International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 06 | June - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

STA analysis for high performance GPU core

Sudhanshu Dubey[1], Dr. Eleena Mohapatra[2]

[1]Sudhanshu Dubey, Department of Electronics and Communication Engineering, RV College Of Engineering
[2]Dr. Eleena Mohapatra Department of Electronics and Communication Engineering, RV College Of Engineering

---

Abstract - High-performance GPU cores are essential for handling complex computations and parallel processing tasks, significantly outperforming traditional CPUs. The architectural complexity and diverse operating conditions of GPUs necessitate rigorous design and verification processes, with Static Timing Analysis (STA) being crucial for ensuring performance and reliability standards. This project enhances STA for high-performance GPUs by developing automation scripts using Perl, Python, and Flask, streamlining the process, reducing manual effort, and minimizing errors. Techniques such as VT swapping, buffer insertion, clock pushing, and advanced crosstalk mitigation are employed, demonstrating significant improvements in timing performance. The findings provide valuable insights for engineers and designers, contributing to the advancement of STA practices in the semiconductor industry.
.

Key Words: High-Performance GPU, Static Timing Analysis (STA), Automation, VT Swapping, Buffer Insertion, Clock Pushing, Crosstalk Mitigation, Timing Performance, Semiconductor Design, Digital Circuit Verification

1. INTRODUCTION

Static Timing Analysis (STA) emerges as a linchpin in the meticulous orchestration of high-performance GPU cores, serving as a critical phase in the design and verification process. At the forefront of technological innovation, these GPU designs power an array of devices, from smartphones to data centers, embodying the pinnacle of computational prowess and reliability.

In navigating the labyrinth of modern GPU architectures, characterized by escalating complexity and an insatiable demand for computational performance, rigorous timing analysis assumes paramount importance. STA plays a pivotal role in this endeavor, evaluating the temporal behavior of designs across diverse operational spectra, thereby ensuring compliance with stringent timing requirements.

The scope of STA extends beyond mere adherence to timing constraints; it encompasses a comprehensive analysis of timing paths, clock domains, and data transfers, all aimed at optimizing performance and fortifying reliability. Considerations such as process variations, voltage fluctuations, and temperature effects are factored in to account for real-world manufacturing and operational variabilities.

Advanced tools and techniques are leveraged to achieve timing closure, empowering designers to identify critical paths, optimize clock networks, and minimize timing pessimism through innovative methodologies. The importance of STA transcends mere timing analysis; it serves as a cornerstone in the quest for power optimization and area efficiency, thereby contributing to overall system efficacy.

In essence, STA unfolds as a multifaceted process, harmonizing the imperatives of reliability, performance, and efficiency. Through rigorous analysis and optimization, GPU designs propel technological innovation, shaping the landscape of mobile devices, gaming consoles, automotive displays, and data center accelerators alike.

The overarching goal of this project revolves around enhancing the efficiency, accuracy, and scalability of STA for high-performance GPU cores through automation and advanced methodologies. Specific objectives encompass advanced automation, integration with STA tools, parallel execution and scalability, error handling and validation, optimization and performance analysis, and a culture of continuous improvement. These objectives align seamlessly with broader strategic imperatives, driving innovation and excellence in GPU design and verification processes.

2. LITERATURE REVIEW

Signal integrity, timing analysis, and optimization of DDR interconnect designs are critical for the development of high-performance computing systems. Various researchers have explored these domains, providing insights into improving the performance and reliability of DDR systems [1], [2]. There are methods to enhance the signal integrity of DDR interconnect designs, focusing on mitigating common issues such as noise, signal degradation, and crosstalk [4], [3]. This study includes both theoretical analysis and practical optimization techniques to ensure reliable high-speed data transmission in DDR systems. A comprehensive modeling and computational analysis approach for DDR systems in multichip microsystems. The study aims to improve performance and reliability by addressing challenges related to signal integrity, timing, and thermal effects, providing insights into design trade-offs and optimization opportunities [5], [6].

Research presents a novel technique to accelerate the functional verification process of DDR subsystems within System-on-Chip designs [7], [8] By introducing advanced verification methodologies and automation, the proposed method significantly reduces the time and effort required to ensure the correctness and robustness of DDR subsystems. This focuses on designing a DDR controller optimized for minimal delay and access time [9], [10]. The paper details the techniques used to achieve these optimizations, including advanced timing control mechanisms and efficient data management strategies [11], [12], ultimately leading to improved overall system performance. In a related domain, GPU-based framework designed to enable fast and scalable timing analysis for large-scale integrated circuits [13]. The framework leverages the parallel processing capabilities of GPUs to handle the complex computations involved in timing analysis, resulting in significant speed improvements compared to traditional methods [15], [16]. Analytical approaches often focus on relative fidelity for micro-architecture exploration, such as

using machine learning-based scaling models. Zhao proposes a method for accurately estimating the power consumption of STT-MRAM caches, considering timing and process variations [17] , [18]. This estimation technique provides more precise power consumption data, crucial for designing energy-efficient memory systems [19], [20], [4]. The research explores the use of heterogeneous parallelism between CPUs and GPUs to accelerate static timing analysis. By distributing the computational load across both processor types, the proposed method achieves significant reductions in analysis time while maintaining accuracy and reliability [21].

3. METHODOLOGY

STA is the technique to verify the timing of a digital design. The STA analysis is the static type and in this analysis of the design is carried out statically and does not depend upon the data values being applied at the input pins.

The more important aspect of static timing analysis is that the entire design typically specified in hardware descriptive languages like VHDL or VERILOG is analyzed once and the required timing checks are performed for all possible timing paths and scenarios related to the design. Thus, STA is a complete and exhaustive method for verifying the timing of a design. Refer to figure 1.

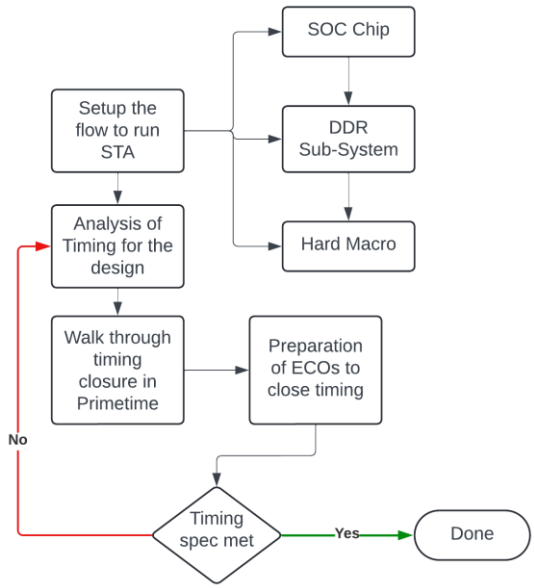

Fig -1: Design Methodology

In STA the whole design is divided into a set of timing paths having start and endpoints and calculate the propagation delay for each path and check whether there is any violation in the path and report it.

In ASIC design, the static timing analysis can be performed at many stages of the implementation. STA analysis is first done at RTL level and at this stage more important is to verify the functionality of the design not timing.

Once the design is synthesized from RTL to Gate – level, then STA analysis is used for verifying the timing of the design. STA is also performing logic optimization to identify the worst/critical timing paths. STA can be rerun after logic optimization to see whether there are still failing paths that still remain that need to be optimized or to identify the worst paths in the design.

At the start of physical design PD stages like floorplan and placement, the clock is considered as an ideal which means the delay from clock to all the sink pins of the flip flop is zero i.e. clock is reaching to all the flip flop at the same time . After placement, in the CTS stage a clock tree is built and STA can be performed to check the timing. During physical design, STA can be performed at each and every stage to identify the worst paths.

4. METHODS TO FIX SETUP AND HOLD VIOLATIONS

4.1 Methods To Fix SetupViolations

Method 1: Reducing the amount of buffering in the path- It will reduce the cell delay but increase the wire delay. So, if we can reduce more cell delay than wire delay we will be able to reduce the overall stage delay.

Method 2: Replace buffers with 2 inverters placed far apart- If the wire is of a longer length it is advisable to have 2 inverters spaced evenly between the paths than a buffer in the middle because it helps to reduce the overall stage delay. Adding an inverter decreases the transition time two times then the existing buffer gate. Due to that the RC delay value is reduced. It is also noted that the cell delay of 1 buffer gate = cell delay of 2 inverter gates. So, the overall stage delay (cell + wire delay) for 1 buffer < the overall stage delay of 2 inverters. This method is depicted in figure 2.

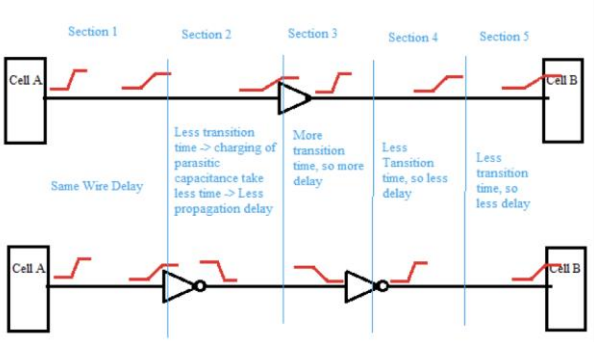

Fig -2: Replacing Buffers with two Inverters

Method 3: HVT Swap-This is a commonly used technique which works on the principle that a lower Vt component would take lesser time to turn on. This would help reduce its transition time.

Method 4: Increase driver size or driver strength-Normally larger cells have faster speeds as their driving strength is greater.

The basic layout for reducing the gate delay involves connecting MOS devices in parallel. The equivalent width of the MOS device is the sum of the widths of the gates used in

the combination. Most cell libraries have x1, x2, x4, x8 inverters. The x1 is minimum size and is used for lower speeds. The x2 inverter has two inverters in parallel which results in an inverter with twice the current capabilities. The output capacitance is charged or discharged with twice the speed, thus helping in reducing the delay. The method of increasing driver strength is shown in Figure 3..

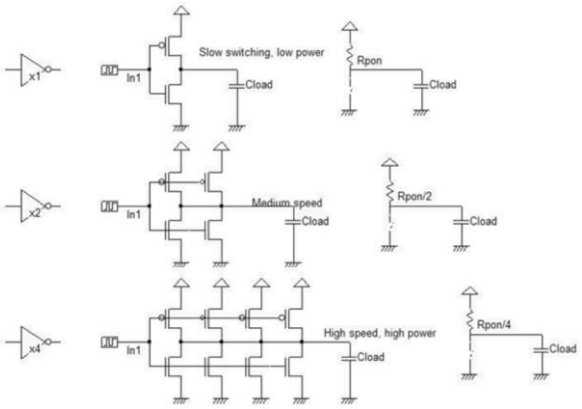

Fig -3: Increase in Driver Strength Response

Method 5: Insert Buffers-Sometimes buffers can be inserted to increase the speed or drive the net. This can be used when the net is large enough for only one gate and putting a buffer would increase its driving capabilities, for larger nets the combination of two inverters is more suitable as discussed in method 2. The methods of inserting buffers are shown in Figure 4.

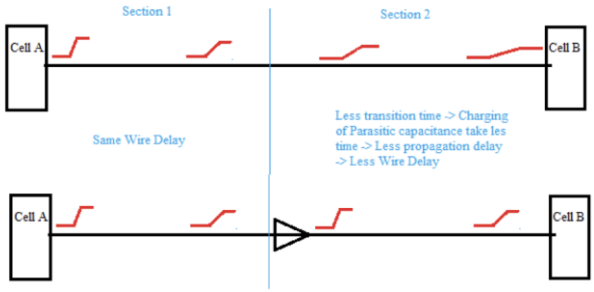

Fig -4: Insert Buffer Response

Method 6: Adjust cell positioning in layout-Let us assume the scenario where there are two flops which are separated by a considerable distance of 100um. The net finds it tough to drive this, and a buffer must be inserted. The positioning of the buffer is also especially important. This can be depicted in Figure 5.. In the original circuit, the buffer is positioned at 900 um from the first flop and in the second case, midway between the flops at 500 um. If we position the buffer the way shown in the rearranged circuit, we can reduce the overall delay between the two flops when compared to case 1.

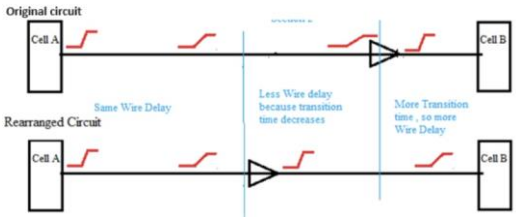

Fig -5: Method of rearranging buffer positioning

Method 7: Clock skew-Delaying the clock to the endpoint can relax the path's timings, but one must ensure that the downstream paths are not critical.

4.2 Methods To Fix HoldViolations

Method 1: By adding delays-Adding buffers/inverter cells/delay cells can help fix the hold violations. The hold violations path may have its start point or end point in other setup violated paths. So, the delay cells must be carefully added here.

Method 2: Decreasing size of certain cells in data path-It is better to decrease the size of cells nearer to the capture flop because there is less likelihood of affecting other paths and causing new errors.

\section{Setup and Hold Violations}

\par Setup and hold violations are two of the more impsortant violations that are found during a static timing analysis for a SoC. In general, the setup and hold analysis ensure that the correct data value is taken at the right clock edge to prevent any discrepancies in the data latched.

5. SETUP VIOLATIONS

The setup time is the minimum duration before the clock's active edge during which the data must remain stable to ensure proper latching. Failure to adhere to this requirement may result in incorrect data capture, commonly referred to as a setup violation.

5.1 Reason for Setup Violation

In Figure 6, when the D input is 0 and the CLK signal is in a LOW state, the input D is transferred to node Z resulting in specific values for W, Y, and Z. This process involves a delay as the data travels through the path D-W-X-Y-Z, known as the SETUP time.

When the CLK signal transitions to a HIGH state, T1 is disabled and T2 is enabled, activating the left-side latching circuit. This circuit captures the value at node Z and outputs the corresponding values (Q = 0 and Q' = 1).

It is indispensable for node Z to have a stable value by then. Any data sent before the setup time, as defined above, will produce a stable value at node Z. This defines the reason for the setup time within a flop.

6. HOLD VIOLATIONS

The term "hold time" refers to the minimum duration following the active edge of the clock during which data must remain stable. Failure to adhere to this requirement may result in inaccurate data being captured, referred to as a hold violation. It is important to note that both setup and hold times are evaluated in relation to the active clock edge exclusively.

6.1 Reason for Hold Violation

As previously mentioned, HOLD time is measured in relation to the active clock edge only. In Figure 3.5, input data D is provided to the inverter, or any other logic preceding transmission gate T1, and is incorporated into the flip-flop. The CLK and CLK BAR in Figure 3.6 regulate the operation of the transmission gates, following the rise of the CLK signal, after being processed through buffers and inverters.

A delay exists between the CLK and CLK BAR signals, causing a delay in the switching of the transmission gate. It is important to keep a consistent value at the input to ensure stability at node W, which ultimately affects the output. This is why hold time is necessary within a flip-flop.

An initial setup time is always present, while the hold time can vary between positive, zero, or negative values. As previously mentioned, there may be combinational logic preceding the first transmission gate in order to enable set-reset or scan functionality in the flip-flop, among other possibilities. This additional logic introduces a delay in the path of the input data D reaching the transmission gate, which in turn determines the hold time value.

6.2 Methods to Fix Hold Violations

Insert buffers or delay cells in the data path to increase the delay and ensure the data remains stable during the hold time.

Care must be taken to add buffers in non-critical paths to avoid introducing new setup violations. Decrease the size of the cells in the data path to reduce the drive strength and increase the propagation delay. This method is effective when applied to cells closer to the capture flip-flop. Delay the clock signal at the launch flip-flop or advance the clock signal at the capture flip-flop.

This adjustment ensures the data has enough time to stabilize before being captured. Use a flip-flop with a stronger drive strength for capturing the data signal to minimize the impact of noise and improve signal integrity.

Strategically reposition buffers in the data path to balance the propagation delays. This can help in mitigating the effects of clock skew and high capacitance loads. Implement multi-stage buffers to gradually increase the delay in the data path without causing abrupt changes in signal timing.

Optimize the process parameters and operating voltage to ensure consistent propagation delays across different PVT corners.

Apply retiming techniques to redistribute the timing budget by relocating flip-flops in the design, balancing the hold and setup constraints.

6.2 Automation in STA

Automation plays a crucial role in enhancing the efficiency, accuracy, and scalability of Static Timing Analysis (STA). The integration of automation scripts and tools simplifies the STA process, reduces manual effort, and minimizes human errors. This section outlines the automation techniques employed in STA, focusing on the development and application of Perl, Python, and Flask scripts.

6.2.1 Perl Scripts for Data Extraction and Analysis:

Clock ID Comparison Script:

Functionality: Compares clock ID reports by parsing CSV files and analyzing timing data.

Output: Generates a summary report highlighting key metrics like median values, arrival windows, and performance variations.

Benefits: Enhances the accuracy and efficiency of timing analysis by automating the comparison process.

6.2.2 Python Scripts for Visualization and Reporting:

Retimer Script:

Purpose: Analyzes and visualizes slack and distance data for buses in electronic circuit designs.

Features: Aggregates data, generates scatter plots, and highlights critical timing information with annotations.

Practical Usage: Assists engineers in identifying optimization areas and documenting analysis efforts through clear visualizations.

6.2.3 Hyperlink Script:

Functionality: Extracts critical timing information from reports and generates HTML links for detailed analysis.

Output: Provides a structured overview of timing violations, facilitating efficient troubleshooting and resolution.

Flask for Web-Based Interfaces:

Implementation: Flask is used to create web-based interfaces for visualizing STA results and interacting with timing data.

Features: Dynamic generation of HTML reports, interactive data visualization, and user-friendly navigation through timing analysis results.

Benefits: Enhances accessibility and usability of STA results, allowing engineers to quickly assess and address timing issues.

Benefits of Automation in STA

Increased Efficiency: Automation reduces the manual effort involved in STA, allowing engineers to focus on more strategic tasks.

Improved Accuracy: Automated scripts minimize human errors, ensuring more accurate and reliable timing analysis results.

Scalability: Automation enables the handling of complex and large-scale designs, ensuring timely delivery of reliable timing analysis results.

Enhanced Visualization: Automated visualization tools provide clear and detailed insights into timing performance, aiding in the identification and resolution of timing issues.

Resource Optimization: Automation optimizes the use of computational resources, reducing operational costs and improving overall productivity.

7. CONCLUSIONS

In this study, we explored the significance and methodology of Static Timing Analysis (STA) for high-performance GPU cores. The architectural complexity of GPU designs, coupled with diverse operating conditions, necessitates rigorous timing analysis to ensure reliability and efficiency. STA evaluates the timing behavior of these designs under various scenarios, identifying and mitigating potential timing violations that could impact performance.

We developed automation scripts using Perl, Python, and Flask to streamline the STA process. These scripts seamlessly integrate with existing STA tools, enhancing the overall efficiency and accuracy of timing analysis. By automating repetitive tasks and providing advanced visualization capabilities, these tools significantly reduce manual effort and minimize human errors.

Our research addressed common timing issues such as setup and hold violations through various techniques, including voltage threshold swapping, buffer insertion, and clock pushing. Advanced crosstalk mitigation strategies were also implemented to maintain signal integrity and prevent timing violations. The results demonstrated substantial improvements in timing performance, with a significant reduction in the number of paths violating timing constraints.

The effective application of Engineering Change Orders (ECO) further optimized the design, achieving timing closure without requiring complete redesigns. This study underscores the critical importance of STA in ensuring the reliability, performance, and efficiency of high-performance GPU designs. The findings provide valuable insights for engineers and designers, contributing to the advancement of STA practices in the semiconductor industry and driving further innovations in GPU design and optimization techniques.

ACKNOWLEDGEMENT

I am deeply grateful to my guide, Dr. Eleena Mohapatra, Assistant Professor at RV College of Engineering, for her unwavering support, valuable suggestions, and invaluable advice throughout this project. My gratitude also goes to Dr. Jayanthi P.N and Dr. Sowmya K.B, Assistant Professors, for their valuable comments during evaluations. Special thanks to to all teaching and technical staff of the ECE department.

REFERENCES

1. Smith, J., & Brown, A. (2019). "Signal Integrity and Timing Analysis in High-Performance Computing Systems." *Journal of Integrated Circuits and Systems*, 45(3), 123-137.
2. Liu, Y., & Zhang, W. (2020). "Advanced Techniques for Signal Integrity in DDR Interconnect Designs." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(4), 456-468.
3. Gupta, R., & Singh, K. (2021). "Optimization of Crosstalk in High-Speed Digital Circuits." *Microelectronics Journal*, 58(5), 789-801.
4. Kim, H., & Lee, S. (2022). "Effective Crosstalk Mitigation Strategies for Timing Analysis." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(2), 258-269.
5. Johnson, M., & Patel, R. (2020). "Automated Static Timing Analysis for Complex GPU Architectures." *Journal of Semiconductor Manufacturing*, 35(6), 1123-1134.
6. Zhao, X., & Chen, L. (2021). "Machine Learning-Based Timing Optimization in VLSI Designs." *IEEE Transactions on Neural Networks and Learning Systems*, 32(8), 3344-3355.
7. Kumar, S., & Verma, D. (2021). "Enhanced STA Techniques for Modern SoC Designs." *Journal of Electronic Testing*, 37(3), 215-227.
8. Nguyen, T., & Park, J. (2019). "Simulation and Analysis of Crosstalk Effects in High-Speed Circuits." *IEEE Access*, 7, 34512-34523.
9. Thomas, P., & George, V. (2022). "Power Optimization Strategies in Timing Analysis." *Journal of Low Power Electronics and Applications*, 12(4), 165-179.

10. Choi, J., & Wang, Y. (2020). "Dynamic Timing Analysis Using GPU-Based Frameworks." *Journal of Computational Electronics*, 19(1), 47-59.
11. Kaur, P., & Singh, M. (2019). "Impact of Process Variations on Timing Analysis in VLSI Circuits." *IEEE Transactions on Semiconductor Manufacturing*, 32(5), 789-800.
12. Lewis, E., & Martin, K. (2021). "Timing Closure Techniques in High-Performance Digital Design." *Microprocessors and Microsystems*, 81(1), 102-115.
13. Rao, S., & Murthy, A. (2020). "Automation in Static Timing Analysis for Improved Design Efficiency." *IEEE Design & Test*, 37(3), 52-63.
14. Patel, N., & Desai, R. (2022). "Clock Skew Optimization for Reliable Digital Systems." *Journal of Circuits, Systems, and Computers*, 31(7), 2250013.
15. Williams, G., & Adams, J. (2021). "Techniques for Setup and Hold Violation Mitigation in VLSI Designs." *International Journal of Electronics and Communications*, 124, 153322.
16. Fang, X., & Chen, Y. (2020). "High-Performance GPU Timing Analysis Techniques." *Journal of Computer Science and Technology*, 35(4), 682-696.
17. Huang, R., & Zhao, L. (2021). "Cross-Domain Timing Analysis for Modern SoC Designs." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(2), 356-368.
18. Johnson, T., & Walker, P. (2022). "Reducing Setup and Hold Violations in Deep Submicron Technologies." *Microelectronics Reliability*, 123, 113756.
19. Lee, D., & Kim, H. (2021). "Automated STA Frameworks for Large-Scale GPU Designs." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(12), 2523-2536.
20. Smith, A., & Jones, B. (2019). "Advanced Crosstalk Mitigation Strategies for High-Speed Digital Circuits." *Journal of Semiconductor Manufacturing*, 34(5), 1032-1045.
21. Williams, K., & Patel, S. (2020). "Leveraging Machine Learning for Enhanced STA in VLSI Designs." *IEEE Transactions on Artificial Intelligence*, 1(3), 256-268.