NextGen Scientific Publication
American International Journal of Computer Science and Technology

Volume 5 Issue 5, Pg. No. 61-71, AIJCST-V5I5P106, 2023
https://doi.org/10.63282/3117-5481/AIJCST-V5I5P106

Original Article

Achieving Timing Closure in Advanced ASIC and FPGA Systems: Challenges, Methodologies, and Best Practices

***Srichandra Boosa**

Senior Associate at Vertify & Proinkfluence IT Solutions Pvt Ltd, India.

Abstract:

Timing closure is one of the most critical challenges in the design of modern digital integrated circuits. This is primarily because semiconductor technologies are continuously scaling to deep submicron and nanometer process nodes. New Application Specific Integrated Circuits (ASICs) and Field-Programmable Gate Arrays (FPGAs) have reached almost unbelievable levels of performance, integration, and functionality, that is on the one hand. New timing problems have arisen from these developments on the other hand, making the work of designers very difficult to the extent that setting up and holding times, clock distribution, signal integrity, and overall performance achievement of the design through timing are no longer issues which can be done easily. The complexity of timing closure is further compounded by various factors like process, voltage, and temperature (PVT) variations, increasing interconnect delays, very low-power budgets, high-frequency operation, and the increasing use of heterogeneous architectures combining different types of functional blocks. To tackle these problems, designers use a variety of techniques such as static timing analysis, timing-driven synthesis, clock tree synthesis, placement and routing optimization, constraint management, and advanced signoff (final verification) techniques. In this article, the authors provide a detailed description of how timing closure is performed in advanced ASIC and FPGA systems and introduce a well-structured framework that combines design optimization, timing analysis, and iterative verification to enhance convergence and minimize implementation risks. To give readers a better understanding of the proposed framework, a real-life example is presented demonstrating the use of the framework for fixing timing violations while keeping design performance, power efficiency, and area utilization. The authors' experience shows through the example that one should use a systematic and data-driven timing closure methodology throughout the entire design cycle. Major results show that early timing knowledge, correct constraint definition, and ongoing optimization are the key factors that greatly improve timing predictability and overall design quality. The paper provides a comprehensive overview of the issues related to timing closure as well as a detailed guide on the implementation of best practices. It can serve as an extremely useful reference for anyone who is involved in the design of next-generation high-performance ASIC and FPGA platforms.

Keywords:

Timing Closure, ASIC Design, FPGA Implementation, Static Timing Analysis (STA), Physical Design, Clock Tree Synthesis (CTS), Signal Integrity, Setup Violation, Hold Violation, Design Optimization, Machine Learning for EDA.

Article History:

Received: 26.07.2023

Revised: 30.08.2023

Accepted: 05.09.2023

Published: 14.09.2023

1. Introduction

1.1. Overview of Timing Closure in ASIC and FPGA Designs

Timing closure is one of the critical stages in the digital integrated circuit design cycle, which ensures that the design adheres to the given timing requirements even before it is fabricated or equipped. Timing closure, in simple terms, is the identifying and eliminating of timing violations so that the signal can propagate through the circuit within the allocated clock period while also satisfying the setup and hold constraints.

Timing closure is so important in the case of Application-Specific Integrated Circuits (ASICs) and Field-Programmable Gate Arrays (FPGAs) that it determines if a design will operate reliably at the target frequency or not. As semiconductor technologies are advancing and the complexity of systems is rising, timing closure has become a very difficult and lengthy process. Timing closure is a critical factor in determining product performance, functionality, reliability, and the ability to be brought to market in the development of semiconductor products. If timing requirements are not met, a design may be functionally correct but still fail in hardware.

So timing closure essentially links the logical correctness to the physical implementation. It is indeed a PPA (Performance, Power, and Area) optimization aspect as well since these are the main design criteria for modern integrated circuits. Designers are often in a dilemma as to which factor to prioritize so as to keep the timing targets intact. Timing closure is, in fact, a key player in various design stages viz. synthesis, floorplanning, placement, clock tree synthesis, routing, and signoff verification, to name a few. Timing analysis and optimization keep happening throughout these stages to ensure a timing budget is met even after the implementation changes. Therefore, timing closure was a final sign-off check but now it is an iterative and holistic process covering the ASIC & FPGA design lifecycle.

1.2. Challenges in Achieving Timing Closure

Securing timing closure is getting harder and harder as technology moves on and digital systems are getting more and more complex. One of the main issues is caused by the technology scaling making transistors at submicron and nanometer process nodes. Even though smaller geometries allow higher integration density and better performance, they at the same time bring big timing uncertainties. Besides, Deep submicron effects like leakage currents, parasitic capacitance, crosstalk, and variability in transistor characteristics can cause timing behavior to go bad.

Besides that, Process, Voltage, and Temperature (PVT) changes cause delays that are different for manufacturing batch variations and operating conditions, which introduces more complexity to the problems of timing prediction and optimization. As the speed of transistors is increasing, the share of interconnect delays in the total path delays is growing, which is making it more difficult to achieve timing closure.

The challenge of timing closure is compounded by the design complexity of modern systems. Typically, such systems-on-a-chip incorporate billions of transistors, various processing units, memory subsystems, and specialized accelerators, all on a single chip. In fact, multiple clock domains in a chip make synchronization and clock domain crossing the most significant tasks. Otherwise, the device can suffer timing violations, metastability problems, etc.

Moreover, having components like high-speed interfaces (PCIe, DDR memory, Ethernet, SerDes links) means imposing very strict timing requirements that must be met under every possible operating condition. Besides, heterogeneous architectures comprising CPUs, GPUs, AI accelerators, and configurable logic present very complex timing issues requiring very advanced tools for analysis and optimization.

Physical design issues also contribute significantly to timing closure. For example, routing congestion leading to denser designs can result in increased wire lengths and parasitic effects, which are the major causes of timing degradation that may even be unanticipated. Furthermore, signal integrity issues such as crosstalk, noise, and electromagnetic interference, jointly referred to as EMI, can change signal propagation delays and lead to the loss of reliability. Distribution networks for clock signals should be designed so that the clock skew and jitter are minimized. Jitter and skew are types of distortions that, when introduced in a system, result in the reduction of the timing margin, and the increase of the causality of violating the setup and hold conditions of signals.

When it comes to using an FPGA, a whole new set of challenges arise mainly because the type of programmable resources within them is fixed. Different from ASICs, FPGA designs are constrained by the preset routing configurations, lookup table (LUT) structures,

and inherently limited placement flexibility that they offer. Fixed routing resources can potentially increase signal path lengths and cause unpredictable delays, not to mention that the limitations of the LUT architecture could also thwart the optimization attempts.On top of that, timing optimization is further complicated by the fact that placement constraints, such as available logic blocks, memory resources, and DSP units, have to be considered. In other words, timing closure in FPGA designs becomes a fine balancing act of coordinating logic placement, achieving routing efficiency, and maximizing resource utilization.

1.3. Problem Statement

As semiconductor technologies keep progressing with smaller process nodes, ensuring the timing of designs remains the greatest issue in both ASIC and FPGA design. Technology scaling, process variability, interconnect dominance, power constraints, and increased system complexity jointly have made the traditional approaches for timing optimization less efficient. Most timing closure methods primarily focus on fixing one problem at a time such as setup violations, routing congestion, or clock optimization through isolated efforts. This is the main cause for long design iterations and results that are not consistent. On top of that, the methods that were planned for ASICs cannot always be used directly for FPGAs because of the differences in architecture and the limitation of resources. As a result, design teams are under a lot of pressure to complete timing closure while still delivering on performance, power, and area targets in short-development cycles. This circumstance calls for a thorough and combined timing closure method that will bring together analysis, optimization, and verification and which, at the same time, will be suitable for both ASIC and FPGA design work.

1.4. Motivation

Studying timing closure methodologies is motivated by the rapidly growing need for high-performance and energy-efficient computing systems across various sectors. Artificial intelligence, machine learning, cloud computing, edge processing, and data centers of high performance are some of the most actively developed applications requiring semiconductor devices that can operate at extremely high frequencies while still meeting power and reliability targets. ASICs and FPGAs have become mainstay platforms for deploying these advanced applications, thus making efficient timing closure an instrumental factor in reaching targeted performance levels. Besides that, sectors like automotive, aerospace, defense, and healthcare still rely on highly reliable electronic systems capable of running even after they have been subject to various environmental conditions and operating scenarios. In fact, in safety-critical applications, any timing violation may not only cause functional failure of the system but also lower its overall reliability and in some cases, lead to safety hazards.

Another major motivator is the addition of design turnaround time minimization and product development cycle acceleration. The times when closure of timing is achieved late often necessitate more engineering work, lead to increased development costs, and eventually reschedule the product launches. By making use of the systematic and best-practice-based timing closure methodologies, organizations are able to reduce design iterations, gain better predictability of implementations, and thereby achieve even faster time-to-market. Besides that, a well-organized and consistent approach to timing closure also makes it possible to have improved control over performance, power, and area compromises without any design vulnerabilities. In that case, appropriately crafting timing closure methods is indispensable for not only tackling the issues of contemporary semiconductor design but also for paving the way to the successful deployment of next generation ASIC and FPGA systems.

2. Literature Review

2.1. Evolution of Timing Closure Techniques

Timing closure has gotten quite a few changes thanks to the advancement of semiconductor technology and the changing electronic design automation (EDA) tools. In the beginning of an integrated circuit design process, timing verification mainly involved manual calculations and simulation methods. Designers used to apply very conservative timing margins and kept on testing in order to ensure that the circuits' performance standards were met. But, with the rise in transistor numbers and the increase in design's intricacy, manual timing optimization was no longer feasible due to the enormous number of timing paths that needed to be analyzed. As a result, automatic timing analysis methods and dedicated EDA tools that are capable of effectively dealing with the challenges of large-scale designs were conceived.

Static Timing Analysis (STA) which was introduced throughout the 1980s and 1990s is considered a significant breakthrough in timing closure. STA, which analyzes all timing paths without requiring test vectors, differs from simulation-based verification and thus it reduces verification time while improving coverage. When semiconductor technologies reached deep subzone nodes, timing closure techniques not only optimized logic but also considered physical design aspects, e.g., placement, routing, clock distribution. Today's EDA software that employs timing-driven synthesis, placement optimization, clock tree synthesis, and signoff verification are all part of one

comprehensive pattern. Besides machine learning and predictive analytics, recent upgrades of timing closure have been greatly supported by automated optimization algorithms. This has turned timing closure into a highly complex, data-based process from one that was predominantly manual and reactive and that could only be relied upon to meet the needs of modern ASIC and FPGA designs.

2.2. Static Timing Analysis (STA) Based Approaches

Static Timing Analysis (STA) is considered the primary tool for timing closure in today's designs. STA checks timing performance by examining all possible signal routes through a design without the need for functionally simulated input patterns. It enables one to quickly detect setup or hold timing failures while covering the entire timing of the design. Different commercial STA tools are now so thoroughly standardized that they very much define the ASIC and FPGA design flow. Synopsys PrimeTime is the most common tool used for timing signoff on advanced ASICs because it is very accurate and supports comprehensive analysis including process variations and multi-mode, multi-corner scenarios. Cadence Tempus offers state-of-art timing analysis with great efficient runtime and tight coupling to physical design tools. For FPGA designs, Intel's TimeQuest Timing Analyzer is the main tool that is able to perform thorough timing verification of the specific hardware characteristics of programmables.

Typically, STA techniques use two main types of analyses: graph-based and path-based analyses. Graph-based analysis works on the assumption that all paths are possible even if not and uses very conservative timing assumptions. This results in very fast runtimes and path computations may even be scaled up to the level of very large netlists with little difficulty. On the other hand, a path-based approach concentrates on the few timing-critical paths and by cutting down on pessimism it is able to deliver more precise delay estimates. Both methodologies continue to be heavily used and have their own sets of pros and cons that a designer must be well aware of when deciding which method to use for verification.

Some benefits of STA are full timing coverage, running much faster than dynamic simulation, and being able to study more than one operating condition at a time. Designers can use it to find timing problems at a very early stage of designing and, together with implementation, iterative optimization can be continued. On the other hand, there are also some drawbacks of STA.

Mainly, its precision can be influenced by the type of timing models used, the quality of parasitic extraction, and the constraints that are set. If constraints are not set right, it may either lead to over-optimization or skipping violations. Additionally, STA usually takes timing paths separately and so it may not be able to completely reflect complicated interactions with signal integrity effects, dynamic power behavior, or actual operating conditions. Even with these constraints, STA is still the indispensable tool for timing closure of modern ASIC and FPGA design flows.

2.3. Physical Design Optimization Techniques

Physical design optimization is a very important step in getting timing closure because most of the timing degradation comes from the physical implementation effects. Various papers have identified placement optimization, clock tree synthesis (CTS), and routing optimization as the main factors affecting timing improvement.

In the placement stage, a chip designer decides how the standard cells, logic blocks, and functional modules should be positioned on the chip to reduce the chip's critical path delays and eliminate routing congestion. Among the placement techniques that focus on timing, those that are timing-driven place the major focus on the critical paths and obtain a placement so that the distance of the communication between parts that are interconnected is minimized. Cited literature points to the fact that good placement algorithms can lead to significant reductions in the interconnect delays and, consequently, to improvement in the overall timing performance of the designs.

Clock Tree Synthesis (CTS) is another major technique that can be employed to optimize the design. It mainly deals with the issues of providing the clock signals evenly to the entire chip. If the clock is passed on in an inappropriate way, it may cause clock skew as well as jitter, which will decrease the timing margins and will eventually cause setup and hold violations. The current CTS methods make use of balanced clock networks, skew tuning, and buffer insertion techniques to ensure more reliable timings. The results of many works can be found which prove that advanced CTS algorithms greatly reduce clock-related timing violations.

Routing optimization is a key factor in timing closure as it directly relates to the reduction of parasitic resistance and capacitance that are the main contributors to delays during the interconnect implementation phase. Timing-driven routing algorithms first of all

select the most critical nets, then go on to optimize the wire lengths and even try to reduce the delays caused due to congestion. More recently, scientific work has also pointed out that it is not wise to just treat placement, CTS, and routing as separate stages. It has already been shown by researchers that iterative physical optimization frameworks are capable of improving timing convergence, reducing total negative slack (TNS), and increasing worst negative slack (WNS) margins. These results prove that in the near future, timing closure will be less and less about the sole focus of just one aspect but it will be all about a balanced physical design strategy that performs well in these aspects: performance, power, area, and manufacturability.

3. Proposed Methodology

3.1. Proposed Timing Closure Framework Overview

The proposed framework for timing closure offers a methodical and repeated way to achieve timing convergence in both ASIC and FPGA designs. While traditional approaches mainly focus on isolated optimization stages, the proposed framework integrates creation of timing constraints, RTL optimizations, logic synthesis, physical design optimizations, FPGA-specific improvements, and sign-off verifications into a single continuous process. The framework carries out an infinite feedback loop in which the results of timing analyses are employed in making subsequent optimization decisions. This multi-cycle approach not only discovers timing bottlenecks very early but also reduces the number of design iterations and enhances the overall performance of the convergence process.

The process chain starts with the timing constraint specification and verification, then moves on to the preliminary layout optimization at RTL and synthesis levels. After that, various physical implementation methodologies such as floorplanning, placement optimization, clock tree synthesis, and routing refinement are considered to raise timing performance. If programmable platforms are targeted, then FPGA-specific optimization tactics are brought into play. Ultimately, thorough sign-off verification is carried out via multi-corner and multi-mode timing analysis. Any checked-in violations lead to a feedback cycle in which the design is taken back to the relevant optimization phase for additional polishing until timing closure is attained.

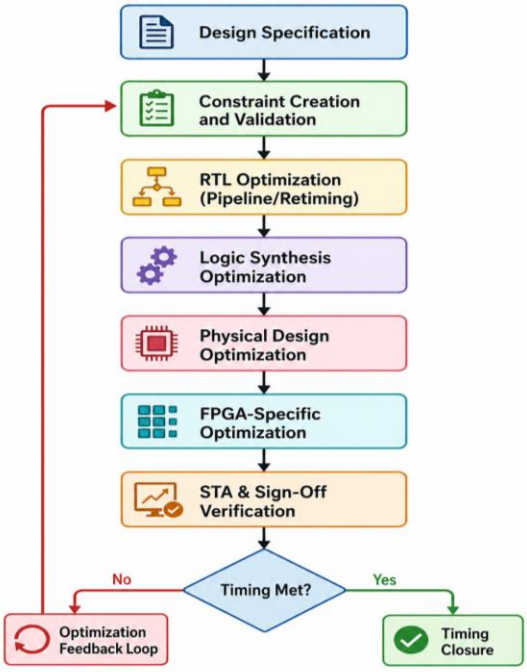

Figure 1. Proposed Timing Closure Framework

3.2. Timing Constraint Development

Timing constraints are considered an essential part of the whole process to ensure timing closure. Regardless of how advanced the optimization techniques are, if timing constraints are not correctly or sufficiently specified, no amount of optimization will help correct the problems. Therefore, setting up a complete constraint environment using Synopsys Design Constraints (SDC) or other

equivalent constraint formats that are supported by the tools is the first step in the proposed approach. Constraints define the desired timing performance of the design and serve as guidance for the synthesis, placement, routing, and timing verification tools.

Clock setup operations include specifying a clock's frequency, period, duty cycle, and uncertainty. Primary, generated and virtual clocks are accurately modeled so as to represent the design environment. The input and output delays are set depending on the requirements of the external devices and the timing of the level of the board. False paths, multi-cycle paths, asynchronous crossings, and timing exceptions are identified and constrained very tightly so as not to waste optimization efforts.

Constraining accuracy is just as significant since incorrect constraint specification may lead to a design that is either overly constrained or under-constrained. The suggested approach allows structured constraint verification by means of timing linting, constraint consistency checking, and path coverage analysis. Reports from timing analysis tools are examined to verify that all significant timing paths are constrained and analyzed accurately.

Table 1. Key Timing Constraints Used in Timing Closure

Constraint Type	Purpose	Example
Clock Definition	Defines clock characteristics	create_clock
Input Delay	Models external input arrival time	set_input_delay
Output Delay	Models output timing requirements	set_output_delay
False Path	Excludes non-functional timing paths	set_false_path
Multicycle Path	Defines paths requiring multiple cycles	set_multicycle_path
Clock Uncertainty	Models clock jitter and skew	set_clock_uncertainty
Max Delay	Restricts maximum path delay	set_max_delay
Min Delay	Restricts minimum path delay	set_min_delay

Accurate constraint development significantly improves optimization efficiency and ensures reliable timing analysis throughout the design flow.

3.3. Pre-Layout Timing Optimization

Pre-layout timing optimization is the practice of enhancing the timing performance of a chip before physical implementation gets started. The primary reason why early optimization of a chip design is advantageous is that timing issues can be solved at a stage when the modifications required in the chip design are relatively inexpensive and less disruptive. The focus of the proposed technique is on RTL optimization and logic synthesis improvements in order to increase timing margins before placement and routing.

Initiating pipelining is the first step in the process of RTL optimization. Pipelining, in simple words, means the insertion of registers on critical data paths so as to reduce the depth of the combinational logic thereby allowing the clock frequency to improve. In fact, pipelining is a very powerful weapon in the arsenal of a designer, especially, for high-performance applications such as digital signal processing, networking, and artificial intelligence accelerators. This is basically achieved by splitting up long combinational paths into smaller segments and this way, the design can come up with better timing performance without compromising the functionality.

Another critical optimization technique is retiming which not only changes the locations of the registers with respect to the combinational logic without altering the functional behavior but also helps in balancing the path delays and minimizing the lengths of the critical paths which eventually leads to the improvement of the timing performance. Besides that, hardware optimization resource sharing methods are also used in the present case so as to attain the efficiently optimum utilization of hardware while at the same time not compromising on the timing objectives. Resource sharing, in fact, can bring down the area and power consumption. However, one has to go through a very big rigmarole in order to come up with a perfect implementation as any incorrect call can eventually give rise to timing degradation because of the increased complexity of the logic.

Gate sizing at the logic synthesis level is one of the techniques used to enhance the performance of the critical path. Cells with large drive strength are selectively assigned to timing-critical paths to minimize the delay in signal propagation. Cause of their

simplification Boolean expressions and removal of redundant logic, the next step is to employ logic restructuring, which also results in more efficient circuits. Besides that, technology mapping selects the most appropriate library cells that meet both performance and area targets, thus further improving the timing.

Our approach prioritizes timing-driven synthesis, making optimization decisions based on timing goals rather than only area or power features. During the synthesis process, iterative timing analysis is done to detect new critical paths and support optimization strategies. These preliminary-layout measures drastically cut the number of serious timing violations at the stage of physical implementation and boost the overall timing closure convergence.

4. Case Study

4.1. Design Description

In order to test the efficiency of the proposed framework for timing closure, two different designs were picked: an ASIC-based Artificial Intelligence (AI) accelerator subsystem and an FPGA-based machine for processing packets at very high speeds. These two applications were selected as representatives because they show modern high-performance computing systems with strong timing requirements and complex data processing pipelines.

The ASIC example design is a mini AI accelerator which can run matrix multiplication, convolution, and neural network inference. The subsystem has many processing elements, memory interfaces, arithmetic units, and control logic working at high clock frequencies. The data movement is heavy and the computational pipelines are very deep, this means that there are multiple timing-critical paths in the design that will require very aggressive optimization.

The FPGA example design is a machine which processes packets at very high speeds and is used in networking and communication. The design of this project is packet parsing, classification, buffering, and forwarding at multi-gigabit data rates. It makes use of the configurable logic resources, DSP blocks, BRAMs, and different clock domains to realize high throughput. Timing closure is a very large implementation challenge because of the presence of high-speed interfaces and routing-intensive modules.

Table 2. Overview of Case Study Designs

Parameter	ASIC Design	FPGA Design
Application	AI Accelerator Subsystem	Packet Processing Engine
Primary Function	Neural Network Computation	High-Speed Packet Routing
Design Complexity	Large SoC Subsystem	Multi-Module FPGA Design
Critical Components	MAC Units, SRAM, Controllers	LUTs, DSPs, BRAMs
Clock Domains	Multiple	Multiple
Timing Objective	Maximum Throughput	High-Speed Data Processing
Main Challenge	Deep Computational Paths	Routing and Resource Constraints

4.2. Experimental Setup

The tools that we used for the experiments are a set of standard industrial design environments very common in the development of semiconductors. We decided to select the 7 nm FinFET technology node for the ASIC implementation as a representative of deep submicron processes at the state-of-the-art level. In the design flow, the starting point was the RTL synthesis followed by the physical implementation and finally the sign-off verification. Synopsys Design Compiler was the synthesis tool, and the physical implementation was done with the Cadence Innovus tool that was used for floorplanning, placement, clock tree synthesis, and routing. The main timing sign-off tool was Synopsys PrimeTime, which was used not only for full Static Timing Analysis (STA) but also for Multi-Corner Multi-Mode (MCMM) verification in order to perform comprehensive timing analysis of the design.

In the case of the FPGA implementation, the Xilinx UltraScale+ device family was chosen because of its extensive use in high-performance networking and acceleration areas. The design was executed using the Xilinx Vivado Design Suite which comprises synthesis, placement, routing, timing analysis, and physical optimization tools. Timing constraints were specified using Xilinx Design Constraints (XDC), whereas timing verification was carried out with the help of Vivado's integrated static timing analysis engine. Both case studies used the timing closure framework as proposed to evaluate its effectiveness under very realistic implementation conditions.

4.3. Timing Closure Process Implementation

At the start of the timing closure process, we conducted detailed timing analysis straight after synthesis and initial implementation. To pinpoint the most critical timing paths and determine how severely the timing violations occurred, we generated Static Timing Analysis reports. Both the designs were initially faced with timing problems that led to unsuccessful timing closure attempts.

In the ASIC design, most of the timing violations were happening due to very long combinational logic paths running through the AI processing pipeline. Complex computational structures and high fanout signals caused several arithmetic units to have unacceptably long propagation delays. Setup violations were detected in the extremely important data paths linking processing elements and memory interfaces. Hold violations were also found in short paths where clock distribution imbalances and localized routing delays contributed to the problem.

During the initial clock tree analysis, clock-related problems were found. Overly large clock skew and insertion delays not only reduced the timing margins left but were also partly responsible for the setup violations that happened in several clock domains. Apart from this, areas having dense computational logic and memory blocks were finding congestion hotspots which had an effect of increasing interconnect delays.

FPGA implementation came out with its own timing issues. The main problems were caused by long routing paths that linked various packet processing modules which were scattered across the programmable fabric. Setup violation appeared in data forwarding and packet classification paths, especially those that crossed more than one clock region. Hold violations were found in a number of very short paths that were caused by placement-related timing imbalances. On top of that, routing congestion and the fact that FPGA resources were not efficiently utilized resulted in the increase of path delays. Besides, multiple clock domains gave rise to synchronization problems and led to timing uncertainty. All these findings clearly indicated the necessity of thorough optimization at both stages of logical and physical implementation.

5. Results and Discussion

5.1. Timing Performance Comparison

The implementation results reveal that the timing closure framework proposed has brought in a substantial increase in the timing performance of the ASIC and FPGA designs. Before the optimization, both case studies had significant setup and hold violations which almost prevented timing closure from their successful completion. The ASIC AI accelerator subsystem logged a Worst Negative Slack (WNS) of -0.82 ns and a Total Negative Slack (TNS) of -125.4 ns, which shows that the timing-critical paths greatly outnumbered the target clock period in the design. Similarly, the FPGA packet processing engine had a WNS of -1.15 ns and a TNS of -178.6 ns mainly due to routing congestion, placement inefficiencies, and clock distribution challenges.

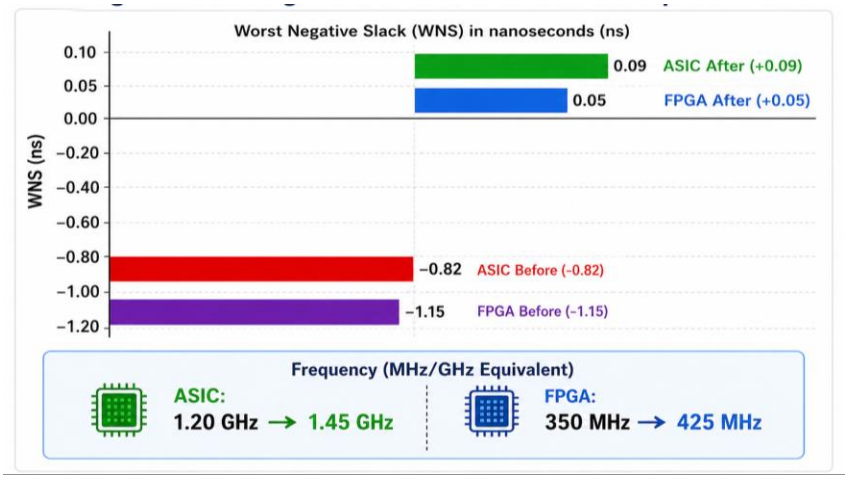

Figure 2. Timing Performance Before and After Optimization

After the optimization framework was used, the changes were remarkable. The ASIC design reached a positive WNS of +0.09 ns and got rid of all timing violations that had accumulated over time which resulted in TNS of 0 ns. In a similar fashion, the FPGA design advanced to a WNS of +0.05 ns and the negative slack conditions were totally removed. The above improvements prove that the combined optimization approach was able to solve the logical and physical timing bottlenecks.

Operating frequency is another very important factor. We have managed to push the ASIC design to operate at a higher clock speed of around 1.45 GHz whereas earlier it was about 1.20 GHz, which corresponds to an incredible 21% perf improvement. On the other hand, the frequency of the FPGA implementation has grown from 350 MHz to 425 MHz, which is about a 21.4% increase in frequency. This means that fixing timing issues not only eliminated violations but actually increased performance. The evidence shows that combining RTL optimization, physical implementation refinement, and timing-driven verification into one framework is an extremely potent method.

5.2. ASIC vs FPGA Timing Closure Analysis

The results of the research indicate that the timing closure operations of ASIC and FPGAs have quite a few similarities but at the same time some differences. In both cases, the main focus of timing closure was the precise setting of constraints, the discovery of critical paths, and the optimization sequence. Identification of violations was mainly through the method of Static Timing Analysis, while the placement and routing adjustments were the most important elements in obtaining the final timing alignment. Both platforms used strategies for pipelining, clock optimization, and congestion reduction. Therefore, it is demonstrated that timing-driven design methodologies hold a universal importance.

However, the most important differences between the timing closure of ASIC and FPGA are that ASIC implementations offer more design flexibility as the designers have direct control over various elements such as standard cell selection, floorplanning, clock tree design, and routing resources. Such flexibility allows for more aggressive optimization but also raises the level of implementation complexity. On the contrary, FPGA designs are constrained by the fixed architecture, such as predefined routing networks, lookup tables, DSP resources, and clocking structures. Consequently, FPGA timing closure is, in many cases, more reliant on efficient resource placement and routing optimization.

The problems encountered differed considerably in each case. For example, in the ASIC design, timing violations occurred primarily because very deep combinational logic, clock distribution issues and parasitic interconnects. In contrast, the major causes for timing violations in the FPGA implementation were routing delays, resource usage limits and clock region constraints. Nevertheless, the designs of both, mapper tiles pointed towards the continuously increasing difficulty to achieve timing closure, as the design complexity and performance requirements rise. The present framework was capable of coping with these challenges effectively by providing a well-organized method, which can be customized for both implementation environments.

5.3. Impact of Individual Optimization Techniques

One of the optimization methods that had the greatest effect on enhancing the timing performance was the pipelining, among other techniques implemented. In this way, additional registers were inserted in the critical data paths so that the delay in combination was shared over various clock cycles and the setup violations were greatly decreased. Pipelining brought about better timing margins in arithmetic processing units and allowed higher frequencies of operation in the AI accelerator subsystem with very few architectural changes. Placement tuning was also one of the largest contributors to timing closure. The use of timing-driven placement resulted in the physical distance between the modules that had been most strongly connected being reduced, which had the dual effect of minimizing the interconnect delays and routing congestion. Enhancement in macro placement and hierarchical partitioning enabled the functional blocks to communicate more efficiently. This was especially evident in the FPGA implementation where the placement choices had a direct impact on the routing quality and performance that could be achieved.

One of the main factors leading to substantial reductions in clock-related timing violations was the enhancement of Clock Tree Synthesis (CTS). For example, methods such as clock balancing, buffer insertion, and useful skew optimization were instrumental in decreasing clock skew as well as increasing setup and hold timing margins. As a result, the reconstructed clock distribution networks exhibit a more consistent timing behavior even when multiple clock domains are involved. Routing improvements revealed another layer of performance enhancements mainly through the shortening of wire delays and the improvement of signal integrity. By employing congestion-aware routing, the occurrences of routing bottlenecks were drastically decreased leading to a more efficient use of the

available routing resources. Besides that, critical net routing, which gives highest priority to timing-sensitive paths, has become a very efficient way of improving Worst Negative Slack as well as overall timing convergence. On the whole, these various optimization techniques, when considered together, make it quite clear that a single comprehensive timing closure plan is necessary which can effectively address both logical as well as physical design issues.

6. Conclusion and Future Scope

6.1. Conclusion

The issue of timing closure is turning into one of the biggest struggles in both ASIC and FPGA design as a result of continuous shrinking of features on a chip, more and more complex circuits, higher performance demands, and also the desire for less power consumption. With even smaller technology nodes come new problems such as variations in semiconductor manufacturing, different levels of voltage and temperature, longer delays due to wires, crowded routing, clock skews, problems with the quality of the signal, and the need to handle multiple clock domains, etc. All these factors make it even more difficult to ensure that all timing requirements are met while not compromising performance, power, and area (PPA) aspects. Hence, timing closure that was once only a final checking task has now transformed into an extensive optimization process covering the whole design flow. In this paper, we have looked at a wide range of timing closure methods available in the literature, such as Static Timing Analysis (STA), techniques of physical design optimization, methods for timing improvements specific to FPGA, and people are also using machine learning for timing closure nowadays. Our study shows that to be able to achieve timing closure you will probably need a combination of logic optimization, better physical implementation, correct handling of constraints, and thorough verification. Although isolated methods can bring about certain improvements, their effectiveness is raised to a quite appreciable level once they are merged into a well-organized and iterative scheme.

6.2. Future Scope

Timing closure will likely rely heavily on new semiconductor technologies and smarter design automation. EDA tools equipped with Artificial Intelligence are being developed for various purposes including predicting timing violations, estimating congestion, and making optimization decisions. Machine learning can be a major factor in reducing the number of implementation iterations and also help with timing convergence by way of intelligent decision-making and predictive analysis. Besides,3D integrated circuits (3D ICs), chiplet-based architectures, and cutting-edge packaging are examples of new developments that bring timing challenges that can hardly be handled by the conventional single-die design methods. Timing analysis methods responding to such issues must deal with complex interconnect structures and heterogeneous integration environments.

References

[1] Saurabh, S., Shah, H., & Singh, S. (2019). Timing closure problem: Review of challenges at advanced process nodes and solutions. *IETE Technical Review*.

[2] Golshan, K. (2020). *The art of timing closure*. Springer International Publishing.

[3] Allenki, S. S., & Lee, N. (2022). Performance Tuning Cloud-Hosted Databases: Resource Allocation & Query Optimization. International Journal of AI, BigData, Computational and Management Studies, 3(4), 152-163. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I4P116

[4] Zuchowski, P. S., Reynolds, C. B., Grupp, R. J., Davis, S. G., Cremen, B., & Troxel, B. (2002, November). A hybrid ASIC and FPGA architecture. In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design* (pp. 187-194).

[5] Suryadevara, S. S. K. (2021). Generative AI–Powered Authoring Assistant for Enterprise Content Management. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 2(2), 103-113. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I2P111

[6] Muppaneni, K. (2021). HTTP/3 & REST Latency Improvement. International Journal of Emerging Research in Engineering and Technology, 2(1), 122-132. https://doi.org/10.63282/3050-922X.IJERET-V2I1P113

[7] Simpson, P. (2010). *FPGA design*. Berlin/Heidelberg, Germany: Springer.

[8] Shiramalla, R. (2022). Design of a Unified API Interface Using Workato for Cross-Platform Data Orchestration Between Salesforce and Oracle ERP. International Journal of Emerging Trends in Computer Science and Information Technology, 3(1), 157-168. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I1P118

[9] Kumar Doodala, A. N. (2022). Strategic Migration for JBoss to IIBM WAS: A Framework for Enterprise-Grade Modernization. International Journal of Emerging Research in Engineering and Technology, 3(2), 161-170. https://doi.org/10.63282/3050-922X.IJERET-V3I2P117

[10] Kilts, S. (2007). *Advanced FPGA design: architecture, implementation, and optimization*. John Wiley & Sons.

[11] Muppaneni, R. K. (2021). Securing the Enterprise: How Dynamics 365 Meets Global Compliance Standards. International Journal of Emerging Research in Engineering and Technology, 2(1), 133-143. https://doi.org/10.63282/3050-922X.IJERET-V2I1P114

[12] Kuon, I., & Rose, J. (2006, February). Measuring the gap between FPGAs and ASICs. In *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays* (pp. 21-30).

[13] Gaddam, R. R. (2022). Cost-Aware Autoscaling for Batch vs. Online Inference. International Journal of Emerging Trends in Computer Science and Information Technology, 3(4), 134-143. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P113
[14] Vppalapati, M. (2022). The Storage Stack Nobody Draws: Cabling, Panels, and the Illusion of Isolation. International Journal of Emerging Research in Engineering and Technology, 3(2), 211-220. https://doi.org/10.63282/3050-922X.IJERET-V3I2P121
[15] Kuon, I., & Rose, J. (2010). *Quantifying and exploring the gap between FPGAs and ASICs*. Springer Science & Business Media.
[16] Katangoori, Sivadeep, and Sushil Deore. "Lakehouse Architecture and the Semantic Revolution: Bridging Analytics and Governance With AI." *The Distributed Learning and Broad Applications in Scientific Research* 8 (2022): 275-300.
[17] Kumar Doodala, A. N., & Thatraju, S. (2022). NLP-Driven Benefits Interpretation Engine for Personalized Member Communication. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(1), 173-183. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P118
[18] Rodríguez-Andina, J. J., Valdes-Pena, M. D., & Moure, M. J. (2015). Advanced features and industrial applications of FPGAs—A review. *IEEE transactions on industrial informatics*, *11*(4), 853-864.
[19] Shiramalla, R. (2022). Predictive Record Assignment Engine in Salesforce using LWC and Einstein AI. International Journal of AI, BigData, Computational and Management Studies, 3(3), 147-159. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I3P117
[20] Srigadde, B. R., & Devaraju, J. M. (2022). The Wrath of Limitations: Lightning Fields and Their Constraints. International Journal of Emerging Research in Engineering and Technology, 3(2), 201-210. https://doi.org/10.63282/3050-922X.IJERET-V3I2P120
[21] Machado, R., Cabral, J., & Alves, F. S. (2019). Recent developments and challenges in FPGA-based time-to-digital converters. *IEEE Transactions on Instrumentation and Measurement*, *68*(11), 4205-4221.
[22] Muppaneni, R. K. (2021). How Enterprises are Achieving 360° Customer Views with Dynamics 365. International Journal of AI, BigData, Computational and Management Studies, 2(2), 129-138. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I2P114
[23] Kumar Doodala, A. N. (2022). Strategic Migration for JBoss to IIBM WAS: A Framework for Enterprise-Grade Modernization. International Journal of Emerging Research in Engineering and Technology, 3(2), 161-170. https://doi.org/10.63282/3050-922X.IJERET-V3I2P117
[24] Hoozemans, J., Peltenburg, J., Nonnemacher, F., Hadnagy, A., Al-Ars, Z., & Hofstee, H. P. (2021). Fpga acceleration for big data analytics: Challenges and opportunities. *IEEE Circuits and Systems Magazine*, *21*(2), 30-47.
[25] Allenki, S. S. (2022). Securing Databases in the Cloud with RBAC and Encryption Best Practices. International Journal of Emerging Research in Engineering and Technology, 3(3), 173-182. https://doi.org/10.63282/3050-922X.IJERET-V3I3P117
[26] Parakala, A. (2022). Integrating Salesforce and UiPath: Cross-System Intelligent Automation. International Journal of Emerging Trends in Computer Science and Information Technology, 3(4), 88-99. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P109
[27] Bobda, C., Mbongue, J. M., Chow, P., Ewais, M., Tarafdar, N., Vega, J. C., ... & Tessier, R. (2022). The future of FPGA acceleration in datacenters and the cloud. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, *15*(3), 1-42.
[28] Vppalapati, M., & Talasila, P. K. (2022). Correlated Independence: Why Redundant Storage Systems Share the Same Fate. International Journal of Emerging Trends in Computer Science and Information Technology, 3(1), 169-179. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I1P119
[29] Kumar Doodala, A. N., & Thatraju, S. (2022). NLP-Driven Benefits Interpretation Engine for Personalized Member Communication. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(1), 173-183. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P118
[30] Suryadevara, S. S. K. (2021). AI-Driven Multi-Cloud Orchestration System for Enterprise Digital Experience Delivery. American International Journal of Computer Science and Technology, 3(1), 21-34. https://doi.org/10.63282/3117-5481/AIJCST-V3I1P103
[31] Prabakaran, B. S., Mrazek, V., Vasicek, Z., Sekanina, L., & Shafique, M. (2020, July). ApproxFPGAs: Embracing ASIC-based approximate arithmetic components for FPGA-based systems. In *2020 57th ACM/IEEE Design Automation Conference (DAC)* (pp. 1-6). IEEE.
[32] Srigadde, B. R. (2021). Future Methods, Most Underrated Apex Features. American International Journal of Computer Science and Technology, 3(1), 35-45. https://doi.org/10.63282/3117-5481/AIJCST-V3I1P104
[33] Gaddam, R. R. (2022). Advanced Data & Model Drift Detection at Scale. International Journal of AI, BigData, Computational and Management Studies, 3(2), 124-136. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I2P113
[34] Monmasson, E., & Cirstea, M. N. (2007). FPGA design methodology for industrial control systems—A review. *IEEE transactions on industrial electronics*, *54*(4), 1824-1842.
[35] Katangoori, Sivadeep, and Sushil Deore. "Predictive Drift Detection and Adaptive Reconciliation in Multi-Cloud Data Environments." *The Distributed Learning and Broad Applications in Scientific Research* 8 (2022): 247-274.
[36] Davis, R. I., & Cucu-Grosjean, L. (2019). A survey of probabilistic timing analysis techniques for real-time systems. *Leibniz Transactions on Embedded Systems*, *6*(1), 03-1.
[37] Muppaneni, K. (2021). Cross-Browser Debugging Strategies. American International Journal of Computer Science and Technology, 3(5), 25-36. https://doi.org/10.63282/3117-5481/AIJCST-V3I5P103
[38] Parakala, A. (2021). Building Analytics-Driven Bots: RPA Meets Business Intelligence. International Journal of Emerging Research in Engineering and Technology, 2(1), 77-87. https://doi.org/10.63282/3050-922X.IJERET-V2I1P109
[39] Amos, D., Lesea, A., & Richter, R. (2011). *FPGA-based prototyping methodology manual: Best practices in design-for-prototyping*. Happy About.