

**COEN 6541: Functional Hardware Verification**

Verification Report for Calc1 & Calc2

Date of submission –April 24th, 2019

------------------------------------------------------------------

**SUBMITTED BY**

|  |  |  |
| --- | --- | --- |
| Sreekanth Reddy Akkala | Anil Kumar Bandi | Sandeep Chanda |
| (Student ID: 40070952) | (Student ID: 40036715) | (Student ID: 40064690) |

**SUBMITTED TO**

Prof. Dr. Otmane Ait Mohamed

# Contents

[Table of figures 2](#_bookmark0)

[Introduction 3](#_bookmark1)

[Design Specification of Calc 1 3](#_bookmark2)

[Design specification of Calc 2 4](#_bookmark4)

[Testbench used for Verification 5](#_bookmark6)

[Transaction 5](#_bookmark7)

[Generator 6](#_bookmark9)

[Interface 6](#_bookmark10)

[Driver 6](#_bookmark11)

[Monitor 6](#_bookmark12)

[Scoreboard and Checker 6](#_bookmark13)

[Environment 6](#_bookmark14)

[Test 6](#_bookmark15)

[Test Scenarios 7](#_bookmark16)

[Test case 1 7](#_bookmark17)

[Test case 2 8](#_bookmark20)

[Test case 3 9](#_bookmark23)

[Test case 4 10](#_bookmark26)

[Test case 5 11](#_bookmark29)

[Test case 6 12](#_bookmark32)

[Test case 7 13](#_bookmark35)

[Test case 8 14](#_bookmark38)

[Test case 9 15](#_bookmark41)

[Test case 10 16](#_bookmark44)

[Test case 11 17](#_bookmark47)

[Current progress on Calc2 17](#_bookmark49)

[Bug Report 19](#_bookmark52)

[Conclusion 19](#_bookmark53)

[References 20](#_bookmark54)

# Table of figures

[Figure 1 Calc1 input/output description 4](#_bookmark3)

[Figure 2 Calc2 input/output description 4](#_bookmark5)

[Figure 3 Testbench for the design 5](#_bookmark8)

[Figure 4 Transcript for four concurrent addition command 7](#_bookmark18)

[Figure 5 Output Waveform for four concurrent addition command 7](#_bookmark19)

[Figure 6 Transcript to show that no port has higher priority 8](#_bookmark21)

[Figure 7 Output Waveform to show that no port has higher priority 8](#_bookmark22)

[Figure 8 Transcript for higher order 27 bits are ignored testcase 9](#_bookmark24)

[Figure 9 Output Waveform for higher order 27 bits are ignored testcase 9](#_bookmark25)

[Figure 10 Transcript for testcase of overflow condition 10](#_bookmark27)

[Figure 11 Output Waveform for testcase of overflow condition 10](#_bookmark28)

[Figure 12 Transcript for addition of two numbers whose sum is FFFFFFFF 11](#_bookmark30)

[Figure 13 Output Waveform for for addition of two numbers whose sum is FFFFFFFF 11](#_bookmark31)

[Figure 14 Transcript for subtraction of two equal numbers 12](#_bookmark33)

[Figure 15 Output Waveform for subtraction of two equal numbers 12](#_bookmark34)

[Figure 16 Transcript for testcase of underflow condition 13](#_bookmark36)

[Figure 17 Output Waveform for testcase of underflow condition 13](#_bookmark37)

[Figure 18 Transcript for shift by 0 places testcase 14](#_bookmark39)

[Figure 19 Output Waveform for shift by 0 places testcase 14](#_bookmark40)

[Figure 20 Transcript for shift by 31 places testcase 15](#_bookmark42)

[Figure 21 Output Waveform for shift by 31 places testcase 15](#_bookmark43)

[Figure 22 Transcript for illegal commands 16](#_bookmark45)

[Figure 23 Output Waveform for illegal commands 16](#_bookmark46)

[Figure 24 Output Waveform for the reset condition 17](#_bookmark48)

[Figure 25 Transcript for Calc2 18](#_bookmark50)

[Figure 26 Output Waveform for Calc2 19](#_bookmark51)

Introduction

The main objective of this project is to write a verification functional test plan based on a design specification, creating a verification environment for the given design using the test plan so that it can be reusable for multiple designs and to find errors in the design. In this project, Calc 1, Calc 2and Calc 3 designs are verified using different verification tests in the same environme nt that is created. Different test scenarios were tested and detailed description of the results are provided. System Verilog is used to code the project and is performed on Questasim.

Design Specification of Calc 1

Calc 1 design specifications include handling 4 requests having equal priority in parallel. It can perform 4 functions Add, Subtract, Shift left and Shift Right. Each port must wait for its response before sending in the next command. Works on FCFS algorithm and allows only 1 add/subtract and 1 left/right shift operation to perform at a time. Should use only the lower order 5 bits of the second operand for shift operation and arithmetic operations are unsigned.

## Pin Description:

**c\_clk:** Gives the clock to drive the input command, data/get the output data, response.

**reqX\_cmd\_in<0:3>:** It is the command bit that defines the operation to be performed with the data provided where

**‘0’** specifies **No Operation**

**‘1’** specifies **addition** of Operand 1 and Operand 2

**‘2’** specifies **subtraction** of Operand 1 and Operand 2 **‘5’** specifies **Shift left** Operand 1 by Operand 2 places **‘6’** specifies **Shift right** Operand 1 by Operand 2 places

**reqX\_data\_in<0:31>:** Operand 1 data arrives with a command bit and Operand 2 arrives on the following cycle.

**reset<0:7>:** Resets the data to ‘1111111’b at the start of test case for seven cycles. Outputs are ignored during this period.

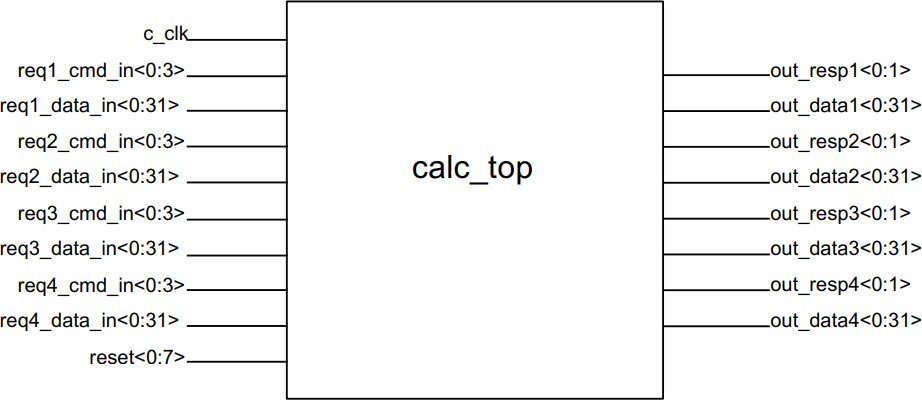
**out\_respX<0:1>:** 2-bit output response bits defines the operation performed at the output where

**‘0’** specifies **No response**

**‘1’** specifies **successful operation completion**

**‘2’** specifies **invalid command or overflow/underflow error ‘1’** specifies **internal error**

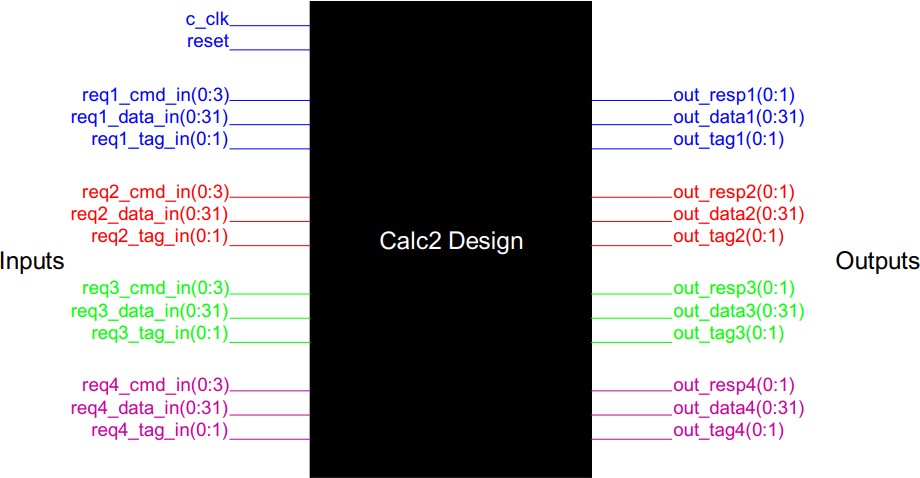
**out\_dataX<0:31>:** Gives corresponding valid data accompanied by response.



*Figure 1 Calc1 input/output description*

Design specification of Calc 2

Calc 2 design specifications are similar to that of Calc 1 except each port can handle up to 4 operations at a time. So, a total of 16 commands are handled at a time. Each command from a port is sent one at a time, but the calc log may respond “out of order” depending upon the interna l state of the queues. A 2-bit tag is accompanied with the command provided so as to track the corresponding data at the output.



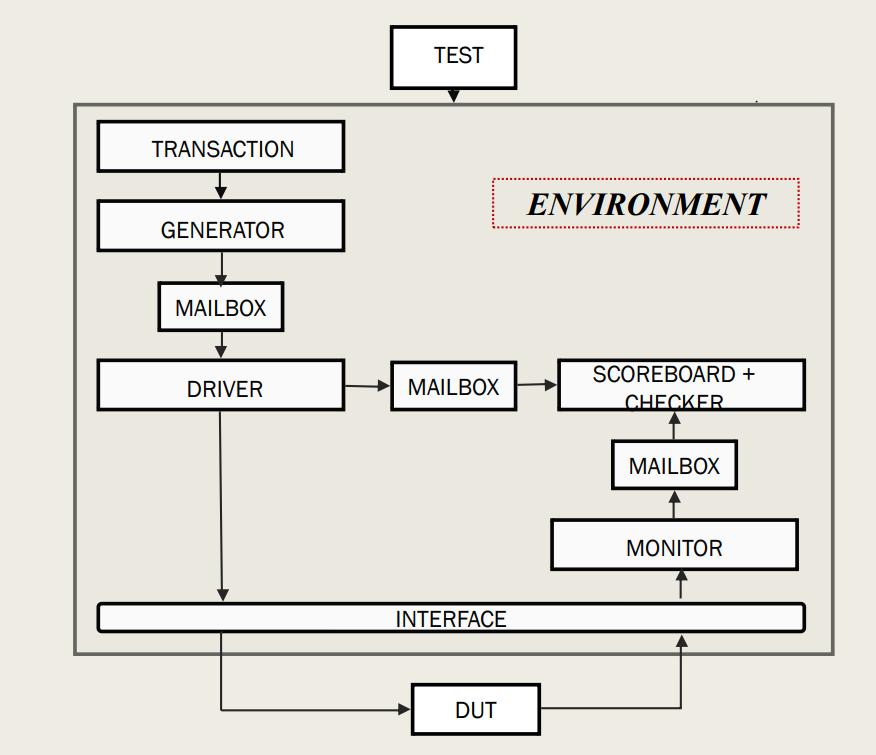
*Figure 2 Calc2 input/output description*

## Pin Description:

**reqX\_tag\_in(0:1):** The tag bus is the two-bit identifier for each command from the port. The port requestor can reuse the tag as soon as the Calc2 HDL responds to the command.

**out\_tagX(0:1):** The output tag bus corresponds to the command tag sent by the requester. It is used to identify which command the response if for.

Testbench used for Verification



Transaction

*Figure 3 Testbench for the design*

The transaction class that we created generates the randomized stimulus. The stimulus can be constrained to get the required values/range. Command and tag bits are also generated in this class respectively for all the ports. The expected tag bits, captured tag bits, output data bits, expected data bits, output response bits, expected response bits, are also declared in this class. Copying current random values for constraints not to repeat the values are also done in this class.

Generator

Generator class that we created randomizes the transaction class that is coded to create multiple input stimulus to be driven to DUT via driver class. This class adds a control to the loop,

* 1. ., the number of iterations done by the loop is controlled by declaring a variable. Also displays whether the randomization succeeded or not. Mailbox, gen2drive is generated in this class to send the randomized stimulus from generator class to the driver. An event is also declared to note the completion of transactions.

Interface

Multiple signals in a device are placed in a single band called interface for easy handling, reusability of the input-output signals. In the interface class, all the input-output signals are declared. Clocking blocks for the monitor and the test class are also declared at the posedge of the clock in this class. Also, the direction of the signals for monitor, DUT and test are declared in the modport in this class.

Driver

The driver class creates a mailbox driv2scb in addition to the mailbox gen2driv. A custom constructor is created in order to pass the values of the mailbox. The instance of the virtual interface is declared in this block. Timing information is defined within the task provided in the class for all the 4 ports. Reset is declared in this class. Expected data, expected response and expected tag bits are calculated in this class. The randomized stimulus is driven to the DUT.

Monitor

The monitor class creates a new virtual interface intf2 and mailbox mon2scb. A custom constructor is used to pass the values of the interface and mailbox to the scoreboard. Output data, Output response and output tag bits are captured in this class for all the 4 ports from the DUT. All the tasks to capture the data, response and tags are parallel.

Scoreboard and Checker

Scoreboard gets the expected data, response and tag bits from the driv2scb mailbox using the get() function and stores in it. It also gets the captured data from the DUT through mon2scb mailbox and compares them. It displays the error message if there is an overflow, underflow, equal and inequal output values.

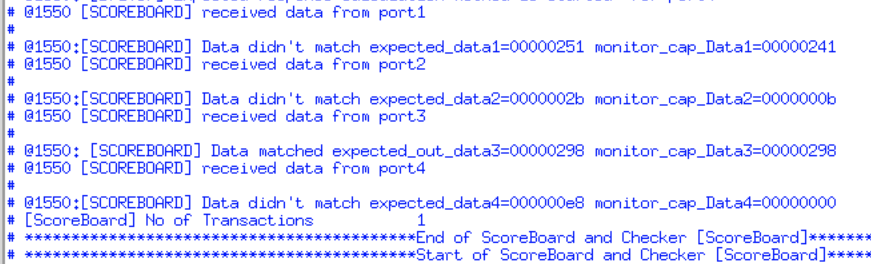
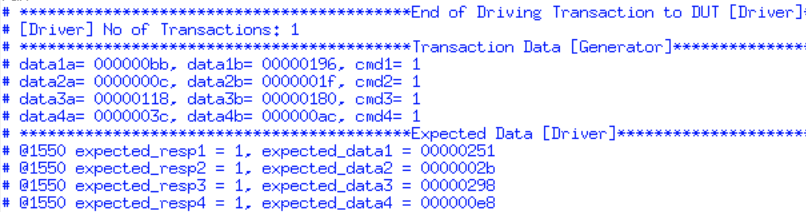
Environment

Instances of all the classes are created and components are connected in this class. Generator, Scoreboard and driver iterations are done in this and displayed. All the components are run in parallel in this class using a fork-join.

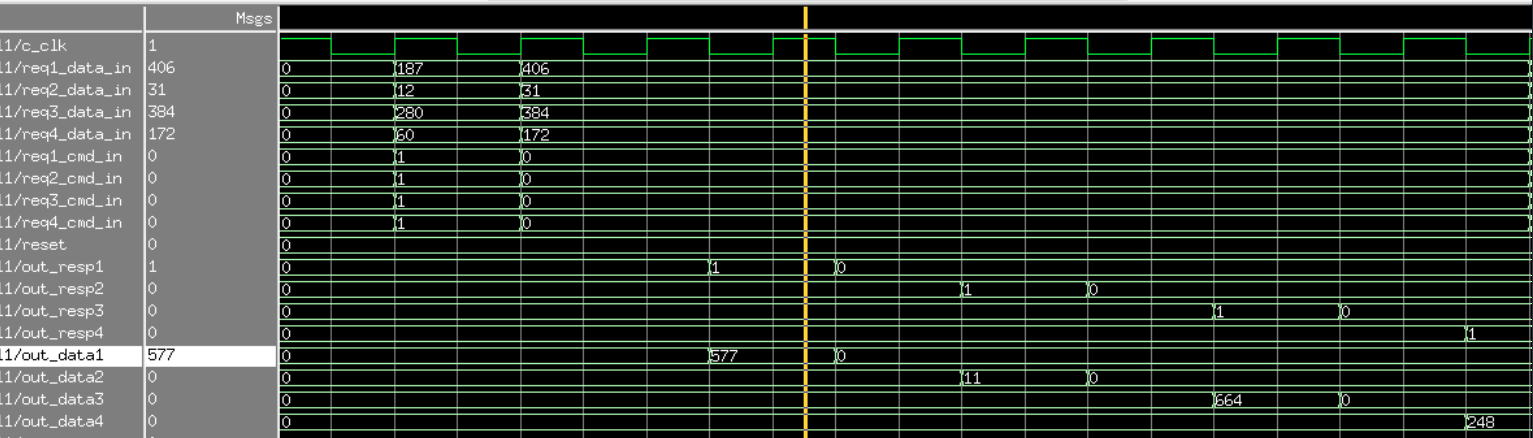
Test

Test class instantiates a copy of the environment class and uses it to control all the components of the design. Environment class is invoked in this class. Number of times the loop has to be iterated is declared in this class.

Test Scenarios: Test case 1

Across all ports (e.g**., four concurrent adds** do not interfere with each other), check that each command can have any command follow it without leaving the state of the design dirty, such that the following command is corrupted.

*Figure 4 Transcript for four concurrent addition command*

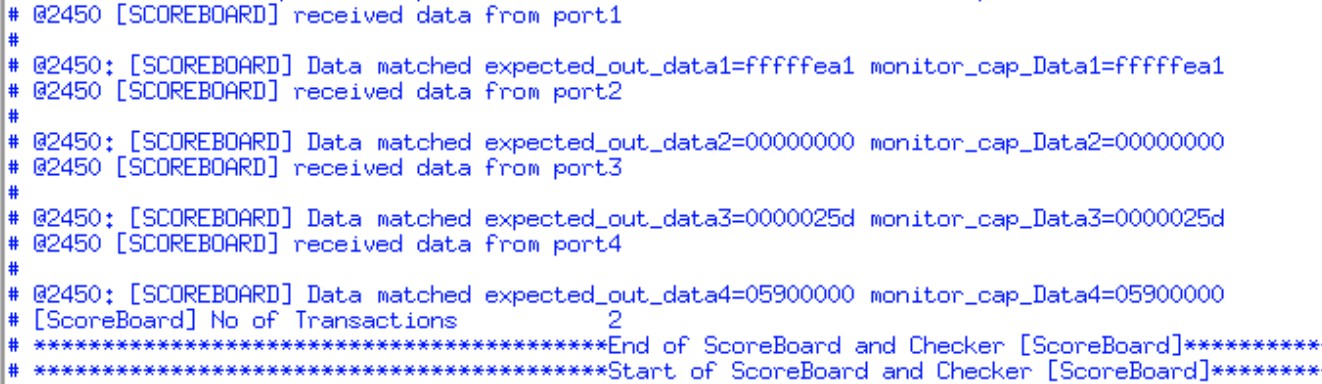
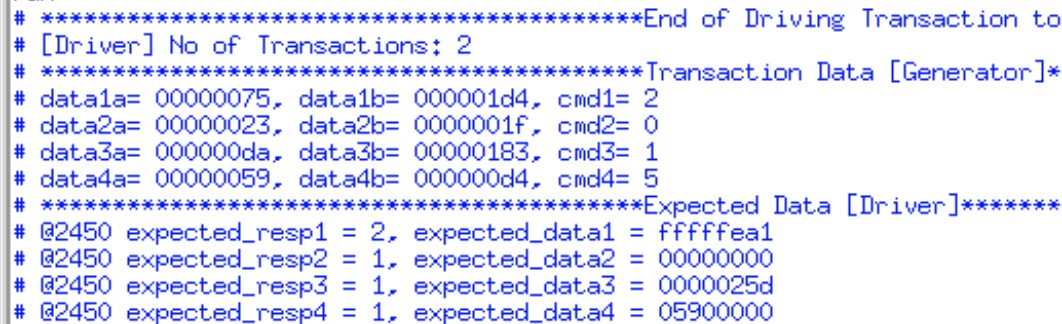


*Figure 5 Output Waveform for four concurrent addition command*

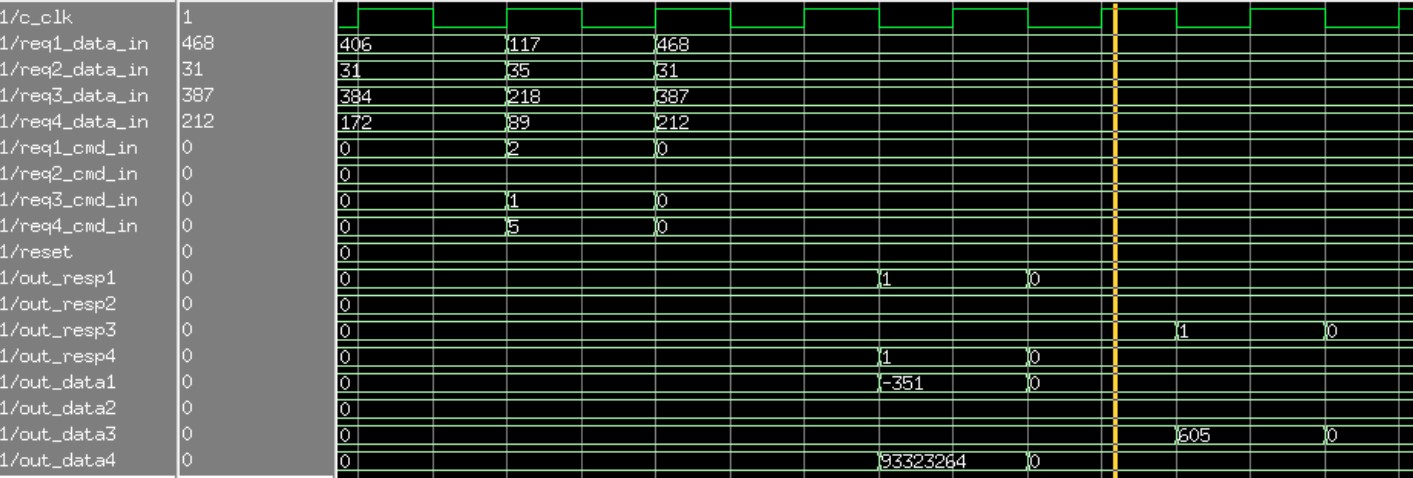
## Explanation & Observation:

Values for both the operands are randomly generated and added simultaneously. When four concurrent additions are given, we’re getting unexpected outputs for 3 of the ports. There is an error in addition datapath.

Test case 2

Check that there is fairness across all four ports such that **no port has higher priority** than the others.

*Figure 6 Transcript to show that no port has higher priority*

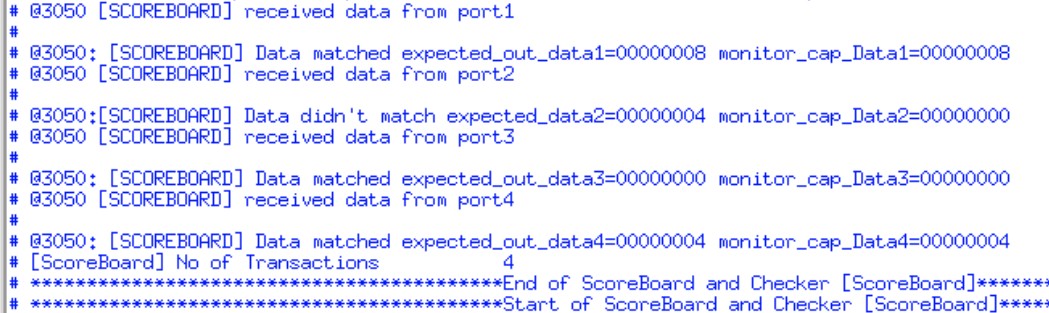
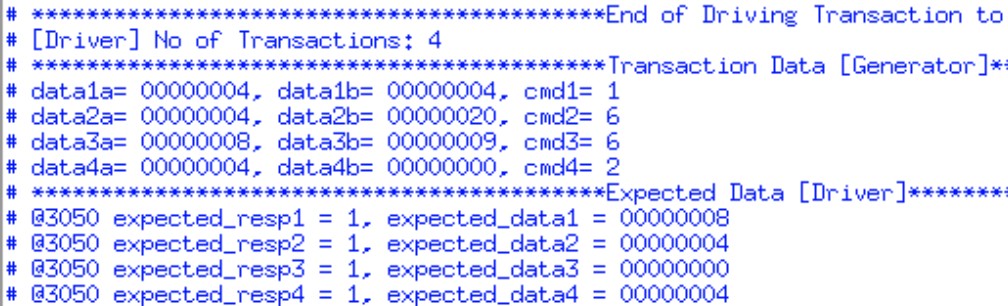


*Figure 7 Output Waveform to show that no port has higher priority*

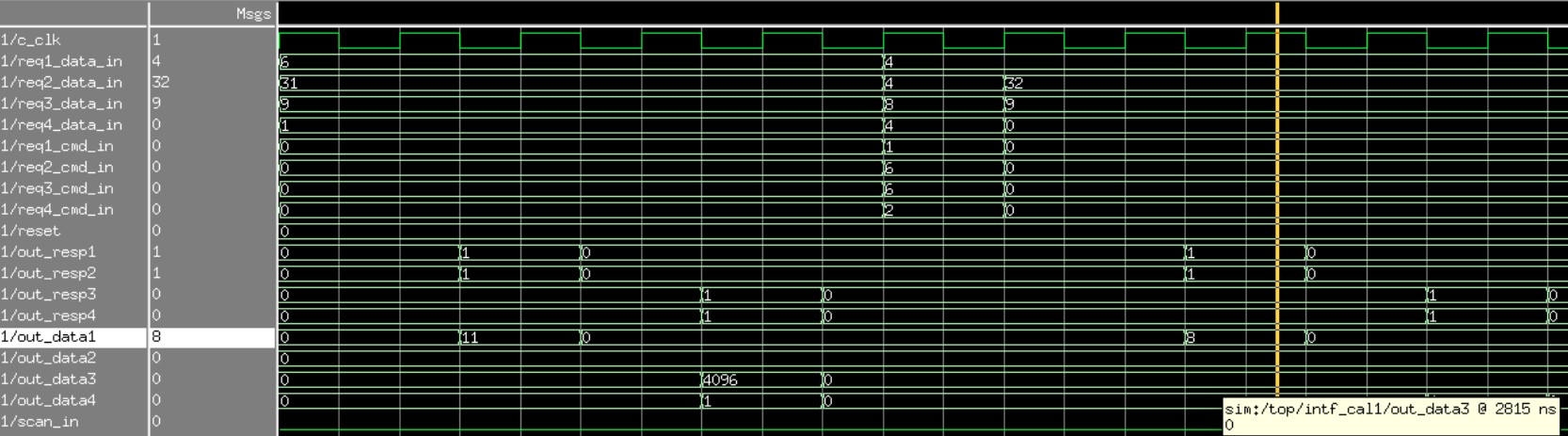
## Explanation & Observation:

Random data is generated and multiple commands are given simultaneously. It is observed that there is no priority among the four ports – All ports have equal priority.

Test case 3

Check that the **high-order 27 bits are ignored** in the second operand of both shift commands

*Figure 8 Transcript for higher order 27 bits are ignored testcase*



*Figure 9 Output Waveform for higher order 27 bits are ignored testcase*

## Explanation & Observation:

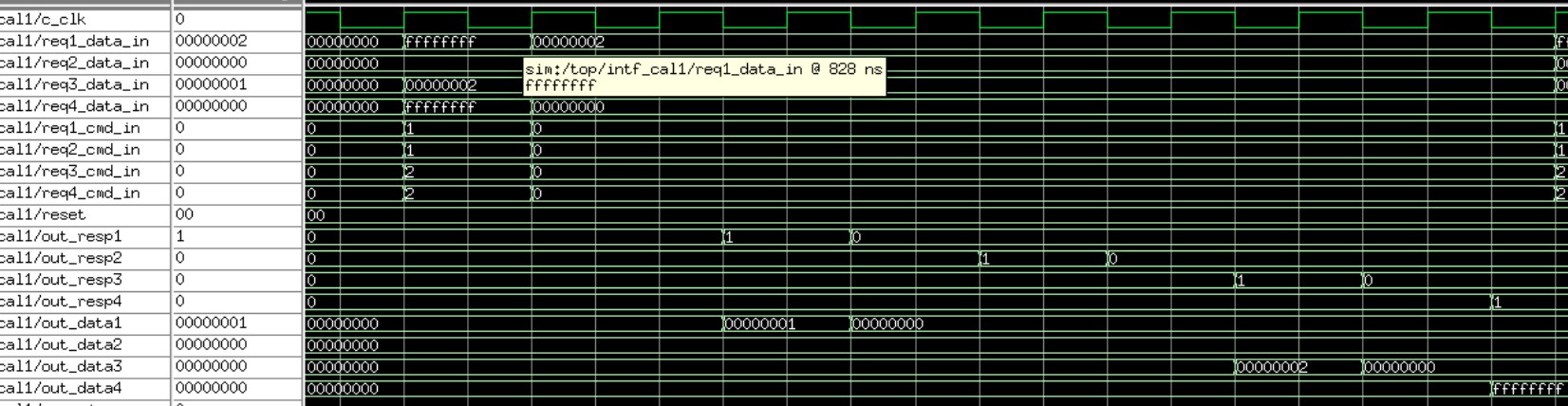
Higher order 27 bits are not ignored in the second operand. In port 2, the expected data is 4, but the resultant data is 0. We found a bug in this case and is explained in the bug report.

Test case 4

Data dependent corner case: **Add two numbers that overflow by 2(“FFFFFFFF”X+2)**



*Figure 10 Transcript for testcase of overflow condition*



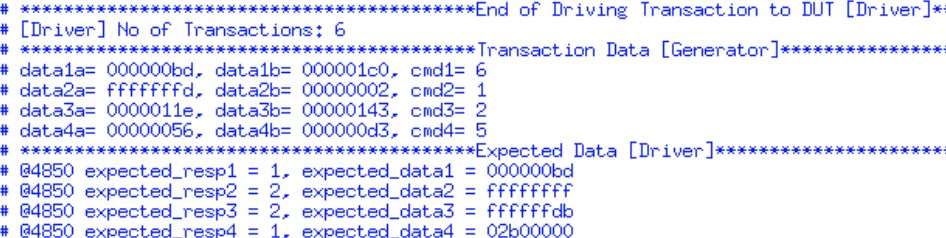
*Figure 11 Output Waveform for testcase of overflow condition*

## Explanation & Observation:

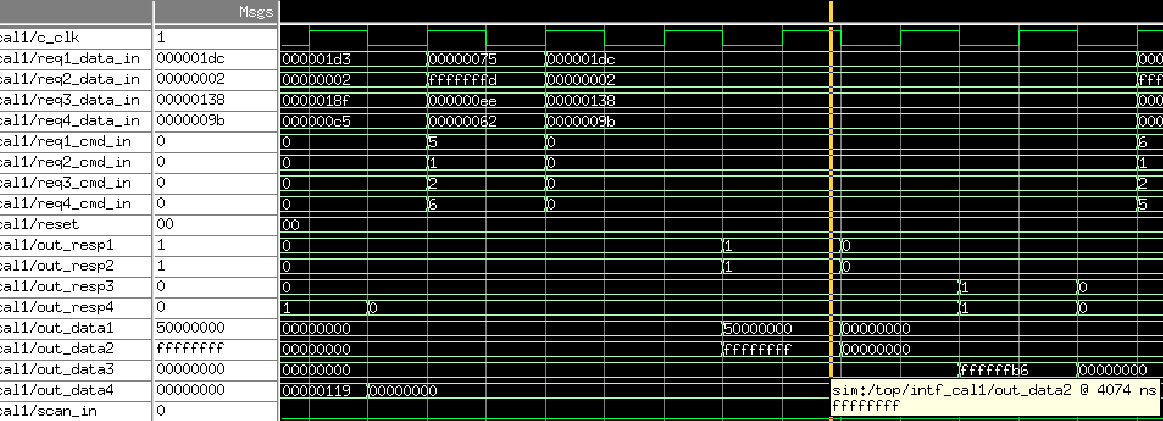
We observed that the two numbers are overflown by 2 – Overflow operation is working as expected but the response is 1 instead of 2.

Test case 5

Data dependent corner case: **Add two numbers whose sum is “FFFFFFFF”X**



*Figure 12 Transcript for addition of two numbers whose sum is FFFFFFFF*



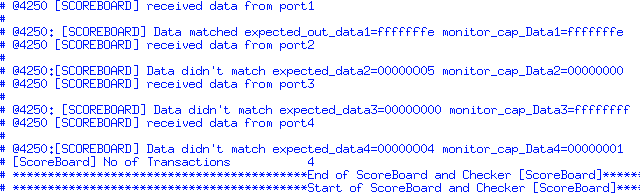
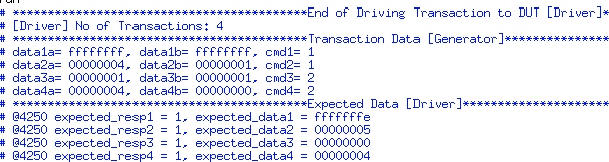
*Figure 13 Output Waveform for for addition of two numbers whose sum is FFFFFFFF*

## Explanation & Observation:

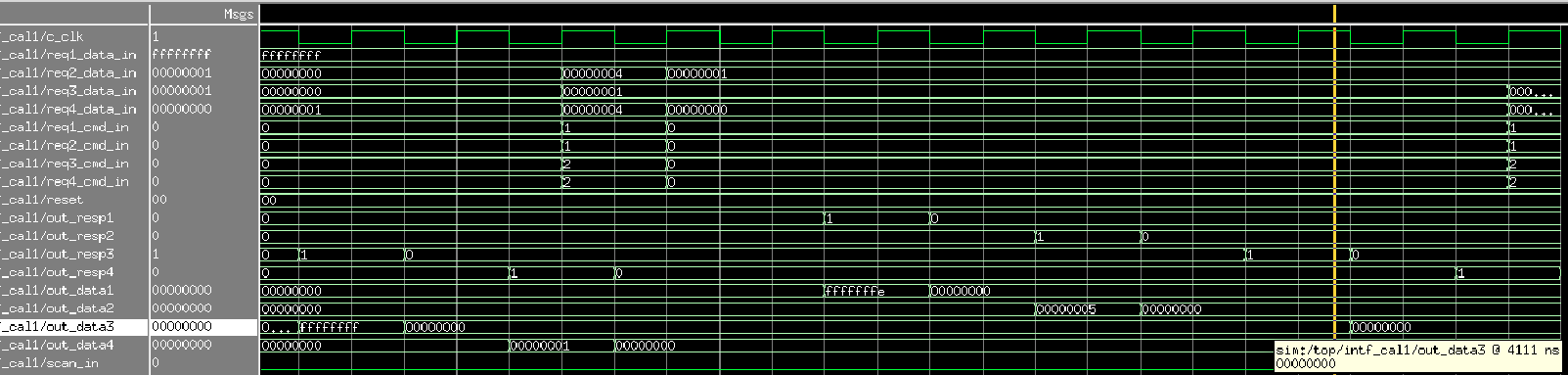
We observed that the Added two numbers – (FFFFFFFD + 00000002) gives the expected result (FFFFFFFF)

Test case 6

Data dependent corner case: **Subtract two equal numbers**



*Figure 14 Transcript for subtraction of two equal numbers*



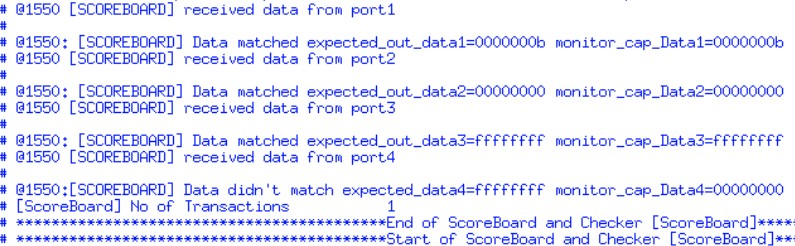
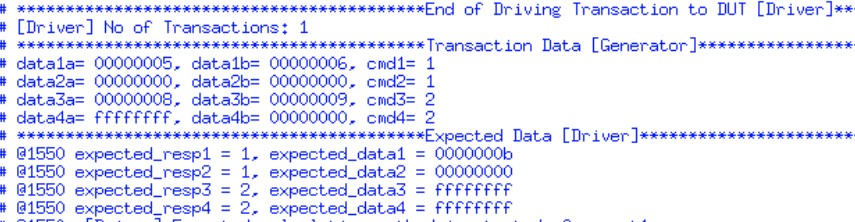
*Figure 15 Output Waveform for subtraction of two equal numbers*

## Explanation & Observation:

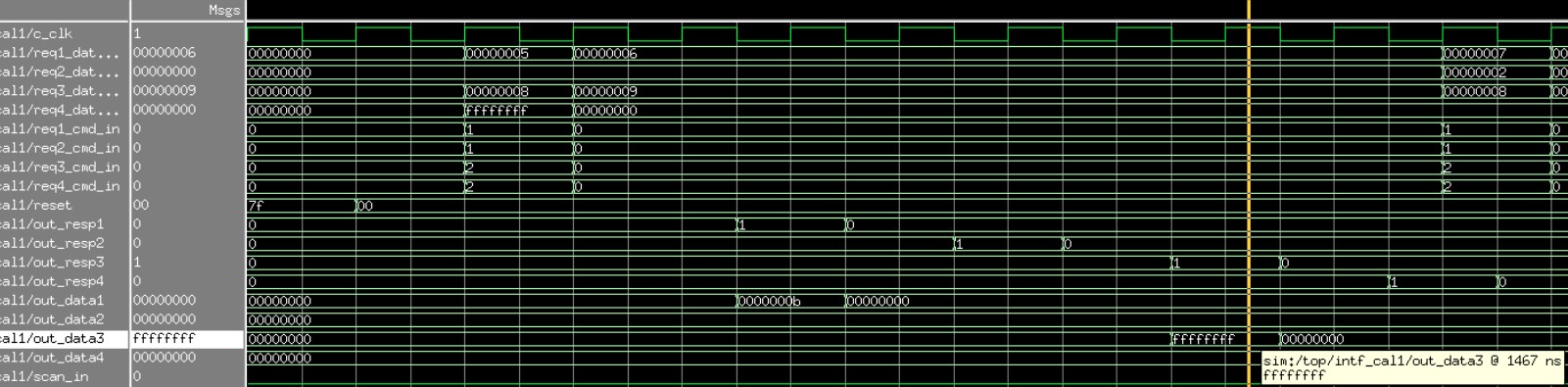
We have observed that the transcript isn’t producing the expected results but the output waveform is giving the required results.

Test case 7

Data dependent corner case: Subtract a number that underflows by 1



*Figure 16 Transcript for testcase of underflow condition*



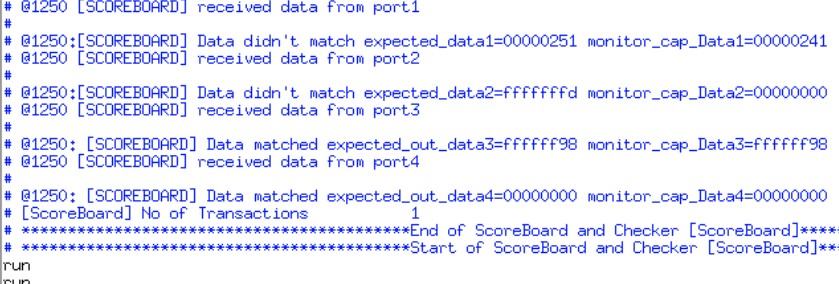
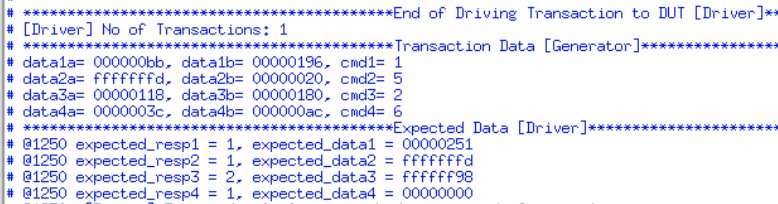
*Figure 17 Output Waveform for testcase of underflow condition*

## Explanation & Observation:

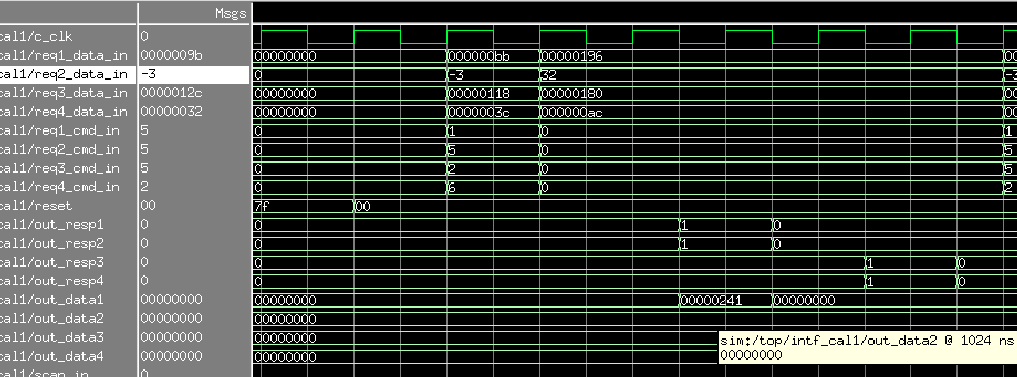
We observed that the subtracted number underflows by 1(00000008- 00000009=FFFFFFFF) as expected but we have found the error in response bits which is explained in the bug report in detail.

Test case 8

Data dependent corner case: **Shift 0 places**



*Figure 18 Transcript for shift by 0 places testcase*



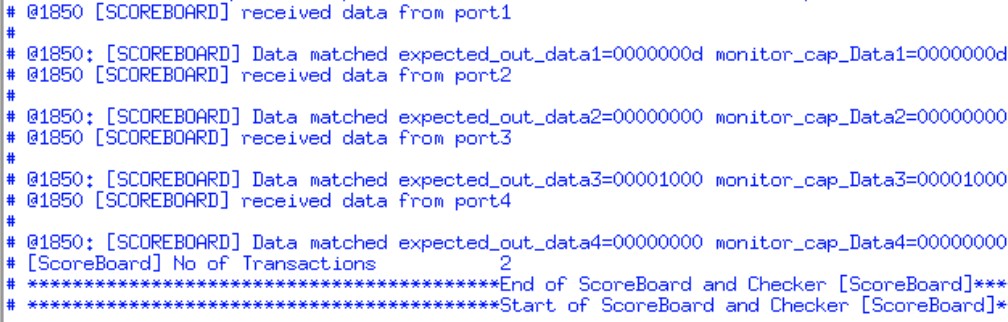
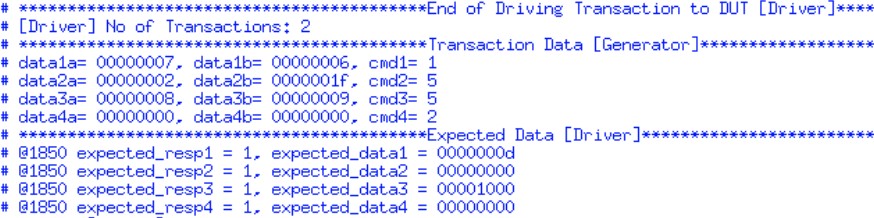
*Figure 19 Output Waveform for shift by 0 places testcase*

## Explanation & Observation:

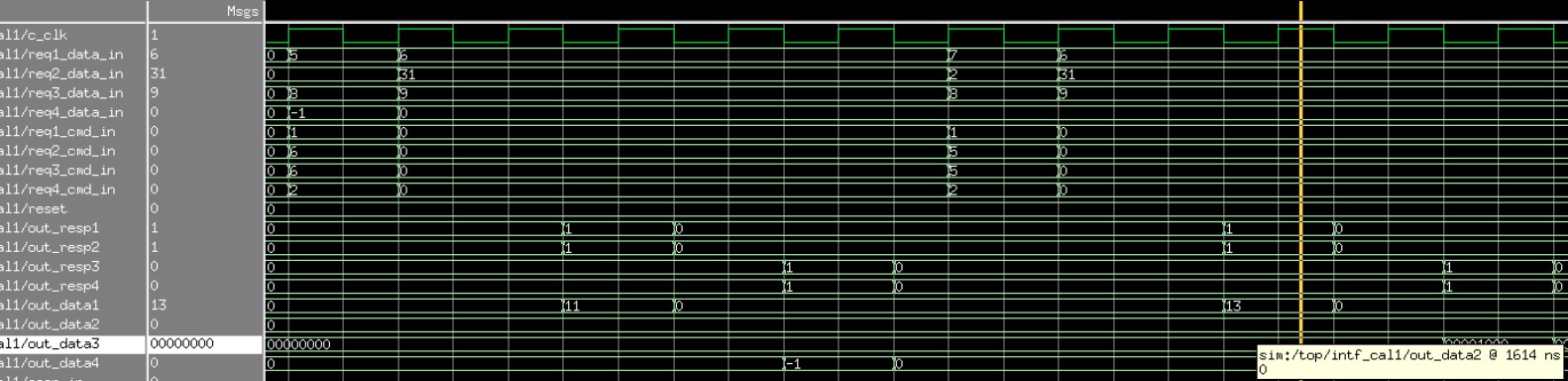
fffffffd if shifted by 0 places, expected data should be same, but the resultant data is 00000000. The bug found is explained in the bug report in detail.

Test case 9

Data dependent corner case: **Left Shift 31 places**



*Figure 20 Transcript for shift by 31 places testcase*



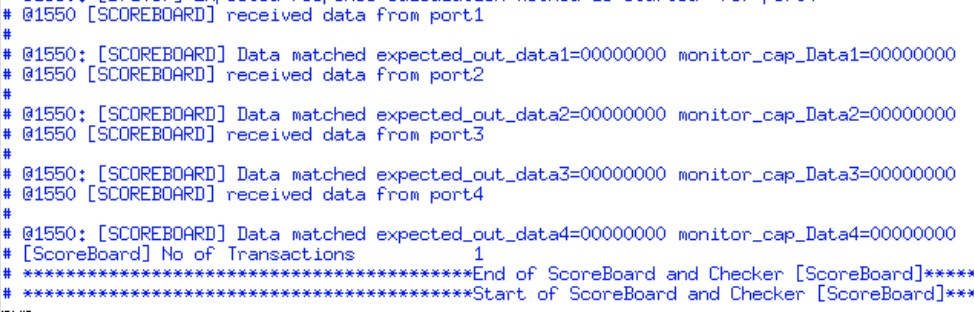
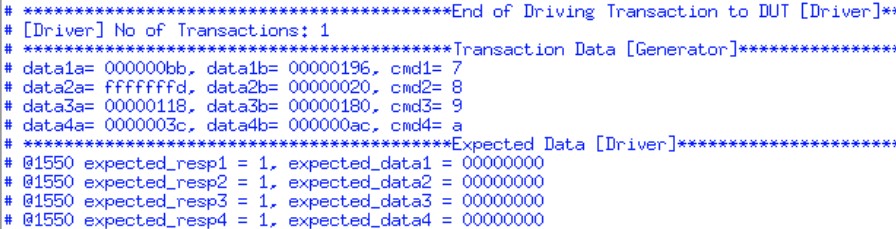
*Figure 21 Output Waveform for shift by 31 places testcase*

## Explanation & Observation:

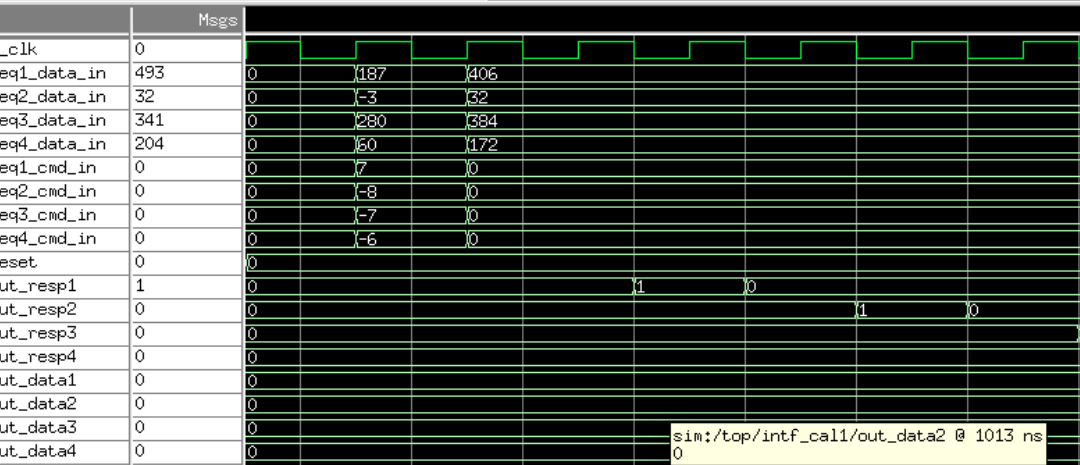
We have observed that the data is left shifted by 31 places as expected.

Test case 10

Check that the design correctly handles **illegal commands**.



*Figure 22 Transcript for illegal commands*



*Figure 23 Output Waveform for illegal commands*

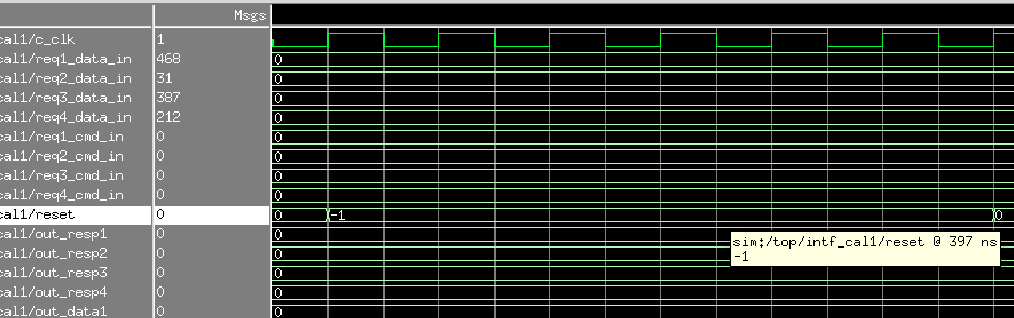
## Explanation & Observation:

For illegal commands, obtained result is **0** as expected, but the response is 0 instead of 2.

Which is explained in the bug report.

Test case 11

Check that the **reset function** correctly resets the design.

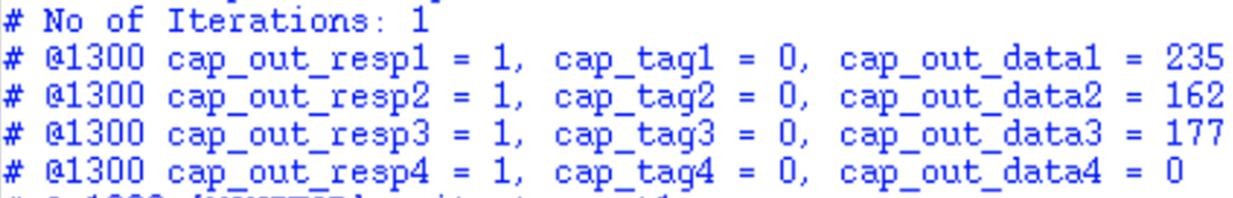
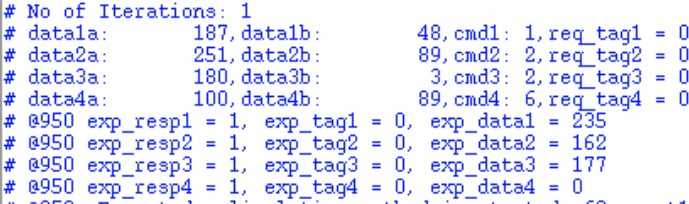


*Figure 24 Output Waveform for the reset condition*

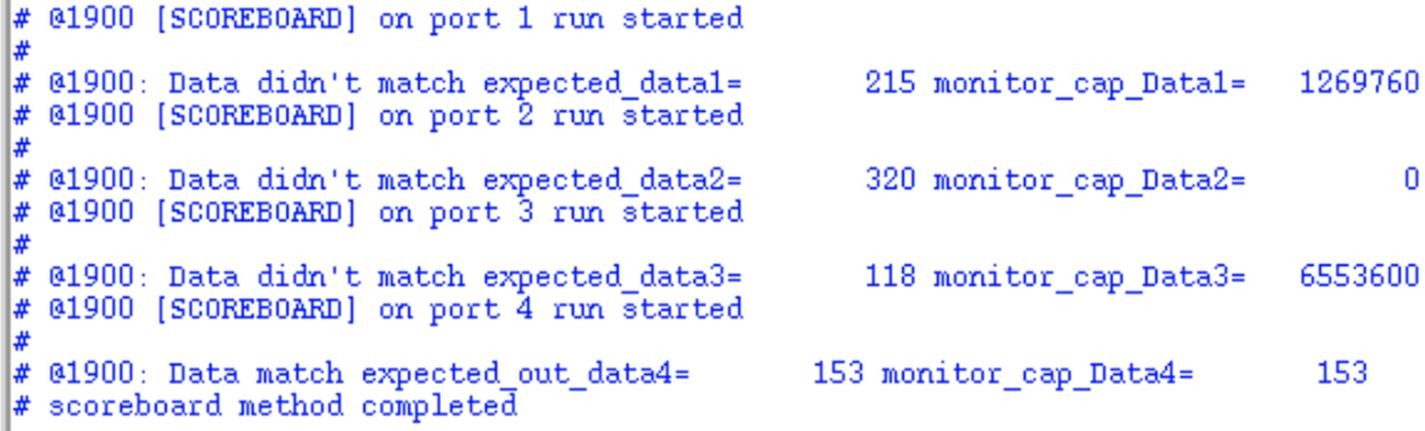
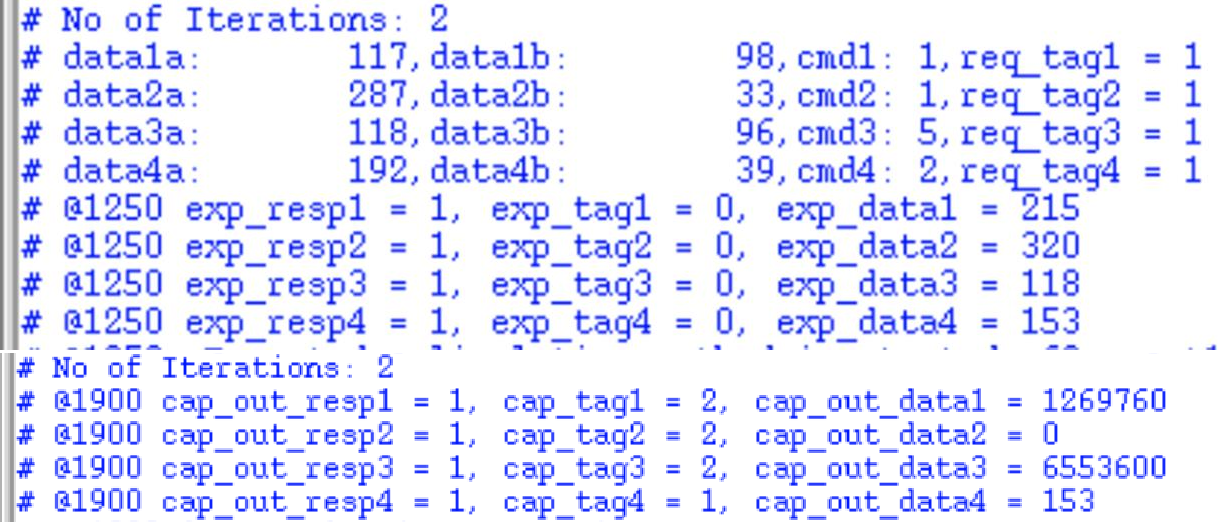
## Explanation & Observation:

We have observed that the reset function holds high for 7 cycles.

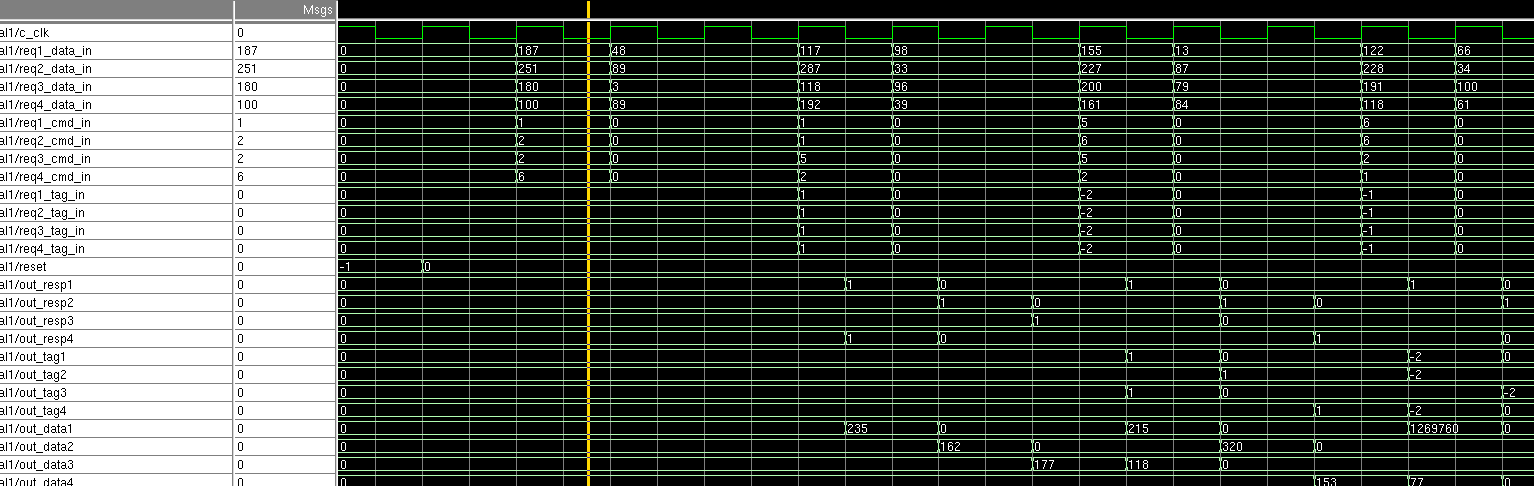
Current progress on Calc2

We have given tag bits along with random input stimulus. The captured data and captured tag bits are matching for the first iteration and it is not matching from the second iteration.





*Figure 25 Transcript for Calc2*



*Figure 26 Output Waveform for Calc2*

Bug Report

|  |  |  |
| --- | --- | --- |
| **BUG** | **TITLE** | **EXPLANATION** |
| 1 | Higher order 27 bits are not  ignored | In the test case 3, the 2nd operand is 20(Decimal – 32,  Binary -100000), which has ‘00000’ as the first 5 bits, meaning the output is supposed to shift 0 places. But, the code is considering 6th bit as well (100000) for  decimal value 32. It is working fine until decimal 31. |
| 2 | Overflow operation | Test 4 show that the overflow operation produces the  error in the response. The response is 1, where as the expected is 2. |
| 3 | Underflow operation | Test 7 shows that the underflow operation produces the  error in the response. The response is 1 instead of 2. |
| 4 | Shift by 0 places for 32 | Test 8 demonstrates that the shift by 0 places command  is producing the error because DUT is considering some random operands. Similar to Bug1 |
| 5 | Illegal commands | Test 10 shows that the given the illegal commands the  output response as 1 which should be 2. |

Conclusion

The environment is created for the given specifications of Calc1 and Calc2. Test bench and test plan are also specified as per the requirements. A detailed bug report is also provided for Calc1.Though we haven’t fu**l** y completed Calc2, we are getting the correct outputs except for the synchronization errors. This project has helped us in learning how an industry design can be verified in real-time.

References

* + - Lecture Notes
    - <https://www.verificationguide.com/p/home.html>
    - <https://www.chipverify.com/systemverilog/systemverilog-simple-testbench>
    - Online Resources