MIPS Pipeline Processor

Computer Architecture And Microprocessors(CSN221)
Indian Institute Of Technology, Roorkee

Shourya Shukla
Arnesh Agrawal
Mahak Bansal

Karma Dolkar
Jyotsnaa Gupta
Tanya Kumari

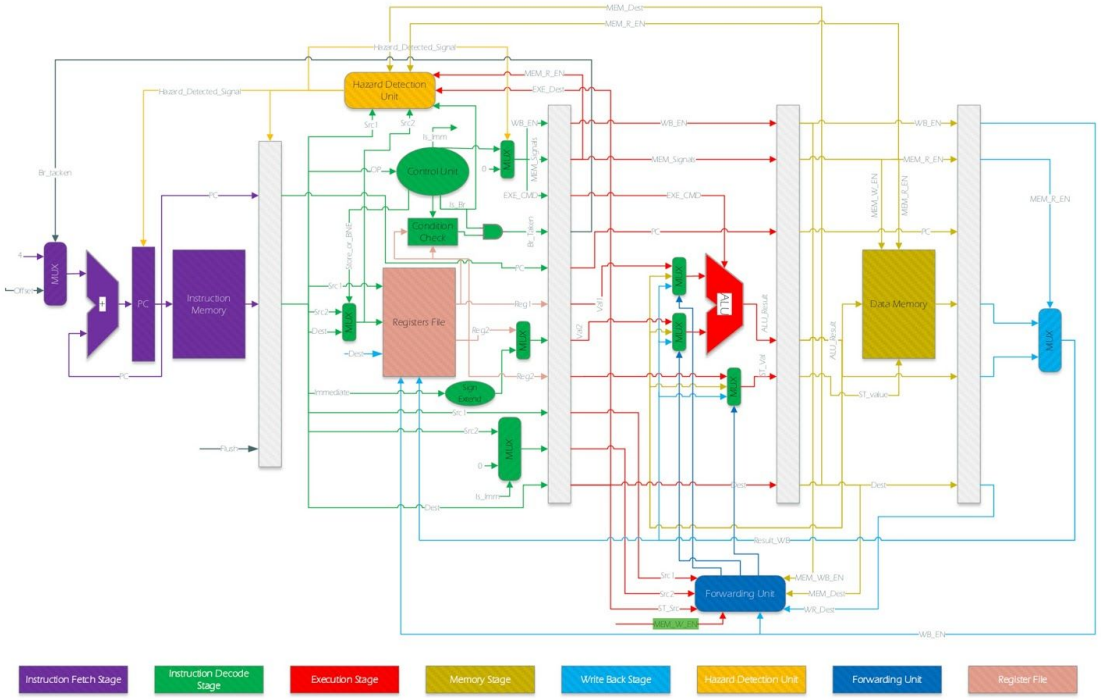
Abstract

To increase CPU performance, we can either have faster hardware, or have a mechanism whereby many instructions are simultaneously executed. Pipelining is a process of arrangement of hardware elements of the CPU to increase its overall performance. It is an implementation technique that allows overlapping of multiple instructions during execution.Today, pipelining is the primary implementation technique used to make fast CPUs. Simultaneous execution of many instructions takes place in a pipelined processor. But, there are a few drawbacks of said process, which include complexity and pipeline stalling due to multiple hazards.

MIPS is a RISC ISA that is used for computing. Through this project, we aim to discuss and implement pipelining using the MIPS architecture.

We have designed a 32-bit MIPS Pipelined Processor to serve the purpose.

A diagram illustrating a 5-stage MIPS Pipeline Processor:



Why this topic caught our intrigue

Pipelining is interesting for a vast multitude of reasons. Primarily, pipelining helps execute multiple instructions simultaneously. Therefore the time it takes to execute an instruction is the same, the exciting part is that multiple instructions are executed at the same time.

A workshop (on Parallel Computing) conducted by Nvidia in our campus under the aegis of our professor Dr. P Sateesh Kumar also played a major role in motivating us to choose this

topic because we felt a bit familiar with the keyword ‘Pipelining’ and hence decided that it would be easier to approach the topic.

A fun trivia about MIPS is that the application processing (or host processing) for most consumer devices (like audio/video player) is usually handled by a programmable core such as a MIPS processor. MIPS is an ISA that helps physically realize (i.e. implement) pipelining in this case. Using something as resourceful, intriguing, and as widely used as MIPS is an interesting facet of our project.

Methods used to evaluate the topic

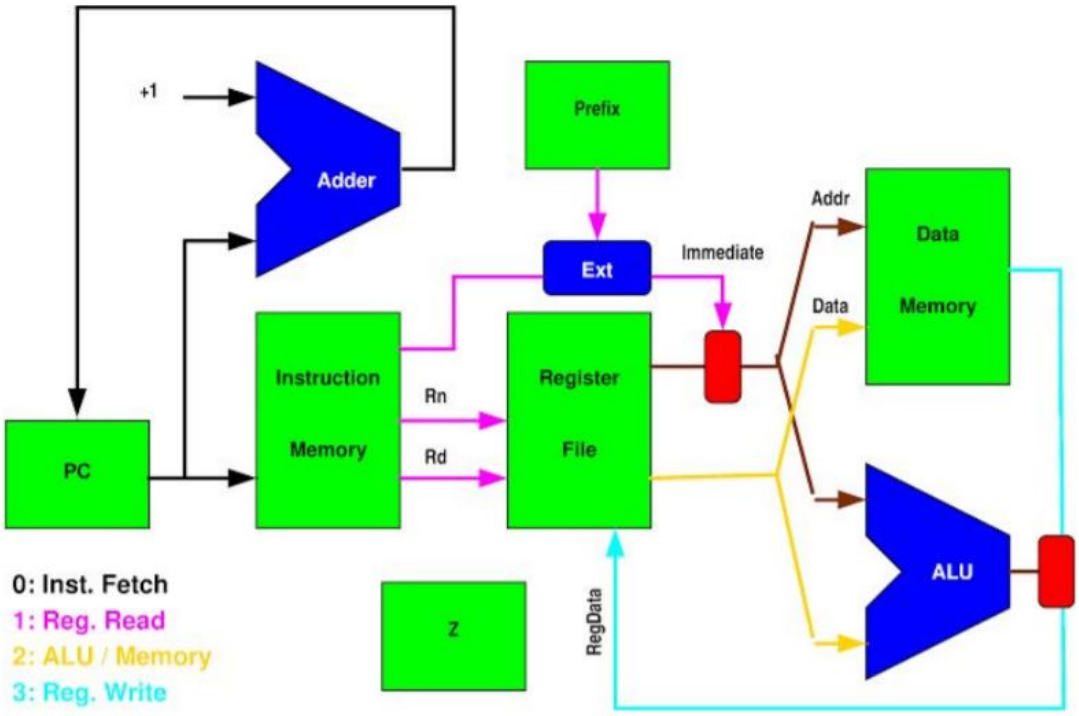
We have used “Microsoft Visual Studio” to write our code in the Verilog programming language. Also, we have used Git for version control and GitHub as a platform to host our code online. GanttProject has been used to create a project timeline and hence bring a better workflow in the project.

We will first work with design and execution of single-cycle MIPS processor and 32-bit 5-stage pipelined MIPS Processor in C++ and then analyse and compare both.

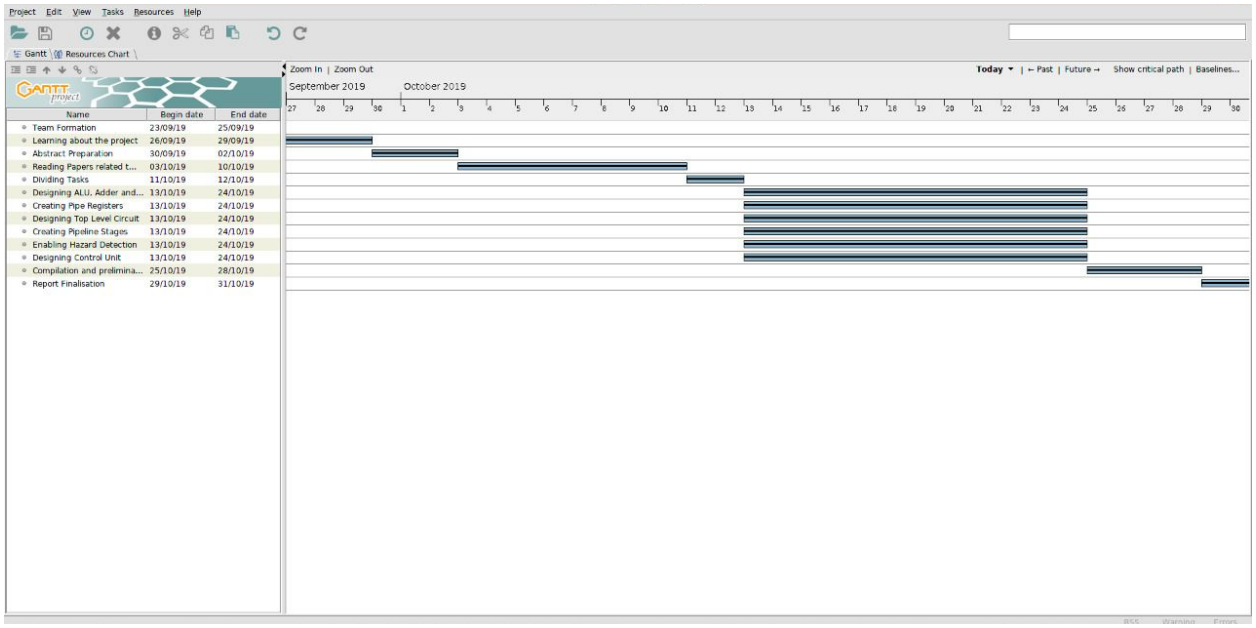
So the implementation of our project consists of **two parts**:

In the **first part**, we are designing and implementing single-cycle MIPS processor. From the I, the single-cycle datapath with the control unit of the MIPS Processor is obtained.

In the **second part**, pipelined registers are added to the single-cycle datapath and make it become a pipelined Processor. Special designs such as forwarding, stall control, and flush control unit are needed to solve hazards in the pipelined MIPS Processor.
The design flow for the pipelined processor:



GanttChart



Result

This design consists of five pipeline stages which are: **Fetch stage, Decode stage, Execute stage, Memory stage and Write back stage**.

The first problem with the single-cycle MIPS is wasteful of the area which only each functional unit is used once per clock cycle. Another serious drawback is that the clock cycle is determined by the longest possible path in the Processor. Thus, the pipelined MIPS came out to solve those problems by exploiting most functional unit in one clock cycle and improving

the performance by increasing instruction throughput. However, the pipelined MIPS also faces challenges such as control and data hazards.

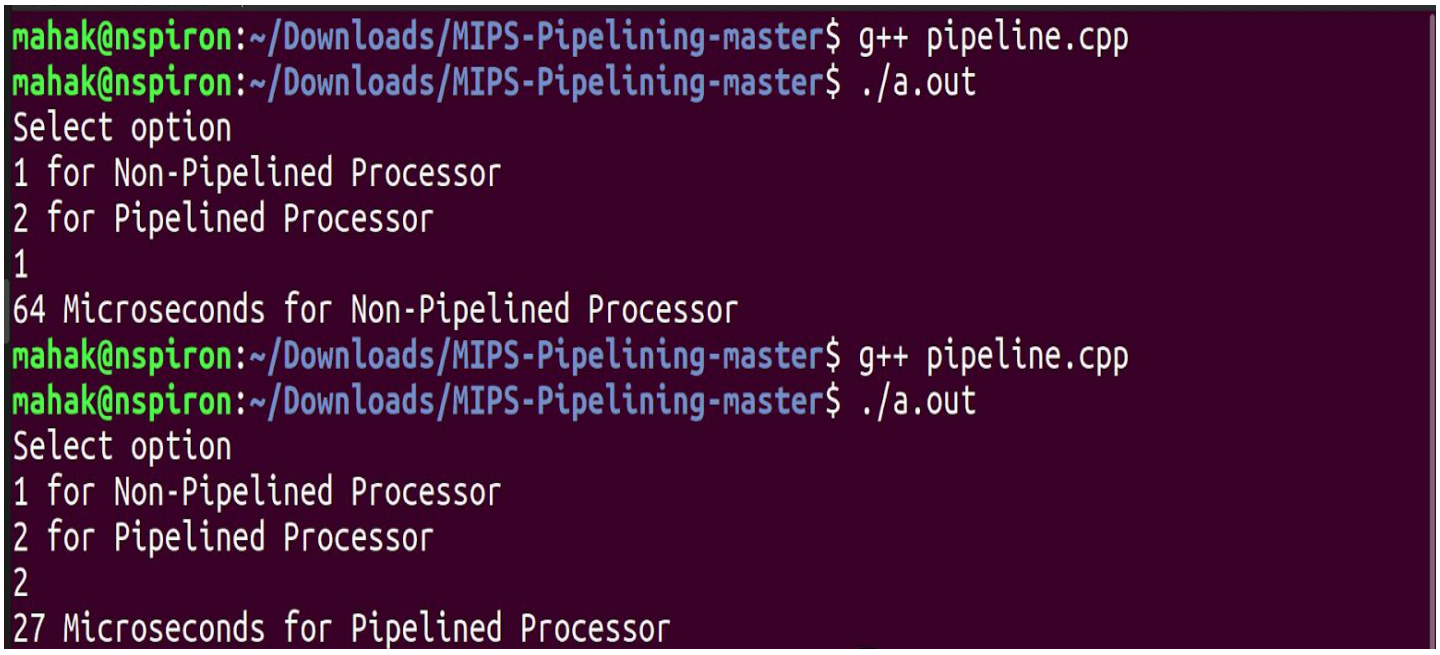
Here is a comparison between a Single-cycle MIPS and a Pipelined MIPS[2]:

processor	Program execution time	No. of clock cycles	Clock period	CPI	Speedup	FPGA area (Number of Slice Registers)
Single-cycle MIPS	1800 ns	90	20 ns	1	1	2,208
Pipelined MIPS	1125 ns	112.5	10 ns	1.25	1.6	2,946

Hence, we observe that even in practice, the Pipelined MIPS processor clearly outperforms the Single-cycle MIPS processor.

Final Outputs of C++ code

These are our outputs of the C++ code we designed. The pipelined and non-pipelined times are mentioned in the screenshot.



References

Links of related research papers :

1. https://ieeexplore.ieee.org/abstract/document/6578243
2. https://link.springer.com/article/10.1007/BF029433
3. https://ieeexplore.ieee.org/abstract/document/5403912/
4. https://www.researchgate.net/profile/Norman_Jouppi/publication/234795328_MIPS_A_microprocessor_architecture/links/00b495185e2fb79958000000/MIPS-A-microprocessor-architecture.pdf