Concordia University

Winter 2022

COEN 413 : Hardware Functional Verification

Final Project

Michael Arabian - 40095854
Max Burah - 40077075
Andre Saad - 40076579

Date Due : April 15th, 2022

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

Test Plan

Design Under Verification

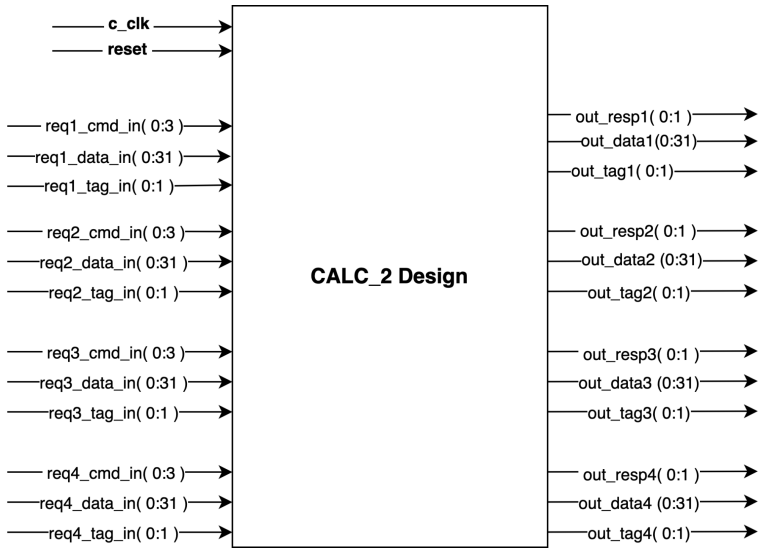

Figure 1. Calculator Design

The provided DUT is a simple calculator which contains 4 ports. Each port in the calculator can have up to 4 commands at one given instance. The calculator is able to perform the following 4 commands : *Add, Subtract, Shift Left, Shift Right*. The desired data is sent along with the command. As this calculator contains 4 ports, there can be up to 16 outstanding commands at one given instance. The input commands as well as their respective descriptions can be seen in *Table 1* below.

Table 1. I/O Description

Command	Description
0	No-Op
1	Add Operand1 and Operand2
2	Subtract Operand1 and Operand2
5	Shift Left Operand1 by Operand2 places
6	Shift Right Operand1 by Operand2 places

Verification Environment

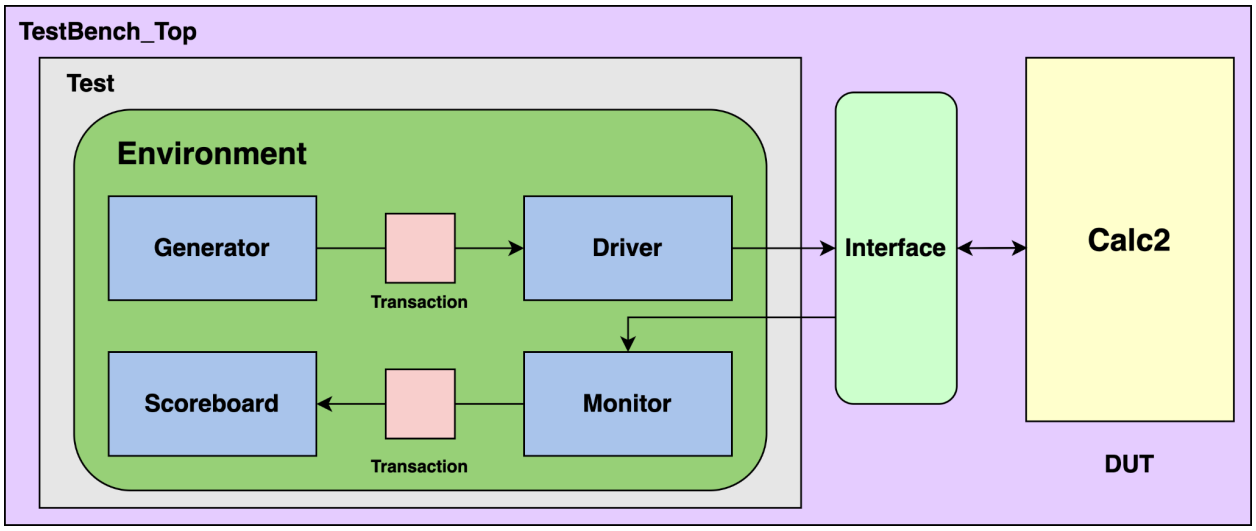

Figure 2. TestBench Architecture

In this section, we'll be describing the classes that create our verification environment. This class based structure allows our code to be as complex as we'd like while keeping each separate class simple to read and understand.

As mentioned previously, our calculator has 4 main operations. Addition, subtraction, shift left, shift right. The *adder.v* class is used to provide ability for the addition and subtraction operations. The *shifter.v* class is used to provide ability for shifting operations.

The *calc2top.v* class combines the classes responsible for the calculator operations such as adder and shifter, alu_input and alu_output. The variables of these classes are combined into a calculator circuit using wires, input data, output data, clock blocks and reset switch.

The calculator circuit is created using the *calc2_if.sv* class which represents the calculator interface. This interface section is where each port's logic and wire variables are initialized. Inputs for each port include a command, data and a tag. A clocking block is synced with the negative edge of the clock and is used to sample the input data for each of the ports. Two

modports are created under the names Master and Slave to specify the inputs and outputs of the modules associated to these modports.

```
task run();

  while(trans_cnt <= max_trans_cnt)
    begin

  addRand();
  addLow();
  addHigh();
  addOverflow();

  subLow();
  subHigh();

  invalidCommand();

  // Generating Random Transactions
      my_tr = randomizeTrans();
      gen2mas.put(my_tr);

      ++trans_cnt;
    end
  ->ended;

endtask
```

Figure 3. Generator Class

Generator.sv serves as the Generator class which is used to generate the stimulus by randomizing the inputs of our system and then sending them to the master. This is achieved by taking a transaction object, randomizing it and then placing it in the gen2mas mailbox. This mailbox stores transaction items and connects the generator class to the driver (master) class. Randomization is a key technique in stimulus testing. It allows our system to handle a wide range of different inputs.

```
class Master;

  bit verbose;
  virtual calc2_if.Master calc_master_if;
  mailbox #(Transaction) gen2mas, mas2scb;


  // Master Concstructor -> Virtual IF, Mailbox (gen2mas, mas2scb)
  function new(virtual calc2_if.Master calc_master_if, mailbox #(Transaction) gen2mas, mas2scb, bit verbose=0);
    this.verbose        = verbose;
    this.calc_master_if = calc_master_if;
    this.gen2mas        = gen2mas;
    this.mas2scb        = mas2scb;
  endfunction: new

  // Main Task for the Master Class
  task main();
    Transaction tr;

    $display("   ---   Beginning Master   ---");

    forever begin

      // Retrieve Transaction object from Mailbox then Drive to ports
      gen2mas.get(tr);
      drive(tr);
    end


  endtask: main
```

Figure 4. Master Class

The *Master* class serves as the driver class and is used to retrieve the values from the generator and then direct the stimulus to the DUT using the inputs from the transaction class and assigning them to interface signals. The constructor for this class uses a virtual interface as a parameter which allows our classes to access signals in the interface through the virtual interface pointer. There are 3 key tasks in this class which are the main task, the reset task and the drive task. The main task retrieves the transaction object from the gen2mas mailbox and places it in, then calls the drive task by passing the same transaction object. The reset task which will initialize the interface signals to their default values. The drive task is used to drive the transaction objects into interface signals. This is achieved by using a master clock and setting the master signal data to the transaction object at each positive edge of the clock.

Monitor.sv is the Monitor class which is used to analyze the environment and sample interface signals of our transactions. The benefit of the monitor is that it allows us to analyze the signals without opening a waveform analyzer. In addition to converting the signals to transaction objects, this class will then send these objects to the scoreboard class through the mailbox mon2scb. The main task within this section uses the positive edge of the clock to sync the results with the transaction object we are creating for each of the 4 ports.

```
task run();
  fork
      forever begin
          mas2scb.get(mas_tr);//input

          request_array[mas_tr.tag] = mas_tr;

          case(mas_tr.cmd)

          //ADD
          4'b0001:
              begin
                  exp_val = mas_tr.data + mas_tr.data2;
              end

          //SUB
          4'b0010:
              begin
                  exp_val = mas_tr.data - mas_tr.data2;
              end

          //LSL
          4'b0101:
              begin
                  exp_val = mas_tr.data << mas_tr.data2[4:0];
              end

          //LRS
          4'b0110:
              begin
                  exp_val = mas_tr.data >> mas_tr.data2[4:0];

              end
```

Figure 5. Scoreboard

Scoreboard.sv is the Scoreboard class which is used as a central location where we handle the data and display the results from our tests. We call transaction objects from the two mailboxes mas2scb and mon2scb to access the results data and the expected data. The main task of this class, takes the transaction object and check to see if it is an addition, subtraction or shift

operation. From there, simple if statements are used to compare the results with what's expected. A string will be displayed to notify whether the results data matches the expected data or not. The number of transactions is also kept track of in this class.

```
class env;

  // Test configurations
  test_cfg    tcfg;


  // Transactors
  Monitor mon;
  scoreboard scb;
  Generator gen;
  Master  mst;

  int tcount = 500;

  mailbox #(Transaction) gen2mas, mas2scb;
  mailbox #(Rslt) mon2scb;

  virtual calc2_if aif;


  function new(virtual calc2_if aif);
    this.aif  = aif;
    gen2mas   = new();
    mas2scb   = new();
    mon2scb   = new();
    tcfg      = new();
    mon      = new(this.aif, mon2scb);
    scb      = new(tcfg.trans_cnt, mas2scb, mon2scb);
    gen      = new(gen2mas, tcount, 1);
    mst      = new(this.aif, gen2mas, mas2scb, 1);
  endfunction: new


  virtual task pre_test();
    scb.max_trans_cnt = gen.max_trans_cnt;
    fork
      scb.run();
      mst.run();
      mon.run();
    join_none
  endtask: pre_test
```

Figure 6. Environment

The *env.sv* class acts as the Environment class which is used to combine all our other classes into a single test environment. This is achieved through the use of object handles which are instantiated for each of our other classes. These handles allow us to access the functions from the following classes: generator, master, monitor, scoreboard and mailbox. There are 3 main tasks within this class which create the testing environment, pre_test, test and post_test. Pre_test task uses threads to run the run() tasks for each of our other classes. The threads are created using the fork and join_none combination. Join_none function allows each thread to execute

without waiting for the previous one to complete execution. Test task serves the purpose of resetting the master class and running a thread of the generator main() task. Post test task uses the keyword wait to suspend execution until the generator and scoreboard events have ended. Our final task is a run task which will call the pre_test(), test() and post_test() task

Test.sv class encompasses the environment class and is where our directed testing takes place. A variable called tcount allows us to insert how many direct tests we'd like to run. An environment handle is created and the run() method is called to run the directed testing.

```
1   `include "env.sv"
2
3
4   //typedef enum {defo, easyA, easyS} mode_t;
5
6   program automatic test(calc_if hif);
7
8
9   env env;
10
11  initial begin
12
13    // Direct Test#1
14    $display("\n ***** START OF DIRECT TESTS \n");
15    env = new(hif);
16    env.tcount = 100;
17
18    env.run();
19
20
21    $display("\n ***** END OF DIRECTED TEST ***** \n");
22
23
24    $finish;
25  end
26
27  endprogram
```

Figure 7. Tests

Test Scenarios

As part of our test plan, we performed two kinds of testing: Direct and Random. The direct tests included corned cases for the 4 operations. For this, we came up and tested the extremities, both maximum and minimum, for all commands. These direct tests can be seen below in Table 2. Our random tests were generated by SystemVerilog's Randomize function. The constraints for the random tests were that the possible commands would only be 1,2,5 or 6.

Table 2. Direct Cases

Test #	Name	Description	Successful
1	Add Max	CMD = 1, Data1 = FFFFFFFF, Data2 = FFFFFFFF	Yes
2	Add Min	CMD = 1, Data1 = 0, Data2 = 0	Yes
3	Sub Max	CMD = 2, Data1 = FFFFFFFF, Data2 = FFFFFFFF	Yes
4	Sub Min	CMD = 2, Data1 = 0, Data2 = 0	Yes
5	Sub D2 > D1	CMD = 2 , Data1 < Data2	No
6	SL Max	CMD = 5, Data1 = FFFFFFFF, Data2 = FFFFFFFF	No
7	SL Min	CMD = 5, Data1 = 0, Data2 = 0	Yes
8	SL Max	CMD = 6, Data1 = FFFFFFFF, Data2 = FFFFFFFF	NO
9	SR Min	CMD = 6, Data1 = 0, Data2 = 0	Yes
10	Add Overflow (data > 8 bits)	CMD = 1, Data1 = 00000000F, Data2 = 00000000F	No
11	Sub Overflow (data > 8 bits)	CMD = 2, Data1 = 00000000F, Data2 = 00000000F	No
12	SL Overflow (data > 8 bits)	CMD = 5, Data1 = 00000000F, Data2 = 00000000F	No
13	SR Overflow (data > 8 bits)	CMD = 6, Data1 = 00000000F, Data2 = 00000000F	No

On top of the above corner cases, we also performed random testing to be able to perform random operations with random data to properly test the DUT. Below in Table 3 are the Random Tests we performed.

Table 3. Random Testing

Test #	Name	Description
1	Rand Port	Random Port → {1,2,3,4}, Count = 100
2	Rand CMD	Random Command → {1,2,5,6}, Count = 100
3	Rand Data	Random Value Given to Data, Count = 100
4	Rand Data2	Random Value Given to Data2, Count = 100

Task Distribution

Team Member	Tasks Completed
Michael Arabian	- Worked on the Interface, Master, Monitor, Env, Scoreboard, Top and Tests files
Andre Saad	- Worked on Transaction and Generator - Worked on Test Cases
Max Burah	- Described the verification environment classes in the report

Bug Report

In our bug report, we found several issues with the provided calculator design. These issues were mostly found when performing LSL and LSR operations. There were also issues when data2 was greater than data and the command was a subtraction.

```
# vsim -do {run -all} -c -novopt top
# //  Questa Sim-64
# //  Version 10.1e linux_x86_64 Jun 11 2013
# //
# //  Copyright 1991-2013 Mentor Graphics Corporation
# //  All Rights Reserved.
# //
# //  THIS WORK CONTAINS TRADE SECRET AND PROPRIETARY INFORMATION
# //  WHICH IS THE PROPERTY OF MENTOR GRAPHICS CORPORATION OR ITS
# //  LICENSORS AND IS SUBJECT TO LICENSE TERMS.
# //
# Refreshing /nfs/home/m/m_arabia/COEN413/413_PROJECT_FINAL_2/work.top
# Loading sv_std.std
# Loading work.top
# Refreshing /nfs/home/m/m_arabia/COEN413/413_PROJECT_FINAL_2/work.calc2_if
# Refreshing /nfs/home/m/m_arabia/COEN413/413_PROJECT_FINAL_2/work.test_sv_unit
# Loading work.test_sv_unit
# Loading work.calc2_if
# Refreshing /nfs/home/m/m_arabia/COEN413/413_PROJECT_FINAL_2/work.test
# Loading work.test
# Refreshing /nfs/home/m/m_arabia/COEN413/413_PROJECT_FINAL_2/work.calc2_top
# Loading work.calc2_top
# Refreshing /nfs/home/m/m_arabia/COEN413/413_PROJECT_FINAL_2/work.adder
# Loading work.adder
# Refreshing /nfs/home/m/m_arabia/COEN413/413_PROJECT_FINAL_2/work.alu_input_stage
# Loading work.alu_input_stage
# Refreshing /nfs/home/m/m_arabia/COEN413/413_PROJECT_FINAL_2/work.alu_output_stage
# Loading work.alu_output_stage
# Refreshing /nfs/home/m/m_arabia/COEN413/413_PROJECT_FINAL_2/work.holdreg
# Loading work.holdreg
# Refreshing /nfs/home/m/m_arabia/COEN413/413_PROJECT_FINAL_2/work.mux_out
# Loading work.mux_out
# Refreshing /nfs/home/m/m_arabia/COEN413/413_PROJECT_FINAL_2/work.priority
# Loading work.priority
# Refreshing /nfs/home/m/m_arabia/COEN413/413_PROJECT_FINAL_2/work.shifter
# Loading work.shifter
# ** Warning: (vsim-3017) top.sv(60): [TFMPC] - Too few port connections. Expected 30, found 26.
#         Region: /top/m1
# ** Warning: (vsim-3722) top.sv(60): [TFMPC] - Missing connection for port 'scan_out'.
# ** Warning: (vsim-3722) top.sv(60): [TFMPC] - Missing connection for port 'a_clk'.
# ** Warning: (vsim-3722) top.sv(60): [TFMPC] - Missing connection for port 'b_clk'.
# ** Warning: (vsim-3722) top.sv(60): [TFMPC] - Missing connection for port 'scan_in'.
# ** Warning: (vsim-3017) calc2_top.v(275): [TFMPC] - Too few port connections. Expected 41,
found 40.
#         Region: /top/m1/priority1
# run -all
#
#  ***** START OF DIRECT TESTS
#
#    ---   Beginning Master   ---
#
# [ CORRECT ]   Port #: 1     Expected value:          8    Received Value:     8
#
# [ CORRECT ]   Port #: 2     Expected value:          8    Received Value:     8
```

```
#
# [ CORRECT ]   Port #: 3     Expected value:          8    Received Value:     8
#
# [ CORRECT ]   Port #: 4     Expected value:          8    Received Value:     8
#
# [ CORRECT ]   Port #: 1     Expected value:          2    Received Value:     2
#
# [ CORRECT ]   Port #: 2     Expected value:          2    Received Value:     2
#
# [  ERROR  ]   Port #: 3     Expected value: 4294967291    Received Value:     2   CMD : 0010
D1 :     0 D2 :          5
#
#
#  Total Errors:           0
#
# [  ERROR  ]   Port #: 4     Expected value: 4294967291    Received Value:     2   CMD : 0010
D1 :     0 D2 :          5
#
#
#  Total Errors:           1
#
#
#                                    ---   INVALID COMMAND   ---
#
#
# [ CORRECT ]   Port #: 1     Expected value: 4294967294    Received Value: 4294967294
#
# [ CORRECT ]   Port #: 2     Expected value: 4294967294    Received Value: 4294967294
#
# [ CORRECT ]   Port #: 3     Expected value: 4294967294    Received Value: 4294967294
#
# [ CORRECT ]   Port #: 4     Expected value: 4294967294    Received Value: 4294967294
#
# [  ERROR  ]   Port #: 1     Expected value:        139    Received Value:     0   CMD : 0001
D1 :    39 D2 :        100
#
#
#  Total Errors:           2
#
# [  ERROR  ]   Port #: 2     Expected value: 4294967294    Received Value:     0   CMD : 0011
D1 :     1 D2 :          1
#
#
#  Total Errors:           3
#
# [  ERROR  ]   Port #: 3     Expected value: 4294967294    Received Value:     0   CMD : 0011
D1 :     1 D2 :          1
#
#
#  Total Errors:           4
#
# [  ERROR  ]   Port #: 4     Expected value: 4294967294    Received Value:     0   CMD : 0011
D1 :     1 D2 :          1
#
#
#  Total Errors:           5
#
# [  ERROR  ]   Port #: 1     Expected value: 4294967294    Received Value:     0   CMD : 0011
D1 :     1 D2 :          1
#
#
#  Total Errors:           6
#
# [  ERROR  ]   Port #: 2     Expected value:        139    Received Value:     0   CMD : 0001
D1 :    39 D2 :        100
#
#
#  Total Errors:           7
#
# [  ERROR  ]   Port #: 3     Expected value:        139    Received Value:     0   CMD : 0001
D1 :    39 D2 :        100
```

```
#
#
#  Total Errors:           8
#
# [  ERROR  ]   Port #: 4     Expected value:        139    Received Value:     0   CMD : 0001
D1 :    39 D2 :        100
#
#
#  Total Errors:           9
#
# [  ERROR  ]   Port #: 1     Expected value:          8    Received Value:     0   CMD : 0001
D1 :     4 D2 :          4
#
#
#  Total Errors:          10
#
# [  ERROR  ]   Port #: 2     Expected value:          8    Received Value:     0   CMD : 0001
D1 :     4 D2 :          4
#
#
#  Total Errors:          11
#
# [  ERROR  ]   Port #: 3     Expected value:          8    Received Value:     0   CMD : 0001
D1 :     4 D2 :          4
#
#
#  Total Errors:          12
#
# [  ERROR  ]   Port #: 4     Expected value:          8    Received Value:     0   CMD : 0001
D1 :     4 D2 :          4
#
#
#  Total Errors:          13
#
# [  ERROR  ]   Port #: 1     Expected value:          2    Received Value: 4294967291      CMD  :
0001    D1 :     1 D2 :          1
#
#
#  Total Errors:          14
#
# [  ERROR  ]   Port #: 2     Expected value:          2    Received Value: 4294967291      CMD  :
0001    D1 :     1 D2 :          1
#
#
#  Total Errors:          15
#
# [ CORRECT ]   Port #: 3     Expected value: 4294967291    Received Value: 4294967291
#
# [ CORRECT ]   Port #: 4     Expected value: 4294967291    Received Value: 4294967291
#
#
#                                    ---   INVALID COMMAND   ---
#
#
# [  ERROR  ]   Port #: 1     Expected value:          0    Received Value:   139   CMD : 0001
D1 :     1 D2 : 4294967295
#
#
#  Total Errors:          16
#
# [  ERROR  ]   Port #: 2     Expected value:          0    Received Value:   139   CMD : 0001
D1 :     1 D2 : 4294967295
#
#
#  Total Errors:          17
#
# [  ERROR  ]   Port #: 3     Expected value:        297    Received Value:   139   CMD : 0001
D1 :   197 D2 :        100
#
#
#  Total Errors:          18
```

```
#
# [  ERROR  ]   Port #: 4     Expected value:        297    Received Value:   139   CMD : 0001
D1 :   197 D2 :        100
#
#
#  Total Errors:          19
#
# [ CORRECT ]   Port #: 1     Expected value:          8    Received Value:     8
#
# [  ERROR  ]   Port #: 2     Expected value: 4294967294    Received Value:     0   CMD : 0011
D1 :     1 D2 :          1
#
#
#  Total Errors:          20
#
# [  ERROR  ]   Port #: 3     Expected value: 4294967294    Received Value:     0   CMD : 0011
D1 :     1 D2 :          1
#
#
#  Total Errors:          21
#
# [  ERROR  ]   Port #: 4     Expected value: 4294967294    Received Value:     0   CMD : 0011
D1 :     1 D2 :          1
#
#
#  Total Errors:          22
#
# [  ERROR  ]   Port #: 1     Expected value: 4294967294    Received Value:     0   CMD : 0011
D1 :     1 D2 :          1
#
#
#  Total Errors:          23
#
# [ CORRECT ]   Port #: 2     Expected value:          8    Received Value:     8
#
# [ CORRECT ]   Port #: 3     Expected value:          8    Received Value:     8
#
# [ CORRECT ]   Port #: 4     Expected value:          8    Received Value:     8
#
# [ CORRECT ]   Port #: 1     Expected value:          2    Received Value:     2
#
# [ CORRECT ]   Port #: 2     Expected value:          2    Received Value:     2
#
#
#                                    ---   INVALID COMMAND   ---
#
#
#  ***** END OF DIRECTED TEST *****
#
# ** Note: $finish    : tests/test.sv(24)
#    Time: 5050 ns  Iteration: 1  Instance: /top/t1
```