**EE 275**

**ADVANCE COMPUTER ARCHITECTURE**

**MINI PROJECT 3 - MIDTERM 1**

**MIPS ISA DOT PRODUCT BENCHMARK**

**WITH “FORWARD CHAINING”**

**TEAM MEMBERS:**

* **JAIMIL PATEL - 014521416**
* **BHAUTIK PATEL - 014371110**
* **ANKITA MOHOLKAR - 013772382**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| SR.NO. | NAME | PAGE NUMBER |
| 1. | Abstract | 3 |
| 2. | Introduction | 4 |
| 3. | History | 5 |
| 4. | Design | 6 |
| 5. | Simulation (Theory of Operation) | 9 |
| 6. | Performance Comparison | 9 |
| 7. | Conclusion | 10 |
| 8. | Simulation Results | 11 |
| 9. | Code | 12 |
| 10. | References | 18 |

**ABSTRACT**

The”main objective of this project is the learn the designing and implementation of a program which utilizes MIPS ISA along with Dot Product program which also uses Forward Chaining. So, starting with the abstract we can try to understand how the process works in MIPS ISA program when we use the feature of forward chaining into it.

Present era of SOC’s contains analog, digital and mixed signal components housing all on the same chip. In such complex environment, a processor plays a vital and vivid role. As we can notice that the computer architecture industry is shrinking at a rapid rate to sub-micrometer technology node, which leads to a huge scope and space for undesirable hazards in processors. And these hazards are extremely dangerous and can ultimately lead to disturbances in area, power and timing which can deviate requirements from the standard desired qualities. Among multiple types of RISC CPU architecture, MIPS is one of the most successful micro-processor and we are focused on improving its performance in this project through Verilog code by using Dot Product and Forward Chaining mechanism.

A Reduced Instruction Set Computer (RISC) is a microprocessor that is designed to perform a small set of instructions operating at higher speed. The project deals with the concepts of computer architecture, understanding the flow of data and control, enhancing the Verilog coding skills, strong understanding of digital logic and using this to design a functional architecture. It incorporates terminologies, techniques and skills gained in the class. Also, to deal the hazard issues, Forward Chains and Hazard detection units are included.

**INTRODUCTION**

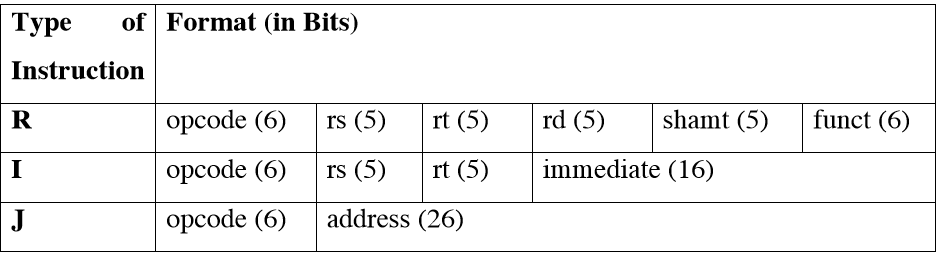
In the very broad field of computer architecture, when several companies have occupied the market, they use one similar thing for MIPS architecture. It is one of the most commonly and most popular technology, being employed in today’s technological world.

MIPS stands for Microprocessor without Interlocked Pipeline Stages and is a reduced instruction set computer (RISC) instruction set. It was developed 33 years ago in the year 1985, by MIPS Technology, a company which was known as MIPS Computer Systems earlier [2]. Nowadays there are multiple versions of MIPS, which includes MIPS I, II, III, IV and V; which also has several sub-divisions within inside it. As stated above, MIPS processors are widely used in embedded systems, which includes residential gateways and routers. Earlier MIPS architecture was solely designed for general computational purposes. Discussing about the architecture of MIPS ISA set, it currently supports up to 4 coprocessors. In its terminology, CP0 is called the System Control Coprocessor, CP1 is an optional Floating point unit (FPU), CP2 and CP3 are optional implementation defined coprocessors. A very common example utilizing MIPS architecture and its components are the PlayStation video game console, in which they are used to accelerate the processing of 3-Dimentional computer graphics, for obtaining a better and good output along with better graphic conditions of the 3D images and scenarios.

MIPS microprocessor with RISC architecture have been used widely in the networking equipment, communication equipment and even in entertainment industry and other fields. Cisco routers, IBM network printer, 4K, 5K, 8K and 9K series of HP laser printer and Sony’s PlayStations and the list goes on, where the MIPS microprocessor is utilized successfully to enhance the performance and other desirable tasks. Also along the time lane, the MIPS architecture has been improved greatly, to make it more adapting and more great output oriented with context of industrial usage; ranging from earlier MIPS16/ MIPS32 architecture to today’s MIPS 64 architecture.

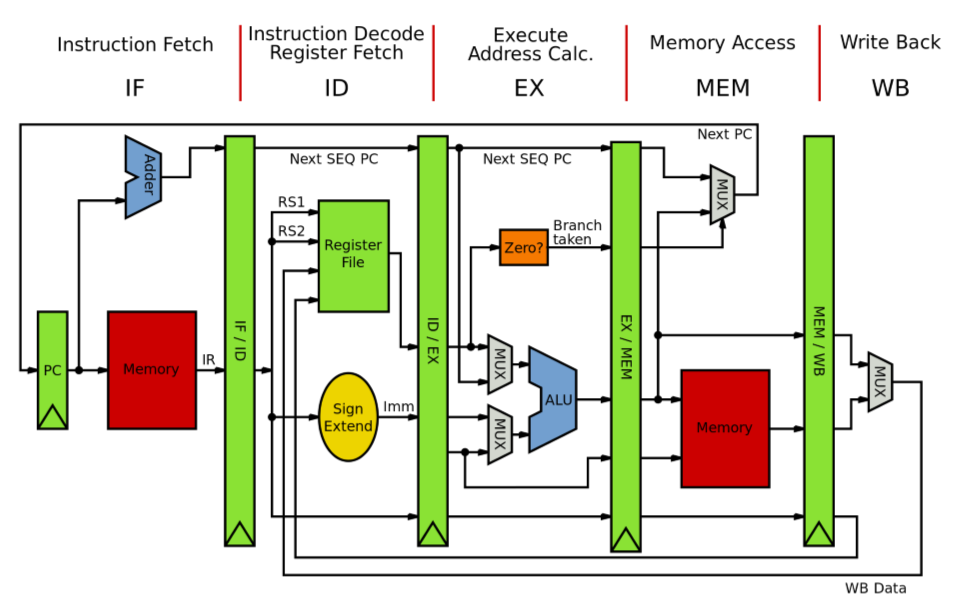
**HISTORY**

The history of MIPS architecture ranges backs to 33 years in the year 1985. It was developed by a company known as MIPS Technology, which was earlier known as MIPS Computer Systems. The very first version of the MIPS architecture was designed for company’s initial R2000 microprocessor. They both were introduced in the same year. Later when MIPS II was introduced, MIPS was renamed to MIPS I, to make a differentiation between the version number and sequence. 8It accepts three different types of instruction formats: R, I and J. Here every instruction starts with a 6-bit opcode. Along with the opcode, they also specify the three registers; which includes a shift amount field; and a function field; I-type instructions gives two registers and a 16-bit immediate value and J-type instructions follow an opcode which has a 26-bit jump target. All these three formats can be illustrated in a tabular format as below-



**DESIGN**

The designing of the MIPS data path is stated as below; and the 5-staged pipelined MIPS data path is demonstrated below; along with all the steps are stated”below:



There”are 5 stages in this RISC architecture and are stated one by one below –

1. Instruction Fetch (IF)

In this stage PC is sent to instruction memory and the current instruction is fetched from memory as well update the PC to next in sequence by adding 4 to the PC (PC = PC+4).

1. Instruction Decode (ID)

Instruction Decode is the second stage of MIPS pipeline. In this stage, the instruction gets decoded and the registers are read as specified in instruction. If there is a branch instruction, the registers are compared as they are read. Moreover, Sign extends the offset field if it is needed. Compute the possible branch target address. Decoding can be done in parallel with reading the registers since the register specifiers at a fixed location.

1. Execute (EX)

In this stage, ALU operations based on the instruction type. In terms of memory instructions, it adds base address and offset to acquire effective address. For register –register operations, as per the ALU – opcode it performs addition, subtraction as it is needed. It also performs operation for register – immediate ALU instructions.

1. Memory access (MEM)

In this stage, load and store instructions are being performed. If it is a load instruction, then it reads an effective address from the memory and in the case of store instruction it writes the data from register into memory.

1. Write Back (WB)

This is the last stage and it performs register – register ALU instruction or LOAD instruction to write the result in to register file (at ID stage), to check whether it comes through load instruction or from ALU when it is a case of ALU instruction.

**Pipelining and Forward Chaining**

A picture containing cabinet, white, sitting, old

Description automatically generated

**Forward Chaining** is the process which works on the basis of available data to make certain decisions. Operand forwarding (or data forwarding) is an optimization in pipelined CPUs to limit performance deficits which occur due to pipeline stalls. A data hazard can lead to a pipeline stall when the current operation has to wait for the results of an earlier operation which has not yet finished. For instance, Data can be taken from memory stage of a particular instruction, before it goes through write-back stage. This avoids stalling. Forward chaining makes it possible for the result of an operation to become available to all functional units without first going through a register. It allows a direct copy of the result of an operation to be given to all the functional units waiting for that result. In other words, a currently executing instruction can have access to the result of a previous instruction before the result of the previous instruction has actually been written to an output register.”

**Theory of Operation**

The code is written in Verilog HDL. The simulation is done successfully with the proof of completion of the task shown by the following simulation waveforms.

Dot Product = Σ𝐴 (𝑖) ∗ (𝑖);

Where A (i) and B (i) are Signed and Unsigned Numbers(vectors) respectively.

The program is given as below –

*addu $r1 $r0 $r0 ; result = 0*

*loop:*

*beq $r7 $r0 done ; done looping?*

*lw $r2 0($r3) ; load a element*

*lw $r4 0($r5) ; load b element*

*mul $r2 $r2 $r4 ; assume this is 1 instr.*

*addu $r1 $r1 $r2 ; result += (\*a) \* (\*b)*

*addiu $r3 $r3 #4*

*addiu $r5 $r5 #4*

*addiu $r7 $r7 #-1*

*j loop*

*done:*

*jr $r31*

**PERFORMANCE COMPARISON**

The”performance of Vector Dot Product is amazing and the number of clock cycles increases linearly with the value of n.

The results are as below:

For length of vector=6, # clock cycles: 69;

For length of vector=5, # clock cycles: 59;

For length of vector=4, # clock cycles: 49;

For length of vector=3, # clock cycles: 39;

For length of vector=2, # clock cycles: 29;

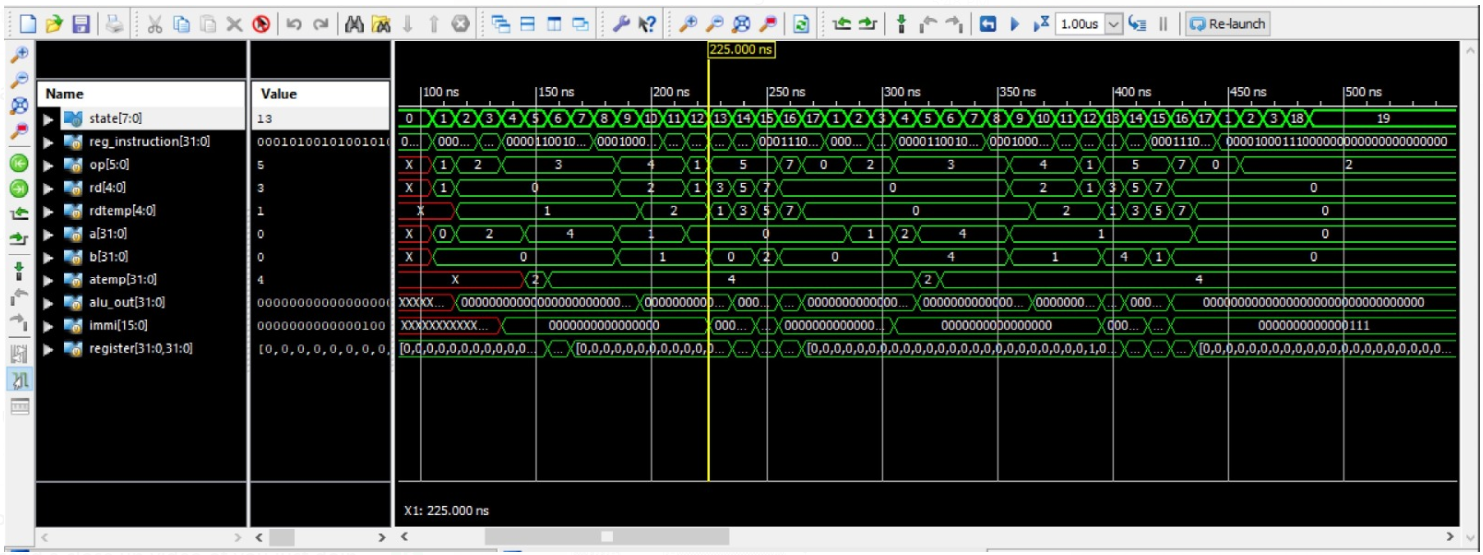
For length of vector=1, # clock cycles: 19.

**CONCULSION**

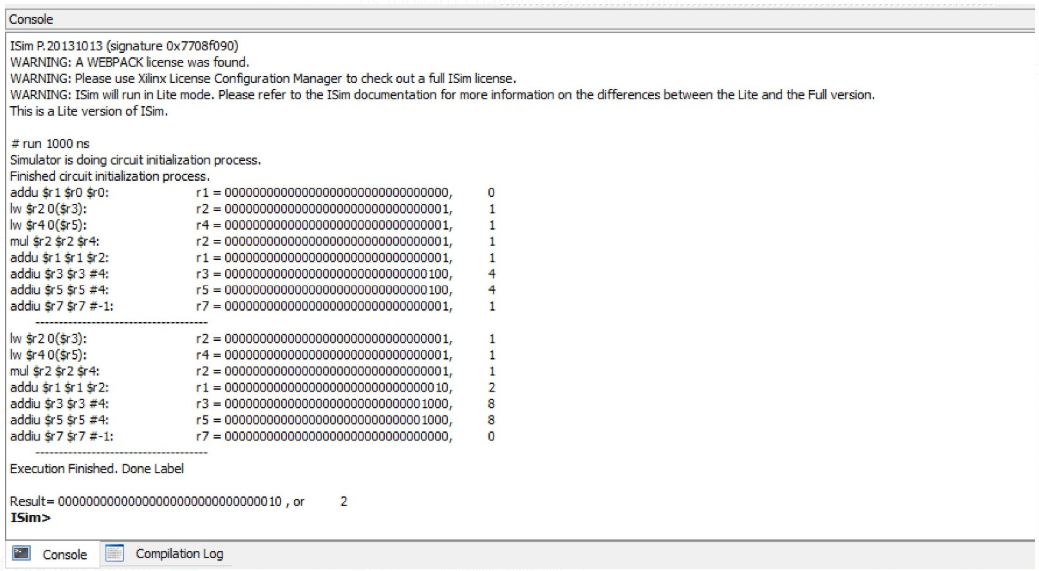
In our project, we have successfully demonstrated the 5 – stage MIPS ISA which executed dot product program. Several data hazard and control hazards are also successfully resolved. Designing hazard detection unit to overcome the data dependencies was critical task and it was implemented successfully. The project involved wide variety of logics to consider during the design. The performance evaluation results showed that the number of clock cycles taken to execute ‘Dot Product’ program is linearly proportional to the value of n. Forwarding and Hazard detection units not only helped to solve data dependency problems, but also improved the performance of the processor; i.e. number of clock cycles to execute any program will be less than the execution without Forwarding and Hazard detection units.

We have successfully achieved the goals for our project by reducing the time spent on the process of addition of two inputs. Also, we have successfully reduced the clock cycles during which the second input is considered for the processing. This is done with the help of pipelining and we have understood the concept of pipelining and implemented it in our project.”

**WAVEFORMS**



**SIMULATION**



**CODE**

**A screenshot of a cell phone

Description automatically generated**

A close up of a piece of paper

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of text

Description automatically generatedA screenshot of a cell phone

Description automatically generated

**References**

[1] <https://cseweb.ucsd.edu/classes/wi14/cse141/pdf/05/04_Elsevier_pipeline.ppt.pdf>

[2] https://minnie.tuhs.org/CompArch/Resources/mips\_quick\_tutorial.html

[3] https://www.ijert.org/design-and-simulation-of-pipelined-double-precision-floating-point-addersubtractor-and-multiplier-using-verilog

[4] DESIGN & SIMULATION OF A 32-BIT RISC BASED MIPS PROCESSOR USING VERILOG. DO - 10.15623/ijret.2016.0511030

[5] Design of High performance MIPS-32 Pipeline Processor. DO - 10.13140/RG.2.1.4883.6567