Portland State
UNIVERSITY

ECE593 Spring 2022 Final Project Report
Group 3

Verification of Pipelined Picoblaze

Yashodhan Wagle
PSU ID: 930027080
ywagle@pdx.edu

Supreet Gulavani
PSU ID: 949311031
sg7@pdx.edu

Ramaa Potnis
PSU ID: 937862463
rgp2@pdx.edu

Table of contents

1. Abstract

In this project, the pipelined Picoblaze is verified using an object oriented SystemVerilog testbench. The microarchitecture consisted of various individual units , such as the register and every stage, the register file, the ALU etc. all of those components were verified using white box testing- the core functionality of those components were known and then they were tested to check for the correct functionality.

Upon construction of a sound testbench, the DUV was then tested. Because the actual functionality of the design was not known completely, gray box testing was needed for its verification purpose.
Both random and deterministic tests were used for the thorough testing of the DUV.

The coverage unit provided a total of 67.44% of coverage.

2. The DUV - in brief

The design under verification is an 8-bit processor. There is a block memory present which can support programs up to 1024 locations. The architecture is a simple one and operates on information stored in the following address space:

- Registers
  A set of 16 8-bit registers are present. They default to the names s0-SF. Initially, all the registers are initialized to zero.
- Instruction Memory
  This is a form of isolated memory from which every new instruction comes forward. The Program Counter pulls out the next instruction to be performed from this memory
- Scratchpad RAM
  A small pool of RAM is used as local memory. This memory can be accessed during the load, store and the fetch instructions.
- I/O ports
  During an Input operation value at the input port is transferred to one of the 16 registers. At an output operation, contents of any of the 16 registers are transferred to the output port.
  - Stack
  There is a hardware stack maintained to track the return address for the subroutine calls and the interrupt handler mechanism. If at all there is an

overflow detected, an internal reset gets generated which defaults the program from address 0x0000.

The picoblaze processor has two flags to determine the details of the ALU functionality- a Z (zero)flag and a C(carry) flag. The Z flag is set when the operation results to zero. The C flag gets set when a carry is generated.
This processor also has an interrupt mechanism, which is enabled by the *interrupt* signal When this signal is enabled, the current routine operations are halted and the interrupt is tended to. Only after the servicing of the interrupt, the pre-existing routine carries forward.

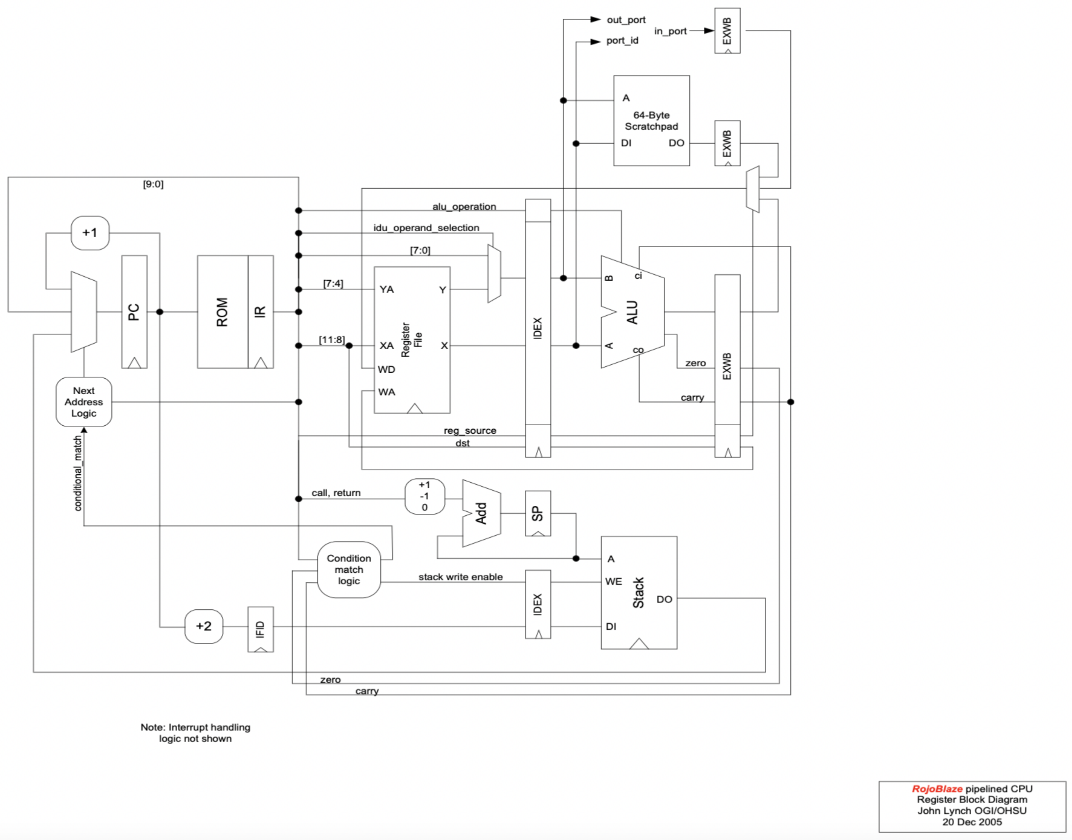


Figure 1. Pipelined Picoblaze datapath

3. Testbench components and interactions

The design was verified bearing in mind the diagram as below. Every module of the diagram is explained in detail in the subsequent section

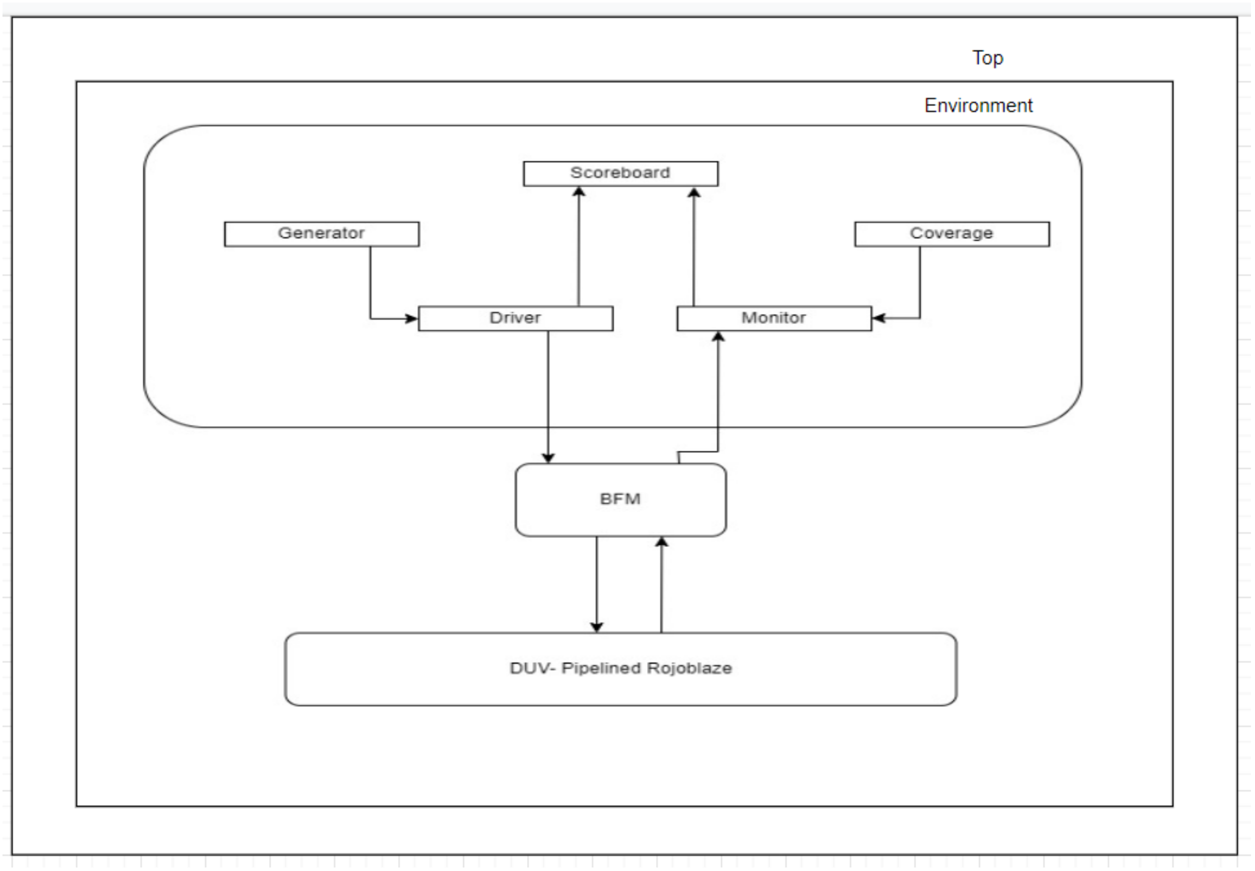


Figure 2. Testbench structure diagram

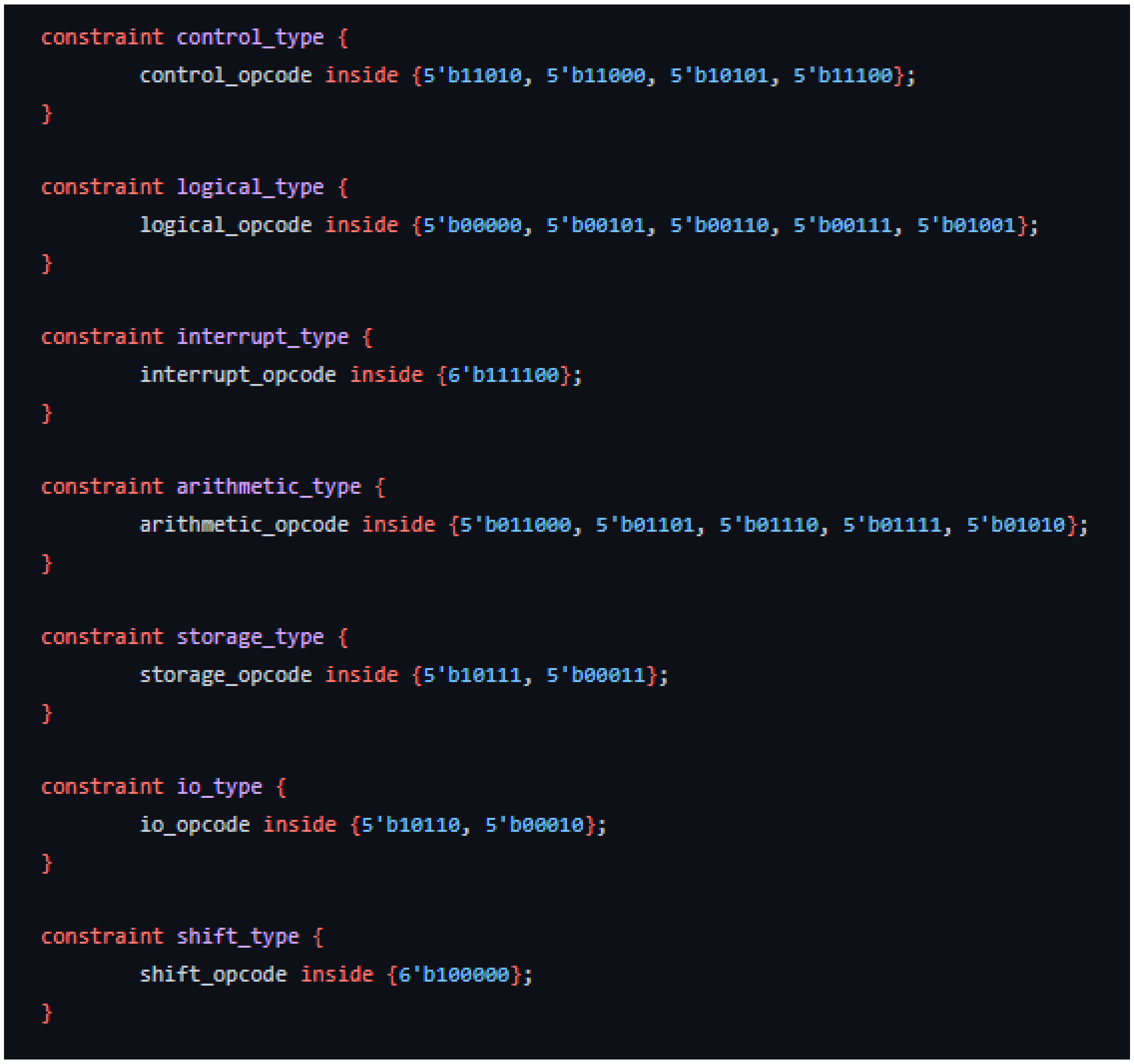
a. Top
The top module sits at the top of the test bench architecture which encapsulates everything. All the signals of the modules are instantiated in these modules.

b. Environment
The environment module has three tasks defined inside it.

- Pre_test - this task triggers off the Driver module to reset.
- Test - This tasks triggers off the driver and the generator modules' main triggered
- Post_test the end of event that occurs

c. Generator
The generator generates stimulus to the DUT and passes it along to the Driver class by the mailbox- gen_driv

There is a main task - ‘task main’ in the generator module which ensures that the transactions that are being passed to it are all randomized. If not, it throws an error message.
The processor has different types of instructions- control, interrupt, logic, storage, input/output and shift and rotate.
For each of the instruction opcode types there is a dedicated constrained randomisation.



```
constraint control_type {
        control_opcode inside {5'b11010, 5'b11000, 5'b10101, 5'b11100};
}

constraint logical_type {
        logical_opcode inside {5'b00000, 5'b00101, 5'b00110, 5'b00111, 5'b01001};
}

constraint interrupt_type {
        interrupt_opcode inside {6'b111100};
}

constraint arithmetic_type {
        arithmetic_opcode inside {5'b011000, 5'b01101, 5'b01110, 5'b01111, 5'b01010};
}

constraint storage_type {
        storage_opcode inside {5'b10111, 5'b00011};
}

constraint io_type {
        io_opcode inside {5'b10110, 5'b00010};
}

constraint shift_type {
        shift_opcode inside {6'b100000};
}
```

Figure 4. Generator module with opcodes that get randomized

To select the type of instruction whose coverage report has to be generated, changes should be made in the MAKE file.

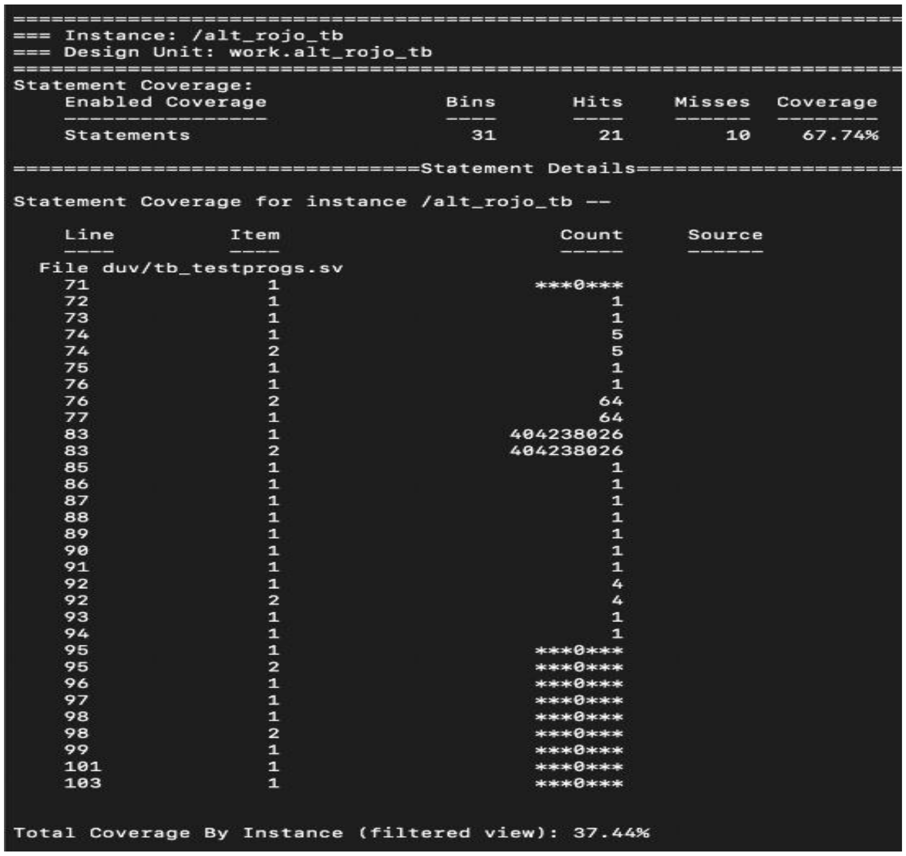
In order to create a custom number of random tests (because the design has only 1024 tests) inside the MAKE file, the number of tests to be implemented are to be entered, the number is then divided by 1024 which will give the number of tests files required.
The fixed loop is then implemented according to how many files are required. This gives "n" number of files each containing 1024 test cases.

d. Driver
The driver wiggles the pins of the DUT through the BFM interface.
The driver receives the stimuli generated by the Generator module via the mail box- 'gen_driv'. The driver also keeps a track of the number of transactions that are taking place. In this module, there are two tasks defined:
   - Task reset - this task resets all the interface values
   - Task main - this task write_mem(), generates .mem files and subsequently increments the transaction count.

e. Monitor
The monitor receives the input signals which are given to the DUv by the BFM. The monitor also does the task of forwarding the signals it receives from the DUV to the scoreboard. The coverage module also feeds the monitor module.

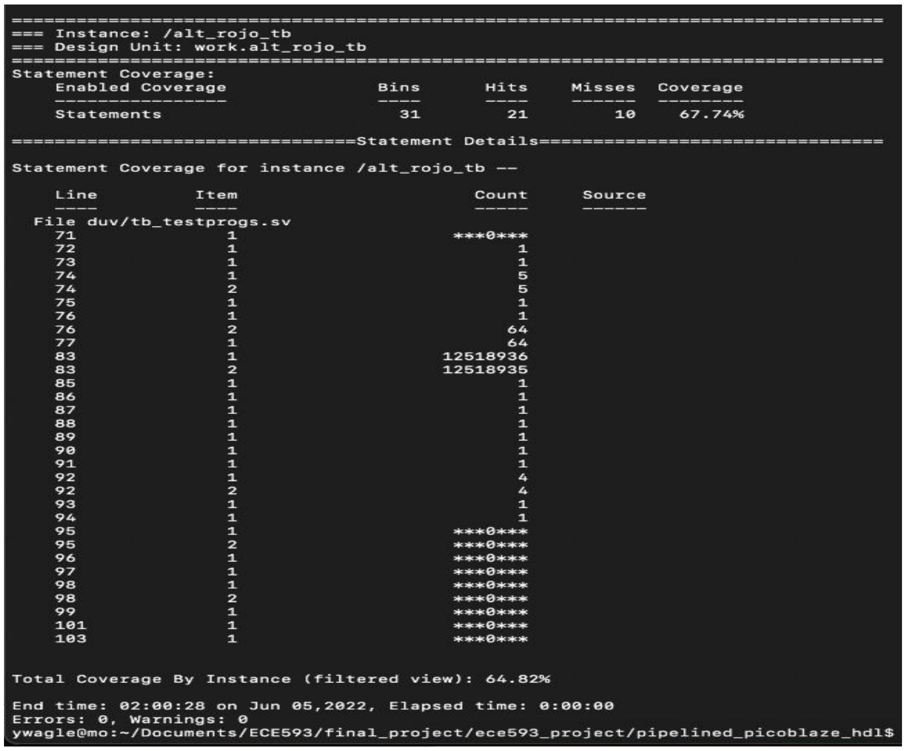
f. Scoreboard
The Scoreboard receives the input and output signals from the DUV which are monitored by the monitor module. Because the design that is being verified is a microcontroller, there is a reference model which is also needed.

g. Coverage
Coverage is based on the signals that are received from the monitor module.
The coverage that was achieved after deploying the above mentioned diagram was 67.44%, which is a fairly decent number given the time constraints and the complexity of the design.



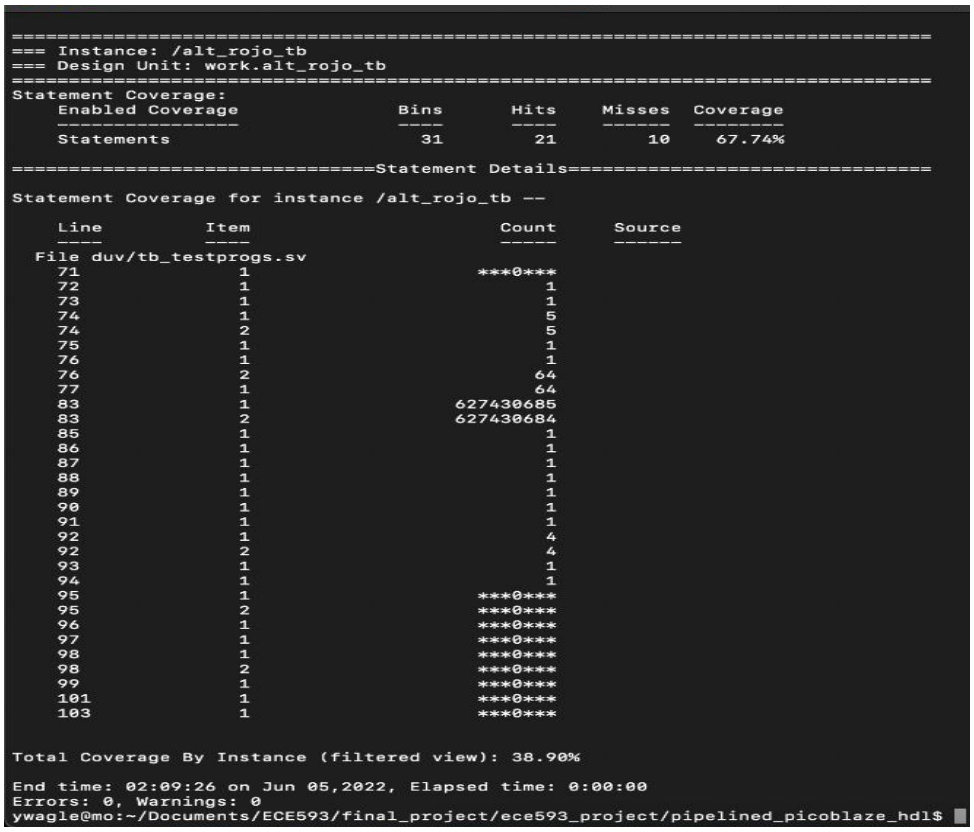
```
=====================================================================
=== Instance: /alt_rojo_tb
=== Design Unit: work.alt_rojo_tb
=====================================================================
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      31        21        10    67.74%

================================Statement Details====================

Statement Coverage for instance /alt_rojo_tb --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File duv/tb_testprogs.sv
    71             1                     ***0***
    72             1                           1
    73             1                           1
    74             1                           5
    74             2                           5
    75             1                           1
    76             1                           1
    76             2                          64
    77             1                          64
    83             1                   404238026
    83             2                   404238026
    85             1                           1
    86             1                           1
    87             1                           1
    88             1                           1
    89             1                           1
    90             1                           1
    91             1                           1
    92             1                           4
    92             2                           4
    93             1                           1
    94             1                           1
    95             1                     ***0***
    95             2                     ***0***
    96             1                     ***0***
    97             1                     ***0***
    98             1                     ***0***
    98             2                     ***0***
    99             1                     ***0***
    101            1                     ***0***
    103            1                     ***0***


Total Coverage By Instance (filtered view): 37.44%
```

Figure 5. Coverage report of the all the instructions



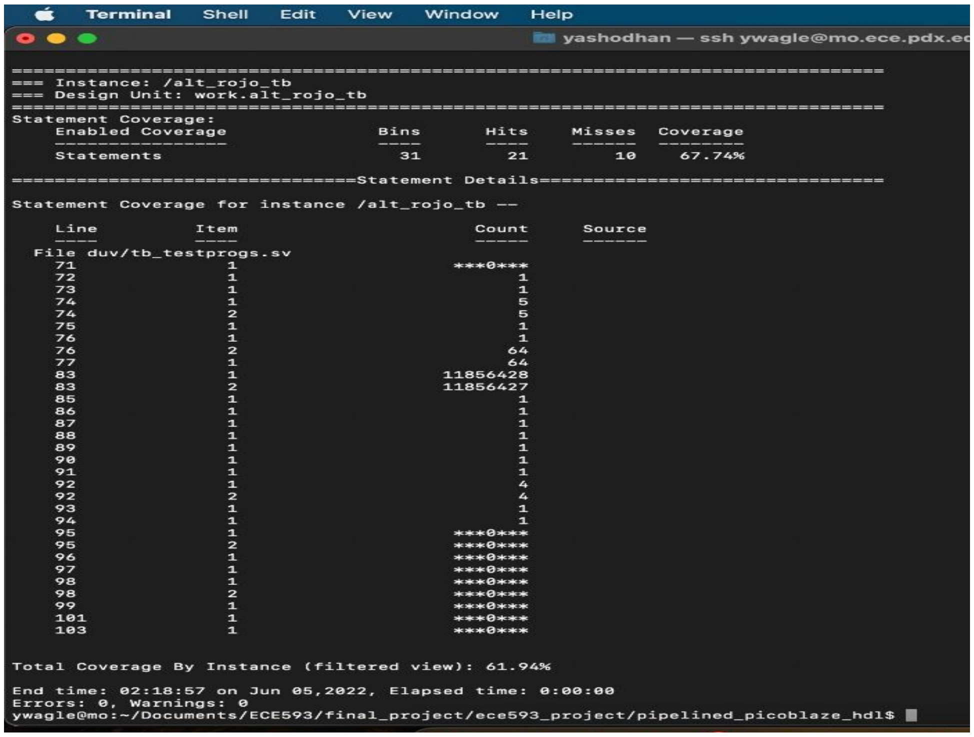
```
================================================================================
=== Instance: /alt_rojo_tb
=== Design Unit: work.alt_rojo_tb
================================================================================
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      31        21        10    67.74%

================================Statement Details================================

Statement Coverage for instance /alt_rojo_tb --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File duv/tb_testprogs.sv
    71             1                     ***0***
    72             1                           1
    73             1                           1
    74             1                           5
    74             2                           5
    75             1                           1
    76             1                           1
    76             2                          64
    77             1                          64
    83             1                    12518936
    83             2                    12518935
    85             1                           1
    86             1                           1
    87             1                           1
    88             1                           1
    89             1                           1
    90             1                           1
    91             1                           1
    92             1                           4
    92             2                           4
    93             1                           1
    94             1                           1
    95             1                     ***0***
    95             2                     ***0***
    96             1                     ***0***
    97             1                     ***0***
    98             1                     ***0***
    98             2                     ***0***
    99             1                     ***0***
    101            1                     ***0***
    103            1                     ***0***


Total Coverage By Instance (filtered view): 64.82%

End time: 02:00:28 on Jun 05,2022, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
ywagle@mo:~/Documents/ECE593/final_project/ece593_project/pipelined_picoblaze_hdl$
```

Figure 6. Coverage report of arithmetic instructions



```
================================================================================
=== Instance: /alt_rojo_tb
=== Design Unit: work.alt_rojo_tb
================================================================================
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      31        21        10    67.74%

================================Statement Details================================

Statement Coverage for instance /alt_rojo_tb --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File duv/tb_testprogs.sv
    71             1                     ***0***
    72             1                           1
    73             1                           1
    74             1                           5
    74             2                           5
    75             1                           1
    76             1                           1
    76             2                          64
    77             1                          64
    83             1                   627430685
    83             2                   627430684
    85             1                           1
    86             1                           1
    87             1                           1
    88             1                           1
    89             1                           1
    90             1                           1
    91             1                           1
    92             1                           4
    92             2                           4
    93             1                           1
    94             1                           1
    95             1                     ***0***
    95             2                     ***0***
    96             1                     ***0***
    97             1                     ***0***
    98             1                     ***0***
    98             2                     ***0***
    99             1                     ***0***
    101            1                     ***0***
    103            1                     ***0***


Total Coverage By Instance (filtered view): 38.90%

End time: 02:09:26 on Jun 05,2022, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
ywagle@mo:~/Documents/ECE593/final_project/ece593_project/pipelined_picoblaze_hdl$
```

Figure 7. Coverage report of control instructions



```
================================================================================
=== Instance: /alt_rojo_tb
=== Design Unit: work.alt_rojo_tb
================================================================================
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      31        21        10    67.74%

================================Statement Details================================

Statement Coverage for instance /alt_rojo_tb --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File duv/tb_testprogs.sv
    71              1                    ***0***
    72              1                          1
    73              1                          1
    74              1                          5
    74              2                          5
    75              1                          1
    76              1                          1
    76              2                         64
    77              1                         64
    83              1                   11856428
    83              2                   11856427
    85              1                          1
    86              1                          1
    87              1                          1
    88              1                          1
    89              1                          1
    90              1                          1
    91              1                          1
    92              1                          4
    92              2                          4
    93              1                          1
    94              1                          1
    95              1                    ***0***
    95              2                    ***0***
    96              1                    ***0***
    97              1                    ***0***
    98              1                    ***0***
    98              2                    ***0***
    99              1                    ***0***
    101             1                    ***0***
    103             1                    ***0***

Total Coverage By Instance (filtered view): 61.94%

End time: 02:18:57 on Jun 05,2022, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
ywagle@mo:~/Documents/ECE593/final_project/ece593_project/pipelined_picoblaze_hdl$
```

Figure 8. Coverage report of logic instructions

Following were the functional coverage coverpoints:

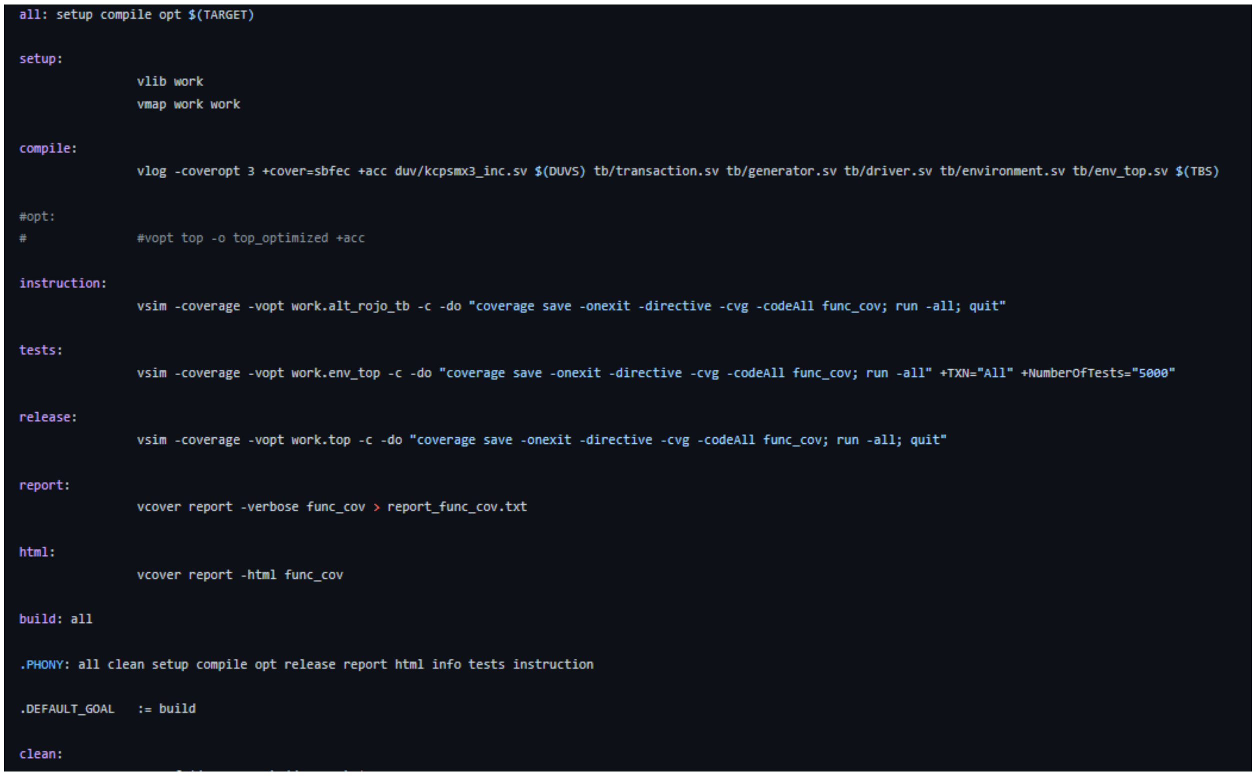
- Cg_input_signals - for collecting the coverage from the input signals
- Cg_output_signals - for collecting the coverage from the output signals

In order to observe the operations, there are certain cross and coverpoints as well.

- rojob_out_zout - for the zero(Z) flag
- Rojob_out_carryout - for the carry (C) flag
- Rojob_out_resultxout - cross point between the result and the Z flag
- Rojob_out_resultcout -   cross point between the result and the C flag
- Rojob_out_portidxreadstrobe - cross between output port id and the readstrobe rojob_out_readstrobe;
- Rojob_out_portidxwritestrobe - cross between output pot id and the write rojob_out_writestrobe;
- rojob_out_portidxinterruptack: cross rojob_out_portid, rojob_out_interruptack;
- rojob_out_readstrobexinterruptack: cross rojob_out_readstrobe, rojob_out_interruptack;
- rojob_out_writestrobexinterruptack: cross rojob_out_writestrobe, rojob_out_interruptack;

4. Results

From whatever modules were designed, and from whatever covergroups were designed, a code coverage of 67.74% was gathered and a functional coverage of 64.82% was gathered.
SystemVerilog Assertions in duv were also added to verify the design functionality more thoroughly.
Although the numbers are not very high, the structure of the testbench is well defined, with a python script to generate various .mem files.



```
all: setup compile opt $(TARGET)

setup:
                vlib work
                vmap work work

compile:
                vlog -coveropt 3 +cover=sbfec +acc duv/kcpsmx3_inc.sv $(DUVS) tb/transaction.sv tb/generator.sv tb/driver.sv tb/environment.sv tb/env_top.sv $(TBS)

#opt:
#               #vopt top -o top_optimized +acc

instruction:
                vsim -coverage -vopt work.alt_rojo_tb -c -do "coverage save -onexit -directive -cvg -codeAll func_cov; run -all; quit"

tests:
                vsim -coverage -vopt work.env_top -c -do "coverage save -onexit -directive -cvg -codeAll func_cov; run -all" +TXN="All" +NumberOfTests="5000"

release:
                vsim -coverage -vopt work.top -c -do "coverage save -onexit -directive -cvg -codeAll func_cov; run -all; quit"

report:
                vcover report -verbose func_cov > report_func_cov.txt

html:
                vcover report -html func_cov

build: all

.PHONY: all clean setup compile opt release report html info tests instruction

.DEFAULT_GOAL   := build

clean:
```

Figure 13.MAKE file which enables to run the entire design

5. Team Efforts

Initially, some time was dedicated to understanding the microarchitecture and the nooks and crannies of the design. Once that was done, the first rendition of the verification plan was drafted. There were certain alterations to be made in the first rendition of the plan,which were done thereby tweaking the verification approach.

For the first code drop(week 8), the team came up with a rough arrangement of all the modules that could help effectively verify the design. When that got approved, all the other modules were constructed bearing that diagram in mind.

Since the code was derived from the class project of ECE 571, there was a basic understanding, but the ask for this project was an object oriented testbench. All the team members spent at least 18-20 hours revisiting and understanding the design.

The hdl files required some changes as with what was given, randomizing the input files was not possible.

Thus after understanding the design, making some changes to it, learning about the object oriented testbenches and after brainstorming the various approaches to verify the design, each team member's contribution was as follows:

- Yashodhan Wagle - Spent more than 25 hours writing different files for class based verification and connection of each file with an interface and mailbox, scoreboard creation,and generation of reference model.

- Supreet Gulavni - Spent more than 25 hours writing the stress cases in the generator, and waveform checking and creating the coverage files and class based debugging.

- Ramaa Potnis - Spent more than 25 hours for proper documentation, deterministic testing, modifying the HDL, creating the environment and creating the top module.

6. Challenges encountered

- The Picoblaze design reads off of a ROM created by the assembler provided in the design files. This meant that to run a program, we needed to create a new memory image file for it to run on the design. This posed a challenge as this along with the limit of program size meant that the generator needed to be customized to create various files containing test cases in the form of a ROM (in our case a .mem file) in order to run the design.

- The design module in itself could produce only 1024 words of program space, that means the number of tests to the design was restricted to 1024 only. Randomizing that number so that proper verification was done was another major challenge.

- Because the design is a microprocessor, the output of it is written to the memory and/or registers. That means there are no output signals to be interfaced with the bfm. This posed a major challenge while collecting the output coverage.

7. Future Scope

- Time was a major constraint for this project, Given more time, the team could have implemented a proper UVM based testbench.

- The functional coverage figure could have been better, should there have been more coverpoints.

- More test case constraints could have also been included.

- Even though we have a model containing bugs for verification purposes, those bugs are not known. This bug ridden design was provided in the ECE 571 class for the purpose of running our testbench. Thus bugs could have been injected in the code to verify the design.

8. References

For this project, we made use of the following references which helped us verify the design more effectively.

- Professor Schubert’s lectures and slides

- Comprehensive Functional Verification Textbook by Bruce Wile, John C Goss and Wolfgang Roesner

- Uvm-primer master by Ray Salemi