EE303B Fall 2017

**Report on Lab#4**

Dec. 16, 2017

Student No. 20160042

Koo InYong

Email: [iykoo010@kaist.ac.kr](mailto:iykoo010@kaist.ac.kr)

**Part I. Design and Simulation of modified traffic-light controller of Figure 17.10 and Figure 17.11 so that north-south and east-west have equal priority**

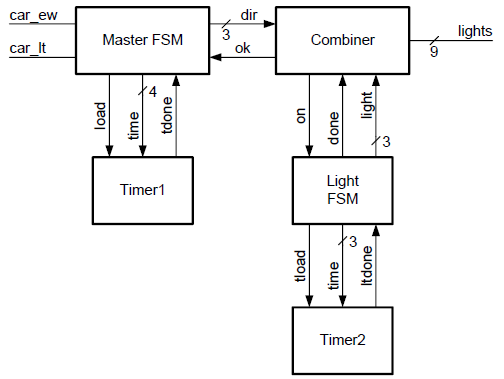
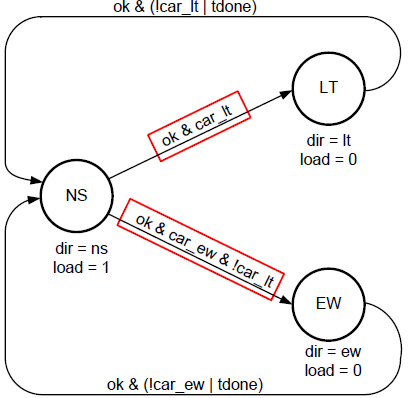
 

Figure 17.10 Block diagram of traffic-light controller Figure 17.11 State diagram of traffic-light controller

***GUIDELINE***

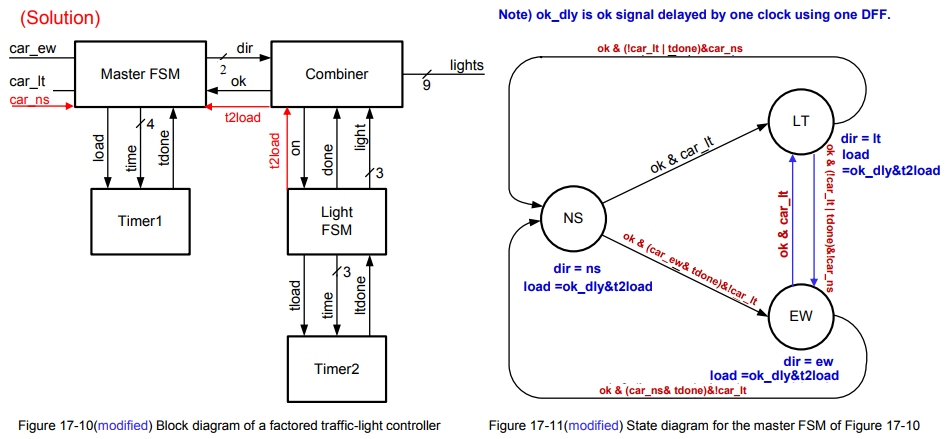
1. *Based on the modified block diagram and state diagram of the master FSM, modify the Verilog of the master FSM including module name changes:*

*-from TLC\_Master to TLC\_Master\_v1*

*-from TB\_TLC\_Master\_Combiner to TB\_TLC\_Master\_Combiner\_v1*

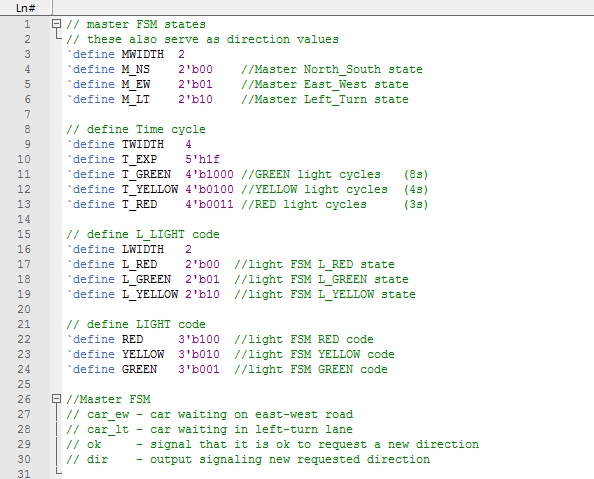
*And etc.*

1. *Compile the Verilog source codes, DUT(Device Under Test) and TB(TestBench)*
2. *Simulate the TB: Obtain the correct waveforms by confirming them with the original timing diagram*
3. *Evaluate the simulation results as a whole to check whether the traffic-light controller with the additional specification function according to the modified diagram for the master FSM*



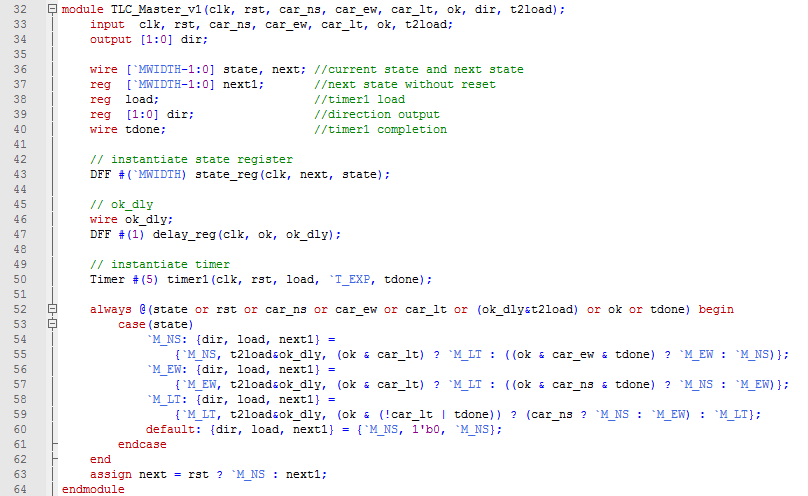
1. **Design (Verilog coding) of DUT (DUT: TLC\_Master\_v1.v)**

First, some predefined values are as following



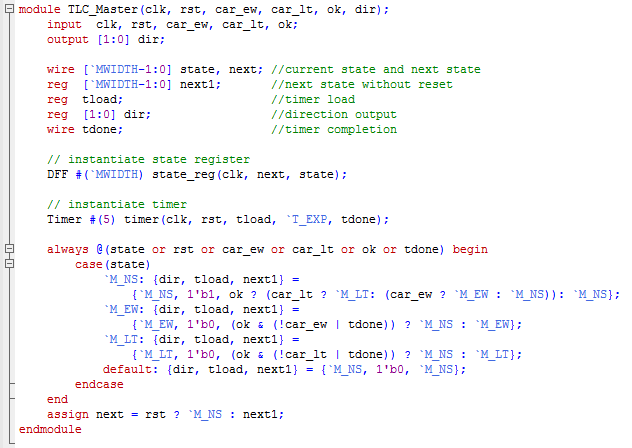
1-1 Definitions for TLC (Traffic Light Control)

Verilog code for module TLC\_Master\_v1 is as following.



1- 2 Verilog coding of TLC\_Master\_v1 (modified master FSM for TLC)

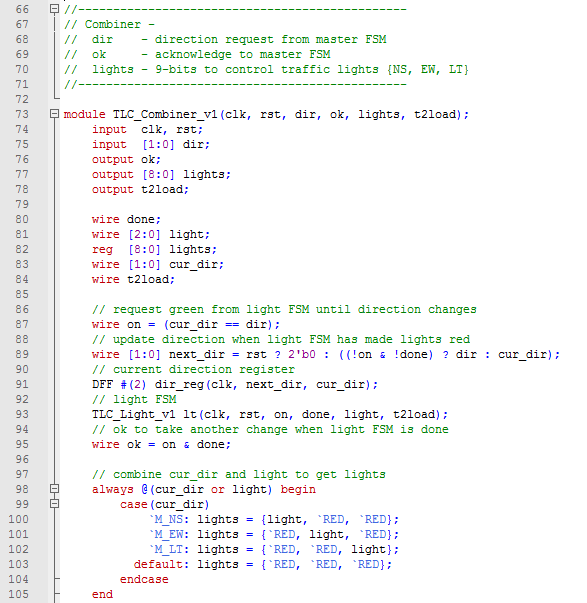
The Verilog code for module TLC\_Master (before modification) is as following.



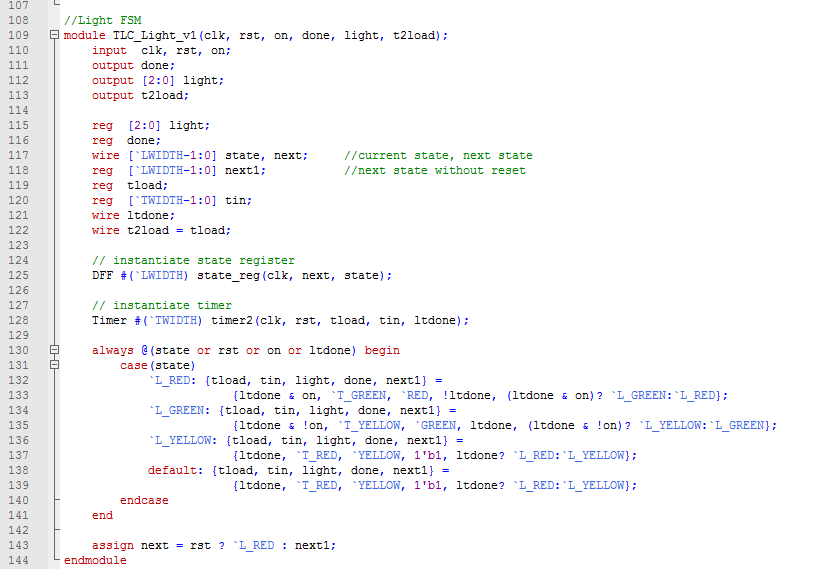
1- 3 Verilog coding of TLC\_Master

We had to modify TLC\_Combiner and TLC\_Light module in order to transfer t2load to TLC\_Master\_v1.

The modified TLC\_Combiner\_v1 and TLC\_Light\_v1 modules are as following.



1- 4 Verilog coding of TLC\_Combiner\_v1

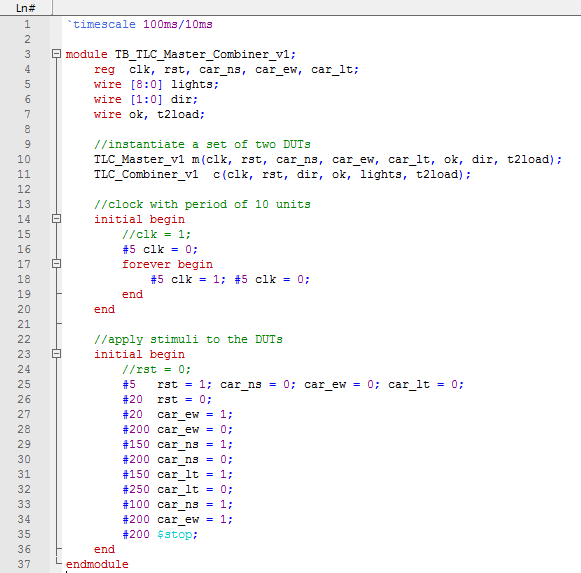


1- 5 Verilog coding of TLC\_Light\_v1

(We skip the original TLC\_Combiner and TLC\_Light module since the modifications are minor.)

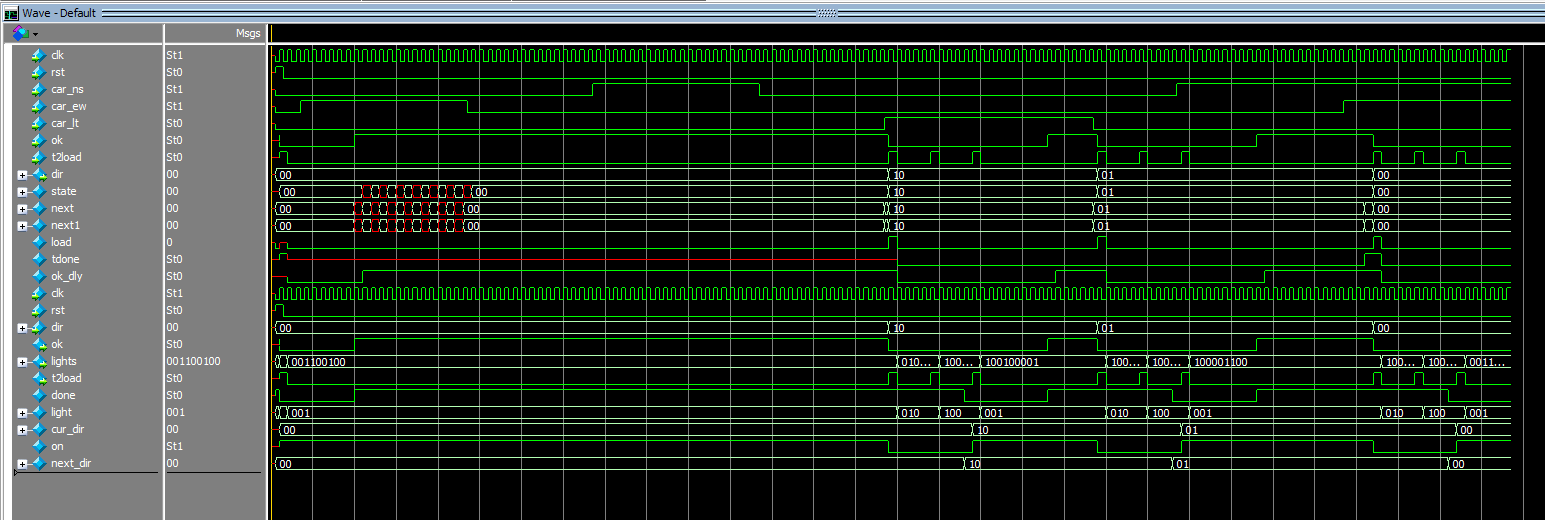
1. **Simulation of the DUT by including the following clock generation and stimuli in the testbench (TB: TB\_TLC\_Master\_Combiner\_v1.v)**

The Testbench code was given on the pdf handout, and is as following.



1- 6 Verilog coding of TB\_TLC\_Master\_Combiner\_v1

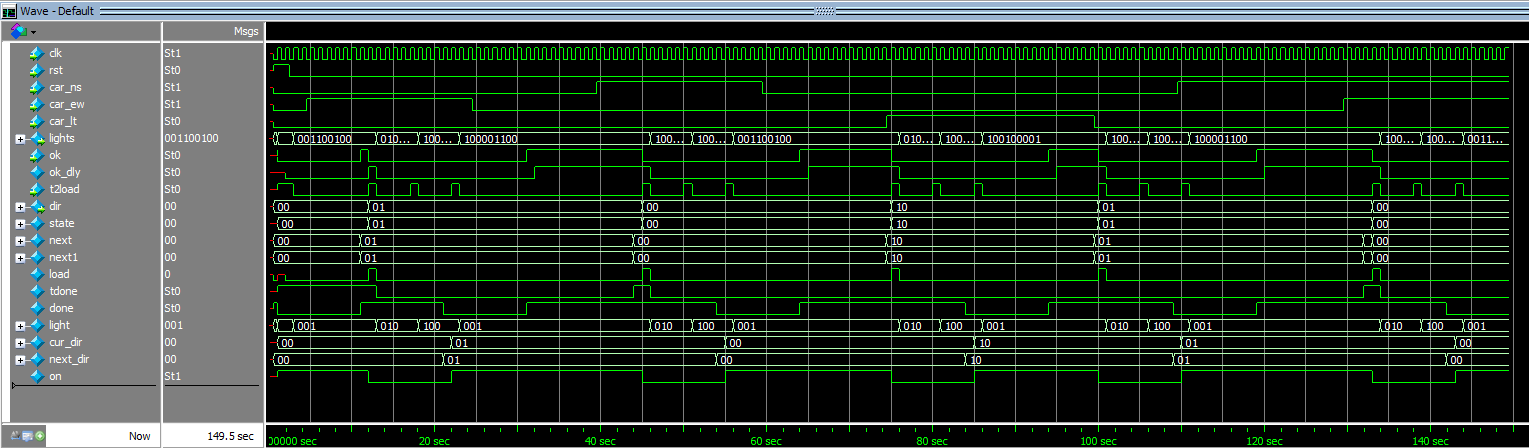
We modified the time interval of reset, from #10 to #20 (on code line 26), as we found error in initializing stage of the simulation as below.



1- 7 Incorrect simulation result of modified TLC (Before modifying rst interval)

*tdone* (output of Timer 1) did not act normally, as we had uninitialized *load* value. *load* is determined as *t2load* & *ok\_dly* value, where *ok\_dly* is a one-clock cycle delayed value of *ok*. Which means, load can be determined after two clock cycles. Thus, we changed the reset interval, and the simulation worked normally.

This is the actual result of the simulation.



1- 8 Simulation result of modified TLC

1. **Evaluation**

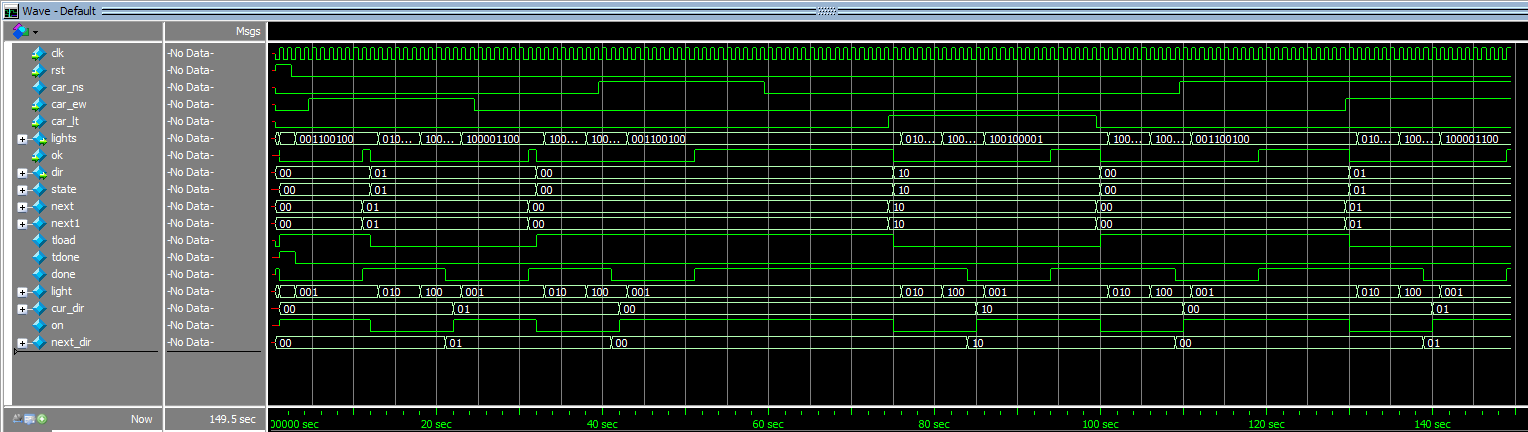
We wanted to compare the result with the original timing diagram, using traffic-controller before modification.

Thus, we modified two lines in testbench as below.



1- 9 Code modification of Testbench for original timing diagram

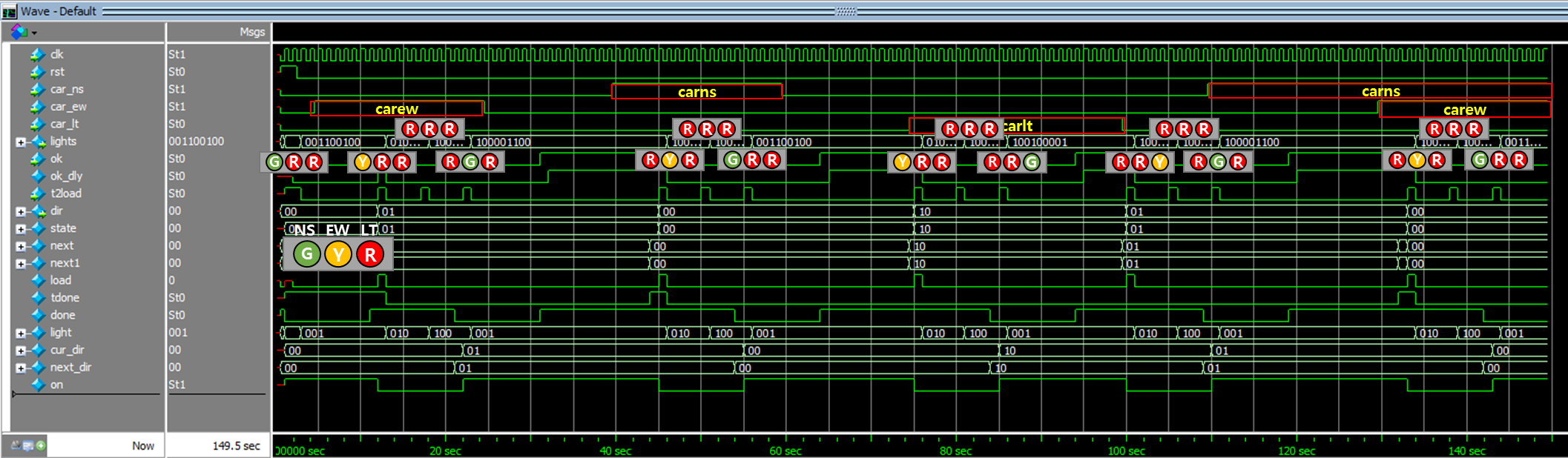
The result showed as following.



1- 10 Simulation result of original TLC

Let’s examine our results with detail.

I applied some description on Figure 1-8 and 1-10 as below.



1- 8 (*detailed*) Simulation result of modified TLC

****

1- 10 (*detailed*) Simulation result of original TLC

In original TLC, car\_ns input was not considered, so lights responded only on car\_ew and car\_lt. We can observe that lights returned to default state (*gns*) after some time interval as stimuli (car\_ew or car\_lt) are gone.

In modified TLC, we could see that lights returned to *gns* around 50 sec. (Original around 38 sec) The lights changed slower because now, *gns* is not the ‘default’ state, so lights remain on *gew* state until car\_ns stimulus is applied.

Also note that around 110 sec, lights shift to *gew,* not *gns*, because according to state diagram of modified TLC, light goes to *gew* after car\_lt stimulus is gone if there is no car\_ns*.* Lights later respond to car\_ns stimulus, and move to *gns* state though car\_ew stimuli is also applied.

**Part II. Design and Simulation of Vending Machine FSM using Microcode**

1. **Design of the DUT**

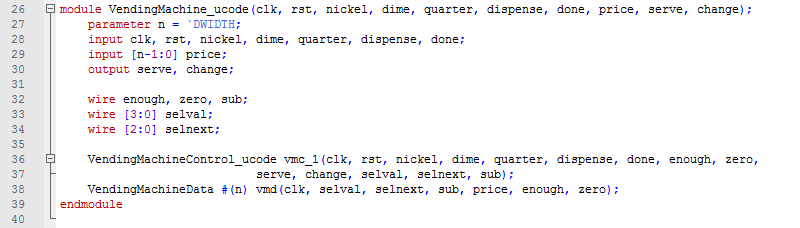
Refer to Lab3 where we handled Vending Machine using pla.

We predefined some values as following.



2- 1 Definitions for VendingMachine. (Reference to Lab3, VendingMachine\_pla.v)

We will modify the some parts in VendingMachine\_pla module, first from changing name to VendingMachine\_ucode.



2- 2 Verilog Coding of VendingMachine\_ucode

As you may see on code line 38, We will not modify VendingMachineData module.

Referring to Lab3, VendingMachineData and related modules are as following.



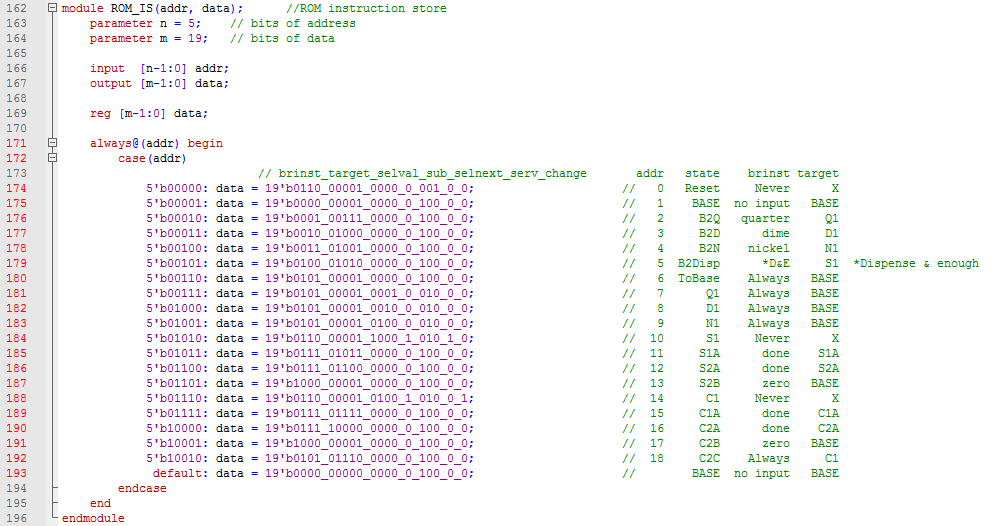
2- 3 Verilog Coding of VendingMachineData

Now, Verilog coding for VendingMachineControl\_ucode is as following.



2- 4 Verilog Coding of VendingMachineControl\_ucode

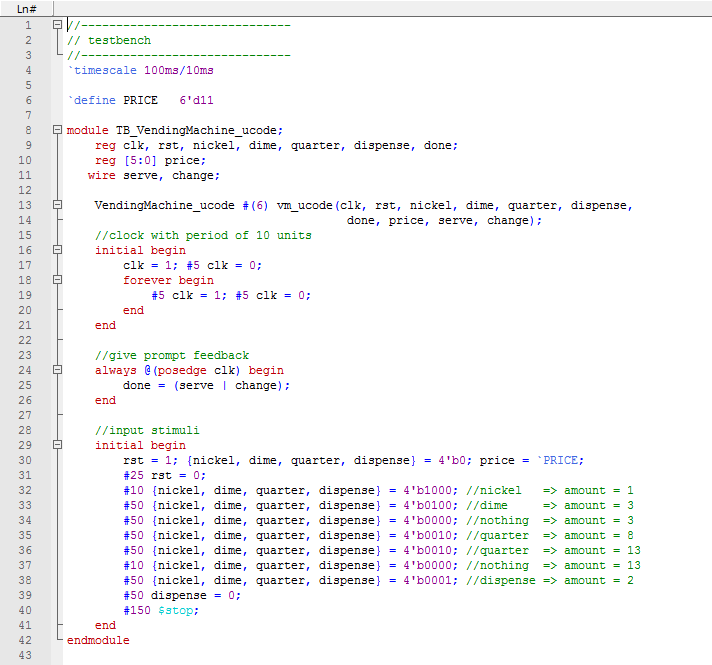
For ROM\_IS, we used the Verilog code given in pdf handout.



2- 5 Verilog Coding of ROM\_IS

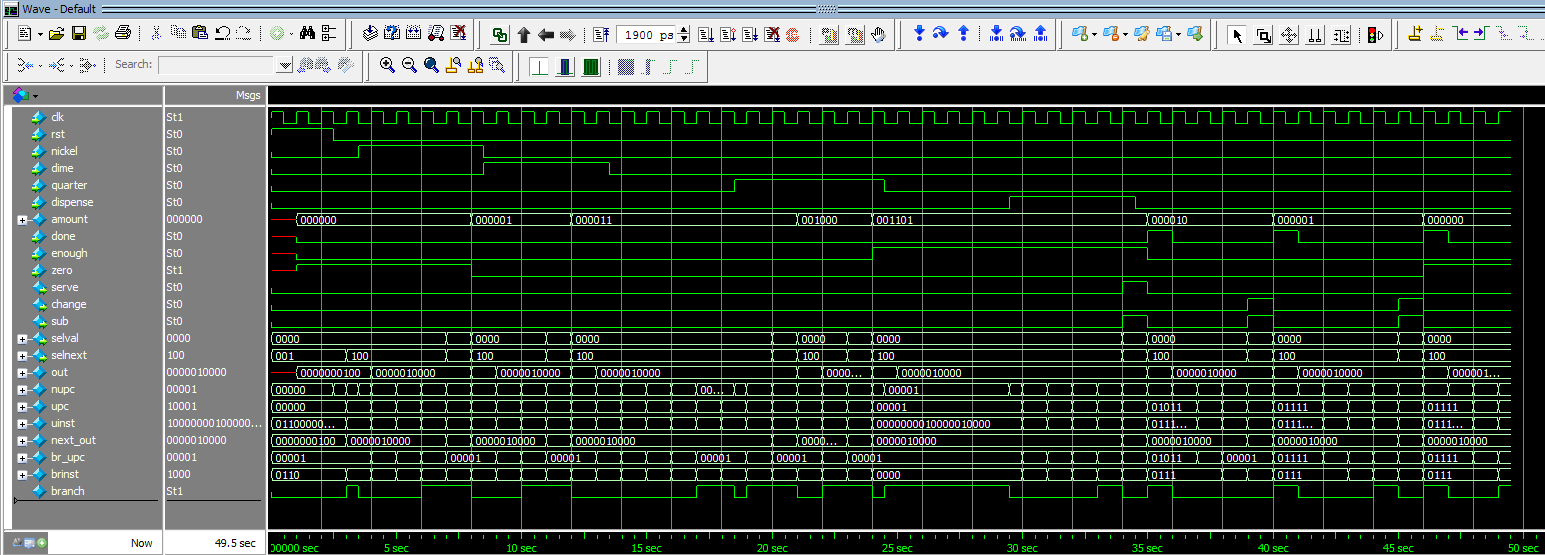
1. **Simulation of the DUT by including the following clock generation and stimuli in the testbench (TB: TB\_TLC\_Master\_Combiner\_v1.v)**

The Testbench code was given on the pdf handout, and is as following.



2- 6 Verilog coding of TB\_VendingMachine\_ucode

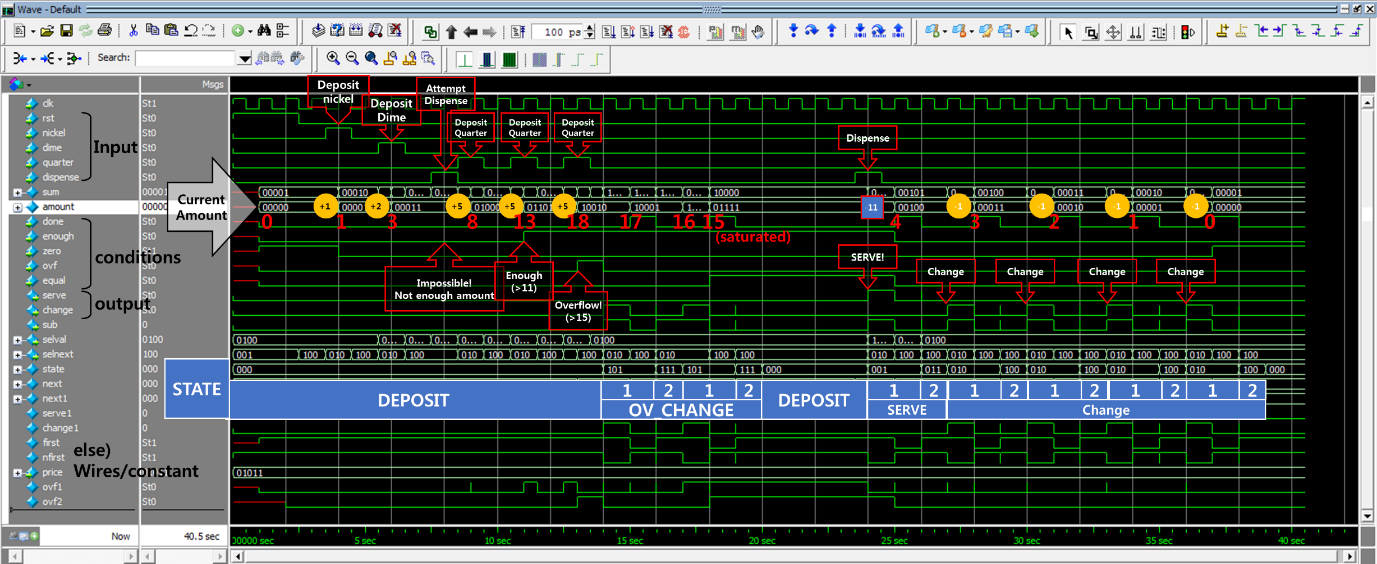
The Simulation result is as following.



2- 7 Simulation result of VendingMachine\_ucode

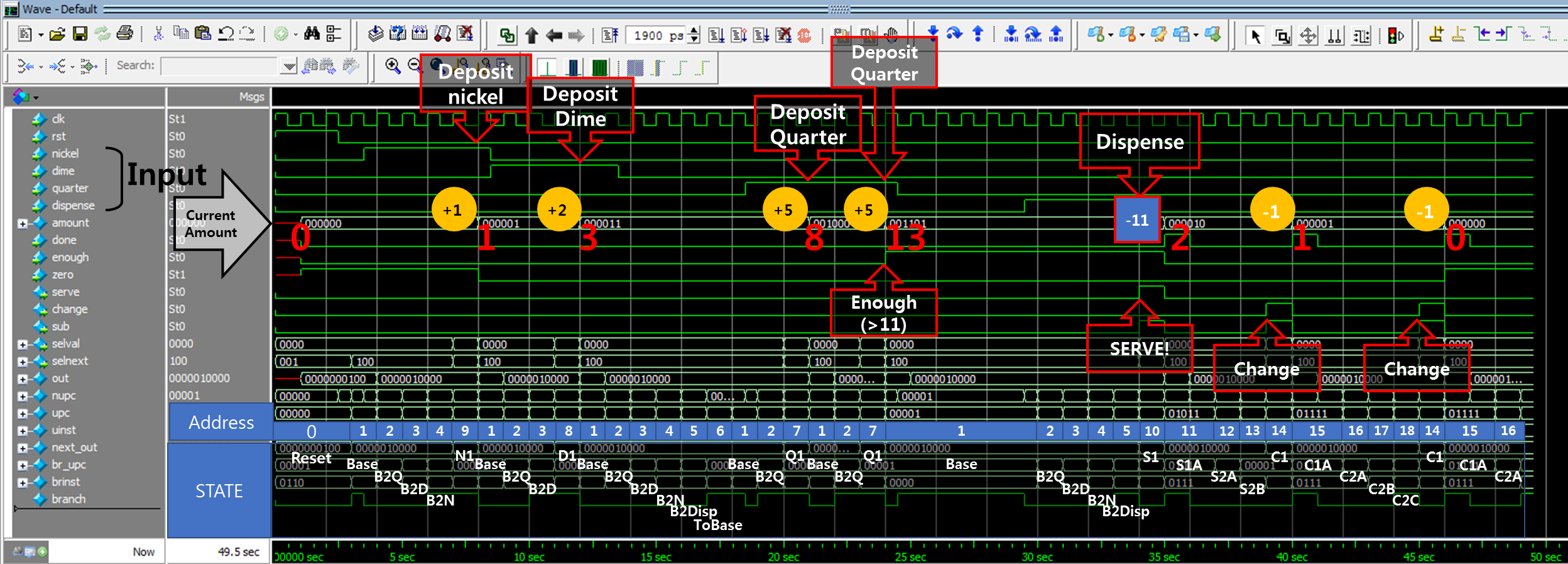
1. **Evaluation**

Recall evaluation of VendingMachine\_pla\_ovf, back in Lab3.



2- 8 Detailed evaluation on Simulation result of VendingMachine\_pla\_ovf

In similar principle, our result was examined with detail.



2- 9 Detailed evaluation on simulation result of VendingMachine\_ucode

Following the *upc*, the *amount* and output of the system (*serve, change)* were observed.

Evaluation showed that our simulation is valid.

End of Report.

조교님, 교수님 감사합니다! ☺