Lab03 report

Vending machine

2019-12172 이재원

1. Introduction

Vending machine을 만드는 것이 목표다. Cpu 전 마지막으로 fsm 연습하는 것이 목적이다.

2. Design

Fsm의 state는 총 coin의 양(current_total)로 할 것이다. 그리고 current_total의 next state인 next_total를 define할 것이다. Sequential logic 부분에서 current_total <= next_total 만 넣고, combinational logic for next state에서 next state인 next_total을 계산할 계획이다. Mealy machine으로 만들어 input과 current_total에 의하여 output을 계산할 예정이다.

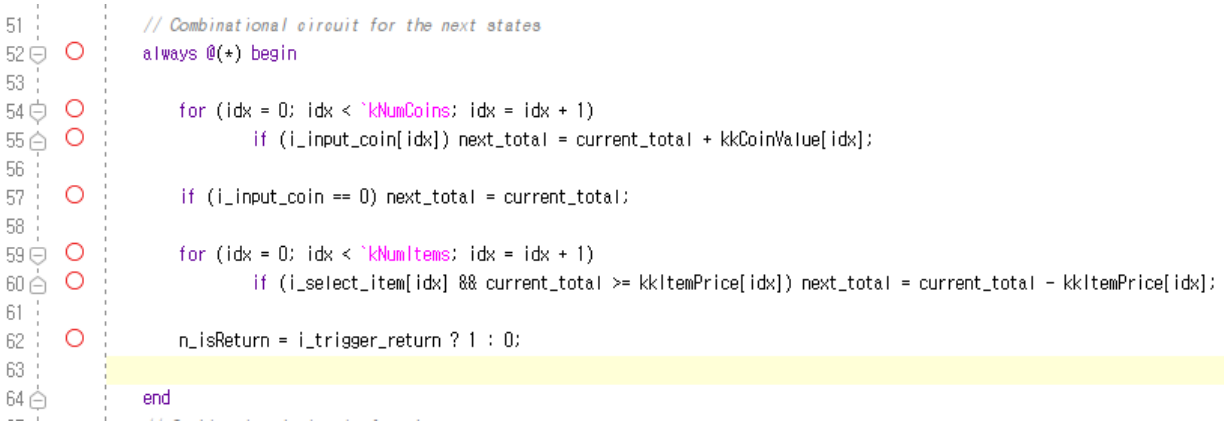
3. Implementation

```
42          // Internal states. You may add your own reg variables.
43          reg [`kTotalBits-1:0] current_total;
44          reg [`kNumItems-1:0] num_items ; //use if needed
45          reg isReturn;
46          reg n_isReturn;
47          reg [`kTotalBits-1:0] next_total;
48
49          integer idx;
```

Figure 1. state declaration of code

Design 단계에서와 다르게 state가 추가되었다. Input에 i_return이 들어온 상태라는 isReturn이라는 state가 추가되었다. Current_total, next_total 처럼 next state는 n_isReturn 이라는 state를 정의했다.

또한, i_select_item을 정의하는 register를 declare했다. Current_total에 따라 buy할 수 있는 물품만 정의한다. Ouput으로 그대로 출력하기 때문에 state라고 볼 수 있다.



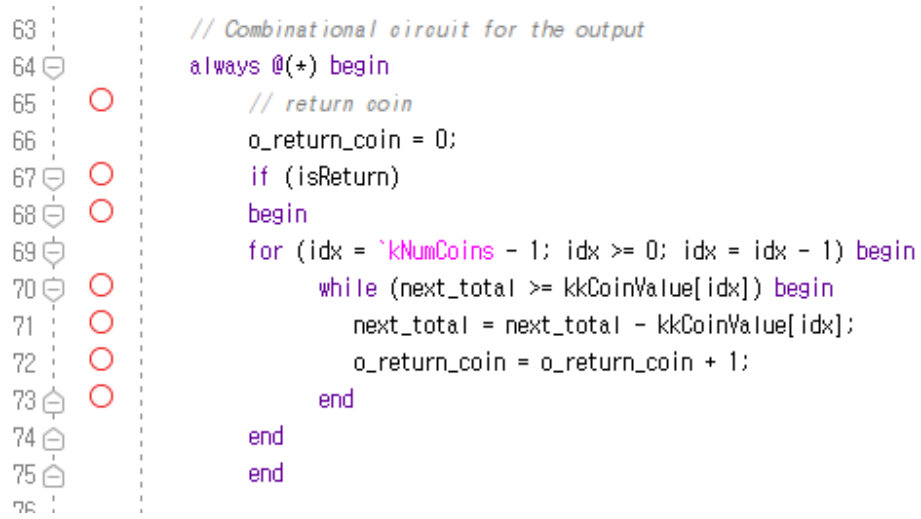
```
51    // Combinational circuit for the next states
52    always @(*) begin
53
54        for (idx = 0; idx < `kNumCoins; idx = idx + 1)
55            if (i_input_coin[idx]) next_total = current_total + kkCoinValue[idx];
56
57        if (i_input_coin == 0) next_total = current_total;
58
59        for (idx = 0; idx < `kNumItems; idx = idx + 1)
60            if (i_select_item[idx] && current_total >= kkItemPrice[idx]) next_total = current_total - kkItemPrice[idx];
61
62        n_isReturn = i_trigger_return ? 1 : 0;
63
64    end
```

Figure 2. Combinational circuit for the next states

next_total은 current_total에서 coin이 들어오면 그 value 만큼 더한다. Coin이 들어오지 않으면 (i_input_coin == 0) next_total은 current_total 그대로 출력하면 된다.

Select_item이 들어와 current_coin보다 작거나 같으면 sold가 되고 next_total은 그 가격 만큼 뺀다.

n_isReturn은 i_trigger_return이 1이면 1이다.



```
63    // Combinational circuit for the output
64    always @(*) begin
65        // return coin
66        o_return_coin = 0;
67        if (isReturn)
68        begin
69        for (idx = `kNumCoins - 1; idx >= 0; idx = idx - 1) begin
70            while (next_total >= kkCoinValue[idx]) begin
71                next_total = next_total - kkCoinValue[idx];
72                o_return_coin = o_return_coin + 1;
73            end
74        end
75        end
```

Figure 3. Combinational circuit for the output o_return_coin

현재 state의 isReturn이 1인 경우 o_return_coin을 계산하면 된다. 그 외의 경우에는 0이다. isReturn은 coinValue가 가장 큰 1000원부터 빼서 계산한다. next_total이 0보다 작아지지 않는 선에서 1000원씩 계속 빼면서 return coin을 1 증가시키면 된다. 이 부분은 while statement로 구현했다. 이 똑 같은 것을 500원, 100원에 대하여 반복한다. 이는 for statement에 해당한다.

이 부분의 경우 ouput을 계산하는 동시에 next state인 next_total을 바꾼다.

Combinational circuit for the next states에 해당한다고 볼 수 있다.

```
79          // o_available_item:
80          for (idx = 0; idx < `kNumItems; idx = idx + 1)
81             if (current_total >= kItemPrice[idx]) o_available_item[idx] = 1;
82             else o_available_item[idx] = 0;
83          if (isReturn) o_available_item = 0;
84
85          // o_output_item:
86          o_output_item = num_items;
87
88
89
90          // current_total
91          o_current_total = isReturn ? 0 : current_total;
92
```

Figure 4. Combinational circuit for the output o_available_item, o_output_item, current_total

o_available_item은 item의 price가 current_total보다 작으면 available하다고 판단하여 1이 나온다. State가 isReturn이면 current_total이 0이기 때문에 항상 0을 출력한다.

o_output_item은 num_items register에 저장되어 있는 값을 그대로 출력한다. 계산은 num_item update하는 sequential logic 부분에 있다.

o_current_total은 current_total을 그대로 출력하나 cycle이 맞지 않는 문제 때문에 isReturn일 때 항상 0을 출력한다.

```
97      // Sequential circuit to reset or update the states
98      always @(posedge clk) begin
99          if (!reset_n) begin
100             // TODO: reset all states.
101             //reset current_total
102             current_total <= 0;
103             next_total <= 0;
104
105             //reset num_items
106             num_items <= 0;
107
108             //reset isReturn
109             isReturn <= 0;
110             n_isReturn <= 0;
111         end
```

Figure 5. Sequential circuit for reset

전부 0으로 reset한다.

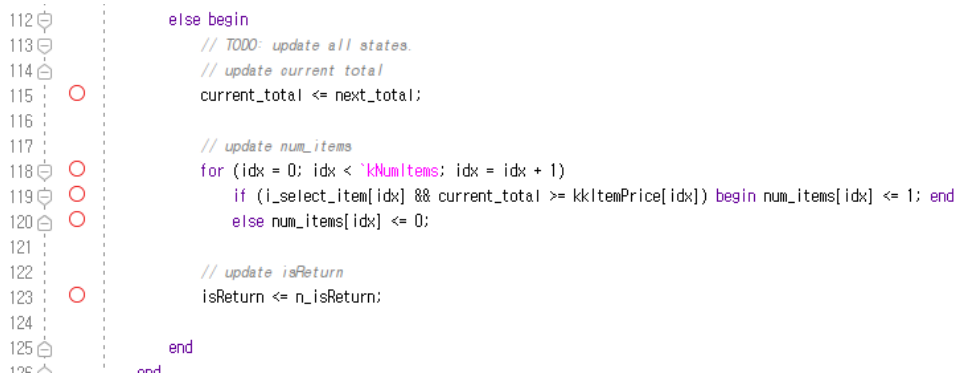


Figure 6. Sequential circuit to update the states

current_total, isReturn state은 next state로 update한다.

num_item은 current_total이 item price보다 큰 경우 buy 할 수 있기 때문에 그 경우에만 저장한다.

4. Discussion

원래 계획했던 것과 달리 i_trigger_return이 들어올 경우 cycle이 맞지 않는 문제가 있었다. 따라서 이 문제를 isReturn 이라는 state를 추가했고, output을 isReturn에 의하여 결정되게 code를 고침으로써 해결했다.

또한, o_return_coin을 계산할 때 next_state인 next_total을 같이 바꾼다는 문제점이 있었다. 원래 의도에서는 next_state를 계산하는 부분과 output을 계산하는 부분을 분리해야 하는 것이 목적이었으나 이를 실행하지 못했다. 하지만 같은 기능을 하는 이 회로를 next_state를 계산하는 부분과 output을 계산하는 부분에 만들면 서로 분리했다고 볼 수 있어서 기능적으로 문제가 없다는 결론을 내렸다.

5. Conclusion

Vending machine을 성공적으로 구현했으며, 구현 중 생긴 여러가지 문제를 해결할 수 있었다.