Static Static Timing Analysis–Aware RTL Design Of A Simple RISC Processor Using Verilog HDL

Prathamesh Pakhale[1], Rahul Ingale[2]
[1, 2]Dept of electronics and telecommunication engineering
[1, 2] Government college of engineering Kohhapur, India

Abstract- *As the demands for higher clock frequencies in modern VLSI systems continue to rise, achieving timing closure has become a significant hurdle in digital design. Many RTL designs struggle to meet setup and hold time requirements after synthesis, often because timing constraints weren't adequately considered in the early stages of design. This paper introduces the design and implementation of a straightforward RISC processor using Verilog HDL, employing a timing-aware RTL methodology. The architecture of the processor features a program counter, instruction memory, register file, arithmetic logic unit (ALU), control unit, and data memory. We conduct Static Timing Analysis (STA) using Xilinx Vivado to assess critical path delay, worst negative slack (WNS), and maximum operating frequency. Initially, we analyze a baseline design to pinpoint timing violations, followed by restructuring the RTL and applying optimization techniques like logic balancing and inserting pipeline registers. The experimental results show a notable reduction in critical path delay and an increase in the maximum achievable clock frequency, confirming the benefits of integrating timing awareness into the RTL design process. This proposed method enhances timing reliability while minimizing the number of design iterations needed after synthesis.*

Keywords: RISC Processor, RTL Design, Verilog HDL, Static Timing Analysis, Critical Path, Timing Optimization, FPGA Implementation.

MOTIVATION*— As the demands for higher clock frequencies in modern VLSI systems continue to rise, achieving timing closure has turned into a significant design hurdle. While many RTL designs may function correctly, they often fall short of meeting the necessary setup and hold time requirements after synthesis. This situation results in multiple rounds of optimization and longer development timelines. By integrating Static Timing Analysis (STA) awareness into the RTL design process, we can minimize timing violations and boost overall performance. Consequently, this work aims to design a straightforward RISC processor using timing-aware RTL techniques to improve critical path performance and maximize operating frequency.*

I. INTRODUCTION

The ongoing advancement of semiconductor technology and the growing complexity of designs have made achieving timing closure a essential challenge in contemporary VLSI systems. Static Timing Analysis (STA) is commonly employed within Electronic Design Automation (EDA) workflows to assess signal delays and detect timing violations without needing exhaustive simulations [1]. Nonetheless, precise timing data usually becomes accessible only after synthesis and physical implementation, which makes fixing timing issues late in the process both expensive and time-intensive.

Recent studies have focused on early-stage timing prediction and optimization at the RTL level by using machine learning and sophisticated modeling techniques [2], [3]. These approaches seek to enhance timing accuracy and shorten runtime, while also supporting more effective design exploration. Besides, optimization methods driven by simulation have shown that introducing performance considerations earlier in the design flow can notably lower latency and boost overall system performance [4].

RISC processor architectures are popular in embedded and low-power applications due to their simplified instruction sets and efficient pipeline designs. Although many RTL implementations of basic RISC processors prioritize functional correctness, there is often insufficient focus on timing-aware RTL design methods that proactively tackle critical path and slack challenges before backend synthesis.

This paper introduces a Static Timing Analysis–Aware RTL implementation of a simple RISC processor using Verilog HDL. The proposed method incorporates timing awareness during RTL development to detect and optimize critical paths early in the design process, thereby enhancing performance and minimizing timing violations after synthesis.

II. LITERATURE REVIEW

[1] Amik and Zhang (2024) proposed a new machine learning framework of early timing prediction on the RTL level. They found the conventional ML methods are typically unreliable

because of uncertain synthesizer outputs and deficit of the technology library features. Their answer combines an RTL-to-graph parser with a Large Language Model and a graph-based ML engine with the aim to enhance prediction accuracy prior to physical synthesis, which can be considered a way of lowering time-to-market.

[2] Guo et al. (2023) resolved the issues of the computational bottlenecks of advanced technology nodes with the proposal of a delay calculator accelerated by the use of a GPU. This is a tool that specializes in model order reduction of Arnoldi and sophisticated effective capacitance algorithms. They designed effective numerical kernels that now perform tasks such as LU decomposition and Krylov subspace computation with up to a 14.03x speed-up over the industrial standards such as OpenSTA and retain close correlation to the golden timing results.

[3] Fang et al. (2024) introduced RTL-Timer, the first general timing estimator, which annotated slack on Verilog code. Having understood that timing information is usually not available at the RTL stage, they adopted graph-based representations and tailor made loss functions to estimate the maximum arrival times at register endpoints. This allows the designers to optimize their code at the beginning of the flow which can be measured in terms of improvement of the overall and worst negative slack.

[4] Fang et al. (2024) also discussed the fine-grained RTL timing analysis, stating that the initial design phase can be optimized most of all. Their experience can be shown by the fact that register-level timing predictions are not only forecasted but also selectively RTL-level optimizations. This methodology has a trade off between the high-level architectural design and the post-synthesis timing reality.

[5] Wadhwa and Shreejith (2024) proposed Simopt, an information flow that derives metadata out of behavioral simulation to drive speculative optimization in FPGA-CAD flows. Simopt can allow the final performance of the design (latency) to be minimized by 38.2 by introducing latency awareness in the Yosys flow placement phase. This has shown that high-level simulation data can be used to drive physical design implementation.

[6] Lv et al. (2025) created an automated debugger RTL timing violations framework, VITAD, based on Large Language Models. Path analysis is done by parsing the Verilog code and timing reports into a Signal Timing Dependency Graph (STDG). Using LLMs to provide root causes and domain-specific debugging information, VITAD automates, to a large degree, the manual and iterative timing closure process that was traditionally manual.

[7] The HeteroSTA (2025) presented in the paper by Guo et al. is a CPU-GPU heterogeneous timing engine that is aimed at providing end-to-end support of industrial design. It has a zero-overhead API and uses industry-standard formats such as .sdc constraints and complicated timing exceptions. It can be used in combination with global routing or placement tools, such as DREAMPlace 4.0 to give an end-to-end acceleration, with graph based and path based timing questions being given an industrial grade of accuracy.

[8] Wang et al. (2025) dealt with the inherent dilemma of accuracy and computational efficiency of timing prediction at the register-transfer level (RTL). They introduced a new cross-stage knowledge distillation model, called RTLDistil, filling the gap with a proposed way of transferring the exact physical properties of a layout-aware teacher model to a high-performing RTL-level student model. This framework will predict the timing accurately at the initial stages and this will greatly minimize errors in comparison with the current state of art models.

[9] Guvvala (2025) examined Static Timing Analysis (STA) as a basic approach to the modern digital design verification, a nanometer-scale semiconductor design verification. The article compared the comparative advantages of STA when compared to the conventional simulation methods with the emphasis on the effect of both on the efficiency and reliability of design verification. It points out that STA is a demanding procedure that is essential in the attainment of high-fidelity outcomes by the examination of thorough physical parameters.

[10] Ramegowda et al. (2025) reiterated that one of the major challenges in the contemporary integrated circuit (IC) design is theinability to maintain precision with regards to timing closure because of the complicated specifications and the dynamism in technology. Their study indicated the need to have simplified (compared to complex) methods to enhance timing closure and physical verification, especially in multi-corner/ multi-mode designs. They showed that by optimizing the physical design tasks like clock tree synthesis (CTS) it is possible to effectively minimize the problems of register to register path violation.

[11] Taraate (2014) suggested the design and construction of a 16-bit 4-stage pipelined Reduced Instruction Set Computer (RISC) IP core on an FPGA. The paper found that pipelined structures are extremely efficient and execution time can be greatly lowered by means of micro-architecture fine tuning.

This was implemented using the Verilog RTL as a controller and tested using standard simulation and synthesis tools.

[12] Manjula et al. (2023) described the design and implementation of a RISC processor with the Microprocessor without Interlocked Pipelined Stages (MIPS) technique on an FPGA. This is a 16-bit processor which is based on the pipelining technique which enhances overall performance in terms of delay and power. This architecture consists of five phases, namely, instruction fetch, instruction decode, execute, memory access, and write back.

[13] Suresh and Ganesh (2014) have talked about the benefits of RISC processors and the fact that they could carry out complicated operations through a pipeline concept which has a high performance as opposed to CISC architectures. They used Verilog HDL to create a 10-bit address bus, 8-bit bi-directional data bus based RISC processor, using MIPS. The simulation and synthesis were successful and the design targeted a Xilinx Virtex-5 FPGA board

[14] In a 32-bit MIPS processor architecture, Tyamanavar and Nidagundi (2019) used an FPGA and got a speed of 290.166 MHz. They incorporated a processor architecture written in Verilog HDL, which incorporated the necessary blocks, which are the memory unit, control unit, program counter, and ALU. This paper shows the high performance and efficiency which are possible when using the modern FPGA tools and timing-aware design methodology.

III. METHODOLOGY

This study aims to assess how well RTL design techniques that are aware of Static Timing Analysis (STA) can enhance the timing performance of a basic RISC processor. Xilinx Vivado is used for synthesis and Verilog HDL is used for implementation. Critical paths, slack violations, and the highest possible clock frequency are determined by analyzing timing reports produced by the STA tool. To enhance the processor's timing characteristics, RTL-level optimization strategies like pipelining, logic restructuring, and pipeline balancing are used.

3.1 Experimental setup & tool environment:

Xilinx Vivado Design Suite (v2023.x) is the primary EDA tool used to develop the complete experimental framework. Hardware description written in Verilog HDL (the standard method for RTL-level modeling of digital systems) prior to synthesis and physical implementation, defines system functionality at the RTL abstraction level [1]. This project's design process includes:

1) Verilog RTL programming,
2) Functional simulation for behavioral verification,
3) Logic synthesis using the Vivado synthesis engine,
4) Static Timing Analysis (STA) post-synthesis.

After synthesis, Vivado generates a timing-analysis compatible technology-mapped netlist (for all the operations performed in the design) on the synthesis level. Early timing analysis at the RTL and synthesis levels helps to reduce the number of cycles needed for the latter stages of design by allowing for the identification of timing-critical paths before physical design begins, as noted in [2]. The timing analysis using user-defined clock constraints is done to evaluate:

1) Critical Path Latency,
2) Worst Negative Slack (WNS),
3) Total Negative Slack (TNS),
4) Maximum frequency at which the design may operate. This structured tool environment allows for layout-level functional correctness to be verified in conjunction with timing performance associated.

3.2 Synopsis of RTL Architecture

The primary design employed in this work is an Arithmetic Logic Unit (ALU) capable of performing arithmetic and logical operations such as these.

1. Involvement
2. Conjectural
3. AND and OR in logic

A program counter, instruction memory, register file, arithmetic logic unit (ALU), control unit, and data memory ma ke up the architecture of the suggested processor.

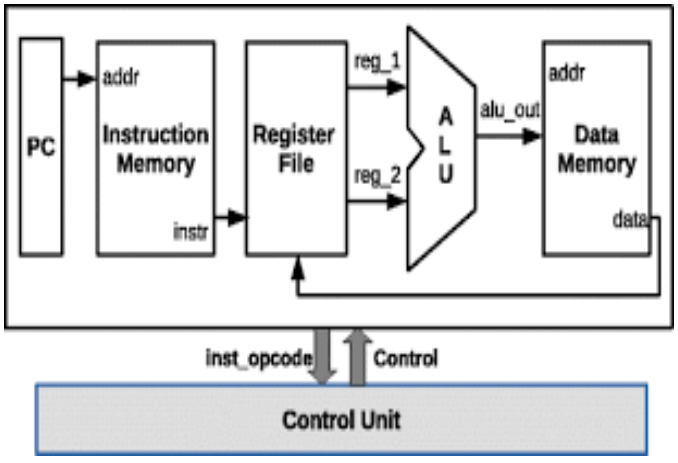


Fig. 1. Block Diagram of the Proposed RISC Processor Architecture

The processor uses a pipelined architecture, which in volves processing instructions in several steps.

Methods of comparison

To improve throughput and reduce combinational delay, the ALU is implemented using a three-stage pipelined architecture. Among the phases of the pipeline are:

- Obtain the Instruction (IF)
- Run/Decrypt (ID/EX)
- Reply (WB)

Pipeline registers are inserted between these steps to divide long combinational paths into shorter segments. The maximum time it takes for data to arrive at sequential elements determines overall timing performance, as described [4]. Pipeline registers reduce the combinational delay between flip-flops, increasing clock frequency.

- How the RTL architecture is configured,
- Each sequential element is triggered by a positive edge of the clock.
- The data path and control path have been modularized.
- The stage boundaries of the pipeline registers help to easily identify where each stage starts and ends.

This modularized and pipelined RTL structure makes timing optimizations more efficient during static timing analysis.

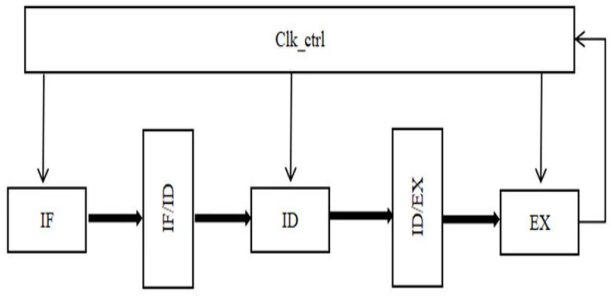


Figure 2: The Proposed RISC Processor's Pipeline Architectur e

3.3 STA Concepts and Constraints:-

STA (Static Timing Analysis) typically is completed after the synthesis phase of a digital design and does not require the use of dynamic simulation to verify that the digital design meets defined timing constraints.

According to [3], STA models the digital circuit as a Directed Acyclic Graph (DAG), where: Nodes represent either pins or registers; Edges represent the delays in the cells and the delays between the cells (interconnect). In this work, the following timing-related and constraint-related factors are the main factors that cause delay in the overall timing of a design:

Timing-Related Factors Combinational Logic Delay Clock to Q Delay Setup and Hold Time Requirements Clock Skew & Clock Uncertainty The setup slack in a design is calculated by using the equation below :

Slack = Required – Actual .

The required arrival time is the longest period of time that the signal can get to the register.

1] Real Arrival Time: the signal path's actual propagation delay.
2] If Slack is less than zero, there is a timing violation.

Your results address WNS and TNS, which makes this formula crucial.

Constraint-Related Factors Clock Period Constraints Input Delay & Output Delay Constraints Multi-Cycle Path & False Path Constraints Accurate timing predictions rely on accurate models of technology libraries and the proper definition of constraints, as indicated in [1]. Clock constraints must be defined explicitly in Vivado by using the equation below :

$$T_{clk} \geq T_{cq} + T_{logic} + T_{setup} .$$

- The Clock-to-Q delay of the launching flip-flop equals Tcq.
- The propagation delay of the combinational logic is described by Tlogic
- The setup time requirement of the receiving flip-flop is defined as Tsetup.

The clock period must be larger than or equal to the total propagation delay between two sequential elements in order for the circuit to operate correctly.

Setup timing violations will occur in the design if the clock period is less than this delay.

prior to completing STA so that a realistic assessment of timing can be achieved. STA analysis will focus on determining: Worst Negative Slack (WNS); Total Negative Slack (TNS); Identification of Critical Paths. With the

WNS, TNS & Critical Path, it is possible to determine where delays are occurring in the ALU pipeline.

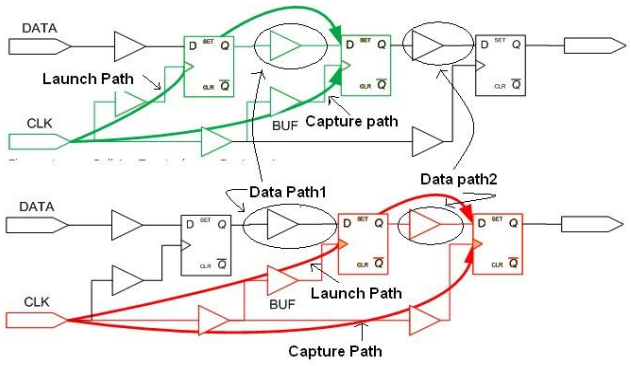


Fig. 3. Critical Timing Path in RTL Design

3.4 Comparative Analysis of the Data The comparative study of STA guided optimizations consists of two parts:

1. the original synthesized design (no timing refinement) and
2. the optimized design (applied STA based optimizations). The original synthesized design serves as the baseline for establishing timing metrics which are as follows: critical path delay, maximum operating frequency and slack. Upon completion of the STA analysis reports, various modifications including pipeline balancing and logic restructuring will be applied to the design before it is re-synthesized and analyzed again. The importance of identifying critical timing paths early through STA is emphasized in [2]. Thus, the two main metrics for comparison between the two designs are

   A. critical path delays that have been reduced,
   B. WNS that has improved, and

3. clock frequencies that are achievable have increased. A timing-efficient and high-frequency design can be established from the re-synthesized and optimized designs that exhibit improved slack and less critical path delay.

IV. RESULTS AND DISCUSSION

We used Static Timing Analysis (STA) to look at the timing characteristics of the RTL design and see how well the proposed RISC processor worked. We looked at timing reports at different points in the design process, such as the baseline RTL implementation, the pipelined architecture before optimization, and the final STA-aware optimized design. We used important timing metrics like Worst Negative Slack (WNS), Total Negative Slack (TNS), and the number of failing endpoints to measure how well the timing worked. The results show that STA-driven optimizations were able to fix timing violations that happened during pipelining. A comparative analysis of these stages underscores the efficacy of the proposed STA-aware RTL design methodology in attaining dependable timing closure.

4.1 ***Baseline RTL Timing Performance***

The RISC processor was first created using standard RTL design techniques in Verilog Hardware Description Language, without any optimizations driven by Static Timing Analysis. The goal of this initial version was to assess the inherent timing characteristics of the processor architecture before making any performance enhancements. Similar RTL-based processor designs have been commonly utilized for FPGA-based processor design and verification in previous studies [12].

Once the processor design was synthesized, Static Timing Analysis (STA) was conducted to evaluate the timing characteristics of the baseline RTL implementation. STA tools produce detailed timing reports that shed light on setup slack, hold slack, and the overall timing performance of the design, all without needing to simulate every possible input combination. This kind of analysis is crucial for pinpointing potential timing bottlenecks in digital processor architectures [11].

The timing analysis revealed some key parameters. The Worst Negative Slack (WNS) was around 0.049 ns, which means the design met the required timing constraints but was operating very close to the maximum allowable clock period. The Total Negative Slack (TNS) was 0 ns, confirming that there were no setup timing violations throughout the design. Additionally, the number of failing endpoints was 0, indicating that all sequential elements in the processor successfully adhered to the required timing constraints.

For finding the fmax the formula is below :

Fmax = 1 / Tcritical

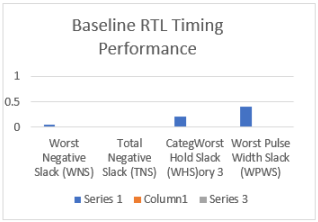
Fmax : Maximum clock frequency at which the processor can operate

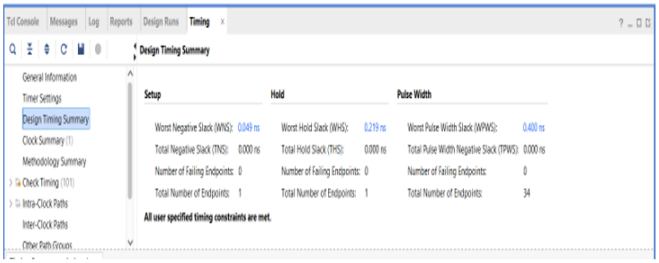
Tcritical path : Longest propagation delay in the circuit

The maximum operating frequency of a digital processor is determined by the critical path delay of the circuit. The critical path of the design shows its maximum

time delay between two successive elements. The maximum clock frequency that the processor can achieve is inversely proportional to this delay. The processor achieves higher clock frequencies because critical path delays decrease through RTL optimization methods that use pipelining and logic restructuring techniques.

The relatively small positive slack value suggests that the design has limited timing margin for any further architectural changes. In real-world processor designs, such limited slack can be influenced by variations in process, voltage, and temperature conditions, or by the introduction of new architectural features in future design iterations [13].





4.2 Timing Issues After Pipelining

Pipelining was added to the RISC processor architecture to enhance throughput of the processor and the general performance of the execution. Pipelining breaks up the instruction execution process into several steps divided by pipeline registers and thus there are several instructions being executed simultaneously in various steps of execution. Such an architectural method is very useful in enhancing processor throughput and general computational efficiency [12].

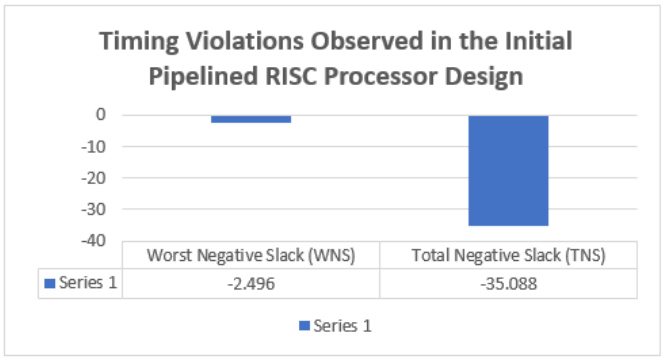
Common pipelined RISC processors utilize many steps which include instruction fetch, instruction decode, execute, memory access and write-back steps to provide parallel processing of instructions [12]. Although pipelining improves the performance, it adds another complexity to datapath timing and register-to-register signal propagation [11].

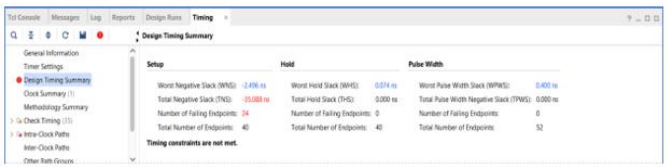
Following the introduction of the basic pipelined architecture, Static Timing Analysis indicated that the processor design had a number of timing violations. The cause of these violations was mainly due to the addition of more delay in combinational between pipeline registers and the added logic in the datapath.

The timing report indicated that the Worst Negative Slack (WNS) had deteriorated to -2.496 ns which showed a major violation of set up timing. The negative slack value would indicate that the propagation delay of some paths would be larger than the desired clock period, and the design would not be able to run at the desired frequency.

The Total Negative Slack (TNS), similarly, was -35.088 ns which was the summation of timing violations of various critical paths of the processor. Also, it was found that there were 24 failing setup endpoints, which means that a number of successive elements could not receive input data in the necessary clock cycle.

Such findings bring to the fore a typical challenge of pipelined processor design. Although pipelining enhances execution throughput, a poor balance between combinational logic in the pipeline stages may result in longer critical paths and timing violation [11]. Thus, timing closure needs to be carefully done by timing analysis and optimization to ensure timing closure in a complex digital processor architecture.





4.3 STA-Based Optimization and Timing Improvement

Having detected the timing violations of the pipelined processor architecture, Static Timing Analysis was applied to explore the critical paths that caused the negative slack values that were observed. STA offers an effective tool of analysing setup and hold timing requirement by considering signal propagation delays of all time paths in the design [11].

The timing datapath blocks that caused timing violations were assessed with the assistance of the timing reports produced by the synthesis tool. As a result of this discussion, a number of RTL level optimizations were made in order to enhance the timing performance of the processor.

These optimizations were:

1. reorganizing the datapath system.
2. balancing pipeline stages
3. maximizing pathways of combinational logic.
4. enhancing the location of registers in the pipeline stages.

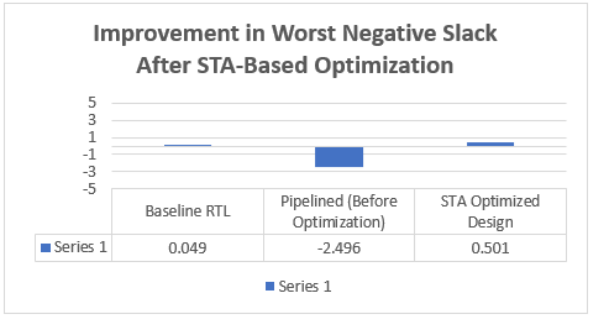
It can be used as an architectural modification to minimize the propagation delay between sequential elements and distribute computation workloads more uniformly through the pipeline stages to enhance timing performance of the processor [13].

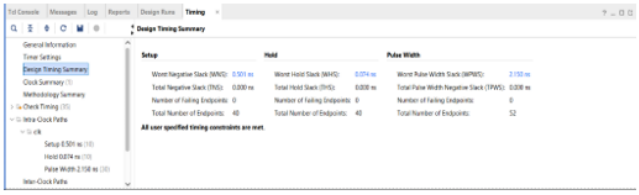
The processor design was re-synthesized and again Static Timing Analysis was run, after the STA-based optimizations.

The revised timing report was much better timing performance. The Worst Negative Slack (WNS) was 0.501ns which meant that a positive timing margin was gained, and the design had met the necessary setup timing requirements.

Total Negative Slack (TNS) was set at zero which showed that all setup timing violations had been removed. Moreover, the failing endpoints were decreased to 0 due to 24 to 0, which validated that all the sequential elements in the processor pipeline achieved the necessary timing constraints at present.

These advances show that the inclusion of STA-conscious design methods at the RTL level can serve conveniently to address timing bottlenecks and to optimize reliably the performance of pipelined processor designs to a large extent.





Comparison table :-

Timing Parameter	*Baseline RTL Design*	*Pipelined Optimization)*	*Design*	*(Before*	*STA Optimized Design*
Worst Negative Slack (WNS)	*0.049 ns*	*-2.496 ns*			*0.501 ns*
Total Negative Slack (TNS)	*0 ns*	*-35.088 ns*			*0 ns*
Worst Hold Slack (WHS)	*0.219 ns*	*0.074 ns*			*0.074 ns*
Worst Pulse Width Slack	*0.400 ns*	*0.400 ns*			*2.150 ns*
Failing Setup Endpoints	*0*	*24*			*0*
Timing Constraints Status	*Met*	*Violated*			*Met*

4.4 Result and Discussion Conclusion.

The experimental findings are a clear indication of the significance of Static Timing Analysis when designing RTL of digital processors. The first version of the RTL implementation was able to satisfy the timing requirements but had a very small timing margin.

Once pipelining was added to improve processor throughput, dramatic timing violations were recorded as a result of more delays in the combinational paths and the lack of logic separation between pipeline stages. The same has been seen in earlier research where pipelined architectures have been shown to enhance performance but brought about more timing issues in the design of datapaths [11][12].

The timing violations were resolved by implementing the STA-based optimization methodology, including datapath restructuring, balancing of pipeline stages and optimization of the critical path. The optimized processor design had positive slack values, no overall negative slack, and zero failing endpoints which confirmed that complete timing closure had been attained.

These findings indicate the relevance of incorporating Static Timing Analysis in the RTL design process of contemporary processors. The identification and optimization of important timing paths early on allows designers to have a consistent operation of the processors at the required clock frequency with high performance and design durability.

V. CONCLUSION

In this project, Static Timing Analysis-Aware RTL Design of a Simple RISC Processor Using Verilog HDL, the importance of static timing analysis (STA) in the enhancement of processor performance and minimization of delay was examined. The analysis of timing is an important process used in the design of digital integrated circuits since it makes sure that the timing behaviour of the circuits is correct, as well as it helps in identifying the critical delay-paths in the circuit and thus does not incur the time-consuming dynamic-simulations [1]. STA can be used to identify timing constraints and signal propagation delays early in the design flow to identify timing constraints and timing violations, as well as to optimize circuit performance prior to fabrication.

Another significant issue that is brought up in the work is the relevance of precise delay modeling in the contemporary digital circuit. Elmore delay and non-linear delay models are traditional delay models used to model gate and interconnect delay, although they are not always effective in modeling advanced technology node complex RC interconnect delay [1]. Thus, the timing analysis techniques and optimization techniques should be more sophisticated to enhance timing and minimize delay in high-performance processors.

The other essential point that has been discussed in this paper is the benefit of carrying out timing-related optimization at a later phase of the design flow. Register transfer level (RTL) analysis offers more flexibility on design optimization that may be done at an earlier stage as there are numerous architecture options that can still be adjusted before synthesis and physical expressions become a reality [2]. Timing behavior prediction at the RTL stage enables designers to influence with better foresight the synthesis and optimization mechanisms involved to achieve better worst negative slack (WNS) and total negative slack (TNS) [2].

Recent studies have also shown that a design-flow optimization can have a great impact on the latency of circuits and its overall performance. Indicatively, metadata produced by behavioral simulation can be applied in a simulation-guided optimization framework to inform placement and routing of FPGA design flows, with significant benefits in circuit timing and latency [3]. Benchmark circuit experimental performance indicates that timing-sensitive optimization methods can boost or halve the latency of certain circuits by up to 38 percent, which illustrates the potential of timing-sensitive optimization methods in digital design [3][4].

Moreover, combining timing analysis with automated design tools and optimization frameworks might give the design evaluation a tremendous boost in performance and load. The high speed performance of advanced STA engines with optimized numerical algorithms and parallel computation methods can be achieved with high timing accuracy [1]. These are used to allow faster design loop times and assist designers to work within rigid timing requirements in complex digital systems.

On the whole, this project indicates that the integration of timings analysis in the schematic phase of the RTL design of a RISC processor can be used to determine the timing paths that are critical, minimise the propagation delay, and enhance common system effectiveness. With the integration of RTL-level timing consideration, and the current optimization strategy and design automation tools, design developers are able to implement more effective and dependable processor structures. Future directions could include the methods of automated STA-driven optimization implemented directly in the RTL design flow, and the investigation of machine-learning-based timing prediction schemes to enhance timing closure and processor performance even further [2].

REFERENCES

[1] Guo, Z., Huang, T. W., Jin, Z., Zhuo, C., Lin, Y., Wang, R., & Huang, R. (2023). Heterogeneous static timing analysis with advanced delay calculator. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[2] Fang, W., Liu, S., Zhang, H., & Xie, Z. (2024). Annotating slack directly on your Verilog: Fine-grained RTL timing evaluation for early optimization. Proceedings of the Design Automation Conference (DAC).

[3] Wadhwa, E., & Shreejith, S. (2024). Simopt: Simulation pass for speculative optimisation of FPGA-CAD flow. IEEE Conference on Field Programmable Logic and Applications.

[4] Lin, Y., Guo, Z., Huang, T. W., & Wang, R. (2023). Advanced delay modeling and acceleration techniques for static timing analysis. Journal of Electronic Design Automation.

[5] Fang, W., Liu, S., Zhang, H., & Xie, Z. (2024). RTL-Timer: A fine-grained timing prediction framework for early stage RTL optimization. ACM Transactions on Design Automation of Electronic Systems.

[6] Lv, W., Xia, Y., Chen, Y., & Kuang, L. (2025). VITAD: Timing Violation-Aware Debugging of RTL Code using Large Language Models. arXiv:2508.13257v1 [cs.AR].

[7] Guo, Z., Liu, H., Shi, X., Hua, S., Zhang, Z., Zhao, C., Wang, R., & Lin, Y. (2025). HeteroSTA: A CPU-GPU Heterogeneous Static Timing Analysis Engine with Holistic Industrial Design Support. arXiv:2511.11660v1 [cs.DC].

[8] Wang, M., Wen, Y., Sun, B., Mu, J., Li, J., Wang, X., Ye, J., Yu, B., & Li, H. (2025). Bridging Layout and RTL: Knowledge Distillation based Timing Prediction. Proceedings of the 42nd International Conference on Machine Learning (ICML).

[9] Guvvala, B. (2025). Static Timing Analysis (STA): Fundamentals and Implementation in Nanometer Digital Design. International Journal of Computer Engineering and Technology (IJCET).

[10] Ramegowda, M., Krishnappa, K. H., Venkatesh, D. Y., & Sreenivasa, K. (2025). Optimizing timing closure and enhancing efficiency in RTL design: a focus on physical design tasks for I2C design blocks. Indonesian Journal of Electrical Engineering and Computer Science

[11] Taraate, V. (2014). FPGA Based Pipelined Controller Design and Implementation. International Journal of Engineering Research & Technology (IJERT).

[12] Manjula, B. B., Vidyashree, H., Kavyashree, M. G., Tulasi, V., & Arpitha, R. (2023). Design and Implementation of RISC MIPS Processor on FPGA. International Journal for Research in Applied Science & Engineering Technology (IJRASET).

[13] Suresh, S., & Ganesh, R. (2014). FPGA Implementation of MIPS RISC Processor. International Journal of Engineering Research & Technology (IJERT).

[14] Tyamanavar, V. A., & Nidagundi, J. C. (2019). FPGA Implementation of a 32-Bit MIPS Processor. International Journal of Engineering Research & Technology (IJERT).