ECE 586

Computer Architecture

(Spring 2024)

A PROJECT REPORT

submitted in partial fulfillment of the requirements
for the award of the degree of

Master of Science
in
ELECTRICAL AND COMPUTER ENGINEERING

Guide: Professor Yuchen Huang

AUTHORS

SUGGU V S S K K L SAI TAGORE (989262683)
SURYA VAMSI TIRUVEEDHULA (934707602)
DIVYA SRI AYLURI (976089658)
RAFATH ACHUGATLA (934857968)

Portland State UNIVERSITY

CONTENTS

OBJECTIVE:

The project's purpose is to replicate MIPS-lite, a simplified version of the MIPS ISA, as well as the in-order 5-stage pipeline. we developed a simulator in System Verilog. As input, the simulator will use the specified memory image.

Two major features will be implemented:

• A functional simulator that emulates the MIPS-lite instruction set architecture and captures the impact of

instruction execution on machine state.

• A timing simulator which models the timing details for the 5-stage pipeline.

The output of the simulator will include:

• A breakdown of instruction frequencies in the instruction traces into different instruction types.

• Final machine state (register values, memory contents etc.).

• Execution time of the instruction trace on the 5-stage pipeline.

OPERATION:

Functional simulator created is capable of simulating following instructions:

● Arithmetic Instructions (ADD, ADDI, SUB, SUBI, MUL, MULI)

● Logical Instructions (OR, ORI, AND, ANDI, XOR, XORI)

● Memory Access Instructions (LDW, STW)

● Control Flow Instructions (BZ, BEQ, JR, HALT)

INSTRUCTION FORMAT:

MIPS supports 3 types of Instruction Formats, they are:

1) R-type

2) I-type

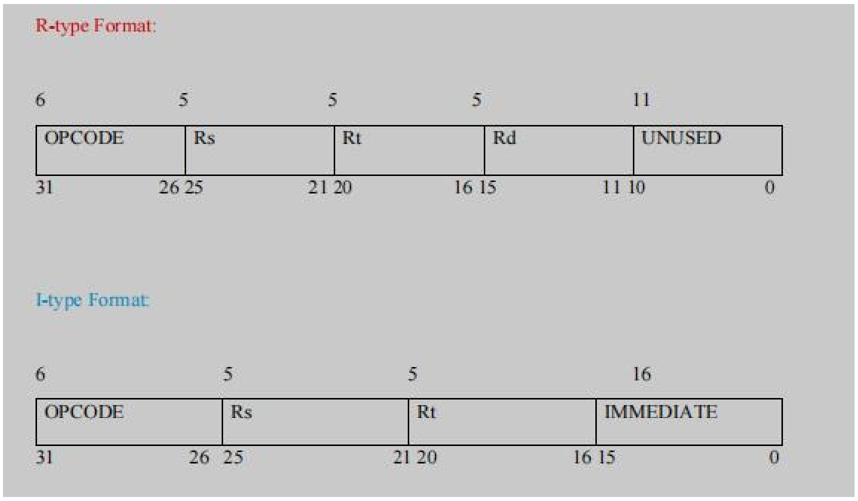
3) J-type (Used for Jump Instructions).

But in our project, we use only two types of Instruction Formats, they are:

1) R-type: This format is used by instructions, ADD, SUB, MUL, OR, AND and XOR.

2) I-type: This format is used by instructions, ADDI, SUBI, MULI, ORI, ANDI,

XORI, LDW, STW, BZ, BEQ, JR, HALT



We simulated MIPS in two modes they are:

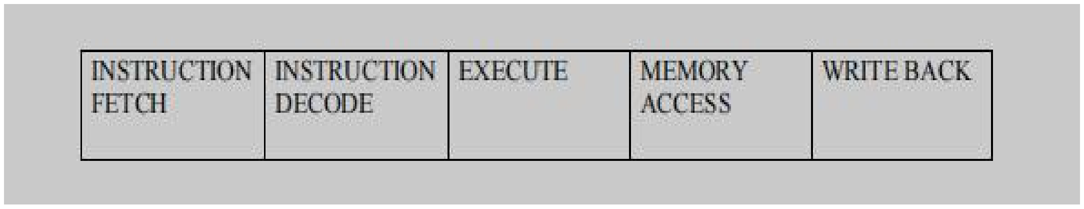
1) Functional Simulator: In this mode, our MIPS runs in normal mode without the pipeline.

2) Timing Simulator: In this mode, MIPS runs in pipelined mode. This is divided into two parts: with forwarding and without forwarding.

All the statistics are collected like RAW hazards, no of clock cycles, Program Counter, etc for every mode.

MIPS PROCESSOR:

MIPS follows 5 stages of inline pipelining as follows:



INSTRUCTION FETCH	INSTRUCTION DECODE	EXECUTE	MEMORY ACCESS	WRITE BACK

Instruction Steps:

1*) **Instruction Fetch***: Fetch the instruction and increment the PC.

2) ***Instruction Decode*****:** Decode the instruction to find the opcode that will be executed. Examine the contents of the source registers.

3) ***Execute*****:** Compute the operation which is determined in the decode stage.

4) ***Memory Access*****:** This stage is used if the instructions are memory-related, such as load and store.

5) ***Write Back*****:** Writes the results into the destination register.

1. Stall Conditions in both 'No Forwarding' and 'Forwarding' conditions:

• In MIPS, there are no WAR or WAW hazards.

• Only RAW hazards require pipeline stalls.

• Techniques like forwarding can be used to reduce stalls.

• Forwarding allows direct passing of data from preceding instructions to dependent instructions.

• By minimizing stalls through forwarding, execution time can be reduced.

2. No Forwarding condition:

• When a consumer instruction follows a producer instruction, there is no stall penalty, except for the load instruction.

• In the case of a consumer load instruction following a producer instruction, there is a stall penalty of 1 cycle.

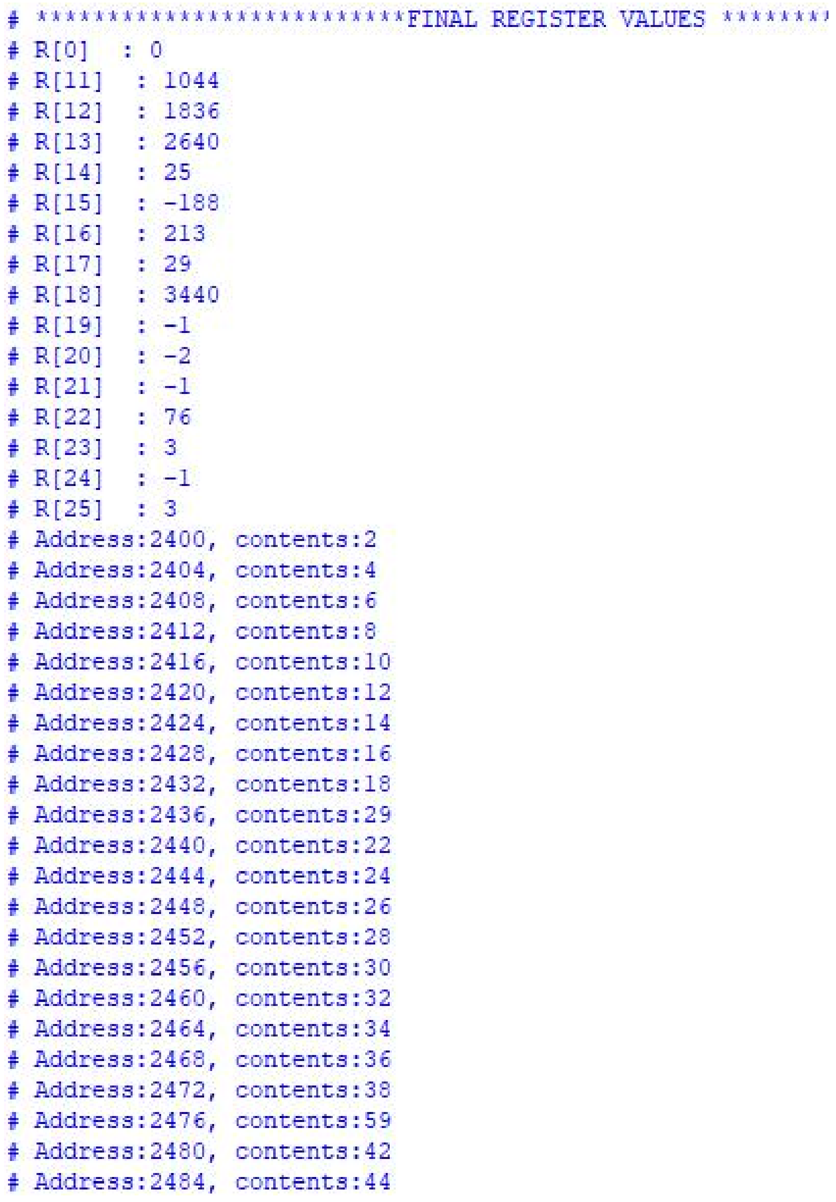
3. Forwarding Condition:

• If the consumer instruction immediately follows the producer instruction, there is a stall penalty of 2 cycles.

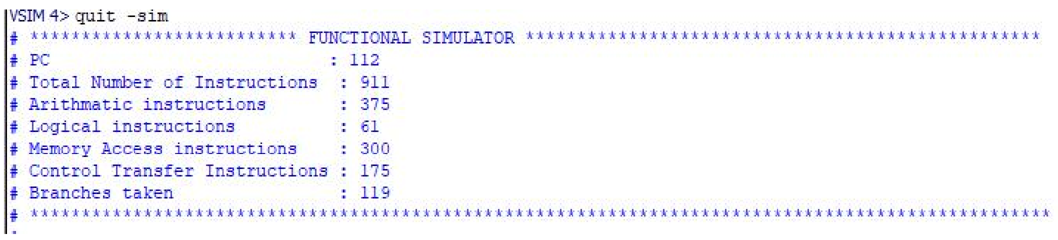
• If there is only one instruction between the consumer instruction and the producer instruction, the stall penalty is 1 cycle.

TRANSCRIPTS:

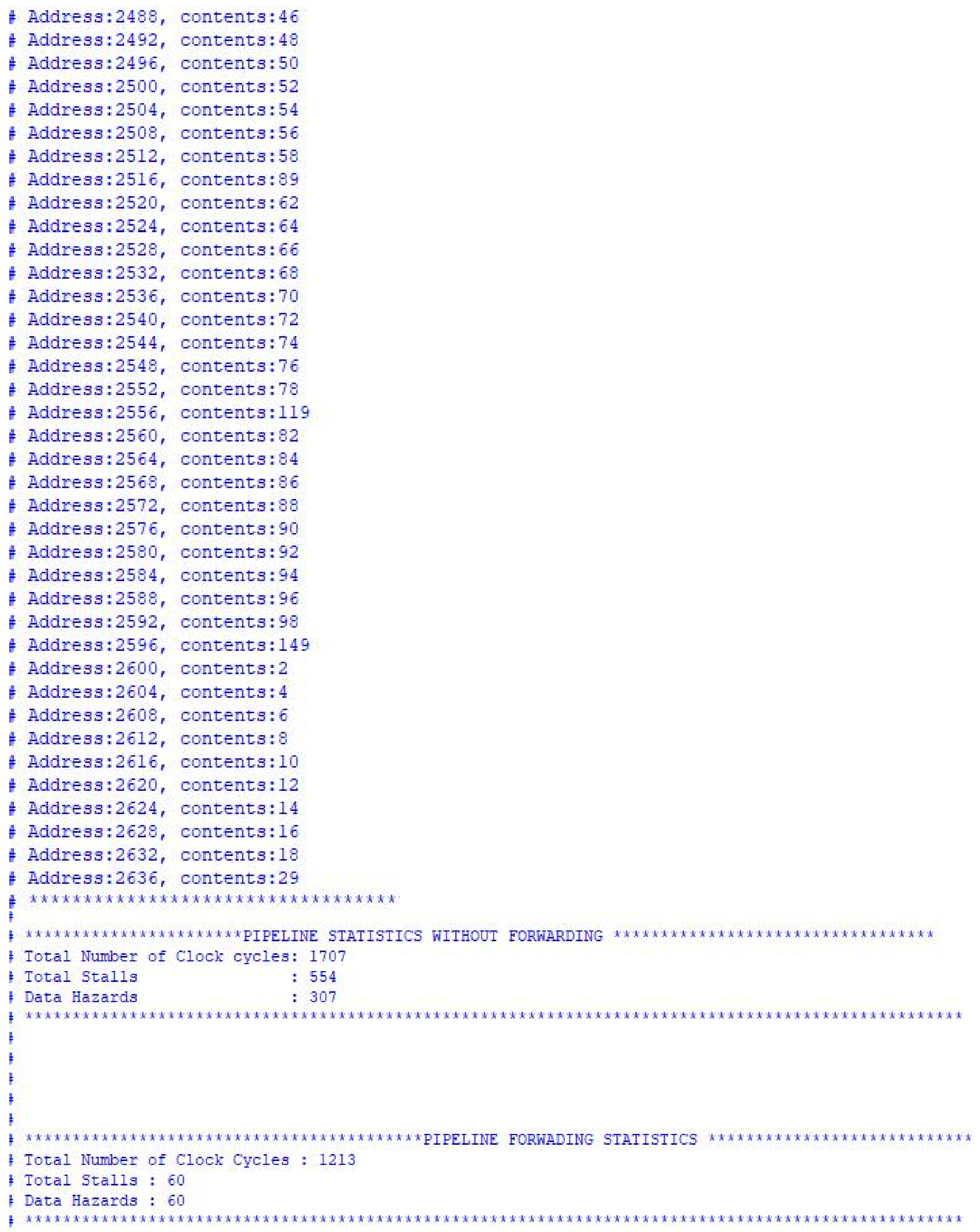
```
# ******************************************* PIPELINE STATISTICS WITHOUT FORWARDING
*************************************
#
#
# The number of clock cycles in non forwarding : 1707
# The number of stall cycles : 554
# The number of Data Hazards : 307
# The number of Single stalls : 60
# The number of Double stalls : 247
# ******************************************* END OF PIPELINE WITHOUT FORWARDING
****************************************
#
#
# ******************************************* PIPELINE FORWADING STATISTICS
**********************************************
# The number of clock cycles during forwarding : 1213

# The number of Data Hazards in forwarding : 60
```



```
# **************************FINAL REGISTER VALUES *******
# R[0]  : 0
# R[11]  : 1044
# R[12]  : 1836
# R[13]  : 2640
# R[14]  : 25
# R[15]  : -188
# R[16]  : 213
# R[17]  : 29
# R[18]  : 3440
# R[19]  : -1
# R[20]  : -2
# R[21]  : -1
# R[22]  : 76
# R[23]  : 3
# R[24]  : -1
# R[25]  : 3
# Address:2400, contents:2
# Address:2404, contents:4
# Address:2408, contents:6
# Address:2412, contents:8
# Address:2416, contents:10
# Address:2420, contents:12
# Address:2424, contents:14
# Address:2428, contents:16
# Address:2432, contents:18
# Address:2436, contents:29
# Address:2440, contents:22
# Address:2444, contents:24
# Address:2448, contents:26
# Address:2452, contents:28
# Address:2456, contents:30
# Address:2460, contents:32
# Address:2464, contents:34
# Address:2468, contents:36
# Address:2472, contents:38
# Address:2476, contents:59
# Address:2480, contents:42
# Address:2484, contents:44
```



```
VSIM 4> quit -sim
# ************************* FUNCTIONAL SIMULATOR ************************************************
# PC                           : 112
# Total Number of Instructions  : 911
# Arithmatic instructions       : 375
# Logical instructions          : 61
# Memory Access instructions    : 300
# Control Transfer Instructions : 175
# Branches taken                : 119
# ***********************************************************************************************
```



```
# Address:2488, contents:46
# Address:2492, contents:48
# Address:2496, contents:50
# Address:2500, contents:52
# Address:2504, contents:54
# Address:2508, contents:56
# Address:2512, contents:58
# Address:2516, contents:89
# Address:2520, contents:62
# Address:2524, contents:64
# Address:2528, contents:66
# Address:2532, contents:68
# Address:2536, contents:70
# Address:2540, contents:72
# Address:2544, contents:74
# Address:2548, contents:76
# Address:2552, contents:78
# Address:2556, contents:119
# Address:2560, contents:82
# Address:2564, contents:84
# Address:2568, contents:86
# Address:2572, contents:88
# Address:2576, contents:90
# Address:2580, contents:92
# Address:2584, contents:94
# Address:2588, contents:96
# Address:2592, contents:98
# Address:2596, contents:149
# Address:2600, contents:2
# Address:2604, contents:4
# Address:2608, contents:6
# Address:2612, contents:8
# Address:2616, contents:10
# Address:2620, contents:12
# Address:2624, contents:14
# Address:2628, contents:16
# Address:2632, contents:18
# Address:2636, contents:29
# ***********************************
#
# ***********************PIPELINE STATISTICS WITHOUT FORWARDING ***********************************
# Total Number of Clock cycles: 1707
# Total Stalls                : 554
# Data Hazards                : 307
# ****************************************************************************************************
#
#
#
#
#
# *******************************************PIPELINE FORWADING STATISTICS *****************************
# Total Number of Clock Cycles : 1213
# Total Stalls : 60
# Data Hazards : 60
# ****************************************************************************************************
```

RESULTS & UNDERSTANDING:

1. Total number of instructions and a breakdown of instruction frequencies for the following instruction types: Arithmetic, Logical, Memory Access and Control Transfer.

Total number of instructions	–	*911*
Arithmetic instructions	–	*375*
Logical instructions	–	*61*
Memory Access instructions	–	*300*
Control Transfer instructions	–	*175*

2. Final state of program counter, general purpose registers and memory

Final state of Program Counter – 112

Final contents of registers

REGISTER	ELEMENT
R11	1044
R12	1836
R13	2640
R14	25
R15	-188
R16	213
R17	29
R18	3440
R19	-1
R20	-2
R21	-1
R22	76
R23	3
R24	-1
R25	3

Final State of Memory

- Memory Address 2432 contents are 18
- Memory Address 2560 contents are 82
- Memory Address 2436 contents are 29
- Memory Address 2600 contents are 2
- Memory Address 2440 contents are 22
- Memory Address 2540 contents are 72
- Memory Address 2444 contents are 24
- Memory Address 2596 contents are 149
- Memory Address 2448 contents are 26
- Memory Address 2584 contents are 94
- Memory Address 2628 contents are 16
- Memory Address 2452 contents are 28
- Memory Address 2592 contents are 98
- Memory Address 2456 contents are 30
- Memory Address 2564 contents are 84
- Memory Address 2460 contents are 32
- Memory Address 2464 contents are 34
- Memory Address 2544 contents are 74
- Memory Address 2468 contents are 36
- Memory Address 2472 contents are 38
- Memory Address 2588 contents are 96
- Memory Address 2476 contents are 59
- Memory Address 2480contents are 42
- Memory Address 2568 contents are 86
- Memory Address 2484 contents are 44
- Memory Address 2488 contents are 46
- Memory Address 2548 contents are 76
- Memory Address 2624 contents are 14
- Memory Address 2492 contents are 48
- Memory Address 2496 contents are 50
- Memory Address 2528 contents are 66
- Memory Address 2636 contents are 29
- Memory Address 2500 contents are 52
- Memory Address 2504 contents are 54
- Memory Address 2572 contents are 88
- Memory Address 2508 contents are 56
- Memory Address 2512 contents are 58
- Memory Address 2552 contents are 78
- Memory Address 2516 contents are 89
- Memory Address 2520 contents are 62
- Memory Address 2532 contents are 68
- Memory Address 2524 contents are 64
- Memory Address 2620 contents are 12
- Memory Address 2400 contents are 2
- Memory Address 2576 contents are 90
- Memory Address 2404 contents are 4
- Memory Address 2616 contents are 10
- Memory Address 2408 contents are 6
- Memory Address 2556contents are 119
- Memory Address 2632 contents are 18
- Memory Address 2412 contents are 8
- Memory Address 2612 contents are 8
- Memory Address 2416 contents are 10

- Memory Address 2536 contents are 70
- Memory Address 2420 contents are 12
- Memory Address 2608 contents are 6
- Memory Address 2424 contents are 14
- Memory Address 2580 contents are 92
- Memory Address 2428 contents are 16
- Memory Address 2604 contents are 4

3.In case of “no forwarding”, the total number of data hazards and average stall penalty per hazard.

Total number of stalls = 554
Total number of data hazards = 307
Single stalls = 60
Double stalls = 247
Average Stall penalty per hazard = Total number of stalls / Total number of data hazards
Average stall penalty per hazard = 554 / 307
= 1.8045602605

Penalties because of branches = 238
Number of branches leading to penalties = 119
Average Branch Penalty = 2

4. In case of “forwarding”, the number of data hazards which could not be fully eliminated by forwarding.

Total number of stalls = 60

5. Execution time in terms of number of clock cycles for the “no forwarding” and the “forwarding” scenarios.

Clock cycles with forwarding = 1213
Clock cycles without forwarding = 1707

6. Speedup achieved by “forwarding” as compared to “no forwarding”.
Speedup = (Execution time) no forwarding / (Execution time) forwarding Speedup = 1.4073

ROLES & RESPONSIBILITIES:

We as in Sai Tagore, Surya Vamsi, Divya Sri & Rafath have worked with a great co-ordination to complete this MIPS Lite Architecture Project, coming to roles and responsibilities, we have divided our work equally among ourselves to complete the project as soon as possible without any errors and without any miss conceptions.
1. **Suggu Tagore -** Gathered all the details and learnt and executed the instruction Decode and

Execute stage related operations and worked on the code and created a pseudo code and documented the data related to instruction decode and execute stage, equally contributed in testbench top creation and documentation.
2. **Divya Sri Ayluri –** Gathered all the details and learnt and executed the instruction fetch stage operations and worked on the code and created a pseudo code and documented the data related to instruction fetch. equally contributed in testbench top creation and documentation.
3. **Rafath Achugatla –** Gathered all the details and learnt and executed the Memory stage related operations and worked on the code and created a pseudo code and documented the data related to instruction memory stage. equally contributed in testbench top creation and documentation.
4. **Surya Vamsi -** Gathered all the details and learnt and executed writeback stage in the pipeline related operations and worked on the code and created a pseudo code and documented the data related to write backstage. equally contributed in testbench top creation and documentation.