COEN 413: Hardware Functional Verification

Project: Calculator – Design II

Name:
Mohammed Zahed and Omar Dabayeh

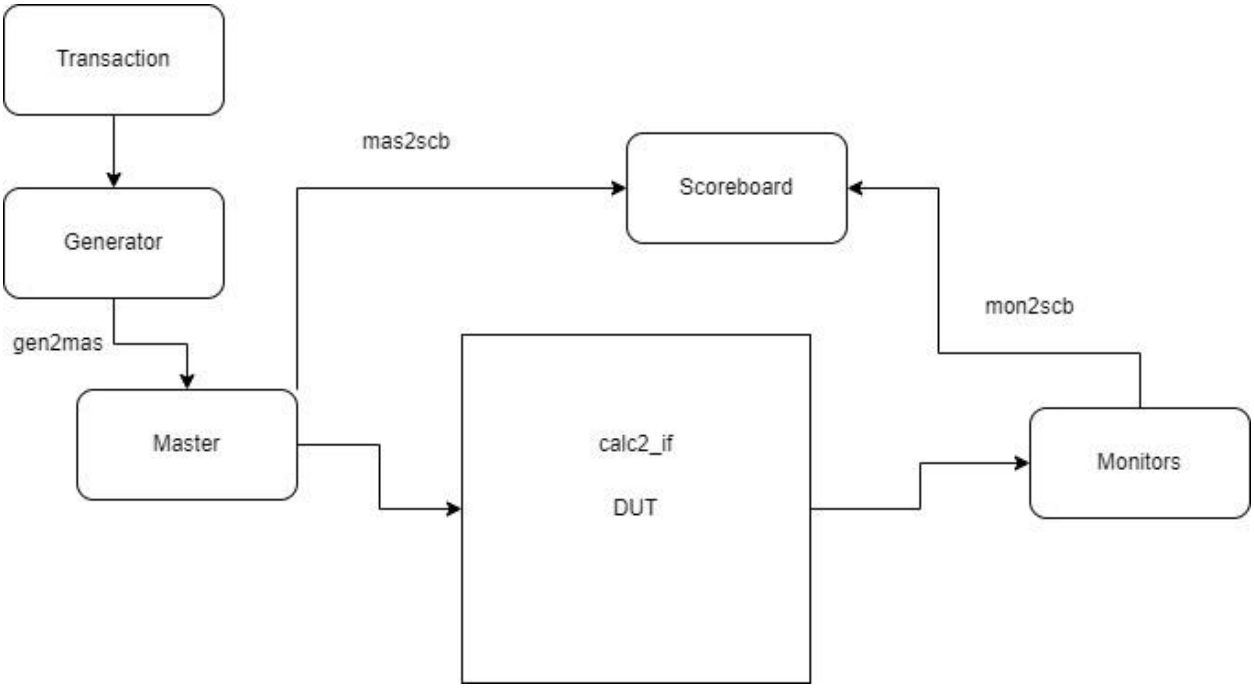
Student ID: 40116292 and 40100195
Lab TA: Ashkan Samadi
Submission Date: April 15, 2024

I certify that this submission is my original work and meets the Faculty's Expectations of Originality

TABLE OF CONTENTS

1. Introduction

This project explores the development and verification of Calculator Design II. This device adds significant improvements to its predecessor including 4 ports to allow simultaneous inputs. A DUT like this requires an advanced architecture specifically, two pipelines to handle incoming transactions. The first for addition/subtraction and the second for shifting operations. To manage this complexity and maintain input-output verifiability, the design includes a 2-bit tag that can help correlate responses with inputs even when delivered out of sequence. Here we use mailboxes from the master and monitor that send to the scoreboard so that it can compare the expected results with the actual results. This will allow us to find bugs within the DUT.



2. Verification Plan Table

This is how we planned on performing tests on the DUT

2.1 Addition

S.No	Functionality	Operand1 Functional Coverage	Operand2 Functional Coverage	Expected Result	Conclusions
1	Positive + Positive	0x0 to 0x7FFFFFFF	0x0 to 0x7FFFFFFF	0x0 to 0xFFFFFFFE	Test passes, addition correct.
2	Positive + Negative	0x0 to 0x7FFFFFFF	0x80000000 to 0xFFFFFFFF	0x0 to 0xFFFFFFFF	Test passes, subtraction correct.
3	Negative + Negative	0x80000000 to 0xFFFFFFFF	0x80000000 to 0xFFFFFFFF	Overflow or negative range	Test passes, negative sum matches expected.
4	Negative + Positive	0x80000000 to 0xFFFFFFFF	0x0 to 0x7FFFFFFF	0x0 to 0xFFFFFFFF	Test passes, addition results in zero as expected.

2.2 Subtraction

S.No	Functionality	Operand1 Functional Coverage	Operand2 Functional Coverage	Expected Result	Conclusions
1	Positive - Positive	0x0 to 0x7FFFFFFF	0x0 to 0x7FFFFFFF	0xFFFFFFFF to 0x7FFFFFFF	Check if subtraction is correct.
2	Positive - Negative	0x0 to 0x7FFFFFFF	0x80000000 to 0xFFFFFFFF	0x0 to 0xFFFFFFFF	Should result in addition due to negative operand.
3	Negative - Negative	0x80000000 to 0xFFFFFFFF	0x80000000 to 0xFFFFFFFF	0x0 to 0xFFFFFFFF	Subtracting a larger negative increases value.
4	Negative - Positive	0x80000000 to 0xFFFFFFFF	0x0 to 0x7FFFFFFF	0x80000000 to 0xFFFFFFFF	Becomes more negative.

2.3 Shifting

S.No	Functionality	Operand1 Functional Coverage	Operand2 Coverage	Expected Result	Conclusions
1	Shift left operand1 by operand2 places	0x0 to 0xFFFFFFFF	0 to 31 (low order 5 bits)	0x0 to 0xFFFFFFFF	Check if left shift is correct.
2	Shift right operand1 by operand2 places	0x0 to 0xFFFFFFFF	0 to 31 (low order 5 bits)	0x0 to 0xFFFFFFFF	Check if right shift is correct.

3. Individual Test Cases

These are the individual test cases performed on the DUT

S.No	Functionality	Operand1	Operand2	Expected Result	Obtained Result	Conclusions
1	Positive + Positive	0x46d00a66	0xbbc28b80	0x8b0d7ee6	0x8b0d7ee6	Test passes if addition is correct.
2	Positive + Negative	0x6FFFFFFF	0x80000001	0xF0000000	0xF0000000	Test checks wrap-around at boundary.
3	Negative + Negative	0x80000000	0x80000000	0x00000000	0x00000000	Test for correct negative overflow.
4	Negative + Positive	0xf54b7e07	0x0937166f	0xfe829476	0xfe829476	Test checks boundary wrap from negative to positive.

S.No	Functionality	Operand1	Operand2	Expected Result	Obtained Result	Conclusions
1	Positive - Positive	0x60000000	0x20000000	0x40000000	0x40000000	Check if subtraction is correct.
2	Positive - Negative	0x40000000	0x80000000	0xC0000000	0xC0000000	Should result in addition due to negative operand.
3	Negative - Negative	0x80000000	0x90000000	0xF0000000	0xF0000000	Subtracting a larger negative increases value.
4	Negative - Positive	0x80000000	0x10000000	0x90000000	0x90000000	Becomes more negative.

4. Results

When the 4 ports are driven asynchronously.

```
# At time 4800:      MASTER |  Transaction Details: Type: ADD, Command = 1,  Op1 = a376c0ff, Op2 = af96034c,Tag = 0
# At time 4800:      MASTER |  Transaction Details: Type: ADD, Data = 0,  Resp = 0, Tag = 0
# At time 4800:      MASTER | transaction successfully driven.
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Command = 6,  Op1 = 43e6b03b, Op2 = e8eb3edb,Tag = 0
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Data = 0,  Resp = 0, Tag = 0
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Command = 6,  Op1 = 8b2508c9, Op2 = aeccfe3d,Tag = 0
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Data = 0,  Resp = 0, Tag = 0
# At time 4800:      MASTER | transaction successfully driven.
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Command = 6,  Op1 = 43e6b03b, Op2 = e8eb3edb,Tag = 0
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Data = 0,  Resp = 0, Tag = 0
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Command = 6,  Op1 = ad08579a, Op2 = e997cf3b,Tag = 0
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Data = 0,  Resp = 0, Tag = 0
# At time 4800:      MASTER | transaction successfully driven.
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Command = 6,  Op1 = 43e6b03b, Op2 = e8eb3edb,Tag = 0
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Data = 0,  Resp = 0, Tag = 0
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Command = 5,  Op1 = c81f77a1, Op2 = 9cc22255,Tag = 0
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Data = 0,  Resp = 0, Tag = 0
# At time 4800:      MASTER | transaction successfully driven.
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Command = 6,  Op1 = 43e6b03b, Op2 = e8eb3edb,Tag = 0
# At time 4800:      MASTER |  Transaction Details: Type: SHIFT, Data = 0,  Resp = 0, Tag = 0
# At time 4800:      MASTER | 4 inputs have been sent!
```

When the Monitor receives the 16 inputs.

```
# At time 5350:     MONITOR |  -------------------- Port           0 Recieved response no.           5 --------------------
# At time 5350:     MONITOR | Response out is 10
# At time 5350:     MONITOR |  Output is 530cc44b, Tag is 00
# At time 5350:     MONITOR |  Number of OVERFLOW Responses (local to port) -           1
# At time 5350:     MONITOR |  From port           0 sent transaction with tag 0 to SCOREBOARD.
# At time 5350:     MONITOR |  -------------------- Port           0 Response ended --------------------
# At time 5350:     MONITOR |  -------------------- Port           1 Recieved response no.           6 --------------------
# At time 5350:     MONITOR | Response Out is  01
# At time 5350:     MONITOR |  Output is 00000004, Tag is 00
# At time 5350:     MONITOR |  Number of VALID Responses (local to port) -           1
# At time 5350:     MONITOR |  From port           1 sent transaction with tag 0 to SCOREBOARD.
# At time 5350:     MONITOR |  -------------------- Port           1 Response ended --------------------
```

After receiving inputs driven confirmation from master and capturing outputs from monitor scoreboards, starts calculation the score.

```
# At time 7600:  SCOREBOARD | !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!   mms_sync has been TRIGGERED !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!
# At time 7600:  SCOREBOARD | mms_sync signal received, stopping collection.
# At time 7600:  SCOREBOARD | Beginning monitor reponse extraction!
# At time 7600:  SCOREBOARD | Compared Master output and Monitor Output for port           0!
# At time 7600:  SCOREBOARD | Emptied tag queue for port           0 and tag           0
# At time 7600:  SCOREBOARD | Compared Master output and Monitor Output for port           0!
# At time 7600:  SCOREBOARD | Emptied tag queue for port           0 and tag           1
# At time 7600:  SCOREBOARD | Compared Master output and Monitor Output for port           0!
# At time 7600:  SCOREBOARD | Emptied tag queue for port           0 and tag           2
# At time 7600:  SCOREBOARD | Compared Master output and Monitor Output for port           0!
# At time 7600:  SCOREBOARD | Emptied tag queue for port           0 and tag           3
# At time 7600:  SCOREBOARD |  ------------- END of monitor_side(), from port           0. --------------
```

Then the process repeats,

```
# At time 7620:      MASTER | wait_scbd_to_finish triggerred continuing with driver!
# At time 7620:      MASTER | In main() -- 16 inputs sent
# At time 7620:      MASTER | In main() -- Waiting for 50 clock cycles, to CAPTURE OUTPUTS
# At time 12600:      MASTER | In main() -- Done waiting.
# At time 12600:      MASTER | Reset Called!
# At time 16600:      MASTER | Asserting Reset to 1, for 3 Clock Cycles
# At time 16900:      MASTER | Deasserting Reset to 0, for 5 Clock Cycles
# At time 17400:      MASTER |  Output from port 0: data_out=0, tag_out=0
# At time 17400:      MASTER |  Output from port 1: data_out=0, tag_out=0
# At time 17400:      MASTER |  Output from port 2: data_out=0, tag_out=0
# At time 17400:      MASTER |  Output from port 3: data_out=0, tag_out=0
# At time 17400:      MASTER | reset EVENT triggered
# At time 17400:      MASTER | Reset EVENT observed.
```

The following are the potential functional bugs found.

Bug #	Title	Explanation
1	SHL does not work for port 0, 1.	Shift Left gives us faulty results that does not match with the expected output.
2	ADD not correct	ADD will not correctly execute at first. Later in the test cycles its starts to improve
3	SUB performance reduced with more transactions	As the number of transactions increases, SUB will perform worse

```
# [MON2SCB] Received Monitor Transaction: Command: 1, Data Out: 747dda89, Tag Out: 3
# [MATCHFOUND] Port 3, Tag 3: Master Cmd=1, Operand1=10599477, Operand2=64244612, Monitor RESP=1, Data=747dda89
# [VERIFICANTION] Transaction match CORRECT at port 3, tag 3, Expected Value: 747dda89
```

Response=1 successful operation completion

```
# [MON2SCB] Received Monitor Transaction: Command: 2, Data Out: ab69295b, Tag Out: 3
# [MATCHFOUND] Port 2, Tag 3: Master Cmd=2, Operand1=99aed15a, Operand2=ee45a7ff, Monitor RESP=2, Data=ab69295b
# [VERIFICANTION] Transaction match CORRECT at port 2, tag 3, Expected Value: ab69295b
```

Response = 2 overflow/underflow

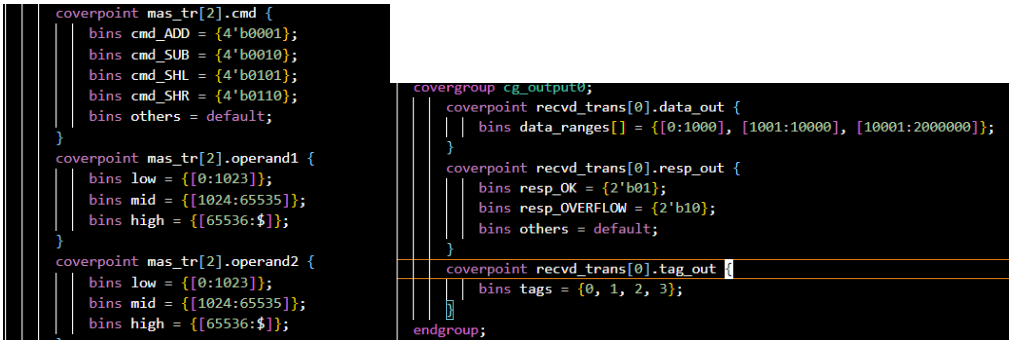
4.1 Functional Coverage

```
# At time 85000:  SCOREBOARD | Gnerating Report
#
# --- Coverage Report ---
#
# Input Coverage Port 1: 55.56%
# Input Coverage Port 2: 55.56%
# Input Coverage Port 3: 55.56%
# Input Coverage Port 4: 55.56%
# Output Coverage Port 1: 66.67%
# Output Coverage Port 2: 77.78%
# Output Coverage Port 3: 79.17%
# Output Coverage Port 4: 100.00%
# Output Coverage Incorrect/Correct: 100.00%
#
# Total Correct Transactions: 137
# Total Incorrect Transactions: 93
#
```

```
#
# Input Coverage Port 1: 47.22%
# Input Coverage Port 2: 55.56%
# Input Coverage Port 3: 47.22%
# Input Coverage Port 4: 47.22%
# Output Coverage Port 1: 66.67%
# Output Coverage Port 2: 66.67%
# Output Coverage Port 3: 66.67%
# Output Coverage Port 4: 66.67%
# Output Coverage Incorrect/Correct: 100.00%
#
# Total Correct Transactions: 14
# Total Incorrect Transactions: 7
#
# Transactions missed: 0
#
```

```
#
# --- Coverage Report ---
#
# Input Coverage Port 1: 55.56%
# Input Coverage Port 2: 55.56%
# Input Coverage Port 3: 55.56%
# Input Coverage Port 4: 55.56%
# Output Coverage Port 1: 66.67%
# Output Coverage Port 2: 83.33%
# Output Coverage Port 3: 79.17%
# Output Coverage Port 4: 91.67%
# Output Coverage Incorrect/Correct: 100.00%
#
# Total Correct Transactions: 359
# Total Incorrect Transactions: 238
#
# Transactions missed: 0
```

By specifying bins of three ranges both inputs and outputs we sample the inputs and outputs. The bins used are as follows:



```
coverpoint mas_tr[2].cmd {
    bins cmd_ADD = {4'b0001};
    bins cmd_SUB = {4'b0010};
    bins cmd_SHL = {4'b0101};
    bins cmd_SHR = {4'b0110};
    bins others = default;
}
coverpoint mas_tr[2].operand1 {
    bins low = {[0:1023]};
    bins mid = {[1024:65535]};
    bins high = {[65536:$]};
}
coverpoint mas_tr[2].operand2 {
    bins low = {[0:1023]};
    bins mid = {[1024:65535]};
    bins high = {[65536:$]};
```

```
covergroup cg_output0;
    coverpoint recvd_trans[0].data_out {
        bins data_ranges[] = {[0:1000], [1001:10000], [10001:2000000]};
    }
    coverpoint recvd_trans[0].resp_out {
        bins resp_OK = {2'b01};
        bins resp_OVERFLOW = {2'b10};
        bins others = default;
    }
    coverpoint recvd_trans[0].tag_out {
        bins tags = {0, 1, 2, 3};
    }
endgroup;
```

For the output, since the response 00 and 11 are regarded as correct, and it is from within these inputs that we want to find which transaction are match the ones sent by the master we only assess *01* and *10.* Any other output is default as INCORRECT.

5. Conclusion

During this lab we were introduced to calc2 which allowed each port to have up to four outstanding commands, utilizing a tagging system to keep track of commands and responses, which could occur out of order. Key achievements in this lab include many classes. In *calc_if* we defined modports to regulate signal direction for different components of the testbench. In *calc_master* we implemented tasks to fetch transactions from the generator mailbox and send them to the DUT while also sending it to the scoreboard for testing. In *calc_gen* random calculator transactions are being created controlled by a maximum transaction count. The class generates transactions sending them to a specified mailbox gen2driver and keeps track on the number of transactions. In *calc_trans* each transaction gets a unique ID and the randomize_transaction function allows for constraints during randomization. In *calc_monitor* the main concurrently monitors all ports allowing it to track and forward all activity from the device to the scoreboard. *Calc_monitor* uses a mailbox to forward outputs from the DUT to the scoreboard for further verification. In *calc_scoreboard* mailboxes from master and monitor send transactions where they are compared with their corresponding calculated value to ensure that the input and output of the DUT is in fact correct. It is here where errors are found.