

논리회로 설계

**VHDL**

**<Finite State Machine을 이용한 자판기 설계>**

Prof.김찬용 교수님

학과 : IT대학 전자공학부

2017115976 이하영

1. 시뮬레이션 결과

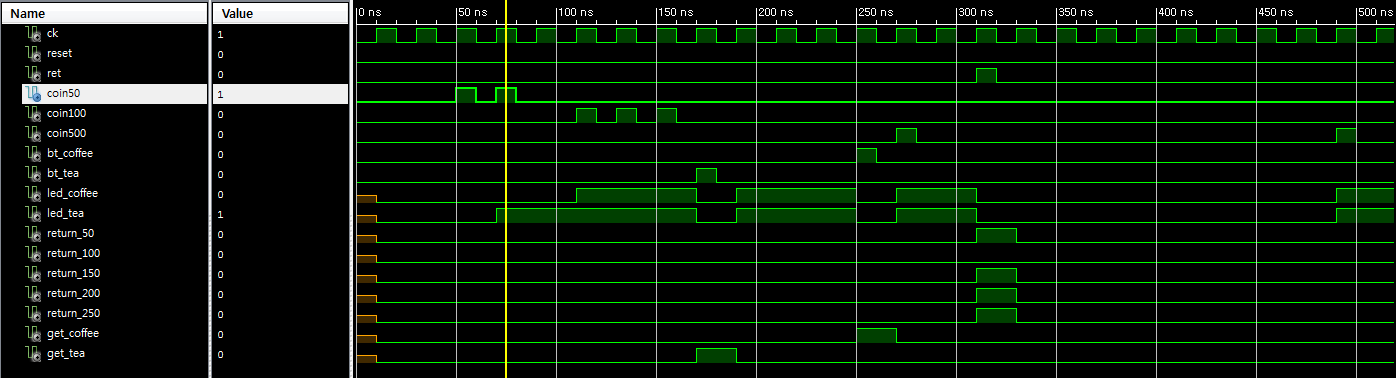


=>50원을 두 번 넣고, 100원을 세 번 넣은 상태에서(400원) tea버튼을 누르면 get\_tea에 의해 자판기로부터 tea가 나오게 된다(현재 상태 300원). 이후에 coffee버튼을 누르면 마찬가지로 get\_coffee가 1로 바뀌고 coffee가 출력이 된다(현재 상태 150원). 그러고 난 후 500원을 더 넣은 상태에서 만약 아무 음료도 안 마시고 싶을 때 ret(return)버튼을 누르면 모든 돈이 반환된다. reset버튼을 누르면 현재 돈 상태, LED 모든 것이 0으로 초기화 된다. 따라서 마지막으로 500원을 더 넣고 reset버튼을 누르면 불이 들어오던 LED들이 모두 꺼지고, 반환되는 돈 없이 모든 상태가 초기화 되는 것을 확인할 수 있다.

(1) 50원을 한 번 넣은 상태

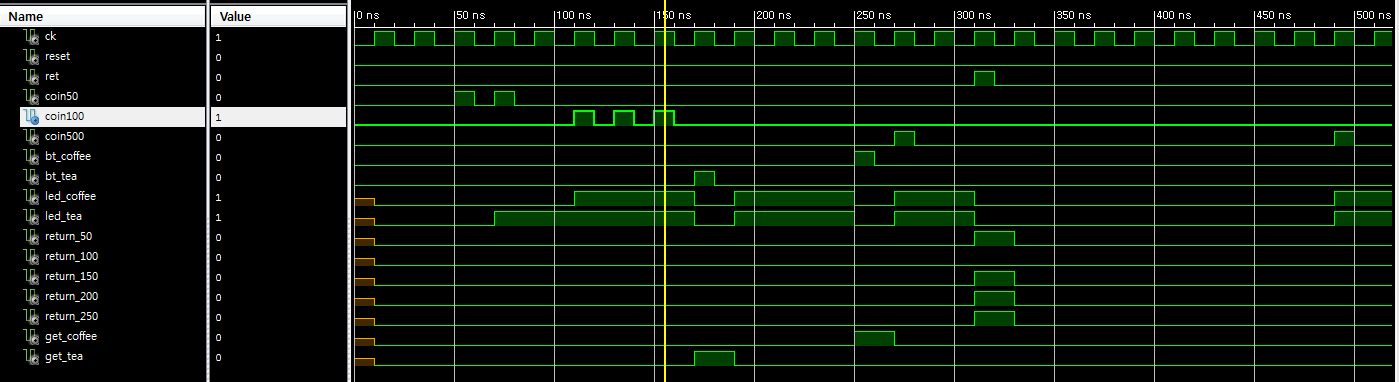
=>처음에 50원 하나를 넣었을 때는 tea와 coffee를 살 수 없으므로 LED에 불이 들어오지 않는다. (tea는 100원, coffee는 150원!!) **WON\_STATE : won\_50**

(2) 50원을 하나 더 넣었을 때



=>50원을 하나 더 넣어서 현재 100원인 상태이다. 따라서 coffee LED는 불이 꺼져있지만, tea는 살 수 있는 상태이므로 LED에 불이 들어온다***.* WON\_STATE : won\_100**

(3) 100원 3번 넣었을 때



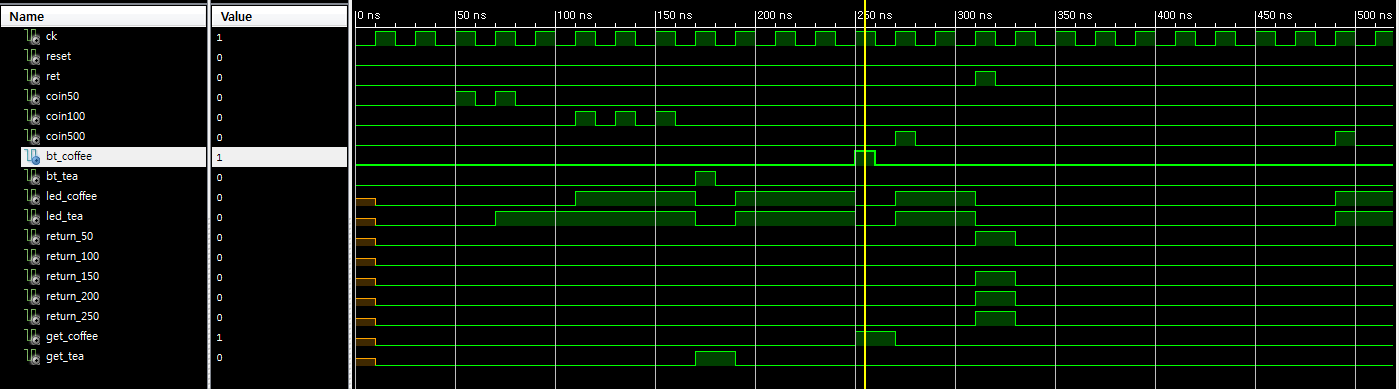
=>위의 두 경우와 마찬가지로 100원을 10ns 간격으로 3번 넣었을 때이다. (2)에서 100원인 상태였으므로 100원을 한번 더 넣은 상태에서 LED불이 들어오는 것을 확인할 수 있다. 현재 위의 상태는 50원 2개과, 100원 3개가 들어온 400원인 상태이다. **WON\_STATE : won\_400**

(4) 400원인 상태에서 tea버튼을 눌렀을 때



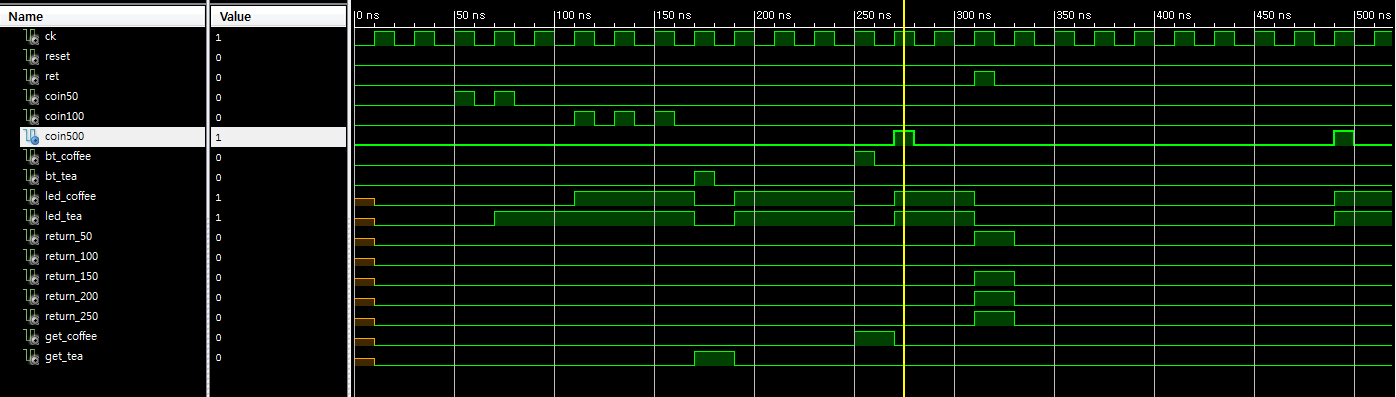
=>400원이 들어있는 상태에서 tea버튼을 누르면 get\_tea에 의해 tea가 나온 것을 확인할 수 있다. 이때 ret(return)버튼을 누르지 않았으므로 남은 거스름돈이 나오지 않는다. 또한 tea가 나올때는 잠시 LED를 꺼두었다. **WON\_STATE : won\_300**

(5) tea 뽑은 후 coffee버튼을 눌렀을 때

****

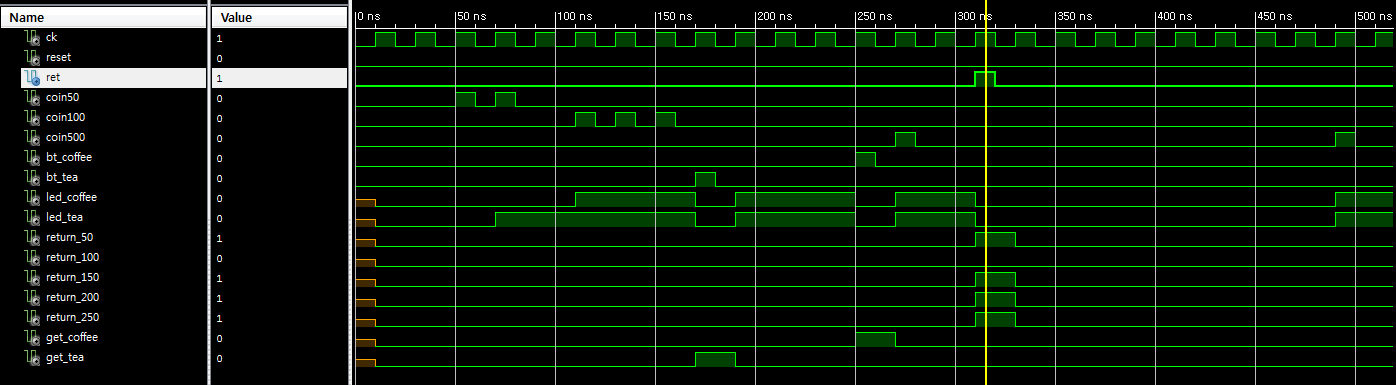
=> 위에서 tea를 뽑고 난 후, 300원이 들어있었는데 coffee버튼을 누르면 get\_coffee에 의해 coffee가 나온 것을 확인할 수 있다. 마찬가지로 ret버튼을 누르지 않아서 반환되는 없다**! WON\_STATE : won\_150**

(6) 500원을 추가로 더 넣은 경우

****

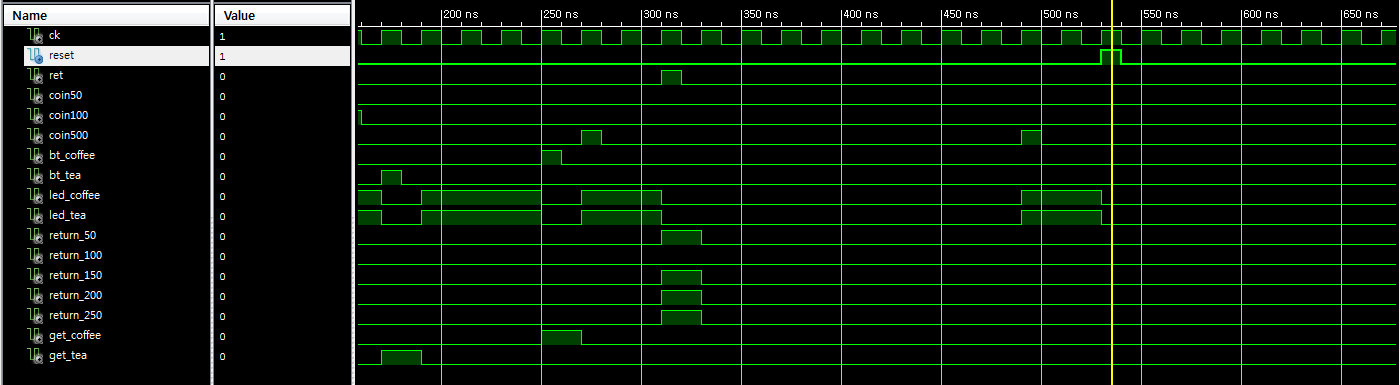
=> 현재 돈 상태가 150원인 상태에서 500원을 더 넣으면 tea 와 coffee 모두 LED가 들어오는 것을 확인할 수 있다. **WON\_STATE : WON\_650**

(7) return버튼을 눌렀을 때



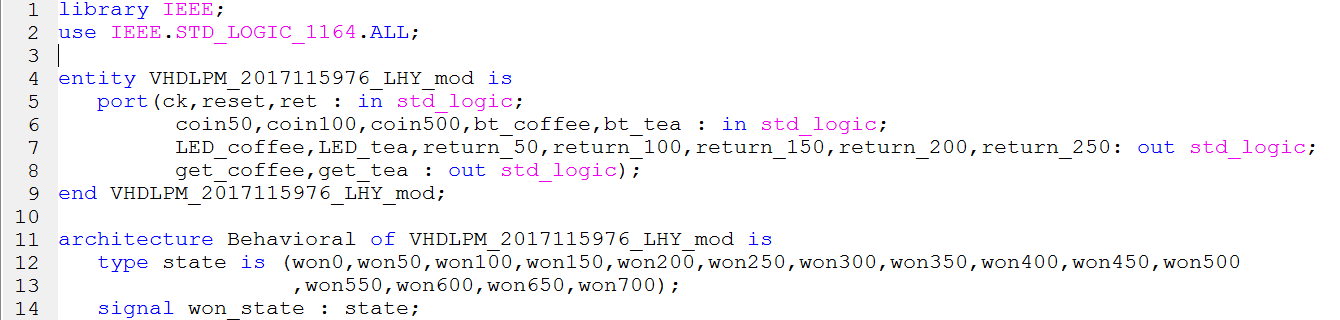
=>return 버튼을 누르면 현재 있는 650원 모두 반환된다. 따라서 return\_250, return\_200, return\_150, return\_50이 1이 되어서 총 650원이 반환되었다. 반환되고 난 후 won\_state는 0이므로 LED 모두가 불이 들어오지 않는다. **WON\_STATE : WON\_0**

(8) reset버튼을 눌렀을 때

=> reset버튼을 누르면 모든 상태가 ‘0’으로 간다. 확인하기 위해 reset버튼 누르기 전에 임의로 500원을 넣어주고 LED에 불이 들어오는 것을 확인하였다. reset버튼을 누르면 반환되는 돈 없이 현재 won\_state와 LED가 ‘0’으로 간다. **WON\_STATE : WON\_0**

2.코드 및 분석-Module

(1) port선언



=>입력신호(in std\_logic)

**coin50, coin100, coin500** : 입력신호로, 사용자가 50원, 100원, 500원 돈을 넣은 것을 의미한다.

**ck, reset, ret** : **ck**는 한 클락을 의미하고 좀 있다 살펴 보겠지만 상승에지 때 작용한다. **reset** 버튼을 누르게 되면 현재 자판기에 있는 돈이 반환 되지 않고 wonstate가 0으로 간다. 또한 LED불도 모두 꺼진다. 사용자가 **return**버튼을 누르게 되면 현재 돈상태가 모두 반환이 되고, wonstate 또한 0으로 간다.

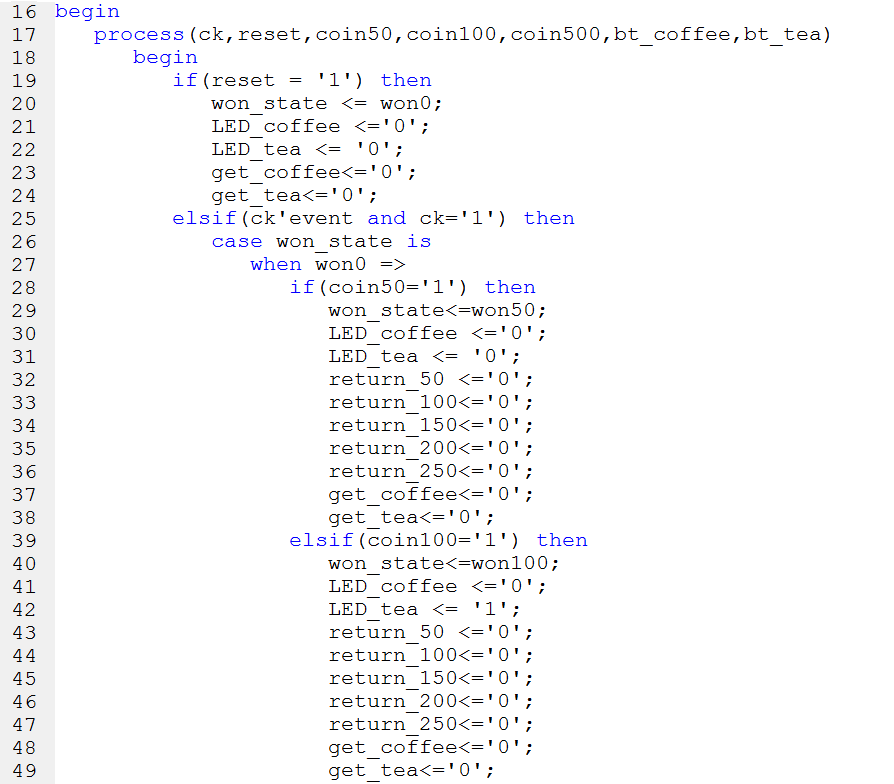
**bt\_tea, bt\_coffee** : 사용자가 tea와 coffee를 사려고 누르면 입력으로 들어가는 신호이다.

=>출력신호(out std\_logic)

**LED\_tea, LED\_coffee** : 만약 각각의 살 수 있는 wonstate가 되면, LED에 불이 들어와 사용자가 살 수 있는 상태를 보여주는 신호이다.

**return\_50, return\_100, return\_150, return\_200, return\_250** : 사용자가 ret버튼을 눌렀을 시, 반환되는 돈을 알려주는 신호이다.

**get\_coffe, get\_tea** : bt\_tea와 bt\_coffee가 입력신호로 들어오면 tea 와 coffee를 각각 출력하는 신호이다.

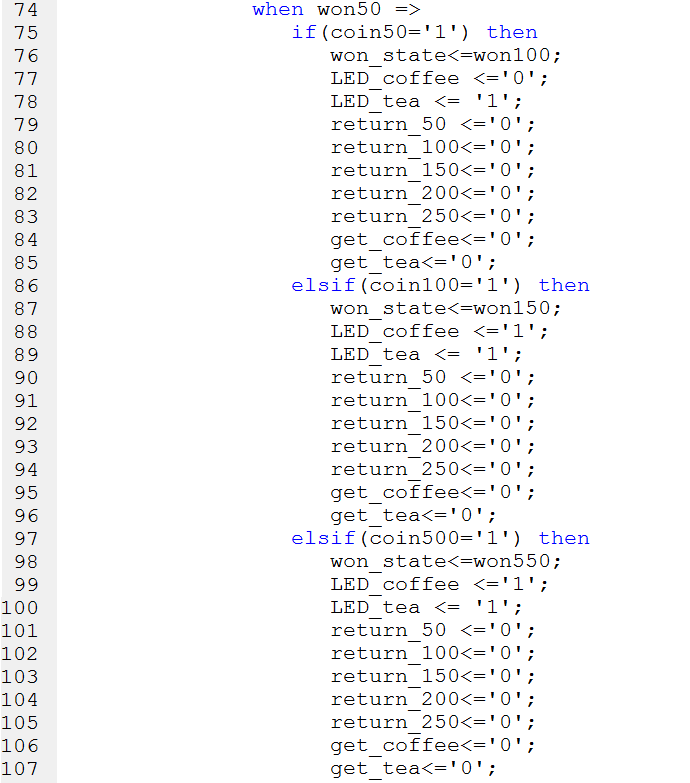
(2) case/when문 시작

=> 기계 동작 중에 **reset**이 ‘1’이 되면 전원 공급이 끊긴 것으로, 기계가 동작을 안하게 된다. 따라서 현재 돈 상태는 0원으로 가고 **LED**불들은 모두 꺼진다. 자판기가 돈을 먹은 상태이므로 기존에 들어있던 돈들은 반환되지 않는다.

* 그렇지 않고 **reset** 이 ‘0’일 때, 즉 전원 공급이 들어왔을 때 비로소 기계가 작동한다.

=> 자판기가 모든 신호에 반응을 하게 되면 전력소모가 크므로 **ck**신호를 만들어서 상승 엣지 일때만 자판기가 반응하게 만들어 준다.

(3) 현재 돈 상태가 50원 일 때

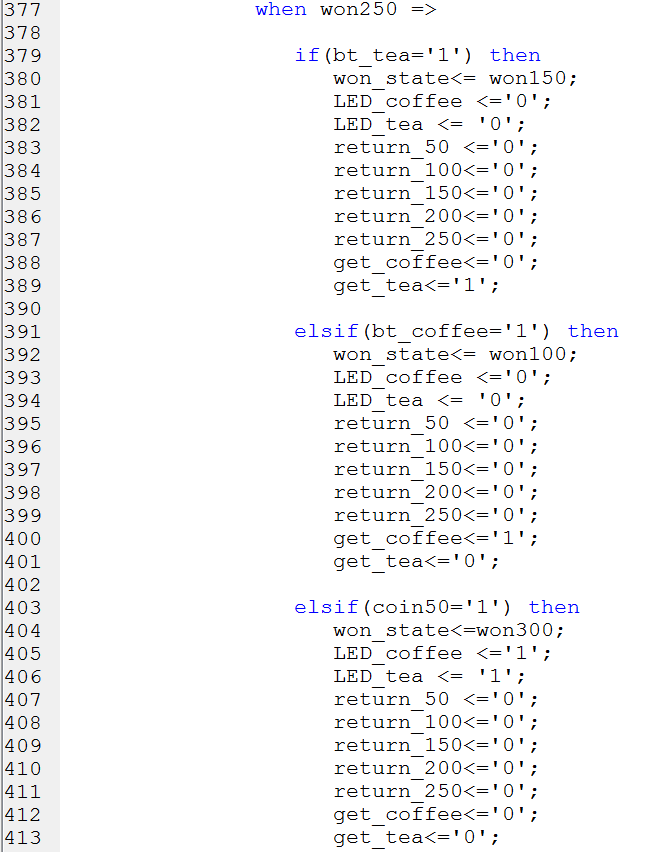


=> 현재 돈 상태가 50원 일 때 동전을 추가로 50원, 100원, 500원을 넣은 경우를 보여준다. 만약 **coin\_50=’1’**이면 50원이 더 들어와서 현재 **won\_state**는 100원으로 간다. 또한 상태가 100원 이므로 tea를 뽑을 수 있다. 따라서 **LED\_tea**에 불이 들어왔다. 마찬가지로 100원이 들어오면 150원, 500원이 들어오면 550원 상태로 이동하고, 두 경우에는 tea 와 coffee 모두에 LED불이 들어온다.

\*\***여기서 질문!!** 만약 500원 넣어서 550원이 됐는데 tea버튼을 누르면 어떻게 되나요? 왜 저기 코드에는 그런 경우는 없는 건가요?

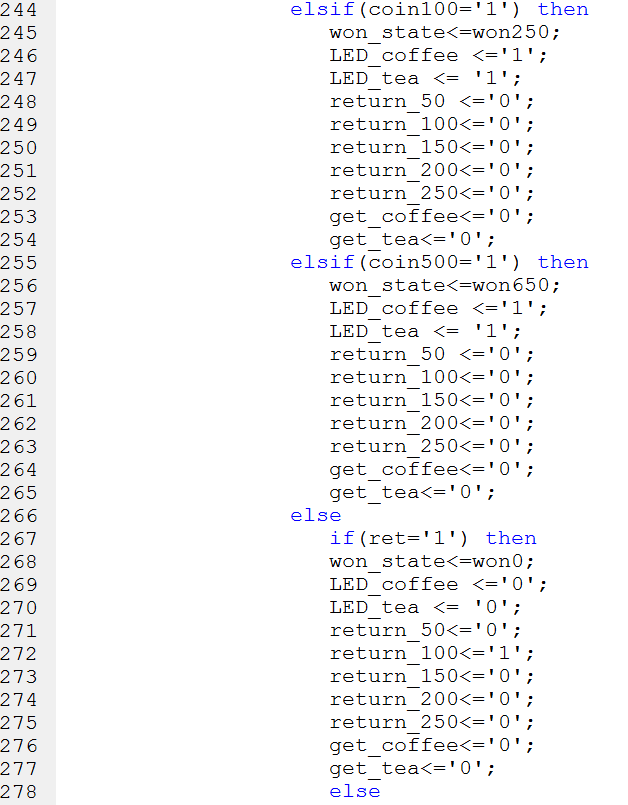
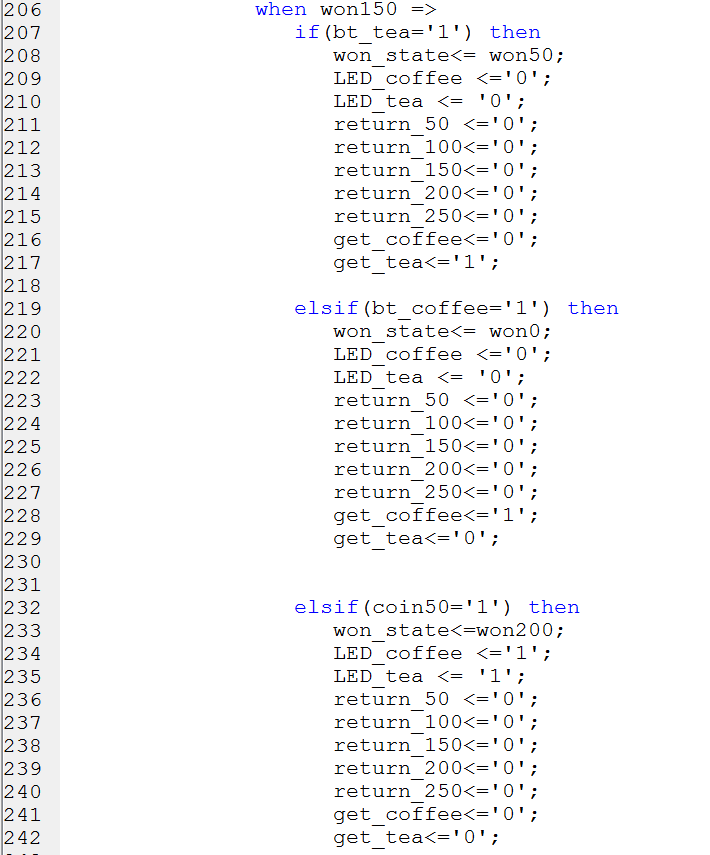
**->답변 :** 50원인 상태에서 500원을 넣게되면 won\_state가 550일때로 갑니다. 따라서 그 경우는 won\_state550으로 가면 존재하는 경우입니다!! 이러한 작동이 바로 유한상태머신(Finite State Machine)을 구현한 것입니다. 즉 하나의 입력을 받으면 그 입력에 의해 현재 상태로부터 다른 상태로 전이가 되어 코드 구현이 비교적 간단해집니다.

(4) won\_state250 일 때 tea 와 coffee 버튼 누른 경우



=> 250원인 경우 tea와 coffee를 모두 살 수 있다. 만약 **bt\_tea**버튼을 누르면 **won\_state150**으로 가고 **get\_tea=’1’**이 됨으로써 tea가 나오게 된다.

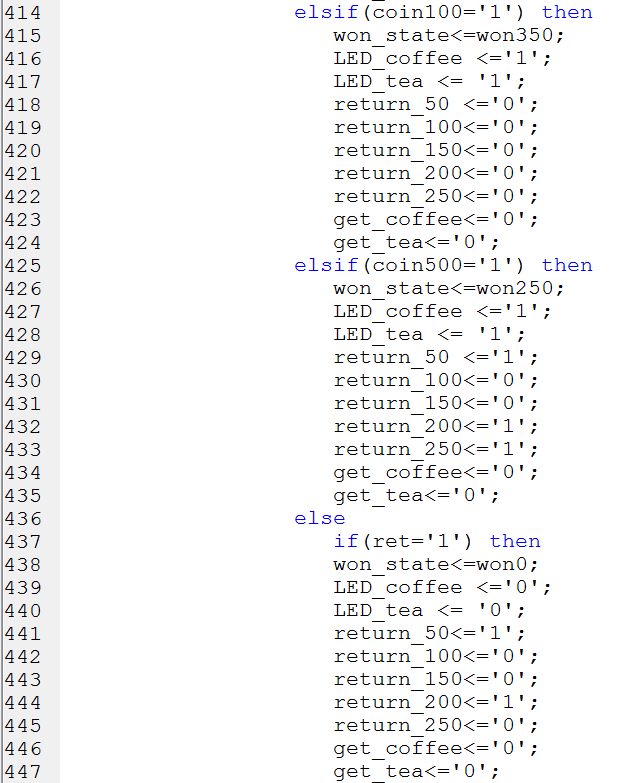
=> (예시)이 상태에서 자판기 내부에 150원이 들어있는데 만약 return받고 싶으면 **won\_state250**인 상태에서 처리하는게 아니라, **won\_state150**으로 전이가 되어 그 상태에서 다시 선택을 하게 된다.



=>위의 사진은 **won\_state150**인 경우로, 처음에 **won\_state250**에서 전이가 되어 왔다. 여기서 동전을 더 넣을수도 있고, tea 와 coffee를 더 뽑을 수도 있고, 남은 돈을 return받을 수도 있다.

=>설명을 다시 **won\_state250**일때로 넘어와서 마찬가지로 **bt\_coffee**를 누르면 **get\_coffee**가 1이 되고, **won\_state100**으로 간다.

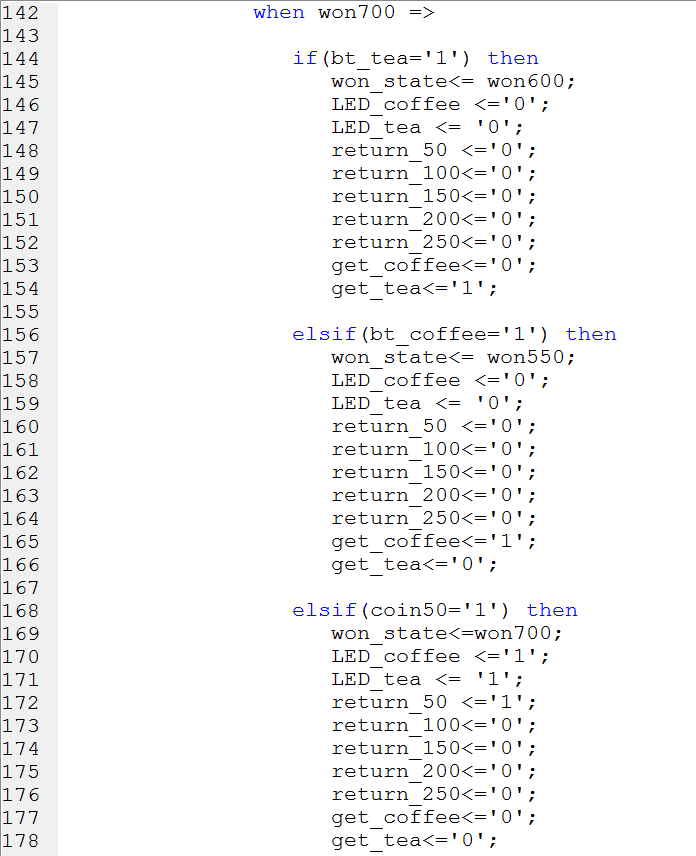
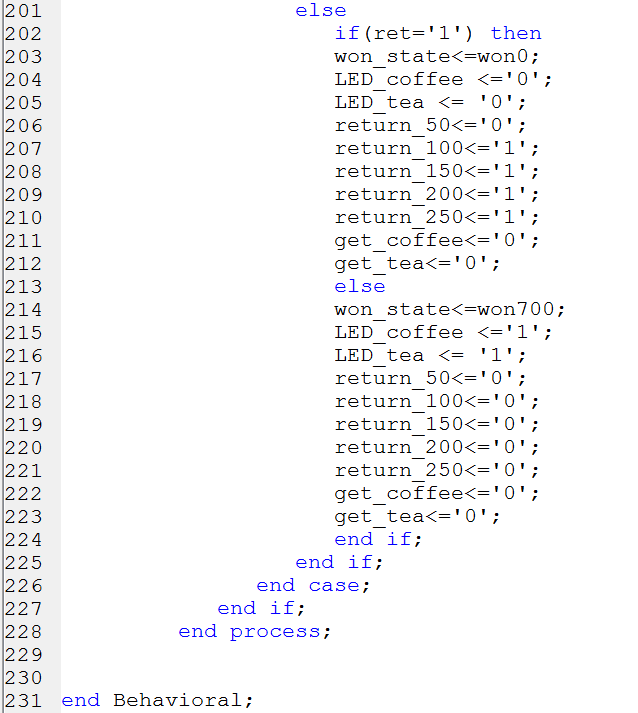
(5) won\_state250 일 때 coin을 더 넣은 경우



=> **won\_state250**인 상태에서 100원을 넣으면 **won\_state350**으로 간다. 이 때 tea 와 coffee를 살 수 있으므로 LED에 불이 들어온다.

=> **won\_state250**인 상태에서 500원을 넣으면 **won\_state750으로 가지 않는다!** 이 자판기는 700원까지가 최대 수용 가능한 돈 범위라서 700원 보다 넘어버리면 넣어주었던 돈을 다시 내뱉는다. 따라서 500원을 넣게 되면 return\_250, return\_200, return\_50 이 모두 ‘1’이 되고 500원을 모두 반환한다.

=> **won\_state250**인 상태에서 **return**을 누르면 250원이 모두 반환된다.

(6) won\_state700인 경우

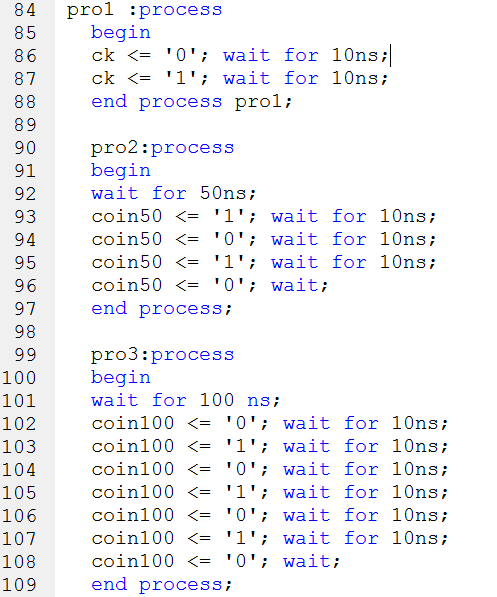
=> 앞의 경우와 마찬가지로, 700원인 상태에서 tea 또는 coffee를 각각 뽑으면, won\_state가 전이되는 것을 확인할 수 있다. Tea 와 coffee를 뽑을 때는 LED에 불이 안 들어오게 설정하였다.

=> 자판기가 수용할 수 있는 돈의 최대 범위는 700원이다. 따라서 **won\_state700**인 상태에서 돈이 더 들어오게 되면 가차없이 들어온 돈이 바로 반환된다.

=> 오른쪽 그림에서 ret=’1’인 경우에는 700원이 모두 반환된다. 만약 ret이 1이 아니면, 즉 ret가 0이면 그냥 LED에 불이 들어온 채로 가만히 있게 된다. 사용자가 무얼 살지 고민하는 시간이라 볼 수 있다.

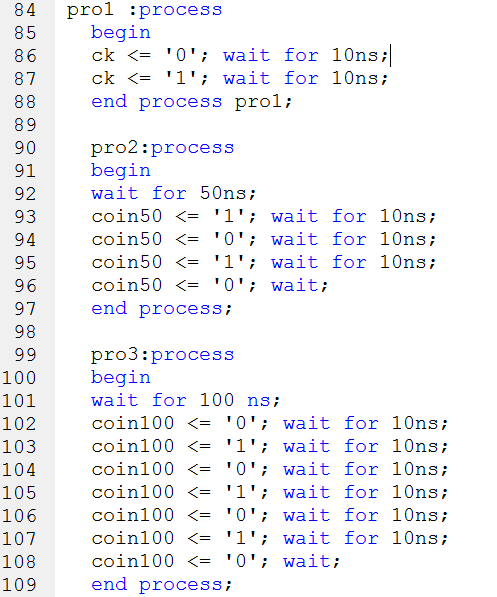
3.코드 및 분석-Test Bench

(1) ck 인가



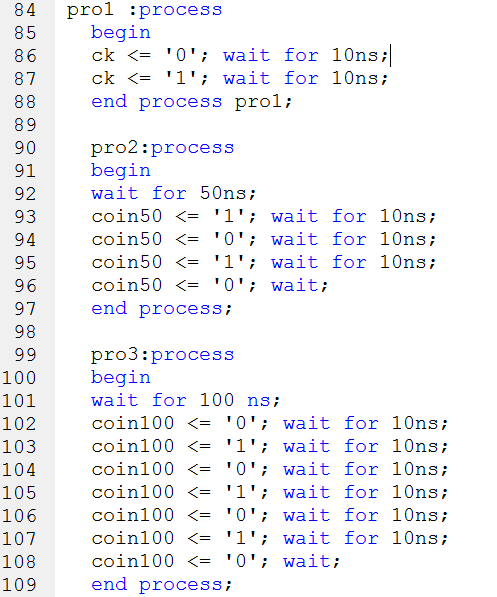
=>ck 신호를 0과 1을 각각 10초씩 반복하며 입력을 가했다. 자판기는 상승에지 때만 동작한다.

(2) coin50 인가



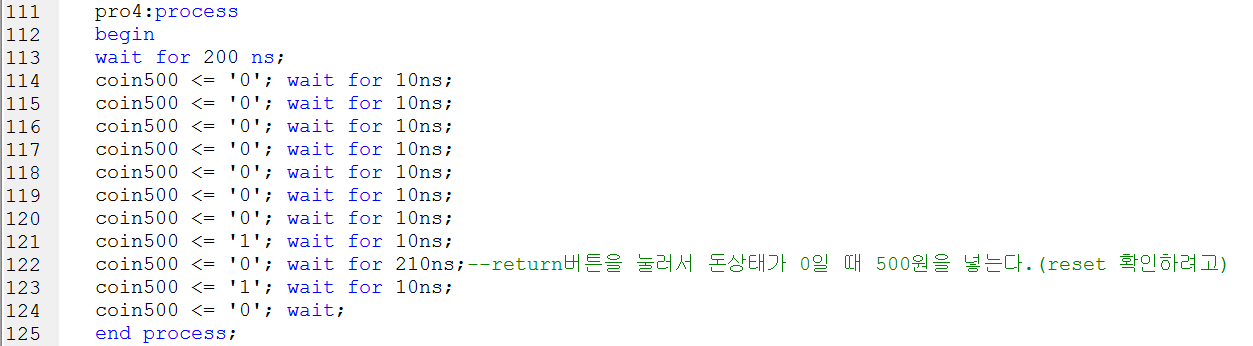
=> 처음에 50ns 기다렸다가 50원을 넣어주었다. ‘1’을 10ns동안 2번 인가하여 만약 ck의 상승에지일 때 coin50신호가 들어온다면 자판기에 50원을 더 넣은 경우로 상태가 전이된다.

(3) coin100 인가



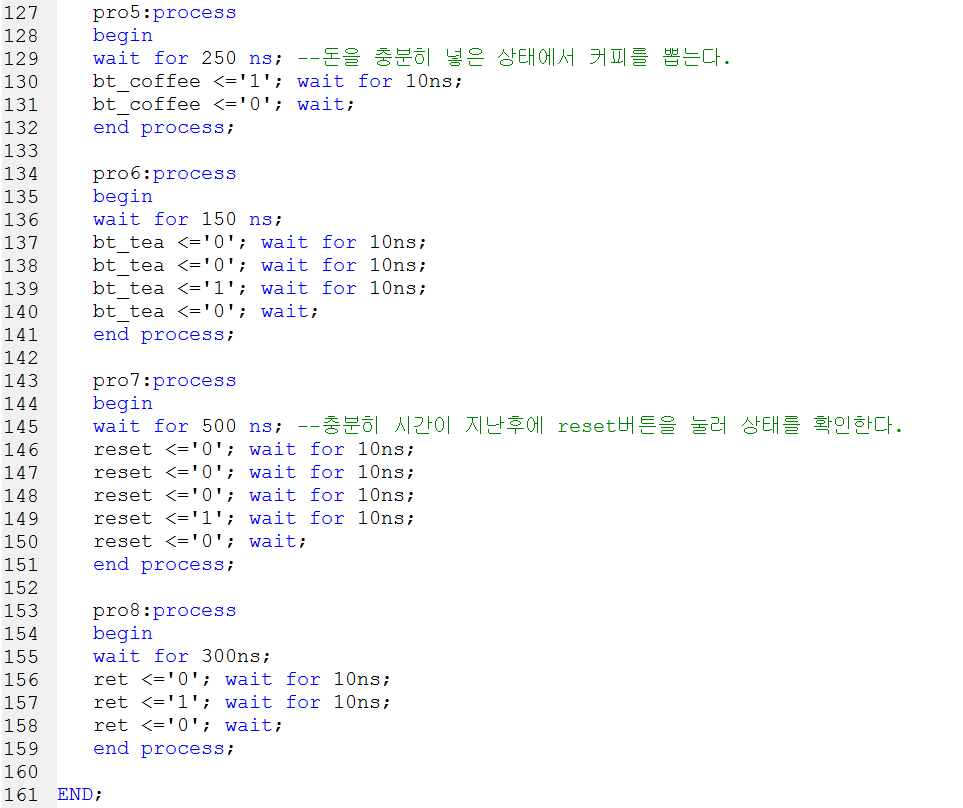
=> 50원을 모두 넣고 난 후에 100원을 넣게 만들어 주려고 100ns를 기다리고 신호를 인가해주었다. 마찬가지로 ck의 상승에지일 때 coin100신호가 들어온다면 자판기에 100원을 더 넣은 경우로 상태가 전이된다.

(4) coin500 인가



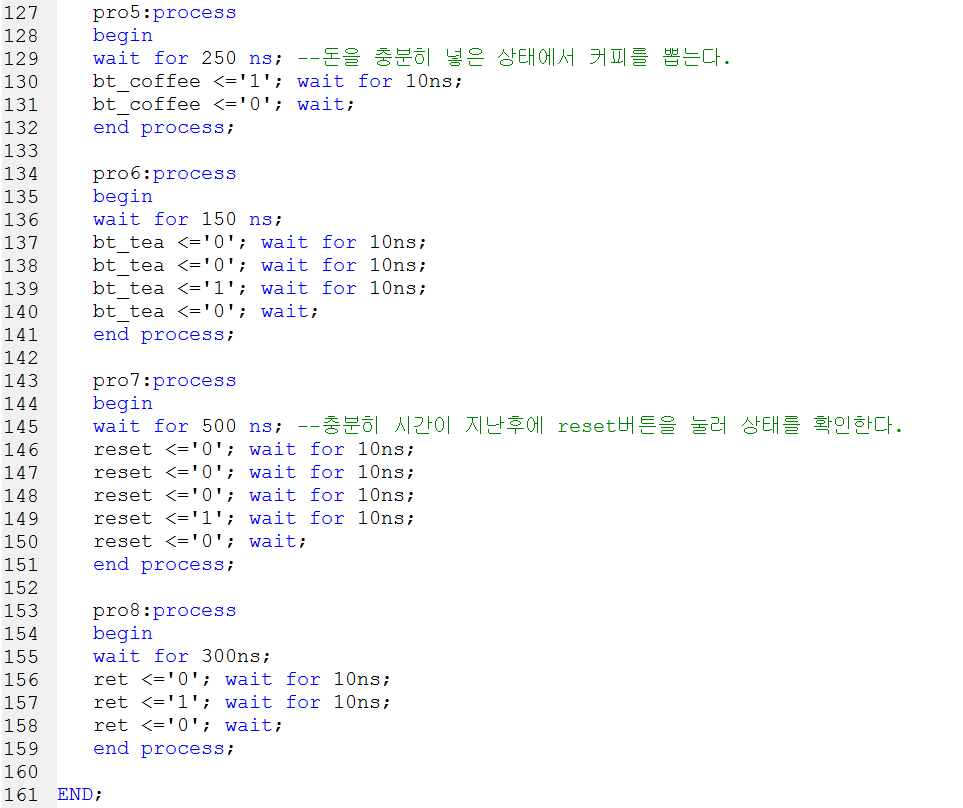
=> 50원 100원을 모두 넣고 난 후에 500원을 넣으려고 200ns를 기다리고 신호를 인가하였다. 중간에 210ns를 기다린 것은, 맨 처음 실행 결과 분석할 때 return을 한번 해서 모든 돈을 반환 받아 won\_state0으로 갔었다. 다시 500원을 넣어서 reset이 실행되는 것을 확인하기 위해 ck의 상승에지와 시간을 맞추었다.

(5) bt\_tea, bt\_coffee



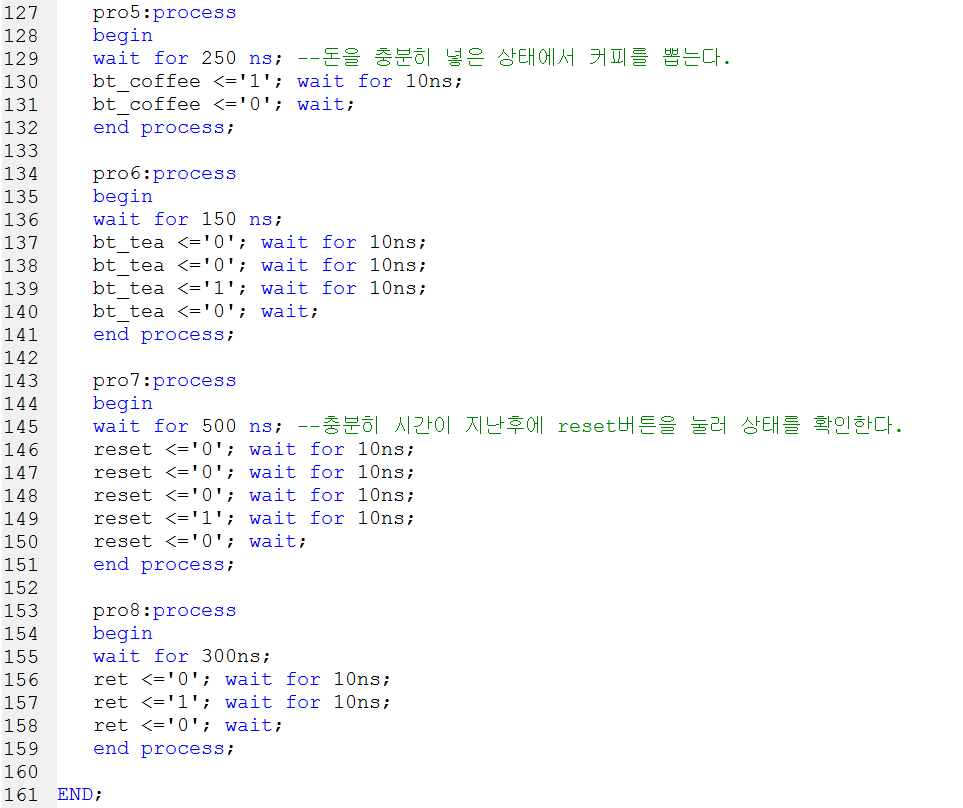
=> 돈을 충분히 넣은 후에 tea 와 coffee를 뽑는다. 따라서 이전의 경우에서 coin을 400원 정도로 넉넉히 넣어준 후 신호를 인가하였다.

(6) reset



=> reset을 누르면 자판기 전원이 끊긴 경우와 마찬가지로 won\_state가 0이 되고 모든 상태가 초기화 된다. 따라서 맨 마지막에 reset 신호를 인가하여 모든 신호를 0으로 가게 하였다.

(7) return



=> return 신호를 인가하여 현재 won\_state에서 들어있는 돈이 모두 반환되는지 확인한다. 돈을 어느정도 넣고난 후에 return 되는 것을 확인하기 위해 300ns를 기다리고 동작하도록 하였다.

*<끝으로 논리회로설계 수업을 듣고 텀 프로젝트 과제를 하면서 느낀점>*

이전의 설계 수업을 들으면서 코딩에 흥미를 붙이고 설계 수업을 더 들어보고 싶어서 논리회로설계를 듣게 되었다. 자바나 C프로그래밍은 소프트웨어적인 성향이 강하다면 논리회로설계는 하드웨어 설계에 직접적인 관련이 있어서 수업시간에 배우면서 새로운 흥미를 느꼈다. 실제 칩을 만들때는 좀 더 복잡하고 어렵겠지만 이렇게 구현을 해서 새로운 칩을 만들 수 있다는게 신기했다.

이번 텀 프로젝트를 하면서 수업시간에 배운 when/case/if 문에 대한 이해도가 한층 더 높아졌다. 과제를 하면서 경우의 수가 많고 또한 유한상태를 구현하면서 찾아봤던 자료들이 많은 도움이 되었다. 한학기 동안 많은것을 배웠고 실제로 구현해보니까 너무 재밌었던 경험이었다.