# Design of Single Cycle MIPS RISC Processor Using Re-timing Technique

Sindhe Sreeja(✉), Gudipati Sneha, Guguloth Ganesh, and Sangeeta Singh

Department of ECE, Vardhaman College of Engineering, Hyderabad, Telangana, India
sreejasindhe17@gmail.com, sangeethasingh@vardhaman.org

**Abstract.** In designing circuits, the speed and power aspects are crucial. Traditional design faces challenges like increased power use, complex timing issues, and inflexibility in changing certain components. Therefore, there's a need for new techniques to make designs more efficient. This paper introduces a fresh approach to increase speed of high-speed systems like the MIPS RISC Processor. It does this by combining advanced methods to adjust the timing of signals with flexible strategies for changing certain components. This combination significantly reduces power usage while making the system perform better. Pipeline Register Re-timing Algorithm, strategically adjusts the placement of certain components to reduce delays in signal movement and the time it takes for the system clock to complete a cycle. This results in an overall improvement in how well the system performs. The design is implemented practically using the Xilinx Vivado tool with the Verilog High Descriptive Language (VHDL). The outcomes of this implementation show a clear reduction in critical path and delay, along with more efficient power usage. This study reveals a 2% decrease in total On Chip power, a 2% reduction in dynamic power, and a 2.5% decline in input signal power.

**Keywords:** MIPS RISC processor · Re-timing · Pipeline register re-timing

## 1 Introduction

The MIPS (Microprocessor without Interlocked Pipeline Stages) architecture is a well-established and influential RISC- based design. RISC, which stands for Reduced Instruction Set Computing. Instead of dealing with complicated instructions, it focuses on doing simple things really fast. Developed by MIPS Computer Systems, it has found widespread use in a variety of applications, from personal computers to embedded systems and networking devices. The MIPS processor is characterized by a streamlined set of instructions that perform basic operations. This simplicity allows for faster execution of instructions. The MIPS architecture utilizes a pipeline structure, where different stages of instruction execution are overlapped. This contributes to improved performance by enabling the processor to handle multiple instructions simultaneously. In MIPS, most instructions operate on register values, and data is loaded from or stored to memory through specific load and store instructions. This design choice enhances instruction


X. Cheng (Ed): BROADNETS 2024, LNICST 601, pp. 189–201, 2025.
https://doi.org/10.1007/978-3-031-81168-5_17

execution speed. MIPS follows a load-store architecture, meaning that most operations involve loading data from memory into registers, performing operations, and then storing the results back in memory. This design simplifies instruction execution. In certain implementations, MIPS processors are designed with a focus on energy efficiency. This makes them suitable for devices with power constraints, such as battery- powered gadgets. MIPS processors find extensive use in embedded systems, powering devices like routers, modems, set-top boxes, and other smart appliances due to their efficient and compact design.

## 2 Literature Survey

The focus is on a 64-bit RISC processor, implying that it processes data in 64-bit chunks, contributing to enhanced computational capabilities [1]. The processor incorporates built-in self-test features, indicating that it possesses the ability to conduct self-assessment procedures. This self- testing mechanism ensures that the processor can verify its own functionality. The paper provides insights into the architecture of the processor, explaining how its components are structured and interconnected. Additionally, it details the data path, which signifies the route that data takes within the processor during operations. The instruction set, comprising the operations the processor can execute, is also outlined. The designed processor is verified using the Xilinx ISE simulator. Verification involves testing the processor's functionality through simulations, ensuring it behaves as intended.

The MIPS (Microprocessor without Interlocked Pipeline Stages) architecture is a well-established and influential RISC-based design [2]. RISC, which stands for Reduced Instruction Set Computing. Instead of dealing with complicated instructions, it focuses on doing simple things really fast. Developed by MIPS Computer Systems, it has found widespread use in a variety of applications, from personal computers to embedded systems and networking devices. The MIPS processor is characterized by a streamlined set of instructions that perform basic operations. This simplicity allows for faster execution of instructions. The MIPS architecture utilizes a pipeline structure, where different stages of instruction execution are overlapped. This contributes to improved performance by enabling the processor to handle multiple instructions simultaneously. In MIPS, most instructions operate on register values, and data is loaded from or stored to memory through specific load and store instructions. This design simplifies instruction execution. In certain implementations, MIPS processors are designed with a focus on energy efficiency. This makes them suitable for devices with power constraints, such as battery-powered gadgets.MIPS processors find extensive use in embedded systems, powering devices like routers, modems, set-top boxes, and other smart appliances due to their efficient and compact design.

A novel design for a 32-bit Reduced Instruction Set Computer (RISC) processor, focusing on optimizing performance and operational speed. The processor employs a multi-cycle architecture, executing each instruction in three distinct stages: Fetch, Decode, and Execution. The Fetch and Decode stages are uniform across all instructions, fostering consistency in the initial processing steps. Notably, the multi-cycle nature of the processor allows for increased flexibility and a streamlined hardware design, contributing to improved efficiency. The 32-bit architecture enhances data processing capabilities, handling information in 32-bit chunks. Through simulations and synthesis using Xilinx 14.7 ISE design suite, the paper validates the proposed design, ensuring its viability for practical implementation. This approach aims to strike a balance between a simplified architecture and multi-cycle execution, ultimately striving for heightened performance and operational speed across a diverse range of instructions [3].

It suggests a solution to this problem, proposing a method that tackles cache coherence without burdening developers. The aim is to maximize the performance of multi-core platforms using chosen RISC-V cores. The paper explains the method's workings, architecture, and hardware implementation in detail. Through prototype development and experiments, the paper demonstrates the effectiveness of this approach, showing its potential to advance the development of efficient RISC-V multi core platforms. The demand for RISC-V multi core platforms has risen, but integrating these cores for optimal performance is challenging due to a lack of cache coherence logic in many open cores [4].

The ultimate goal of these techniques and the MATLAB environment is to achieve a lower clock period and a lower register count while maintaining the original input-output behavior of the circuit. This is done to enhance the overall performance of the sequential circuit. The focus is on techniques such as cut set re-timing, clock period minimization, and register minimization, implemented in MATLAB to achieve improved performance in sequential circuits [5].

The foundation of this algorithm builds upon a one-less-than- previous approach [6]. The algorithm is designed to be straightforward, offering a simple method for multiplying NxN unsigned binary numbers. It utilizes a 2n-1 constant number that is applied recursively to both the multiplicand and multiplier. The re-usability of hardware resources is a key factor contributing to low power consumption. It is suggested that this re-usability also leads to a superior power-delay product when compared to conventional multipliers.

The design employs a register minimization re-timing technique, aiming to optimize the critical path and improve VLSI design metrics such as area, speed, or power. The use of a GUI for component binding further enhances the efficiency of the design process, aiming for minimum design cycle time while meeting specific performance criteria. It compares different adders and multipliers in terms of VLSI design metrics [7].

A fresh perspective on re-timing, a register reconfiguration technique initially proposed by Leiserson and Saxe. Departing from traditional linear programming approaches, this novel method employs loop analysis inspired by network theory to formulate the re-timing problem. By utilizing the circuit's incidence matrix and identifying linearly independent loops to represent diverse register configurations, the algorithm efficiently narrows down the set of re-timing solutions based on design and timing

constraints. This approach offers designers flexibility in selecting an appropriate implementation. The conclusion underscores the algorithm's capacity to evaluate re-timing solutions considering register locations and timing constraints, highlighting its simplicity in formulation and programming compared to other re-timing methods. The potential integration of these algorithms into circuit compilers and interactive design tools is suggested, and future work is hinted at, exploring the synergy of re-timing with other VLSI synthesis and design automation techniques [8].

The significance of re-timing as an optimization technique for sequential circuits within an end-to-end industrial design flow. Re-timing involves strategically moving edge-triggered registers across combinational logic without altering functionality. The paper presents findings from experiments conducted on seven 14 nm industrial designs, assessing the performance of re-timing algorithms in complete design flows. Unlike previous evaluations primarily at the logic level, this study focuses on physical level assessments. Results reveal variations in re-timing algorithm performance across designs, leading to the development of a re-timing-prediction model. This model accurately forecasts the potential improvements in timing, area, and power for industrial designs, offering a cost-effective approach to designing efficient flows and reducing Time-to-Market. The conclusion emphasizes the importance of considering physical-level evaluations for re timing algorithms to avoid potential misinterpretations based solely on logic-level assessments [9].

A global placement method designed to optimize performance in physical planning with re-timing. GEO utilizes a multilevel partitioning approach based on min cut principles, facilitating the top-down assignment of gates to tiles. A notable contribution of this work is the incorporation of re-timing-aware timing analysis (RTA), a tool guiding the placement process by providing timing slack information post-re-timing. This allows for direct minimization of the clock period during placement. By concurrently addressing partitioning and re-timing under the geometric delay model, GEO effectively conceals global interconnect latency, showcasing superior performance compared to traditional methods. The paper concludes by outlining plans for improving wire length results, exploring incremental timing analysis for sequential circuits, and integrating buffer insertion for enhanced delay and power minimization in future GEO iterations [10].

A new approach to improve the timing behavior of digital circuits containing enable registers. Unlike existing methods that optimize single-clock-cycle paths, our approach considers paths spanning multiple clock cycles, revealing greater optimization potential. The study also addresses register relocation in circuits with enable registers and D-Flip flops, presenting a specialized re-timing algorithm for such scenarios. The key outcome is a method that strategically inserts pipe lining into slow logical blocks, achieving local changes in the data path and minor controller adjustments. Experimental results demonstrate a noticeable reduction in the clock period (up to 40%) with a slight increase in area due to added registers. The approach proves beneficial for circuits meeting specific criteria, offering a new design perspective for timing optimization. Future work aims to extend the approach with additional optimization methods for both the data path and controller of circuits [11].

An effective approach for teaching RISC processor design in an undergraduate computer architecture course, aiming to minimize the gap between high-level data path block diagrams and Verilog code specifications. Using a graphical method and an online browser-based simulator, students progress through examples from single-cycle to pipeline designs, actively participating in the processor design process. Emphasizing fundamental characteristics of RISC processor design, the approach provides essential knowledge for computer architecture and serves as a valuable resource with a compact browser-based editor/simulator. While acknowledging challenges in creating Verilog descriptions, the paper offers a straightforward online tool, and future work includes expanding examples and exploring FPGA execution in the cloud to further enhance teaching capabilities [12].

A design method for multiplexers in the context of a single- cycle CPU system using the MIPS instruction set. The design employs both Schematic and VHDL for efficient development, discussing considerations and operational principles in detail. The advantages and disadvantages of Schematic and VHDL inputs are explored, highlighting the intuitive nature of Schematic and the general-purpose capabilities of VHDL. The paper emphasizes the application of multiplexers in various circuits, including binary comparators and generators, graphics occurrence circuits, and order selection circuits within a single-cycle CPU system based on the MIPS instruction set. The design process leverages Quartus II features for correctness, ensuring simplicity and flexibility in adapting to different data input requirements. Simulation results for FPGA-constructed components are presented, demonstrating the effectiveness of the proposed multiplexer design approach [13].

Method for safeguarding the register file of a RISC microprocessor against Multiple Bit Upsets (MBUs). The approach combines matrix code, providing detection and correction capabilities through information redundancy, with Triple Modular Redundancy (TMR), a widely used hardware redundancy technique that masks faults. The 32-bit single- cycle MIPS architecture microprocessor is designed to integrate a crypto module implementing the AES-128 algorithm, creating an application platform. The suggested approach has the capability to identify and rectify a range of errors, including 2, 4, and 8-burst errors or random errors, depending on the configuration of the dataset associated with the registers. Implementation on Xilinx Virtex-5 FPGA is carried out, with a comparison of area and power consumption. The technique proves applicable to register files of various sizes. Simulation results demonstrate effective correction of faults in the original register file and prevention of wrong operations when faults affect parity or check bits. Future research will involve more detailed analysis and overhead reduction [14].

A basic 16-bit RISC and an advanced 32-bit RISC. RISC processors use simple instructions to reduce complexity, cost, and power consumption. The initial 16-bit version faced technical challenges, leading to the development of more efficient 32-bit and 64-bit versions. The study focuses on their instruction sets and performance, comparing factors like speedup and power dissipation. Both processors include General Purpose Registers and Flag registers. An optimized multiplication algorithm is examined to improve the data path, crucial for power efficiency. The models are simulated and integrated using XILINX ISE Design Suite 14.7, with power analysis conducted.

The objective is to draw a comparative study between the processors, considering their instruction sets and performance metrics. Overall, the research aims to understand and improve the efficiency of RISC processors through advancements in architecture and instruction sets [15].

## 3 Design and Implementation

In the present work, Units of MIPS RISC Processor using pipeline Register Re-timing technique is implemented. MIPS RISC architecture is finely optimized to carry out the tasks of fetching, decoding, and executing instructions within a solitary clock cycle, orchestrating a systematic progression through its five well-defined stages: instruction fetch, instruction decode, execute, data memory, and write back.

**Instruction Fetch Stage:** Two registers responsible for equality comparison and a 16-bit offset. This offset is utilized to calculate the branch target address relative to the instruction's address. The computation involves adding the sign-extended offset field to the Program Counter (PC).This mechanism ensures the accurate determination of the branch target address based on the specified offset and contributes to the dynamic control flow within the processor Architecture.

**Instruction Decode Stage:** In this specific phase, instructions go through a decoding process, and the register file is accessed to fetch data from the registers. The results from the general-purpose registers are transferred to two temporary registers, specifically labeled as register 1 and register 2, for utilization in subsequent clock cycles. Concurrently, the lower 16 bits of the Instruction Register (IR) undergo sign-extension and get stored in a temporary register known as IMM, intended for future application in the next cycle.

The processor's register state, comprising 32 registers, is housed within the 'register file', serving as the repository for the machine's register values. This register file is pivotal in maintaining and managing the processor's internal state during the execution of instruction.

**Execution Stage:** During the execution stage, mathematical operations take place utilizing a Data Arithmetic Logic Unit (ALU) and a branch address adder. The ALU is equipped with arithmetic, logic, and shifting functionalities, whereas the branch address adder is specifically designed for PC-relative branch instructions. The branch data path comprises a sign extension unit and an adder responsible for calculating branch target addresses. When it comes to branch instruction comparison, the ALU receives two register operands as input, configured for a subtraction operation. This stage ensures the efficient execution of various computational tasks within the processor.

**Arithmetic and Logic Unit (ALU):** In a pipe lined ALU with re-timing, the goal is to optimize the performance of arithmetic and logic operations by introducing pipeline registers. These registers are strategically placed in the data path to break down the processing of instruction into sequential stages. The re-timing concept involves carefully managing the flow of data through these pipeline registers, enhancing parallelism and overall throughput.

**ALU Operation:** In Fig. 1, the ALU supports basic operations such as bit wise AND, bit wise OR, addition, subtraction, and a logical NOT operation on the most significant bit.The zero flag is set based on specific ALU operations, such as equality checks ((in_a == in_b) ? 1′b1: 1′b0).

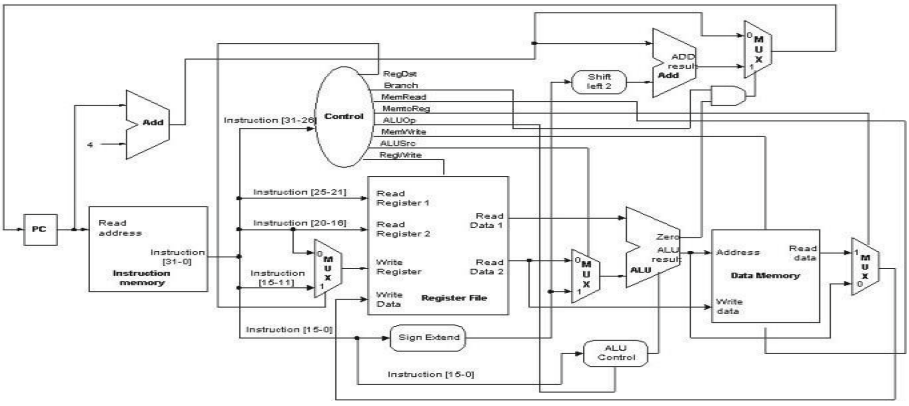


**Fig. 1.** Block Diagram of 32-bit MIPS RISC Processor

The pipeline registers (alu_out_reg and zero_reg) are employed to introduce a single-clock cycle delay, demonstrating the use of re-timing techniques to enhance performance.This module essentially represents a pipe lined ALU with support for various operations and a simplified control mechanism. The use of pipeline registers improves the timing characteristics of the design. For example, when control is 4′b0010, it performs addition (alu_out_reg <= in_a + in_b). The zero_reg is set accordingly based on the ALU operation.The default case (default) handles situations where the control signal doesn't match any specified operation.

**Control Unit:** In a MIPS RISC (Reduced Instruction Set Computing) processor, the control unit plays a critical role in generating control signals that orchestrate the execution of instructions. The control unit interprets the opcode of an instruction and produces signals that control various functional units within the processor, such as the ALU (Arithmetic Logic Unit), register file, and memory. When discussing a control unit with retiming techniques applied in a MIPS RISC processor, we consider the introduction of pipeline registers to enhance performance.

**Operation of Control Unit:** The module takes a 6-bit opcode as input and generates various control signals commonly used in a MIPS processor. Reg_dest, jump, branch, mem_read, mem_to_reg, mem_write, alusrc, reg_write, and aluop are output control signals.The control signals are generated based on the opcode bits. Each signal corresponds to a specific condition defined by the opcode. alusrc, reg_write, and aluop are derived based on specific combinations of opcode bits to control the ALU source, register write, and ALU operation, respectively. The comments next to each signal assignment indicate the opcode values that activate the corresponding control signal.

**Data Memory:** The data memory module provides basic read and write operations on a 64-word by 32-bit data memory. It supports synchronous behavior, where read and write operations occur on the rising edge of the clock, and a reset operation clears the entire data memory. The module is suitable for use as a basic data memory component in a digital system design. The module uses non-blocking assignments ($<=$) inside the always block, ensuring proper simulation behavior. This module represents a basic synchronous data memory with read and write functionality, suitable for use in a digital system design. The data memory array represents a data memory with 64 words, where each word is 32 bits. It is organized as a 2D array with 64 rows and 32 columns. The always @(posedge clk or posedge reset) block triggers on the rising edge of the clock (clk) or the rising edge of the reset signal (reset). If the reset signal is asserted (reset $==$ $1'b1$).

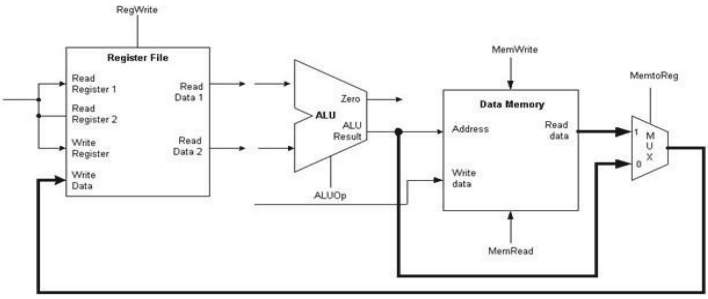


**Fig. 2.** Architecture of write back stage

Re-timing involves optimizing the timing of signals within the processor stages to improve performance. In a re-timed single- cycle MIPS, designers analyze and adjust the timing of signals to make the most of each clock cycle.

From Fig. 2, It is said that some stages that might finish their work early in one clock cycle can pass their results to the next stage sooner, potentially reducing the overall execution time. This optimization requires careful consideration of the dependencies between different stages and the timing constraints of the hardware.

The key difference is how efficiently the processor uses each clock cycle. Without re-timing, every instruction follows the same fixed path, regardless of how quickly each stage completes. With re-timing, designers tweak the timing to allow certain stages to complete faster, optimizing overall performance.

## 4 Result Analysis

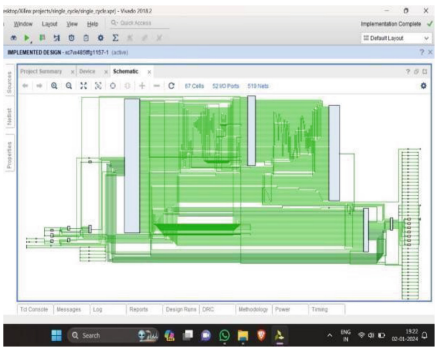

**Fig. 3.** Schematic of RISC Processor without Re-Timing

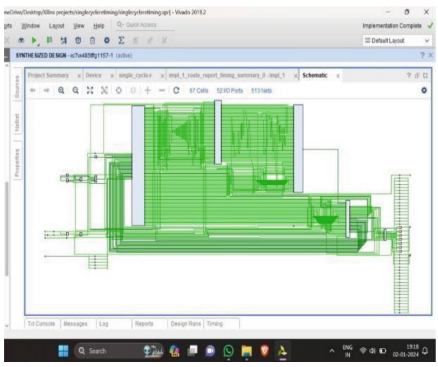

**Fig. 4.** Schematic of RISC Processor With Re-Timing

In Fig. 3, the single-cycle MIPS RISC architecture schematic consists of 67 cells, encompassing various types such as registers and logic gates .Where as in Fig. 4, the re-timed single-cycle MIPS RISC architecture schematic consists of same as without re-timing. There are 52 I/O ports serving distinct input and output functions. The interconnection of components is facilitated by 519 nets, highlighting critical signal pathways. Notable features include custom-designed cells, if any, which play a crucial role in the optimization. The I/O ports are specified with brief descriptions of their intended purposes. The schematic underscores critical nets and pathways, emphasizing high-traffic connections and critical paths. Major modules and components are identified, with their roles and significance outlined. Considerations for performance metrics, including clock frequency and critical path delays, are integral to the schematic information. The visual representation of schematic sections aids in comprehending the architecture, showcasing the impact of re- timing on its overall design and efficiency.

In the above Fig. 5 and Fig. 6, the simulation of the single_cycle_tb test bench for the single_cycle module has been successful, demonstrating expected behavior and validating the functionality of the design. The test bench effectively cycles through different values of switch_select, displaying corresponding reg_read_data_1 values at each positive edge of clkread. The counter count increments correctly, and the control mechanism involving switch_run is well-coordinated. No unexpected behavior are observed, and the simulation completes within the specified time frame. These positive outcomes indicate that the single_cycle module functions as intended, showcasing its reliability and suitability for integration into larger systems or projects.

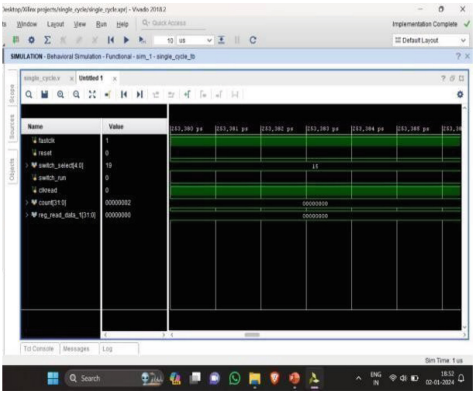

**Fig. 5.** Simulation of single cycle MIPS RISC Processor

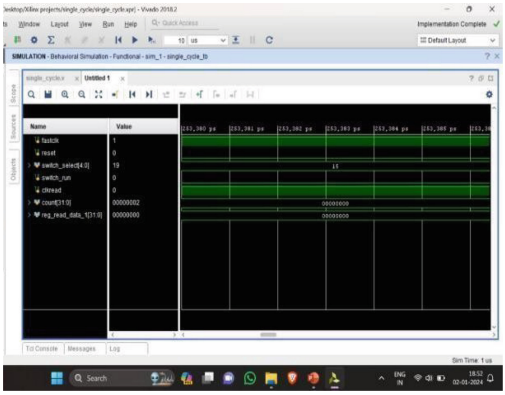

**Fig. 6.** Simulation of MIPS RISC Processor with re-timing

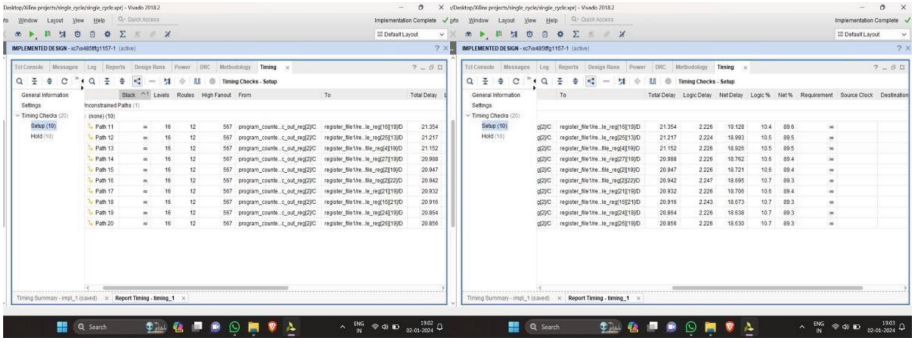

**Fig. 7.** Single cycle MIPS RISC processor set-up timing report

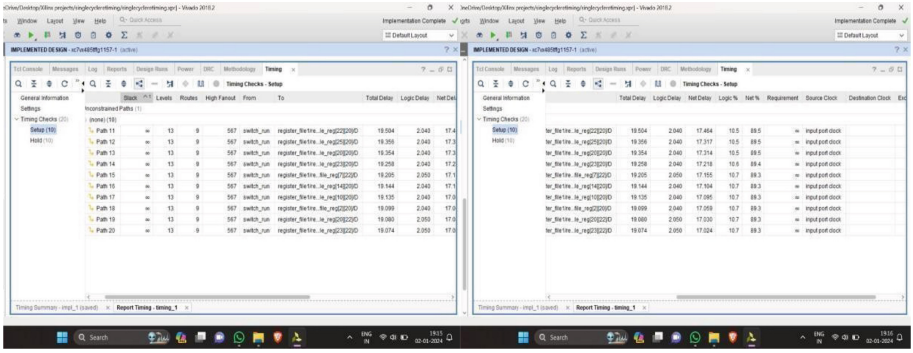

**Fig. 8.** Set up Timing report of Single Cycle MIPS RISC with re-timing applied

The above figures Fig. 7 and Fig. 8 shows set-up timing analysis of a single cycle MIPS RISC Processor with and without applying re-timing technique, possibly involving a stack, levels, routes, and various paths with associated delays. The table outlines different paths (Path 11 to Path 20) and their corresponding total delays, setup times, and hold times. The High Fan-out From and To sections may indicate critical areas for signal distribution. The timing checks include setup and hold checks for the paths. The total delay has been reduced from 21.942 ms to 19 ms.

The critical path in a traditional MIPS design is determined by the slowest stage in the fixed sequence of instruction processing.

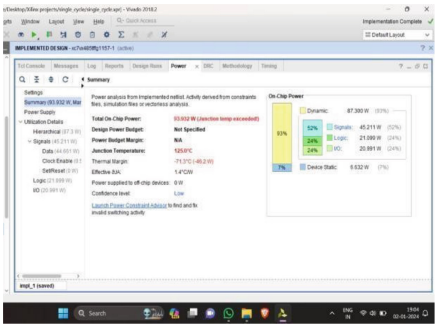

**Fig. 9.** Power report of Single cycle MIPS RISC processor

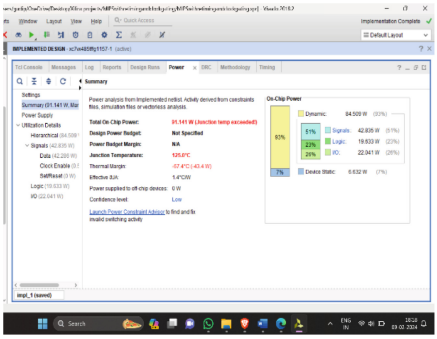

**Fig. 10.** Power report of processor after Re-timing

Figure 9 and Fig. 10 is about the power dissipation of the system, specifically in the ALU control, has been successfully reduced by applying clock gating. Before implementing clock gating, the power dissipation was measured at 93.932 w. After the integration of clock gating logic, the power dissipation has decreased to 91.141 w. This reduction in power dissipation indicates improved energy efficiency in the system, as the ALU control now consumes less power during idle periods, contributing to overall power savings.

Applying re-timing to a MIPS RISC processor, which involves strategically adding registers to improve performance, comes with challenges. First, it makes the design

more complex, needing careful management. The insertion of registers increases the physical size of the processor, impacting efficiency. Changes in clock distribution and possible clock skew issues need attention. Compatibility with existing software could be impacted, and efficient resource use is crucial. Achieving timing closure becomes more challenging, and dealing with clock domain crossing complexities is necessary. Balancing these factors is key to successfully applying re-timing to a MIPS RISC processor (Table 1).

**Table 1.** Power Analysis

| Parameters | Proposed | Existing |
|---|---|---|
| TOTAL ON-CHIP POWER | 91.141 | 93.932 |
| DYNAMIC POWER (IN W) | 84.61 | 87.300 |
| SIGNAL POWER(IN W) | 43.72 | 45.211 |
| IO POWER | 22.074 | 20.991 |
| DEVICE STATIC | 6.632 | 6.632 |
| IO PORTS | 52 | 52 |

## 5 Conclusions

The implementation of retiming in the single cycle MIPS RISC Processor using Xilinx Vivado resulted in some notable benefits, like reducing on-chip power and making the processor more efficient. By adjusting the timing of signals, the setup delay experienced an 8% reduction, showcasing enhanced performance in critical path execution they managed to speed up how quickly the processor performs tasks. The total onchip power of re-timed processor has reduced by 2%, dynamic power has also reduced by 2%. This not only saved power but also made it easier to add specialized features without slowing things down. The flexibility gained from this approach allows designers to create processors that are better suited for specific tasks. Additionally, having control over the clock and timing lets the processor adapt to different needs. In simple terms, using retiming techniques is a big step forward in making processors faster, more efficient, and customizable for today's technology.

## References

1. Sharma, R., Sehgal, V.K., Nitin, N., Bhasker, P., Verma, I.: Design and implementation of a 64-bit RISC processor using VHDL. In: 11th International Conference on Computer Modelling and Simulation Cambridge, UK (2009)
2. Sulik, D., Vasilko, M., Durackova, D., Fuchs, P.: Design of a RISC microcontroller core in 48 hours. In: Embedded Systems (2000)

3. Zaid, M., Mustajab, P.: Design and application of RISC processor. In: IEEE International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT), Aligarh, India (2017)
4. Jang, H., et al.: Developing a multicore platform utilizing open RISC-V cores. IEEE Access (2021). https://doi.org/10.1109/access.2021.3108475
5. Mehra, H., Bhat, M.S.: High-level optimization methodology for high performance dsp systems using retiming technique. In: IEEE 2018 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), Mangaluru (2018)
6. Jalaja, S, Prakash, A.M.V.: Design of low power based VLSI architecture for constant multiplier and high speed implementation using retiming technique. In: IEEE 2016 International Conference on Microelectronics, Computing and Communications (MicroCom) - Durgapur, India (2016)
7. Yagain, D., Vijaya, K.A.: FIR filter design based on retiming automation using VLSI design metrics. In: IEEE International Conference on Technology, Informatics, Management, Engineering & Environment (TIME-E 2013) – Bandung (2013)
8. Simon, S., Bernard, E., Sauer, M., Nossek, J.A.: A new retiming algorithm for circuit design. In: Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94 (n.d.)
9. Yu, C., et al.: End-to-End Industrial Study of Retiming. In: 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI) (2018)
10. Cong, J., Lim, S.K.: Retiming-based timing analysis with an application to mincut-based global placement. IEEE Trans. Comput.-Aided Design Integr. Circ. Syst. **23**(12), 1684–1692 (2004)
11. Martin, H.-G.: Retiming for circuits with enable registers. In: Proceedings of EUROMICRO 1996. 22nd Euromicro Conference. Beyond 2000: Hardware and Software Design Strategies (n.d.)
12. Passe, F., Canesche, M., Neto, O.P.V., Nacif, J.A., Ferreira, R.: Mind the gap: bridging verilog and computer architecture. In: 2020 IEEE International Symposium on Circuits and Systems (ISCAS) (2020)
13. Jiang, L., Huang, C.: Design and realization of multiplexer based on schematic and VHDL language. In: 2010 International Conference on Optics, Photonics and Energy Engineering (OPEE) (2010)
14. Ustaoglu, B., Yalcin, B.O.: Fault tolerant register file design for MIPS AES-crypto microprocessor. In: 2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS) (2015)
15. Kulshreshtha, A., Moudgil, A., Chaurasia, A., Bhushan, B.: Analysis of 16-bit and 32-bit RISC processor. In: 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS) (2021)