Implementation of 32-Bit MIPS Processor with Hazard Control and Forward Chaining

-Eakta Anchlia

The intention of this project is to implement 32-bit MIPS processor in Verilog. The design is pipelined and for the 5-stage pipeline.

The design is tested using dot product program. The test vector is the student ID and each digit of student incremented by 2.

## I. Introduction

The execution of an instruction in a processor can be split up into a few stages. How many stages there are, and the purpose of each stage is different for each processor design. Examples includes 2 stages (Instruction Fetch / Instruction Execute) and 3 stages (Instruction Fetch, Instruction Decode, Instruction Execute). The MIPS processor has 5 stages:

IF: The Instruction Fetch stage fetches the next instruction from memory using the address in the PC (Program Counter) register and stores this instruction in the IR (Instruction Register)

ID: The Instruction Decode stage decodes the instruction in the IR, calculates the next PC, and reads any operands required from the register file.

EX: The Execute stage "executes" the instruction. In fact, all ALU operations are done in this stage. (The ALU is the Arithmetic and Logic Unit and performs operations such as addition, subtraction, shifts left and right, etc.)

MA: The Memory Access stage performs any memory access required by the current instruction, So, for loads, it would load an operand from memory. For stores, it would store an operand into memory. For all other instructions, it would do nothing.

WB: For instructions that have a result (a destination register), the Write Back writes this result back to the register file. Note that this includes nearly all instructions, except nops (a nop, no-op or no-operation instruction simply does nothing) and s (stores).

## Figure 1: Pipelined MIPS Processor

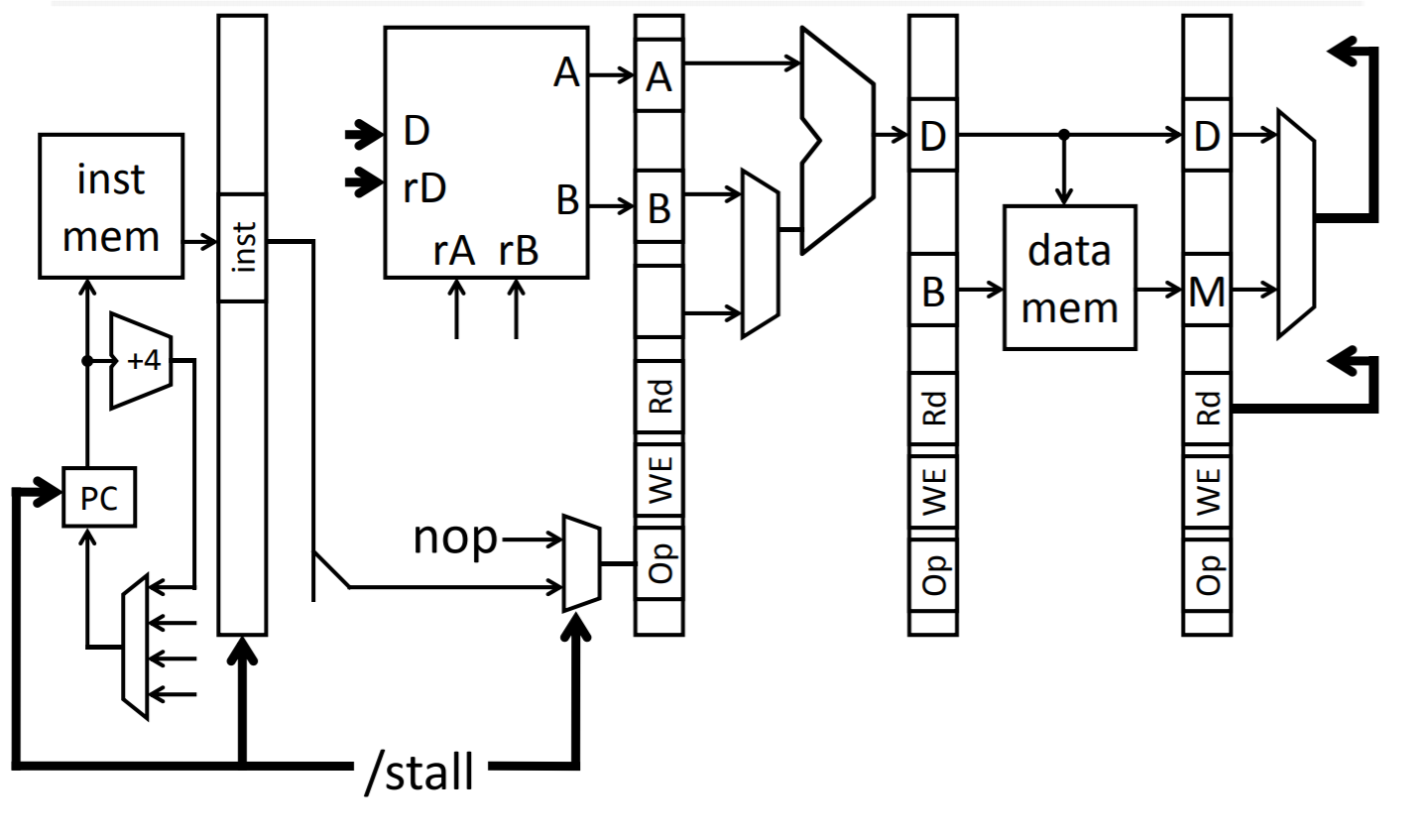


Figure 2: **MIPS Processor with Stall Calculation Module**

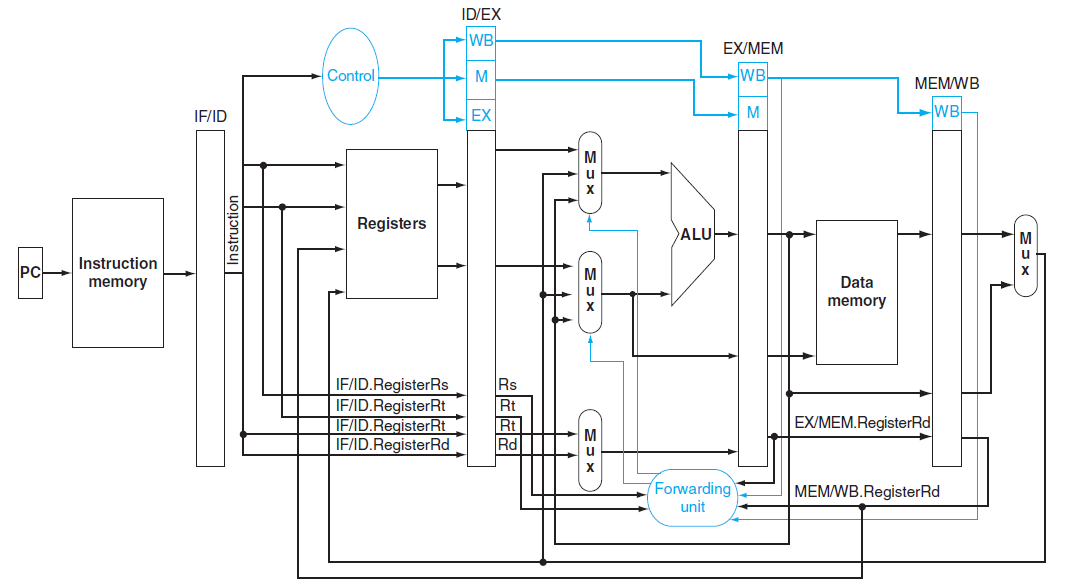


Figure 3: **MIPS Processor with Forward Chaining Module**

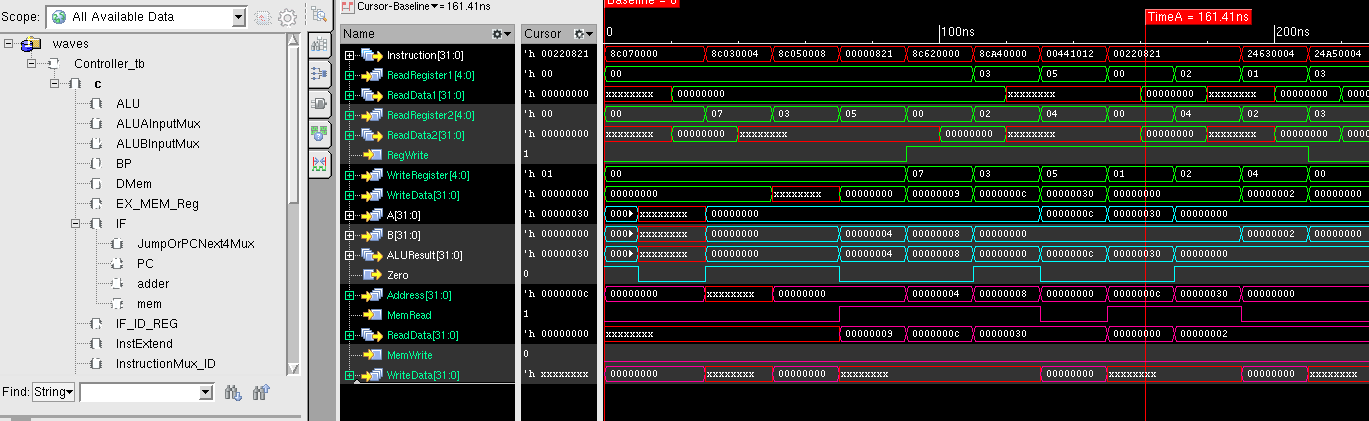
**II. Simulation and Verification Results**

Attached below are waveforms for Dot product commuted using design MIPS processor. The screenshots of simulation have timing completion information for processor with Forward chaining and Hazard control to compare performance.   
The desired result on the waveform is pointed using a yellow square like 🡪

## 

**Figure 4**: Simulation Log for Dot product with Forward Chaining

Test Case 1: Operand 1 = 0, Operand 2 = 2, Dot Product = 0, Result = 0



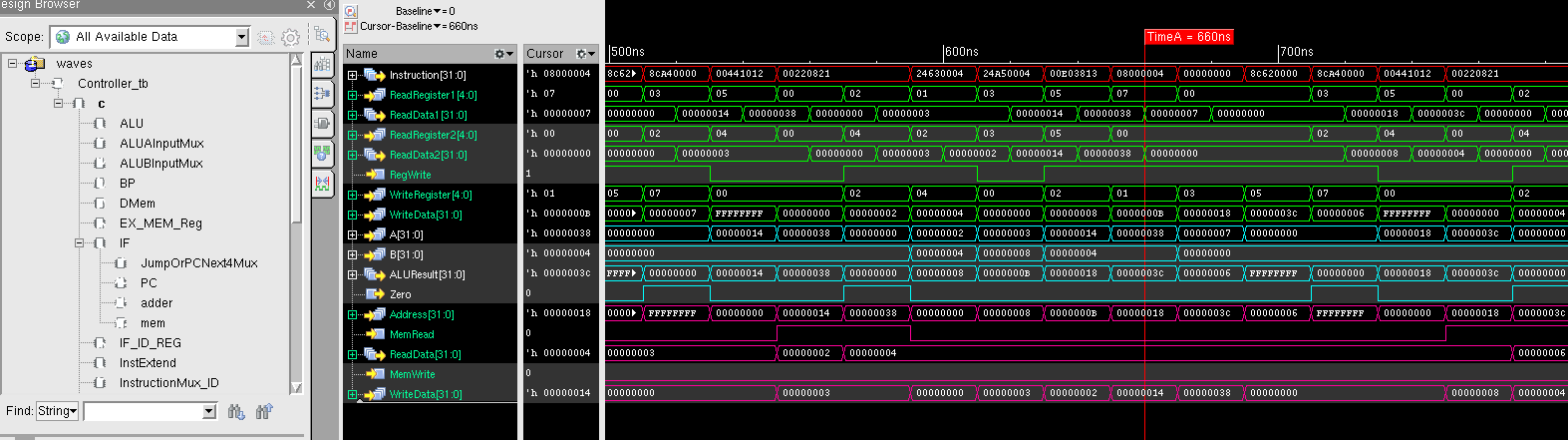
**Figure 5**: Simulation Log for Testcase 1 Forward Chaining

Test Case 2: Operand 1 = 1, Operand 2 = 3, Dot Product = 3, Result = 3



**Figure 6**: Simulation Log for Testcase 2 Forward Chaining

Test Case 3: Operand 1 = 2, Operand 2 = 4, Dot Product = 8, Result = B



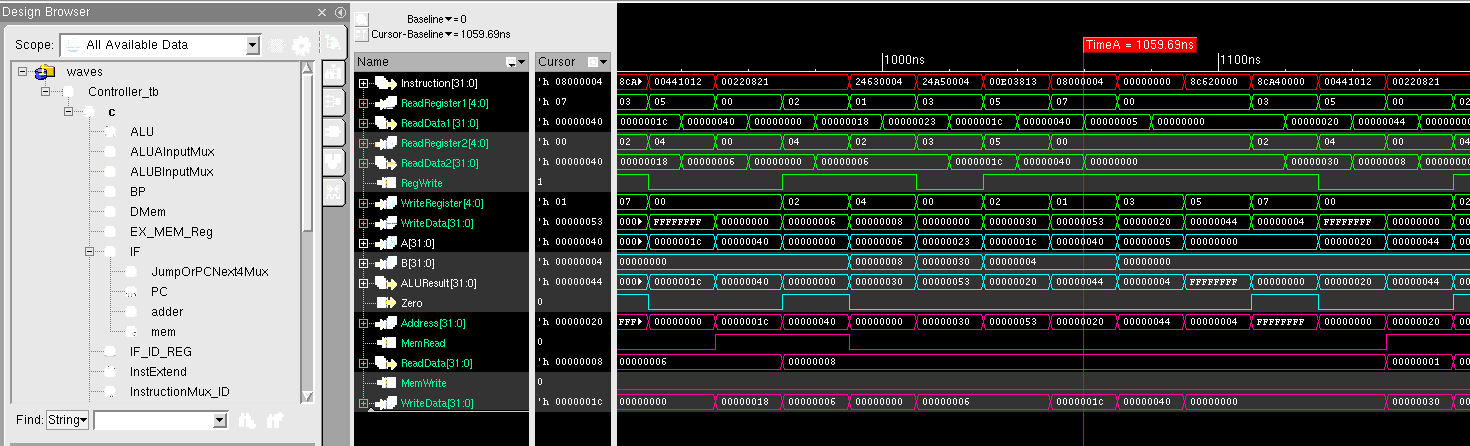
**Figure 7**: Simulation Log for Testcase 3 Forward Chaining

Test Case 4: Operand 1 = 4, Operand 2 = 6, Dot Product = 18, Result = 23



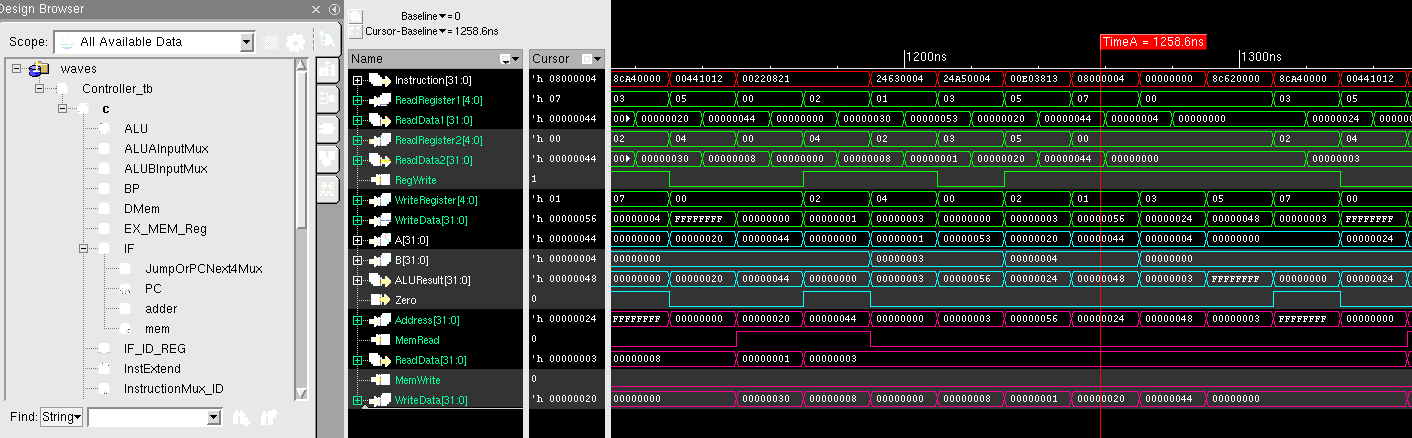
**Figure 8**: Simulation Log for Testcase 4 Forward Chaining

Test Case 5: Operand 1 = 6, Operand 2 = 8, Dot Product = 30, Result = 53



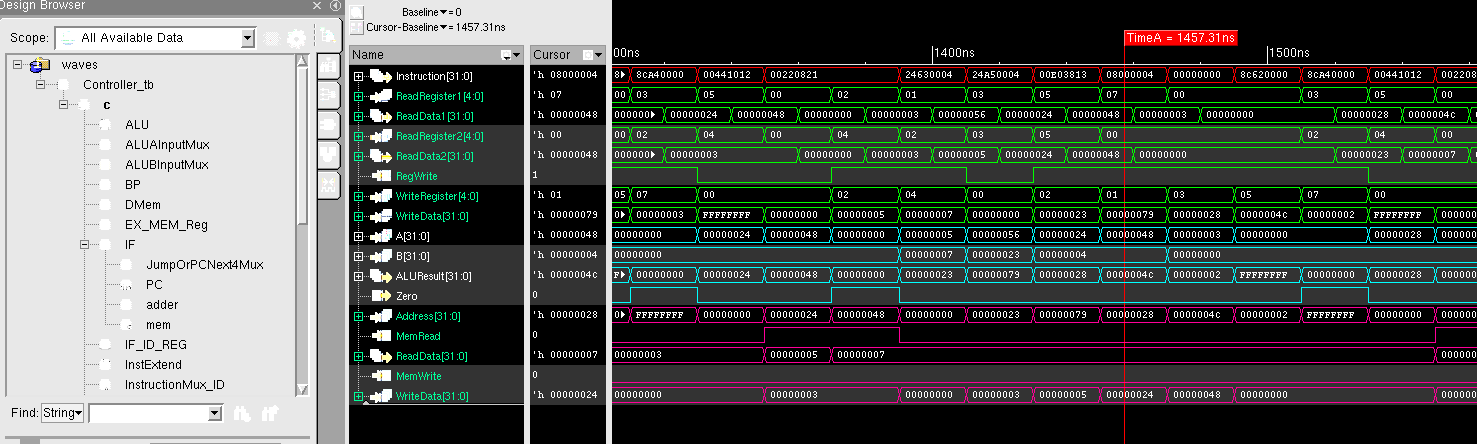
**Figure 9**: Simulation Log for Testcase 5 Forward Chaining

Test Case 6: Operand 1 = 1, Operand 2 = 3, Dot Product = 3, Result = 56



**Figure 10**: Simulation Log for Testcase 6 Forward Chaining

Test Case 7: Operand 1 = 5, Operand 2 = 7, Dot Product = 23, Result = 79



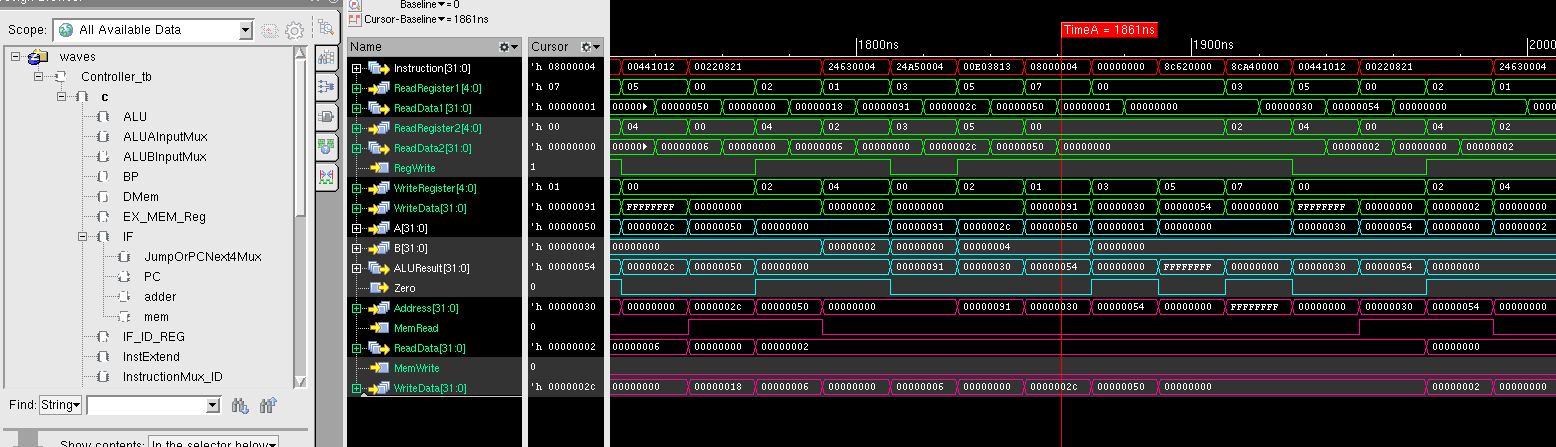
**Figure 1**: Simulation Log for Testcase 7 Forward Chaining

Test Case 8: Operand 1 = 4, Operand 2 = 6, Dot Product = 18, Result = 91

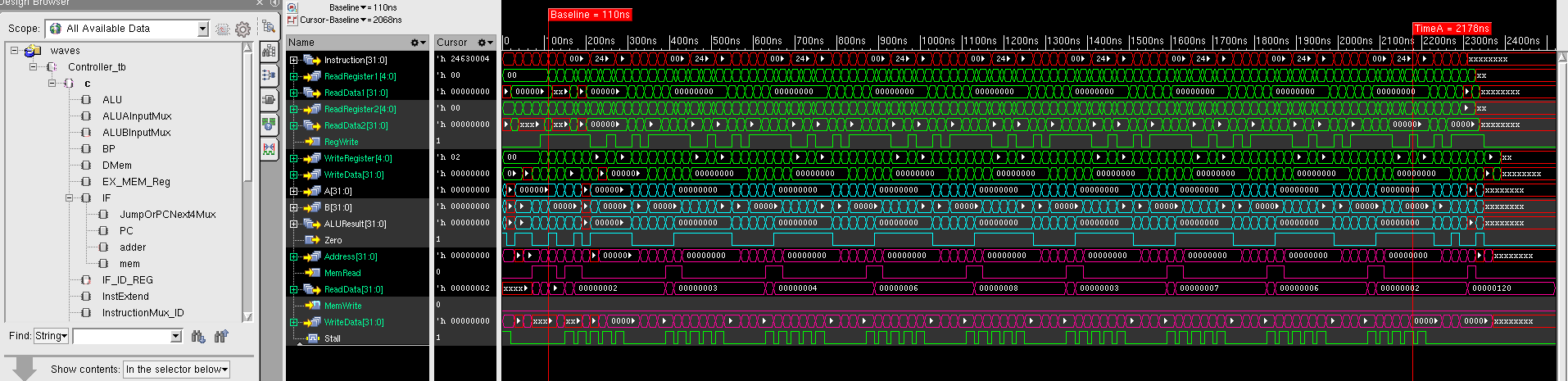


**Figure 12**: Simulation Log for Testcase 8 Forward Chaining

Test Case 9: Operand 1 = 0, Operand 2 = 2, Dot Product = 0, Result = 91



**Figure 13**: Simulation Log for Testcase 9 Forward Chaining



**Figure 14**: Simulation Log for Dot Product with Hazard Control

## 

**Figure 15**: Simulation Log for Testcase 9 with Hazard Control

## III. Conclusion

In practice, I have successfully accomplished building of a 32-bit MIPS processor with pipeline functionalities. Data Hazard and control hazards are resolved successfully by using stalls. In the second stage of design Forward chaining was successfully implemented. The design demonstrates implementation of MIPs CPU capable of handling various R-type, J-type and I-type of instructions.

Also, it can be seen forward chaining gives an optimized processor. Dot product was used as test bench instruction. The processor completes the instructions in 2330 ns by stalling and 1620 ns in forwarding mode, which is 30% gain in performance.

This project provided a vital chance to acquire hands on knowledge on MIPS five stage pipeline processor. The challenge of debugging and fixing the problem helped get more practical knowledge on the subject.