4C1 Integrated Systems Design

REPORT: Lab-2 Advanced Testbenches

-Pulkit Agarwal (19323939)

Introduction:

In this following Lab, I have tried to partition a testbench into stimulus generation and pass/fail assessment using Scoreboard. I have efficiently generated a stimulus using tasks and functions as well as developed a non-synthesizable Verilog model of desired behaviours.

My feature list includes a thorough test of module by considering all possible usage conditions whether expected or not. I also intend to use a log file to clearly state and report all the passed and failed testing conditions.

My test plan includes checking the FSM at their extreme conditions i.e. what happens when a car exit is triggered even when no cars are present. Similarly, what happens when a car tries to enter even though the parking is full.

In addition, I have also planned to test my parking task functions to check the desired output to the observed one. My tasks will include a 5-State-Parking and a 6-State-Parking. The 5-State-Parking has only one possible permutation to enter or exit a parking whereas the 6-State-Parking has 3 possible combinations to enter or exit the parking.

On all the stated above tasks, I will be using a random combination of these tests to check all the possibilities.

Errors (found or introduced):

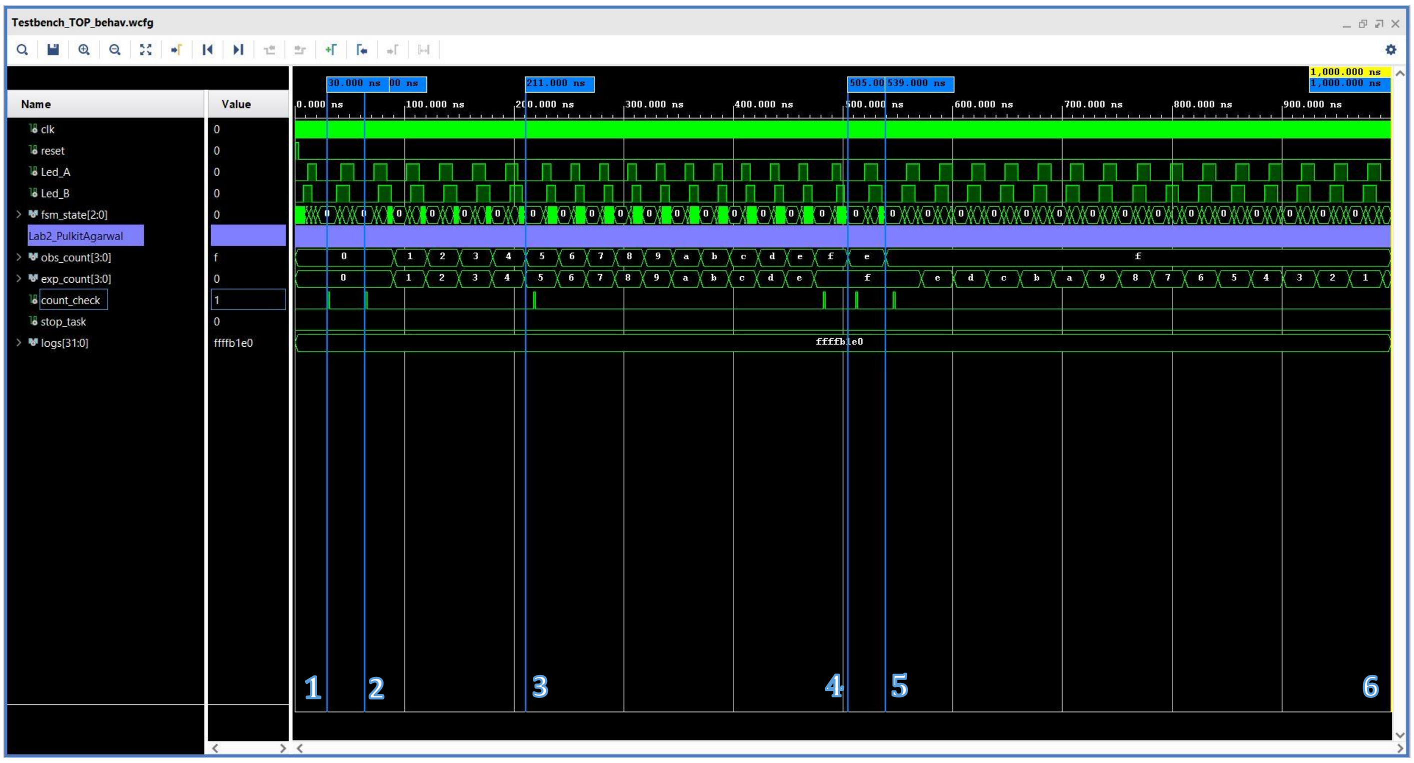
During my testing of the module, I introduced certain errors like checking the outputs at extremities to ensure that the conditions are working perfectly.

Although the 5-State and 6-State Parking functions are working perfectly for the lower limit i.e. 0, I found out that my 6-State-Parking function displayed a top limit of 14 rather than 15 as of that compared with the 5-State-Parking function.

It also came to my attention that my module is not working well decrementing the counter i.e. to say that the counter goes up when incrementing whereas stays constant while decrementing.

For both of the above found errors, I have tried various ways to debug these errors but unfortunately, I have not been successful in doing so.

The waveform below shows the intentional and observed errors during my testing:



Markers 1 & 2 display the intentional errors to check bottom limit for both 5-State and 6-State Parking respectively.

Marker 4 displays intentional error to check top limit for 6-State-Parking.

Marker 5 displays the observed error while trying to test the top limit for 5-State-Parking. The result was supposed to be 15 but rather is 14.

From Marker 5 onwards, the observed count was supposed to decrement till marker 6 but remains constant.

Marker 3 represents the switching of increment from 6-State-Parking to 5-State-Parking.

Below is also a copy of the logs generated during the simulation. The logs confirm the errors explained above in the waveform.

```
Check-1: Exiting when 0 Cars are present
5-State Parking
Check No.        1 ----PASSED
Observed Count:  0  Expected Count:  0
6-State-Parking
Check No.        2 ----PASSED
Observed Count:  0  Expected Count:  0
```

```
Check-2: Count from 0 -> 5 car entries using 6-STATE-PARKING
6-State-Parking
Check No.        3 ---- PASSED
Observed Count:  5
 Expected Count:  5
```

```
Check-3: Count from 5 -> 15 car entries using 5-STATE-PARKING
5-State Parking
Check No.        4 ---- PASSED
Observed Count: 15
 Expected Count: 15
```

```
Check-4: Count for more than 15 car entries 5-State-Parking
5-State Parking
Check No.        5 ----FAILED
Observed Count: 14
 Expected Count: 15
```

```
Check-5: Count for more than 15 car entries 6-State-Parking
6-State-Parking
Check No.        6 ---- PASSED
Observed Count: 15
 Expected Count: 15
```

```
Check-6: Count from 15 -> 10 car exits using 6-STATE-PARKING
6-State-Parking
Check No.        7 ---- FAILED
Observed Count: 15
 Expected Count:  0
```

```
Check-6: Count from 10 -> 0 car exits using 5-STATE-PARKING
6-State-Parking
Check No.        8 ---- FAILED
Observed Count: 15
 Expected Count:  0
```

```
Total Checks conducted:  8
Total Checks passed: 4
Total Checks failed: 4
```

Testing a complex FSM:

To test a complex FSM with large number of states, the initial step would be to alter the DUT by determining all the possible states that might occur. This would allow to compare the FSM with the stimulus to calculate expected and observed outcomes.

Furthermore, the test plan would be altered on a basis that all the test conditions are checked and verified. This could be done by determining the various combinations and introducing tasks to manage all the possible states. Individually checking the tasks in range and at extremities along with a combination of other tasks would allow to test a complex FSM successfully.

Comment on how a verification engineer might ensure the block is adequately tested.

To ensure that a block of code is adequately tested, a verification engineer must have a validation plan i.e. to say they need to plan all the activities beforehand that need to be included in the testing.

Secondly, they should define requirements i.e. to say that the goals and functionality of the FSM should be predefined. This would allow the engineers to set the requirements for the testing and come up with a validation plan to include all of them.

Lastly, the engineers should evaluate the FSM as per the specifications and prepare a validation report. This could include any bugs or missing functionalities found during evaluation.

All these methods would help a verification engineer ensure that the block is adequately tested.