**CSED-311 LAB1 REPORT**

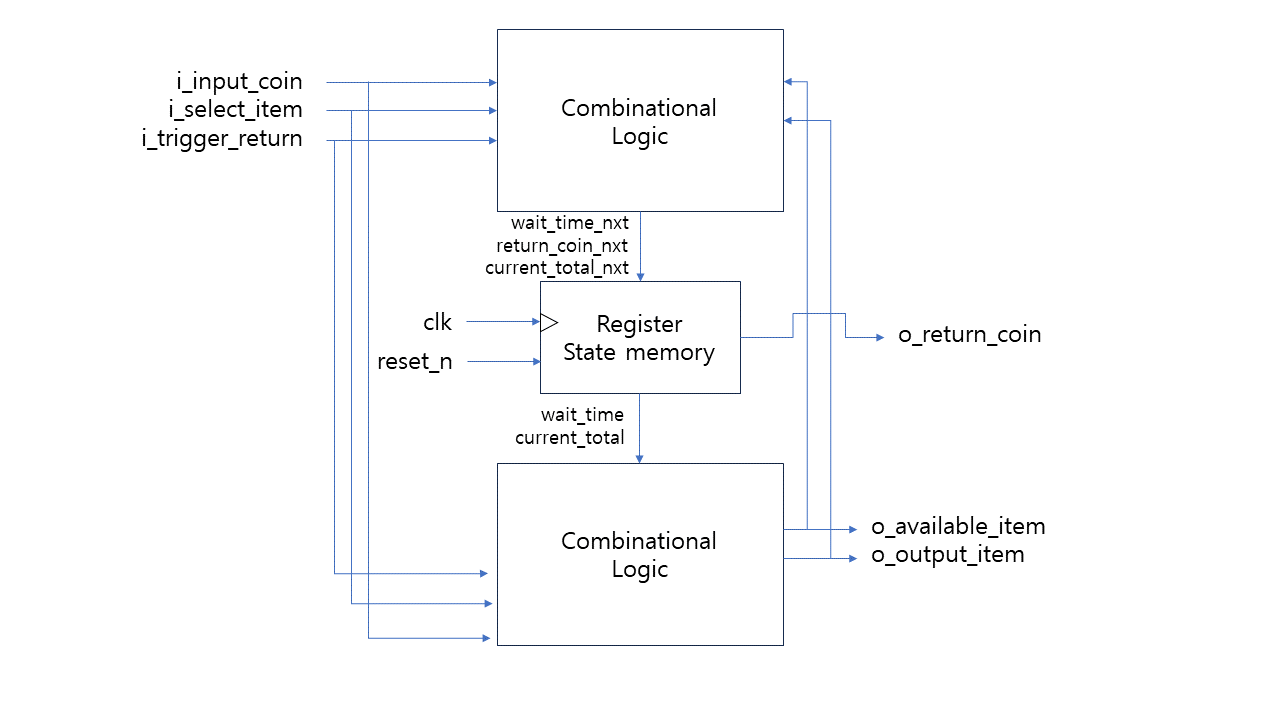
김모세

표승현

**Introduction**

1. Vending machine
   1. Verilog를 이용해 간단한 vending machine을 구현한다. 구현하고자 하는 기능은 아래와 같다.
      1. wait time을 10으로 초기화하고 clk에 맞춰 1씩 감소 시킨다. 동전을 투입하거나 물건을 구입할 때마다 10으로 초기화 시킨다.
      2. vending machine에 들어가 있는 돈의 총량을 계산하여 구매 가능한 물품 리스트를 보여준다.
      3. 구입하고자 하는 물건을 선택하면 해당 물품을 출력하고 물건의 가격만큼의 돈을 current total에서 뺀다.
      4. wait time이 0이 되거나 return trigger를 활성화 하면 남아 있는 돈을 동전으로 환산하여 반환한다.
      5. reset\_n을 활성화하면 모든 상태를 초기화한다.

**Design**

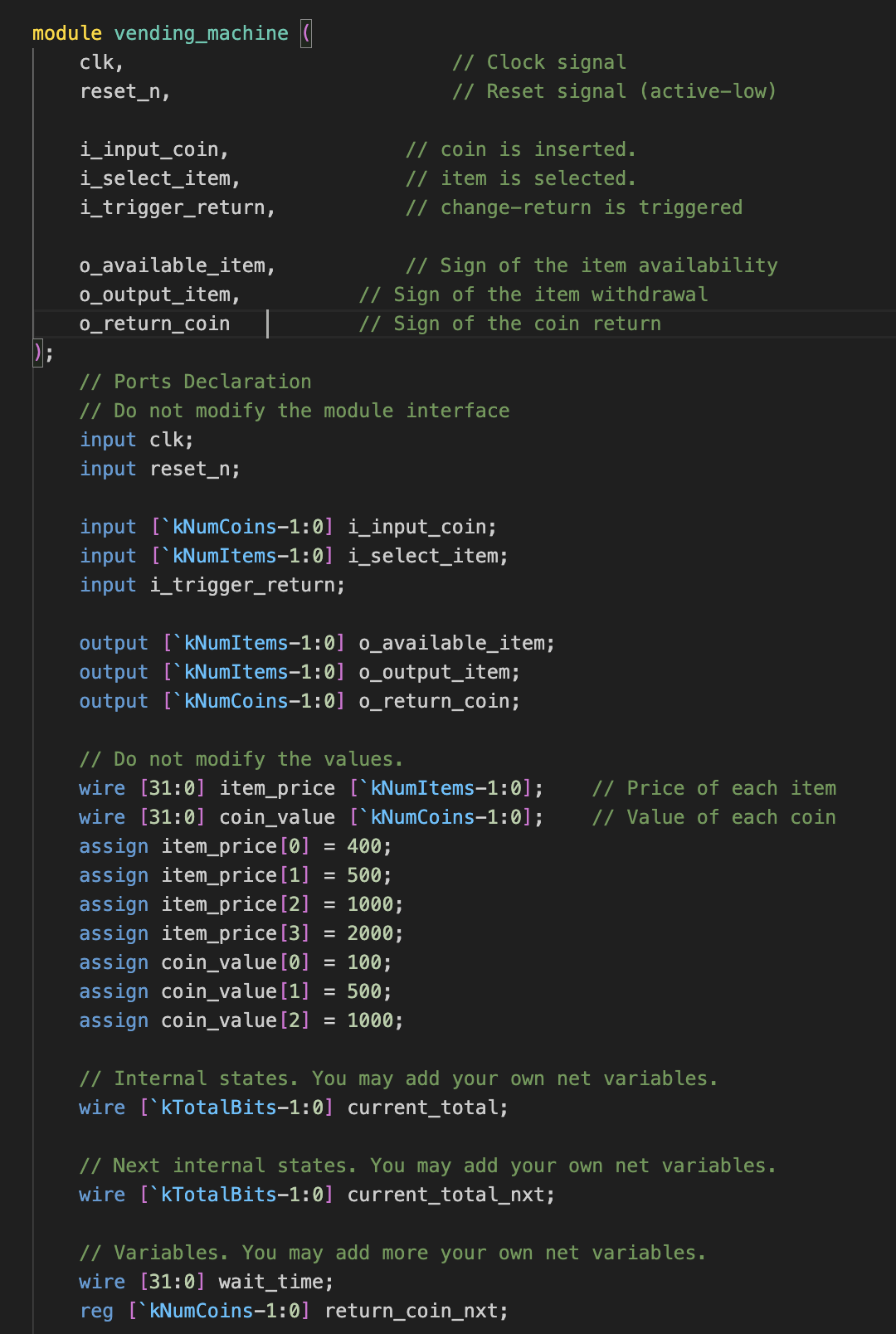
****

<flow of vending machine>

이번 실습에서 제작한 vending machine의 흐름도는 위와 같다. input으로 i\_input\_coin, i\_select\_item, i\_trigger\_return 그리고 clk, reset\_n 신호를 받는다. 입력값을 바탕으로 wait\_time\_nxt, return\_coin\_nxt, current\_total\_nxt를 combinational logic으로 계산하고, clk 신호에 따라 state에 반영한다. 마지막으로 wait\_time과 current\_total 값을 통해 o\_available\_item, o\_output\_item을 도출한다.

output이 input값과 current state의 영향을 받으므로 **Mealy machine**이다.

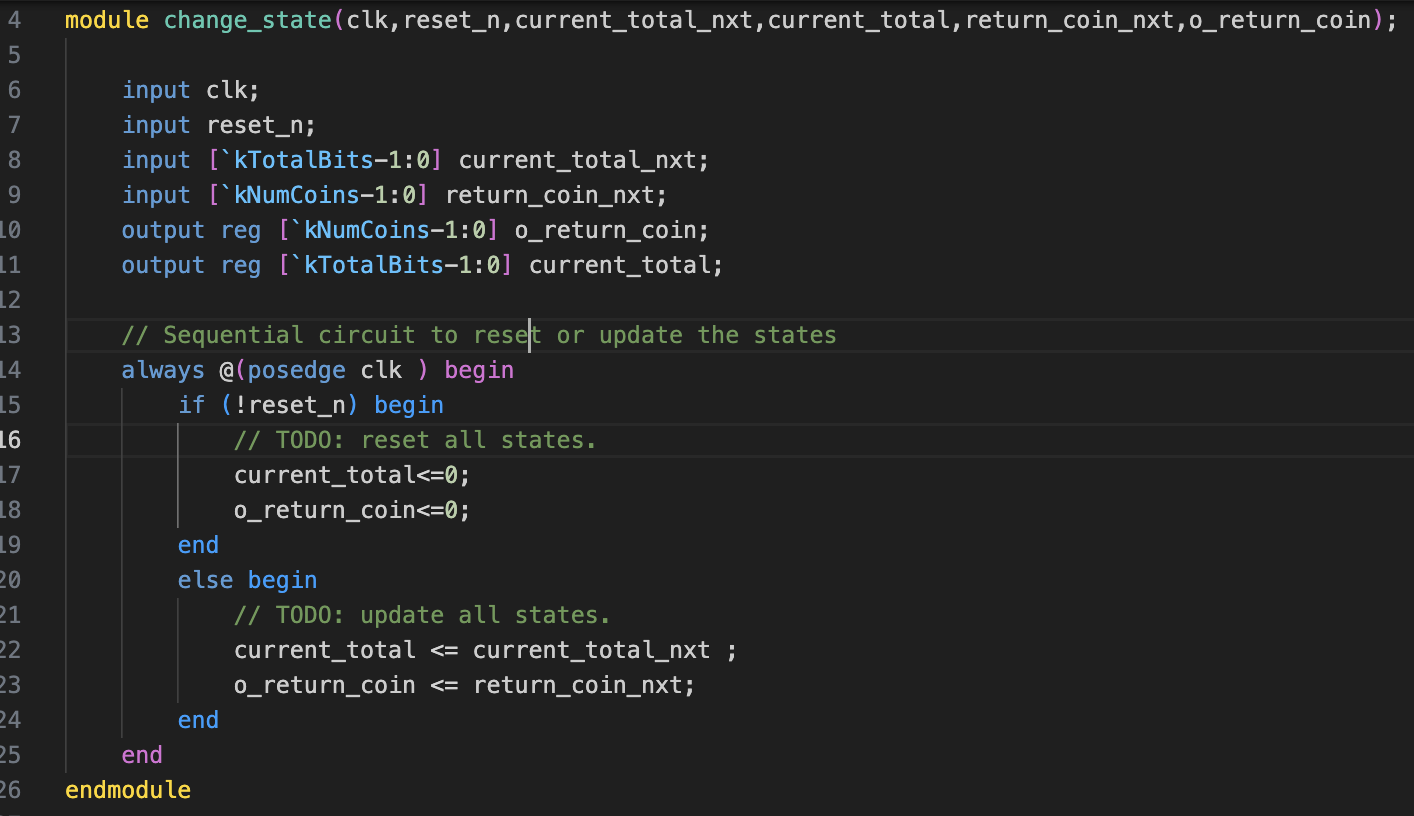
**Implementation**



벤딩머신 최상위 모듈의 포트선언과 내부 변수는 위와 같다. 5개의 input 신호를 받아 3개의 output신호를 내보낸다. current\_total은 현재 투입된 총 금액이며, current\_total\_nxt는 다음 순간 투입된 총 금액이다. o\_return\_coin은 어떠한 코인을 리턴해야하는지 나타내는 ouput reg이며 return\_coin\_nxt는 다음순간에 리턴해야하는 코인을 나타내는 reg이다.

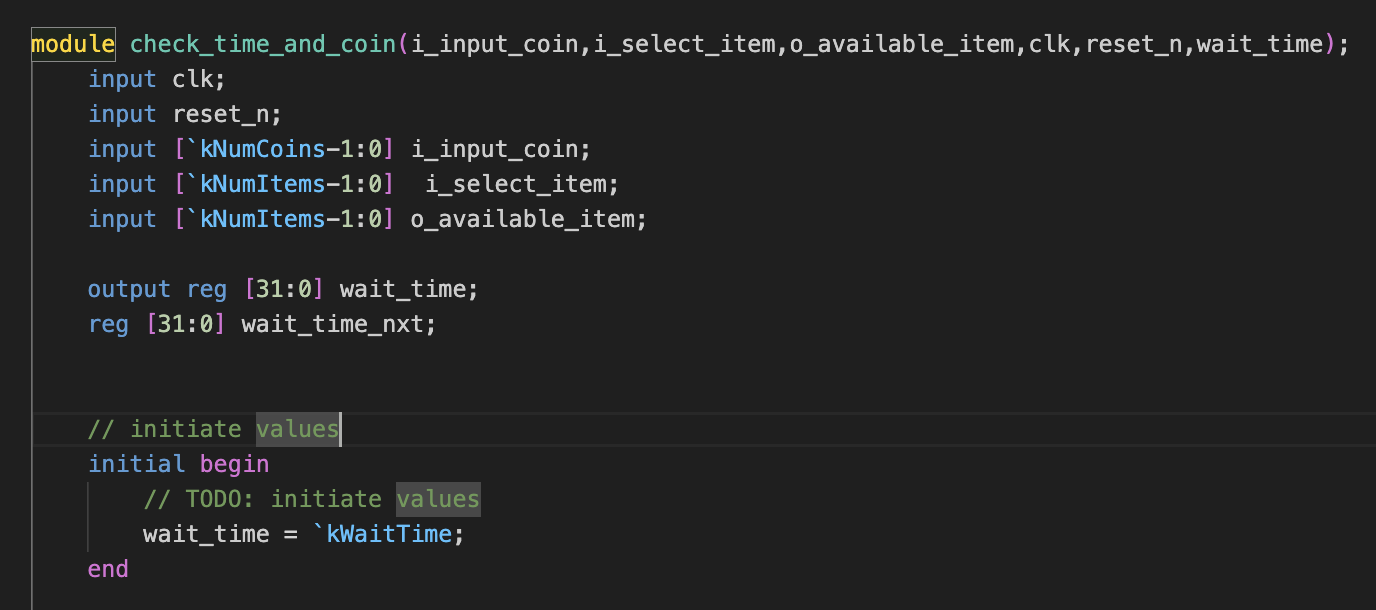
1. change\_state 모듈

change\_state\_module에서 매 posedge clk마다 o\_return\_coin과 current\_total을 업데이트한다.

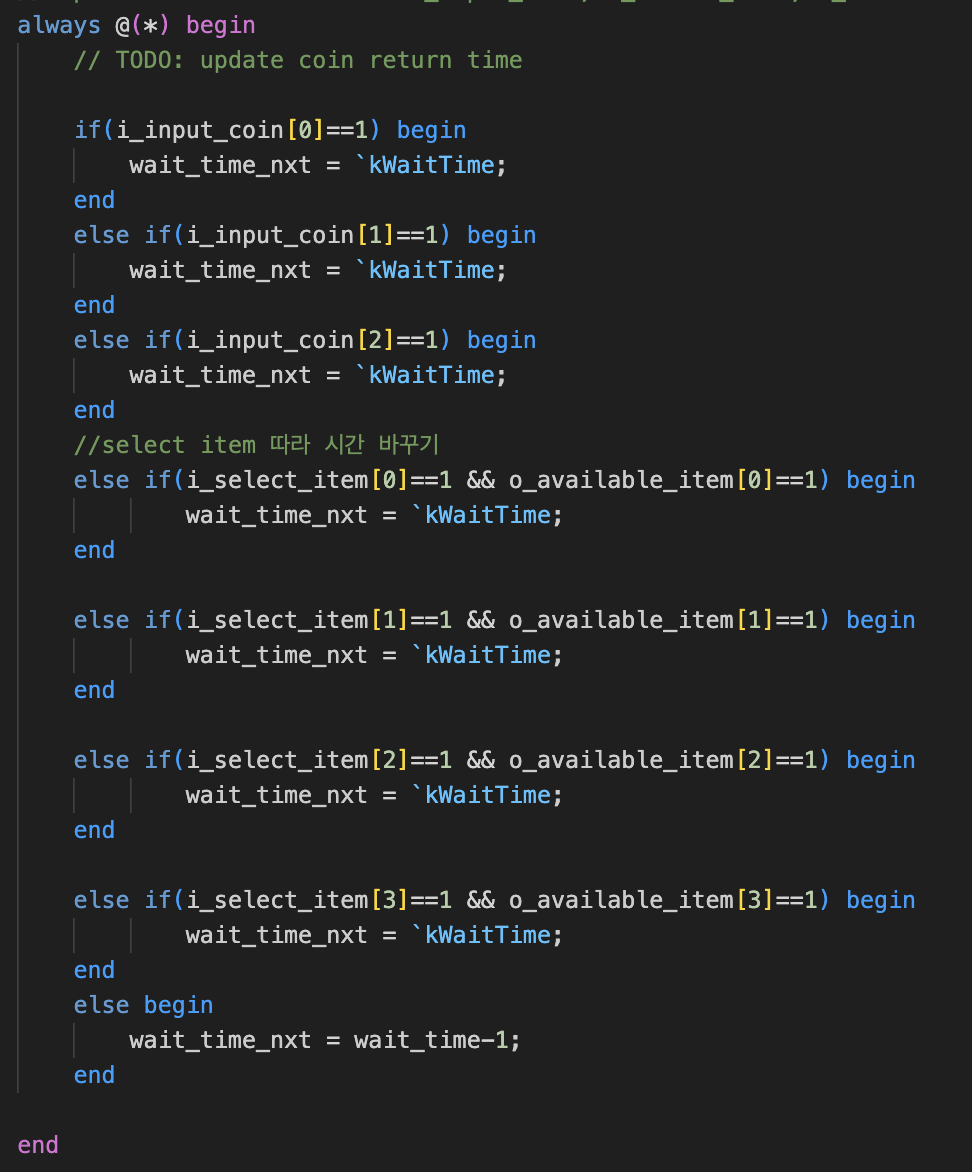


2)check\_time\_and\_coin 모듈

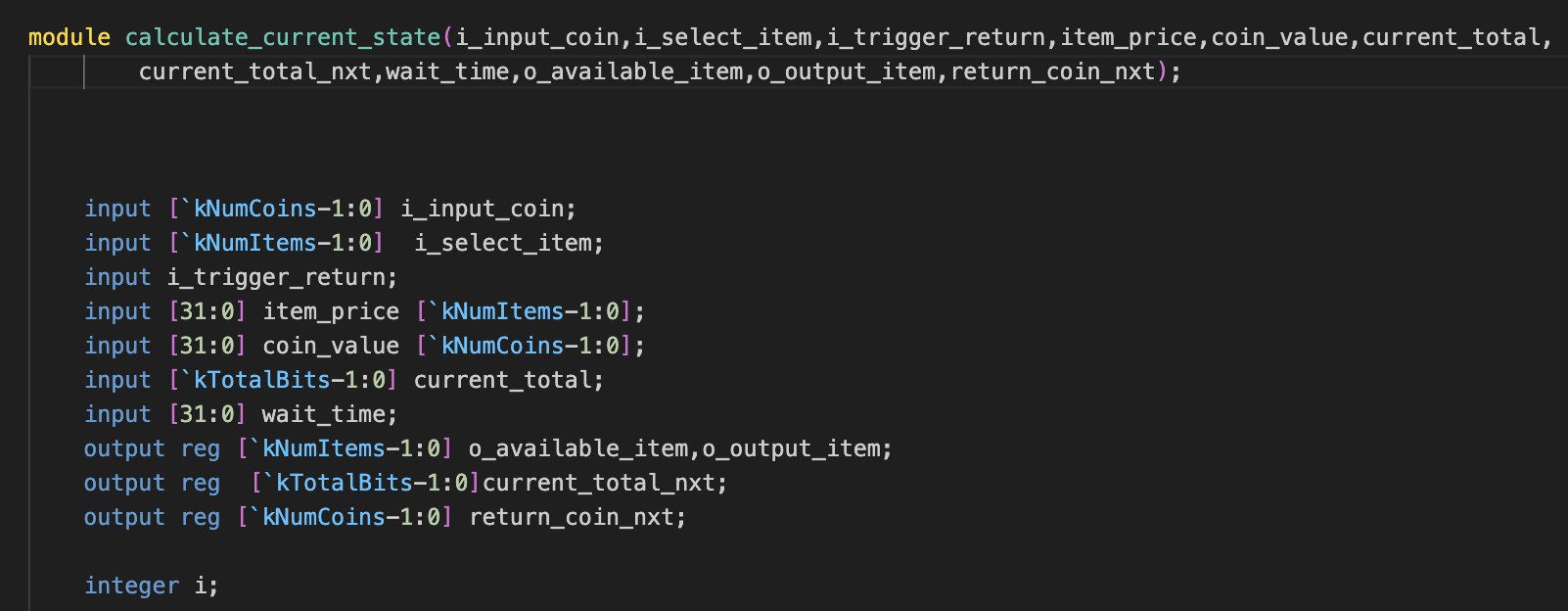
check\_time\_and\_coin 모듈에선 wait\_time을 계산한다.



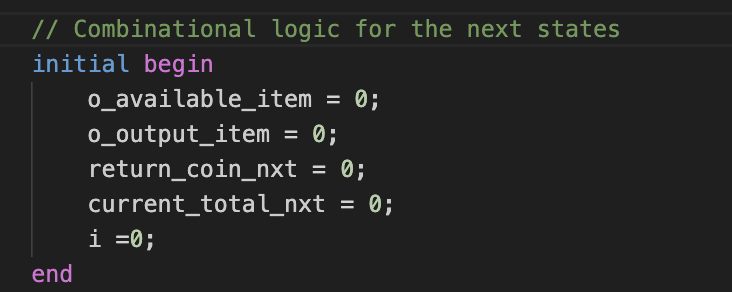
아래와 같이 if, else if, else 구조를 이용해 input\_coin이 바뀌면 wait\_time\_nxt를 10으로 하고, select\_item이 바뀌고 선택된 item이 available하다면 또한 wait\_time\_nxt를 10으로 한다. 위 경우가 아니라면 wait\_time\_nxt는 wait\_time-1이다. 여기서 if, else if, else문 구조를 이용할 수 있는것은 vending machine특성상 input\_coin과 select\_item이 동시에 0이 아닌 수를 가지는 state가 없기 때문이다. 프로젝트 초반에는 아래의 combinational logic에서 wait\_time에 blocking assignment를 이용해 10을 입력하고, 같은 모듈의 다음 sequantial logic에서 non-blocking assignment를 이용해 매 사이클마다 1을 빼주려 했지만, 어떠한 이유에서인지 같은 모듈에서 어떤 변수에 두가지 assignment를 모두 이용할 수 없다는 warning이 발생해 combinational logic에서 wait\_time\_nxt를 계산하고 sequential logic에서 wait\_time에 wait\_time\_nxt를 넣어주는 식으로 구현했다.



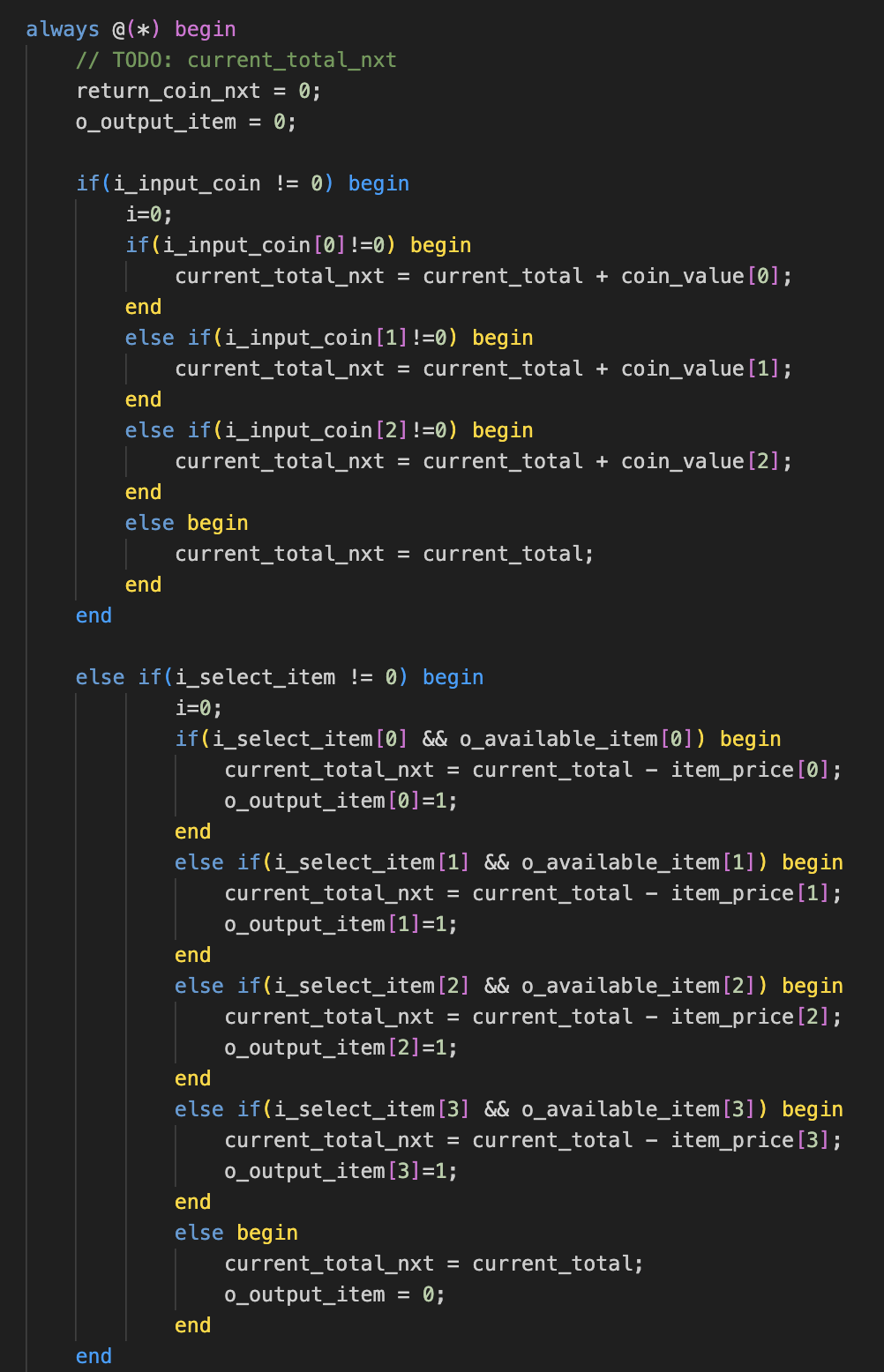
3)calculate\_current\_state 모듈



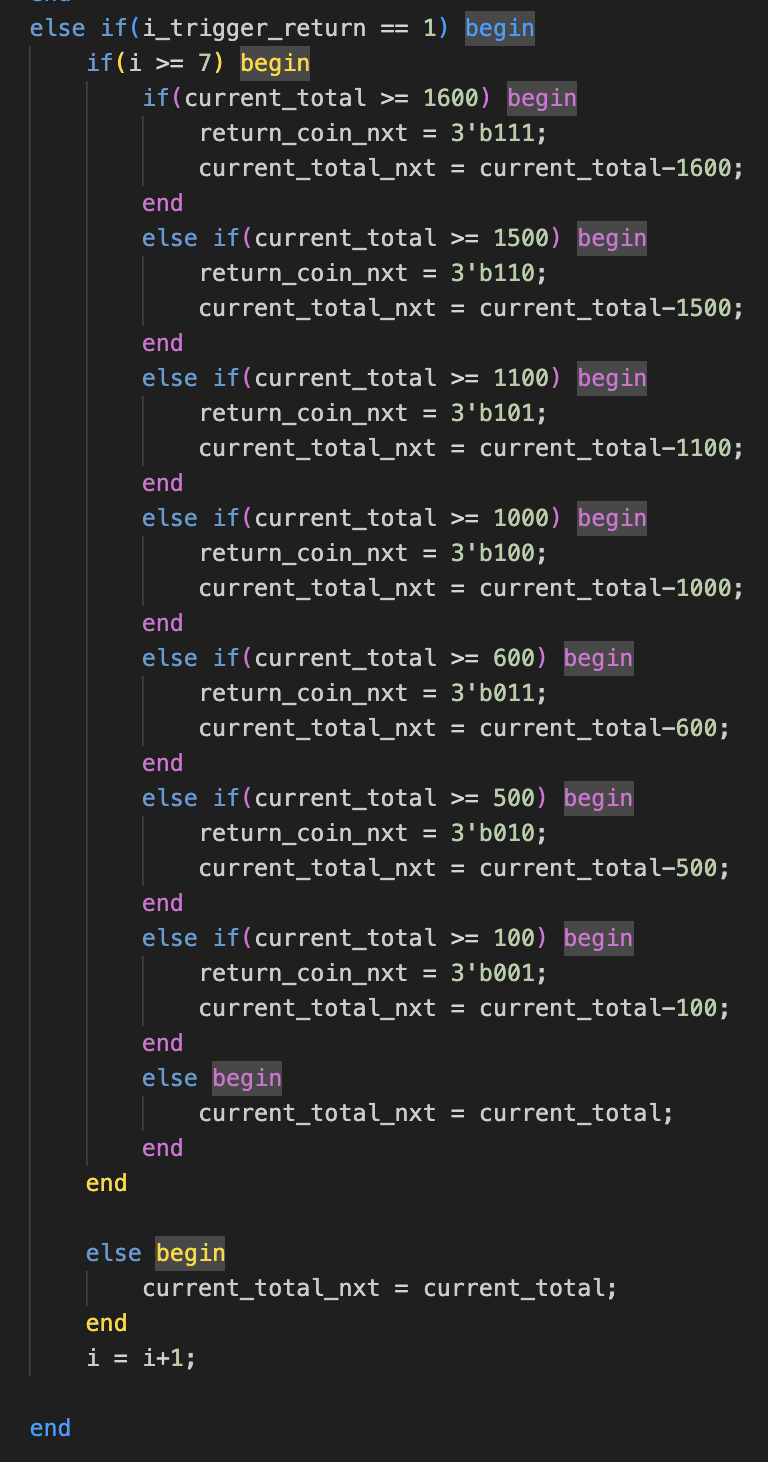
calculate\_current\_state 모듈에선 i\_input\_coin, i\_select\_item, wait\_time, current\_total, item\_price, coin\_value등을 input으로 받아 o\_available\_item, o\_output\_item, current\_total\_nxt, return\_coin\_nxt를 output으로 내보낸다.



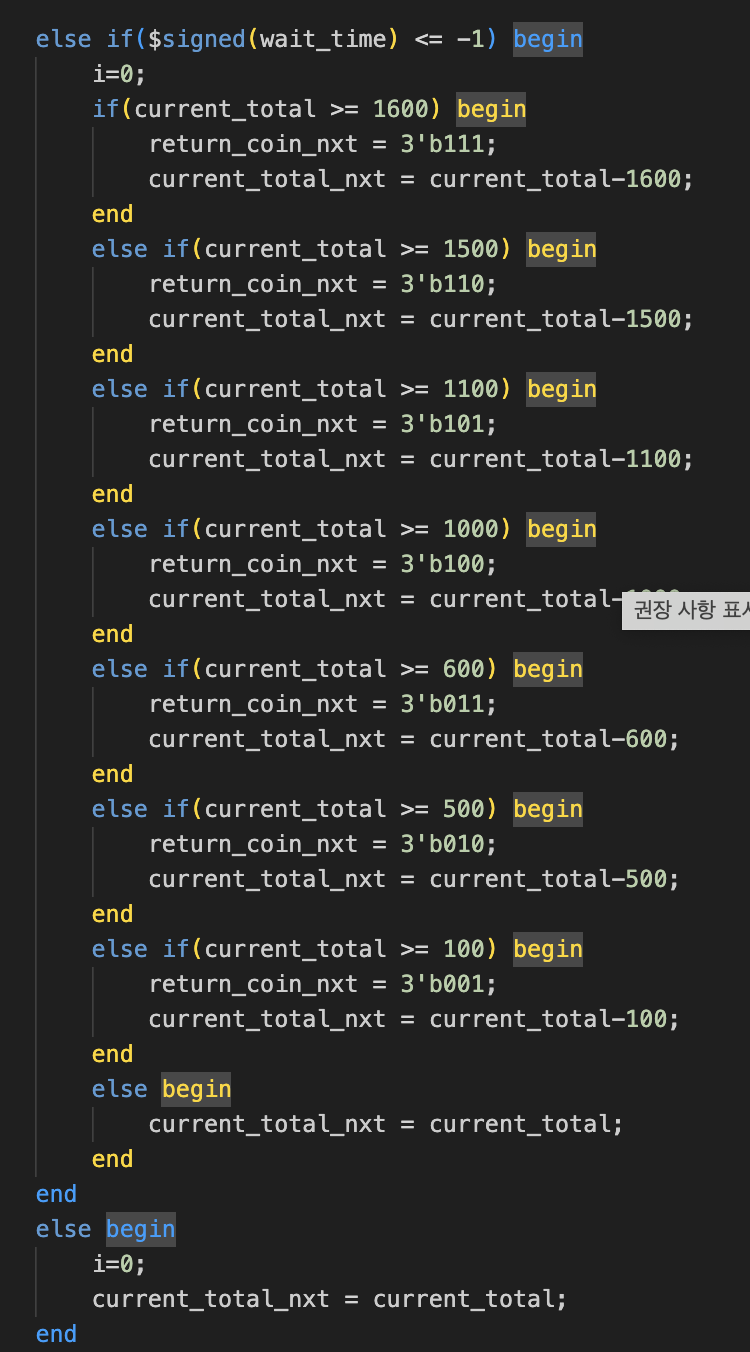
우선 위와 같이 초기값을 설정해준다. 이후 첫번째 combinational logic에서 if, else if ,else문 구조를 이용해 current\_total\_nxt, o\_output\_item, return\_coin\_nxt를 계산한다. 먼저 if문을 이용해 input\_coin이 0이 아니라면 current\_total\_nxt에 해당하는 코인만큼 current\_total에서 더한 값을 저장한다. 다음 else if문에선 select\_item과 o\_available\_item 값에 따라 o\_output\_item과 current\_total\_nxt 값을 계산한다.



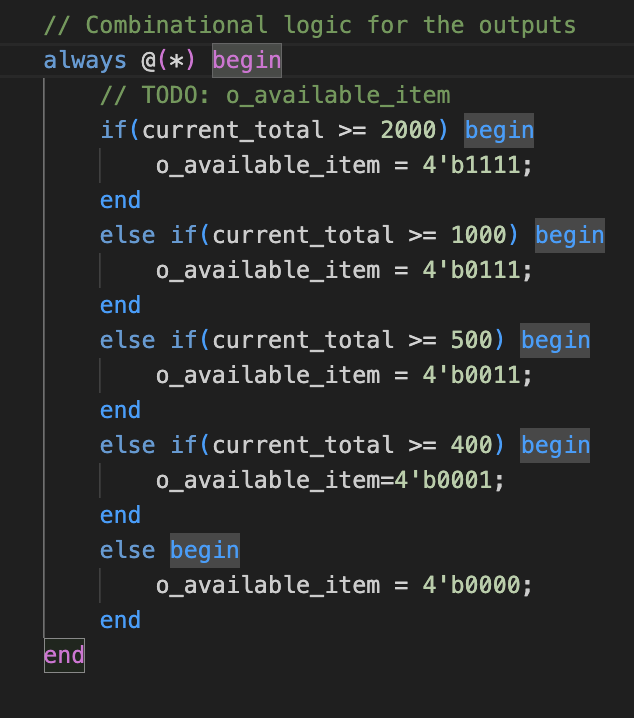
다음 else if문에선 i\_trigger\_return이 1로 세팅되면 리턴되게 하는 부분이다. 테스트벤치에서 i\_trigger\_return이 1로 세팅되면 세번의 사이클을 기다렸다 리턴해야 하므로 모듈에서 integer i를 0으로 초기화하고 i\_trigger\_return이 1인 state 마다 else if 문에서 i에 1씩 더해 세번의 사이클이 돌아야 실행되도록 하였다.



또한 10번의 사이클동안 아무런 행동도 하지 않으면 리턴해야 하므로 wait\_time이 음수가 되면 리턴이 실행되도록 하였다.

****

다음 combinational logic에선 current\_total에 따라 o\_available\_item을 세팅하도록 했다.

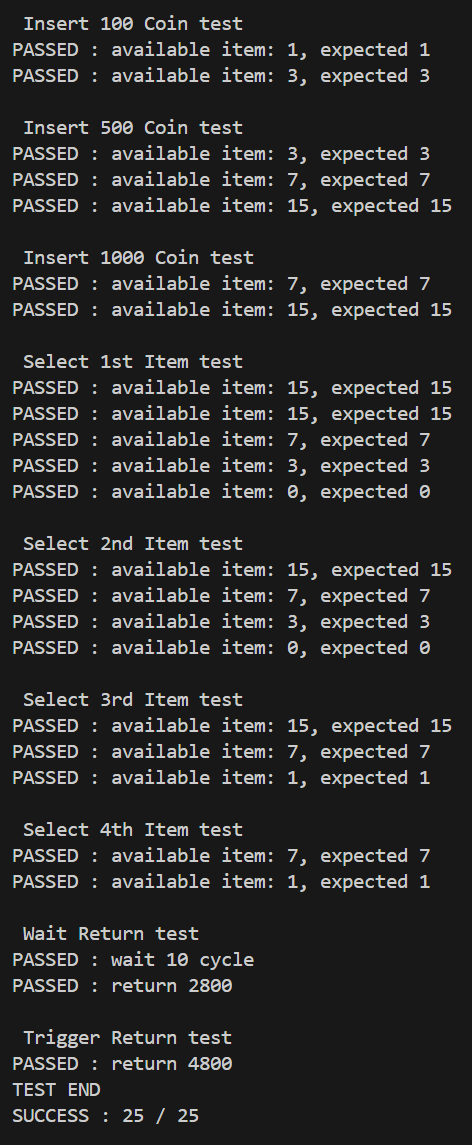


**Discussion**

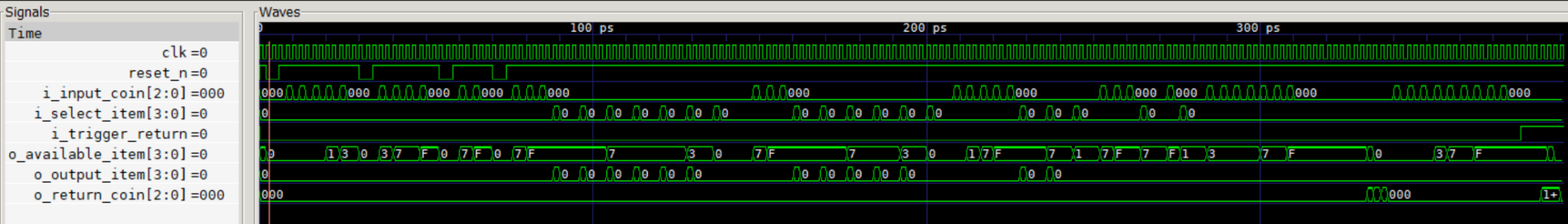
불안정한 코딩 스타일 교정: verilog를 이용해 vending machine을 구현하는 과정에서 C++나 Python과 같이 일반 프로그래밍 언어를 다루는 것처럼 코드를 작성하여 많은 warning과 error를 마주하였다. 이를 해결하며 verilog에 보다 익숙해질 수 있었다.

1. UNUSEDSIGNAL: 정의된 signal이 절대 사용되거나 소비되지 않을 때 발생하는 경고이다. wait\_time을 계산하는 코드를 작성할 때 wait\_time을 모듈에서 사용함에도 불구하고 해당 warning을 발생시켜 어려움을 겪었다.   
   input으로만 사용하던 signal을 다른 모듈에서 output으로도 사용하면 이를 해결할 수 있다는 것을 알아내었다. 다만 편의를 위해 check\_time\_and\_coin 모듈에서 reg로 wait\_time을 선언하여 모듈 내에서 사용하였다.
2. LATCH: 해당 경고는 combinational logic에서 의도하지 않은 latch 구조인 경우 latch 추론이 발생할 수 있음을 알려주는 경고이다. 조건문을 사용하는 경우에 자주 발견되었고, 모든 신호에 값이 할당되지 않아 이전의 값에 의존할 위험이 있을 때 경고가 발생하였다. 이 경우 어떠한 경우에도 값이 할당되지 않는 신호가 없도록 수정하면 해결할 수 있었다.

**Conclusion**

****

주어진 test case를 모두 통과하였다.

****

모델의 파형도 의도한 대로 출력되었음을 확인할 수 있었다.

vending machine을 만드는 과정에서 finite state machine, mealy& moore machine, combinational& sequential logic 등 디지털 시스템의 중요한 개념을 복기하였다. 또한 verilog를 다루며 HDL를 이용한 설계에 익숙해질 수 있었다.