IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, December 2024

# Design and Implementation of a MIPS Datapath

**Mohd Hakimi Zohari[1], Mohd Shamian Zainal[1], Azlina Bahari[1]**
Department of Electrical Engineering Technology, Faculty of Engineering Technology
Universiti Tun Hussein Onn Malaysia, 84600 Pagoh, Johor, Malaysia[1]

**Abstract:** *Finally, the integration of these individual components culminates in the execution of the complete MIPS datapath. The MIPS (Microprocessor without Interlocked Pipeline Stages) architecture serves as a widely recognized and utilized instruction set architecture. Implementing the complete datapath involves orchestrating the flow of instructions through the CPU, from instruction fetch to execution and memory access, showcasing the culmination of our efforts in crafting a functional and efficient processor. In essence, this project not only provides a hands-on experience in HDL-based digital design but also immerses us in the intricacies of CPU architecture, emphasizing the importance of each component in achieving a harmonious and high-performance computing system..*

**Keywords:** MIPS, Computer Architecture, HDL Language, Computer Design, CPU

## I. INTRODUCTION

In the realm of computer architecture and digital design, the creation of a single cycle or pipelined Central Processing Unit (CPU) stands as a fundamental yet intricate challenge. This project aims to explore and implement a CPU using Hardware Description Language (HDL), a specialized language that enables the description and simulation of digital circuits. The significance of designing a CPU lies in its role as the core component responsible for executing instructions in a computer system [1]. By delving into the intricacies of a single cycle or pipelined CPU, we embark on a journey to understand the architectural intricacies that govern modern processors. The project is structured around key components integral to the functionality of a CPU. Starting with the basic building blocks, we delve into the creation of an SR latch, a foundational element for storing binary information. Moving forward, the design includes the development of a Register, the heart of data storage in a CPU [2].

Furthermore, an Arithmetic Logic Unit (ALU) is crafted, responsible for executing arithmetic and logical operations critical for computation. State elements on the datapath, including multiplexers (Mux) and Sign Extension Units, are created to manage data flow efficiently within the CPU [3].A pivotal aspect of the project involves the creation of a Data Memory Unit, responsible for handling data storage and retrieval. Finally, the culmination of these individual components leads to the execution of the complete MIPS datapath. As we progress through this project, we will not only witness the synthesis of a functional CPU but also gain insights into the meticulous design considerations required to achieve seamless and efficient instruction execution in a digital computing environment.

## II. METHODOLOGY

The Set-Reset latch, abbreviated as the SR latch, is a fundamental building block in digital electronics. It operates as a simple memory cell, capable of storing binary information. The SR latch is composed of two NOR gates, and its behavior is controlled by two inputs: Set (S) and Reset (R). When the Set input is activated, the latch stores a '1', and when the Reset input is triggered, it stores a '0'. Crucially, when neither input is active, the SR latch maintains its current state. This latch is classified as a bistable multivibrator due to its ability to exist in two stable states. Its straightforward design and functionality make it a cornerstone in more complex digital circuits, serving as the basis for various sequential circuits and memory units. The SR latch provides a crucial introduction to the principles of memory storage in digital systems, forming the foundation for further exploration into the realms of registers, flip-flops, and advanced memory components.

A register, in the context of computer architecture, serves as a small yet high-speed storage component embedded within the central processing unit (CPU) [4]. It functions as a crucial element in facilitating the active processing of data. Unlike larger, slower memory components like RAM, registers are designed for swift data access and temporary

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, December 2024

storage during the execution of machine instructions [5]. Registers possess the essential characteristic of speed, outpacing other forms of memory due to their direct integration within the CPU. This expedited access is instrumental in enhancing the overall speed and efficiency of a computer's operations. These storage elements are employed for a variety of purposes within the CPU. One primary function of registers is data manipulation. They hold operands required for arithmetic and logical operations conducted by the CPU. As the CPU performs computations, data stored in registers serves as inputs, and the results of these operations are stored back into registers.

A 32-bit Arithmetic Logic Unit (ALU) is a crucial component within a processor, tasked with executing a diverse set of arithmetic and logical operations. The "32-bit" designation specifies that the ALU processes data in chunks of 32 binary digits (bits), aligning with the processor's data width. The primary functions of a 32-bit ALU encompass fundamental arithmetic operations, such as addition and subtraction, and logical operations, including AND, OR, and XOR. Additionally, the ALU excels at bitwise operations, allowing manipulation of individual bits within the binary data. This broader processing capability is particularly advantageous in modern computing environments, where larger data sizes and intricate calculations are prevalent [6].

```
/* Testbench for NAND-based SR latch */

module NANDSR_latch_tb;
  reg  R, S;
  wire Q, Qnot;

  // instantiate the latch:
  NANDSR_latch dut(R,S,Q,Qnot);

  initial
    begin
      // specify test points to be capt
      $dumpfile("latch.vcd");
      $dumpvars(1);
      $display("S R !Q Q");
      $monitor("%b %b %b  %b",S,R,Qnot,

      // excercise the lines to test:

      R=1'b1;  S=1'b1;
  #5  R=1'b1;  S=1'b0;
  #5  R=1'b1;  S=1'b1;
  #5  R=1'b0;  S=1'b1;
  #5  R=1'b1;  S=1'b1;
  #5  R=1'b0;  S=1'b0;
  #5  R=1'b1;  S=1'b1;
  #5  R=1'b1;  S=1'b1;

  #5  $finish;
  end
endmodule
```

Fig. 1 VHDL Coding

## III. RESULT AND DISCUSSION

Based on the coding using VHDL application, the result is shown in Fig. 2 below. The figure shows the results from SR latch based VHDL coding. The coding is using VHDL code and developed based on SR latch implementation. The graphs are results that come out after executing the code. The graphs are shown in the figure.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, December 2024

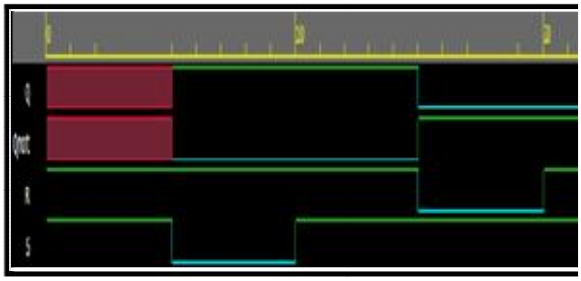

Fig. 2 Result for SR Latch

Another coding is using VHDL application for Register design and the result is shown in Fig. 3 below. The figure shows the results from Register based VHDL coding. The coding is using VHDL code and developed based on Register implementation. The graphs from the results are shown in the figure.

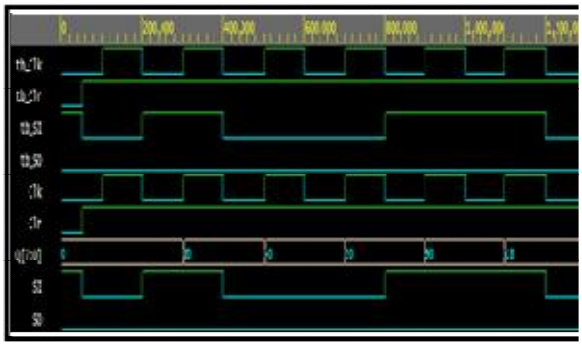

Fig. 3 Result for Register Design

The results of the Arithmetic Logic Unit (ALU) design using VHDL coding are presented in Fig. 4 below. This figure illustrates the output generated by the ALU, which was developed through VHDL code. The graphs displayed represent the results obtained after executing the ALU implementation in VHDL.

Fig. 4 Result for Arithmetic Logic Unit (ALU) Design

In this project, we utilized the EDA Playground platform, a free online tool designed to facilitate the learning process of hardware description language (HDL) code development. EDA Playground allows users to edit, simulate, analyze, synthesize, and share their HDL code. It's particularly useful for small prototypes and demonstrations, although it may not be suitable for extensive FPGA or ASIC design projects.

The initial focus of the coding involved the implementation of an SR latch to generate a desired output waveform. The SR latch, a cornerstone in digital electronics, provides insights into memory storage principles. Its bistable nature, controlled by Set (S) and Reset (R) inputs, enables stable states. The truth table outlines its behavior, emphasizing the

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, December 2024

importance of avoiding the forbidden state (S=1, R=1) for reliable operation. The truth table is based on SR Flip Flop circuit as shown in Fig. 5.

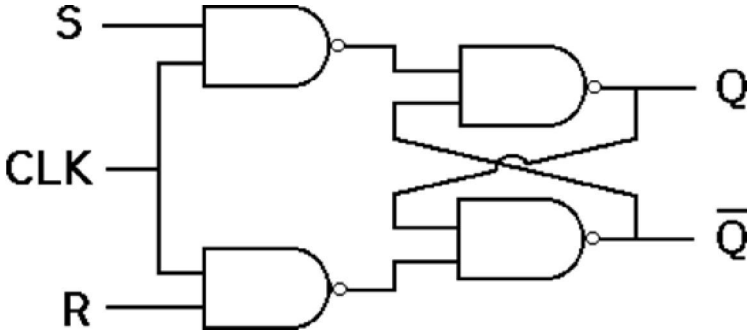


Fig. 5 SR Flip Flop Circuit

Subsequently, we developed a Register, acting as a quick memory storage for data and instructions ready for execution. The SISO register, a type of shift register, introduces the concept of shifting data with a single input and output. The provided truth table illustrates its behavior during various input scenarios, showcasing the versatility of this register type. Moving forward, an Arithmetic-Logic Unit (ALU) was created. The 32-bit ALU, a vital processor component, handles arithmetic, logical, and bitwise operations. Its versatility in processing 32-bit data chunks contributes to the efficiency of modern computing. While the detailed code is provided in the appendix, the ALU's significance lies in its impact on overall processing power, emphasizing speed and computational capabilities.

The implementation of a state element in the datapath, a digital circuit storing and retrieving data in response to input and clock signal changes, was demonstrated. State elements, including the 4-bit State Register, flip-flops, and additional components, form the backbone of the CPU's internal state management. Initialization, state update logic, and output mechanisms ensure coherent data flow and control signal handling within the datapath [7].

Following that, a Data Memory Unit, a digital component for data storage in a computer, was defined. The Data Memory Unit (DMU) serves as a bridge between the CPU and external memory. Its memory array, write logic, and read logic enable data storage and retrieval. The detailed design explanation and simulation flow demonstrate the DMU's functionality in writing and reading data from specified memory addresses.A sign extension unit was then introduced. The Sign Extension Unit (SEU) addresses the challenge of maintaining sign bit integrity when working with signed numbers of varying bit-widths. Its design, encapsulated in a System Verilog module, ensures correct sign extension by replicating the most significant bit.

The 2-to-1 multiplexer enhances datapath flexibility by dynamically selecting between two inputs based on a control signal. Integrated into the overall datapath architecture, the multiplexer adds responsiveness to changing conditions within the system.In conclusion, this project provides a comprehensive exploration of MIPS datapath components, laying the groundwork for understanding advanced sequential circuits in digital electronics. The detailed discussions, along with the provided code and simulations, contribute to a robust learning experience in digital system design.

## IV. CONCLUSION

In conclusion, this project has been a comprehensive exploration of the design and implementation of key components within a MIPS datapath. The journey began with the fundamental SR latch, providing insights into memory storage principles and the importance of stable states. Moving forward, the creation of registers demonstrated their critical role in CPU architecture, offering high-speed data storage for active processing.The centerpiece of the project, the 32-bit Arithmetic Logic Unit (ALU), showcased its versatility in handling various arithmetic, logical, and bitwise operations. This component is pivotal in determining the overall processing power and efficiency of modern computing systems.

The exploration of state elements on the datapath, including the 4-bit State Register, flip-flops, and additional components, provided a deeper understanding of internal state management within the CPU. Initialization, state update logic, and output mechanisms were discussed to emphasize their roles in maintaining coherent data flow and control signal handling. The Data Memory Unit (DMU) emerged as a fundamental bridge between the CPU and external memory, with functionalities for both writing and reading data from specified memory addresses. The Sign Extension

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, December 2024

Unit (SEU) addressed the challenge of sign bit integrity in signed numbers of varying bit-widths, contributing to the reliability of digital circuits.

Finally, the 2-to-1 multiplexer added a layer of flexibility to the datapath, dynamically selecting between two inputs based on a control signal. Integrated seamlessly into the overall architecture, the multiplexer enhanced the system's responsiveness to changing conditions. This project has not only provided a hands-on experience in designing and coding digital components but also emphasized their interconnections within a larger system. The detailed discussions, code implementations, and simulation results contribute to a holistic understanding of MIPS datapath design, forming a solid foundation for further exploration in digital system architecture.

## V. ACKNOWLEDGMENT

This work was supported by Department of Electrical Engineering Technology, Faculty of Engineering Technology, Universiti Tun Hussein Onn Malaysia (UTHM).The project group consist of Foong Kar Chun, Afiq Luqman Bin Jamal and Muhammad Ammar Hafidz Bin Mohamed Azly.

## REFERENCES

**[1].** Aneliya Ivanova, A Concept of Visual Programming Tool for Learning VHDL, IOP Conference Series: Materials Science and Engineering, 2021http://doi.org/10.1088/1757-899X/1031/1/012120.

**[2].** Quek Wei Chun, Pang Wai Leong, Chan Kah Yoong, Lee It Ee, Chung Gwo Chin, VHDL Modelling of Low-Cost Memory Fault Detection Tester, Journal of Engineering Technology and Applied Physics, 2020.

**[3].** https://doi.org/10.33093/jetap.2020.2.2.3.

**[4].** Kadam, S., & Mali, S. D. (2016). Design of RISC Processor Using VHDL. International Journal of Research - GRANTHAALAYAH, 4(6), 131–138. https://doi.org/10.29121/granthaalayah.v4.i6.2016.2646.

**[5].** Kirat Pal Singh,Shivani Parma,VHDL Implementation of a Mips-32 Pipeline Processor, International Journal of AppliedEngineering Research, 2012.http://dx.doi.org/10.5281/zenodo.33247.

**[6].** X. Ji and L. Chen, Design and Implementation of MIPS Experiment Platform Based on FPGA Device,International Conference on Big Data, Information and Computer Network (BDICN), Sanya, China, pp. 742-745, 2022. http://doi.org/10.1109/BDICN55575.2022.00144.

**[7].** S. P. Ritpurkar, M. N. Thakare and G. D. Korde, Synthesis and Simulation of a 32-Bit MIPS RISC Processor using VHDL,International Conference on Advances in Engineering & Technology Research (ICAETR), Unnao, India, pp. 1-6, 2014. .http://dx.doi.org/10.1109/ICAETR.2014.7012843

**[8].** R. Sharma, V. K. Sehgal, N. Nitin, P. Bhasker and I. Verma, Design and Implementation of a 64-bit RISC Processor Using VHDL,11th International Conference on Computer Modelling and Simulation, Cambridge, UK, pp. 568-573, 2009.http://doi.org/10.1109/UKSIM.2009.30