렸는지, 또는 동전 또는 음료수 선택 버튼이 눌렸는지를 감지하여, 반환할 동전의 종류와 동전 반환까지 남은 시간을 계산하는 모듈이다. Input으로는 clk, reset\_n, i\_input\_coin, i\_select\_item, current\_total, item\_price, coin\_value가 들어오며, output reg o\_return\_coin과 wait\_time을 계산한다.

module 내부에서는 동전 반환 버튼(i\_trigger\_return)이 눌렸거나 반환 waiting time이 0이 되었을 때, 현재 동전 총액(current\_total)에 따라 반환할 동전의 종류를 결정하고, 반환 waiting time이 갱신된다. Posedge로 clock 신호에 따라, 리셋 신호가 들어오지 않으면 반환 대기 시간이 갱신되는 식으로 waiting time을 100에서 1씩 줄여 나간다.

1. **Calculate\_current\_state**

본 calculate\_current\_state 모듈은 자판기의 현재 상태를 계산하는 모듈으로, input으로는 i\_trigger\_return, i\_input\_coin, o\_return\_coin, i\_select\_item 등 현재 state 계산을 위한 값들이 들어오고, output으로는 o\_available\_item, o\_output\_item으로 item을 살 수 있는 상태인지, item이 나왔는지를 알려주고, input\_total, output\_total, return\_total, current\_total\_nxt 등 현재 금액 상태에 대한 정보들을 출력한다.

현재 상태를 계산하는 블록에서는 입력된 코인과 선택한 상품을 바탕으로 현재 돈의 합을 계산하고, 반환해야 할 돈을 계산한다. 그리고, 선택한 상품이 현재 돈의 합보다 적은 가격을 가진 경우, 아이템을 출력할 수 있도록 o\_available\_item과 o\_output\_item을 설정하는 식으로 본 모듈은 구현된다.

1. **Change\_state**

본 change\_state 모듈은 신호 clk와 reset\_n을 입력으로 받고, 현재 자판기의 현재 금액을 나타내는 current\_total\_nxt를 입력받아, 현재 자판기의 금액 상태를 저장하는 current\_total 출력을 제공한다. reset\_n이 0이면 모든 상태를 초기화하고, 그렇지 않으면 모든 상태를 업데이트 한다. 이때, always 블록에서 posedge clk 신호를 감지하여 업데이트가 이루어지는 식으로 본 모듈은 구현되었다..

1. **Implementation**
2. **Check\_time\_and\_coin**

본 모듈은 time을 관리하고 현재 coin에 따라 반환 시 얼만큼의 돈을 어떤 type의 동전으로 반환할 수 있을지 계산하는 역할을 하도록 구현하였다.

먼저, initial 부분에서는 waiting time을 0으로 초기화한다. 다음으로, coin return time을 업데이트 하는 부분에는 i\_input\_coin이 신호 또는 i\_select\_item 신호가 들어왔을 때 시간을 100으로 설정해주어야 하므로, ‘kWaitTime으로 wait\_time을 설정해주었다.

다음으로, o\_return\_coin을 계산해주는 block을 구현하였는데, 먼저, 맨 앞에서 o\_return\_coin을 초기화해주고, current\_total 값에 따라 1000원, 500원, 100원 순으로 반환하여야 하므로, current\_total/1000>0, (current\_total%1000)/500 >0, (current\_total%500)/100>0의 세 조건에 따라 o\_return\_coin의 index 2, 1, 0에 각각 1을 설정해주었다.

마지막으로, clock에 따라 시간이 지나고, 그에 따라 wait\_time이 줄어드는 부분을 구현하였다. !reset\_n의 조건만 따로 빼어 o\_return\_coin과 wait\_time을 0으로 초기화하였고, wait\_time>0인 경우에는 1씩 감소시키도록 구현하였다.

1. **Calculate\_current\_state**

본 모듈은 현재 상태를 계산하는 모듈로, current\_total\_nxt를 계산하는 것이 모듈의 주 목적이다. 다음 state를 계산하는 부분과 ouput을 계산하는 combinational logic으로 구성되어 있다.

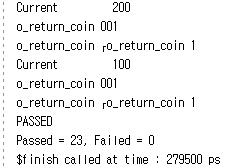
먼저, current\_total\_nxt를 계산하기 위해서 return\_total, input\_total, output\_total을 계산하여야 한다. 각각을 static for loop에서 계산하는데,먼저 return\_total 값은 o\_returncoin[i] 신호가 들어온 경우 해당 index의 coin\_value 값을 더해주어 구현하였다. Input\_total의 경우는 i\_input\_coin 신호의 index을 구하여 해당 index의 coin\_value를 더해주어 계산하였다. 마지막으로 output\_total 값은 i\_select\_item[i]와 (item\_price[i] <= current\_total) 조건을 걸어서 해당 index의 item\_price[i]를 더해주어 계산하였다. 이는 해당 item을 골랐고, 그 item의 가격이 현재 보유액보다 작은 경우 item을 반환할 수 있기 때문이다. 최종적으로 구한 값들을 이용해 current\_total\_nxt는 current\_total에서 input\_total을 더하고 output\_total을 빼준 return\_total을 빼주어 계산하였다.

다음으로 ouput 관련을 다루는 모델에서는 먼저 o\_available\_item과 o\_output\_item을 초기화해준 후 for loop에서 item의 가격이 보유 금액보다 적을 시 available에 1을 넣어 주었고, item이 available하고 해당 item을 골랐을 때, o\_output\_item에 1을 넣어 주는 식으로 구현을 완성하였따.

1. **Change\_state**

본 모듈의 경우 간단하게 구현할 수 있었다. 먼저, reset\_n이 0일 경우에 current\_total을 0으로 초기화해 주었고, 그렇지 않을 경우에는 current\_total에 current\_total\_nxt를 넣어 줌으로써 다음 state로 상태가 넘어가도록 구현하였다.

1. **Discussion**



주어진 모든 테스트를 통과할 수 있었다. o\_return\_coin을 설정하는 부분에서 blocking assignment operator를 사용하였을 때, 마지막 테스트를 통과하지 못하여서 always 구분에서 non-blocking assignment operator을 사용하여 동기적으로 처리될 수 있도록 하였다.

check\_time\_and\_coin module과 calculate\_current\_state module을 실행시킬 때는 signal로 i\_trigger\_return가 input port로 들어갈 수 있도록하였고 해당 시그널에 따라 return\_total이나 o\_return\_coin을 재측정할 수 있도록 하였다.

1. **conclusion**

ALU module을 설계하면서 Verilog 기본 문법과 module에서 signal, input port, output port가 연결되는 과정을 이해할 수 있었다. vending machine을 설계하면서는 전체적인 machine을 디자인하는 과정으로 복잡한 로직을 구현할 수 있었으며, blocking assignment operator, non-blocking assignment operator을 사용하는 기본 원리를 익혀볼 수 있었다.