International Journal on Science and Technology (IJSAT)

E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Reconfigurable and Delay Reduced Level Shifter Circuit for Multi-Process FPGAS

Er. S. Senthazhai[1], Er. S. Madhan[2], A. Nedha[3], P. Nivetha[4], S. Sulthana begam[5]

[1]Associate Prof, Department of ECE, Krishnasamy College of Engineering & Technology
[2]Assistant Prof, Department of ECE, Krishnasamy College of Engineering & Technology
[3, 4, 5] Final Year, Department of ECE, Krishnasamy College of Engineering & Technology

Abstract

Traditional level shifter circuits often suffer from significant propagation delays and power overhead, which limits their performance in high-speed and low-power applications. In response to these challenges, this work introduces a reconfigurable and delay-reduced level shifter circuit specifically designed to improve interoperability between multiple voltage domains in FPGA-based systems. The proposed circuit leverages adaptive threshold control and optimized transistor configurations to minimize delay without compromising signal integrity, ensuring efficient voltage level translation across varying voltage domains. These advancements make the circuit particularly well-suited for dynamic voltage scaling (DVS) and heterogeneous computing environments, where the system operates under varying power and performance conditions. The reconfigurability of the circuit allows for its adaptation to different system requirements, providing a flexible solution for multi-process FPGAs. Additionally, by reducing both delay and power consumption, the proposed level shifter enhances the efficiency of power-aware FPGA designs, extending their applicability in real-time and embedded systems that demand high performance and low energy usage. This innovation promises to significantly improve the viability of FPGA systems in a wide range of modern, energy-conscious applications.

Keywords: FPGA System, Multiprocess Compatibility, Reconfigurable circuit

1. Introduction

1.1 Overview of multiprocessor FPGA

Voltage level shifter is a device which converts one voltage level to another. Voltage level shifters are used to interface various circuit blocks operating at different supply voltages. At the boundaries of different voltage islands on the system-on-chip (SoC) voltage level shifter is used. In modern multi-process FPGA architectures, efficient power management and seamless communication between voltage domains are crucial for optimizing performance and reducing energy consumption. Traditional level shifter circuits often introduce significant propagation delay and power overhead, limiting their effectiveness in high-speed and low-power applications. To address these challenges, this work presents a reconfigurable and delay-reduced level shifter circuit designed to enhance interoperability between multiple voltage domains in FPGA-based systems. By leveraging adaptive threshold control and optimized transistor configurations, the proposed circuit minimizes delay while maintaining signal integrity, making it suitable for dynamic voltage scaling and heterogeneous computing environments. This innovation not only improves the efficiency of power-aware FPGA designs but also extends their applicability in real-time and embedded systems.

1.2 Need for Voltage Level Shifter Circuits in Digital Design

In modern digital design, voltage level shifter circuits play a crucial role in ensuring seamless communication between components operating at different voltage levels. As semiconductor technology advances, integrated circuits (ICs) are designed with varying supply voltages to optimize power consumption, performance, and thermal efficiency. However, direct interfacing between circuits with different voltage domains can lead to signal integrity issues, increased power dissipation, and even damage to low-voltage components.

Level shifters act as intermediaries that safely translate signal voltages between different power domains, enabling compatibility between legacy and modern circuits. They are particularly essential in applications such as multi-voltage FPGAs, system-on-chip (SoC) designs, and energy-efficient embedded systems, where low-power and high-performance components must coexist. Additionally, with the increasing adoption of dynamic voltage scaling (DVS) and power-aware architectures, level shifters contribute to overall system efficiency by allowing flexible voltage adaptation without compromising signal reliability. Thus, these circuits are fundamental to modern digital systems, ensuring robust and efficient data exchange across diverse voltage environments.

1.3 Applications of Level Shifters in Digital Design

Voltage level shifters are essential components in digital design, enabling smooth communication between circuits operating at different voltage levels. Their applications span various domains, including:

Multi-Voltage System-on-Chip (SoC) Designs

Modern SoCs integrate multiple functional blocks with different supply voltages for power efficiency. Level shifters allow these blocks to interact without voltage mismatches.

FPGA-Based Systems

Field-Programmable Gate Arrays (FPGAs) often interface with peripherals operating at different voltage levels. Level shifters ensure proper signal translation between FPGA I/Os and external components.

Interfacing Legacy and Modern Devices

Older devices may operate at higher voltages (e.g., 5V), while newer ones use lower voltages (e.g., 3.3V, 1.8V). Level shifters enable seamless connectivity between them.

Low-Power and Battery-Operated Devices

Mobile devices, IoT sensors, and wearables use different voltage domains to optimize power consumption. Level shifters help maintain compatibility while minimizing energy usage.

Dynamic Voltage Scaling (DVS) Systems

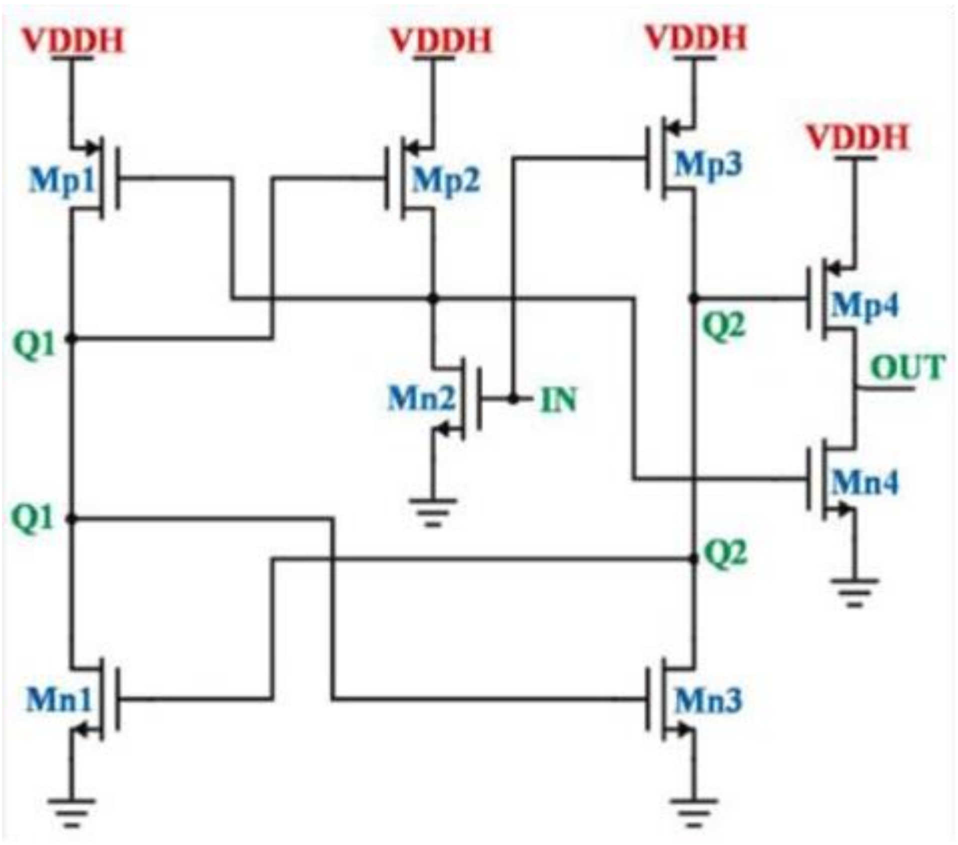
Power-efficient processors dynamically adjust voltage levels based on workload demands. Level

shifters facilitate smooth transitions between these voltage states.
Automotive Electronics

2. Proposed system

2.1 System Design

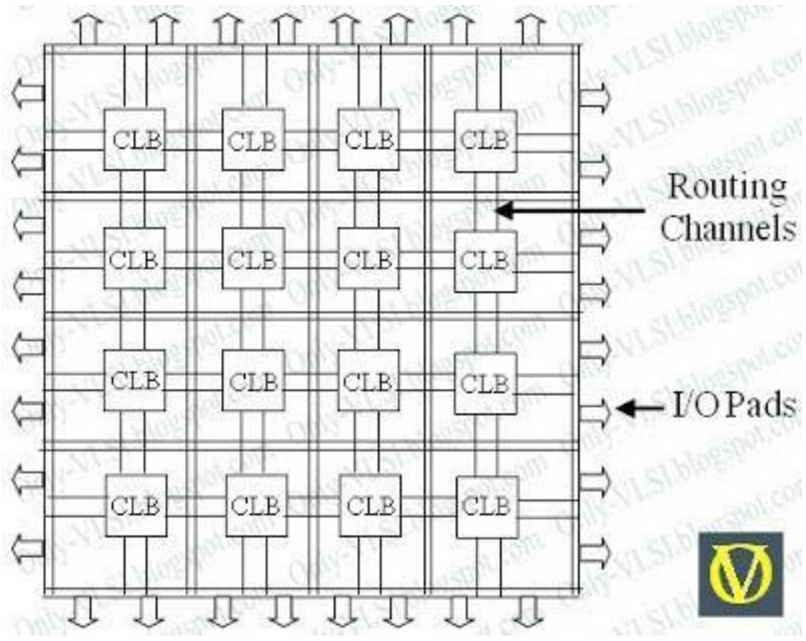
In the proposed approach, a reconfigurable level shifter circuit is introduced, featuring an **N-tunable circuit** that allows for dynamic adjustment of voltage levels. This reconfigurability enables the level shifter to handle various voltage requirements for different operations in FPGA systems. The system utilizes **digital regulator circuits** to generate multiple shifted voltage levels, supporting a wide range of applications across the FPGA. The regulator circuits can precisely control the output voltage, ensuring that each level shift is appropriate for the specific task at hand. The **switching between voltage levels** is achieved through **cross inverters**, which are strategically configured to facilitate the rapid transition between different voltage states. These cross inverters play a crucial role in ensuring the reliability and speed of the voltage level switching, while also minimizing the delay associated with the level conversion process. The reconfigurability of the circuit, coupled with the flexibility provided by the N- tunable design, enables the level shifter to adapt efficiently to varying operational conditions, making it highly suitable for multi-purpose FPGA applications that require both high performance and low power consumption. This innovative approach ensures that the system can handle a variety of voltage domains while maintaining optimal performance across a range of processing tasks.



2.2 FPGA Architecture

Field-Programmable Gate Array (FPGA) is a semiconductor device containing programmable logic components called "logic blocks", and programmable interconnects. Logic blocks can be programmed to perform the function of basic logic gates such as AND, and XOR, or more complex combinational functions such as decoders or mathematical functions. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

FPGA consists of large number of "configurable logic blocks" (CLBs) and routing channels. Multiple I/O pads may fit into the height of one row or the width of one column in the array. In general all the routing channels have the same width. The block diagram of FPGA architecture is shown below.



2.3 Modules

Module 1: Design of Dynamic Clocked Regulator

Regulator is the basic building block for the ADC applications. This application module developed consist of a single-tail this comes under the clocked regenerative Regulator that can make fast decisions with the positive feedback. The main parameters considered here are the clock inputs and with this the double tail Regulator is developed.

Module 2: Design of Double-Reference Regulator

This module is implemented to show the difference between the dynamic clocked regenerative Regulator and the double-tail Regulator. The conventional double tail Regulator is developed and its delay analysis is performed. The analysis results help to find out the disadvantages of the current double- tail Regulator design.

Module 3: Design Digital Low drop out Regulator

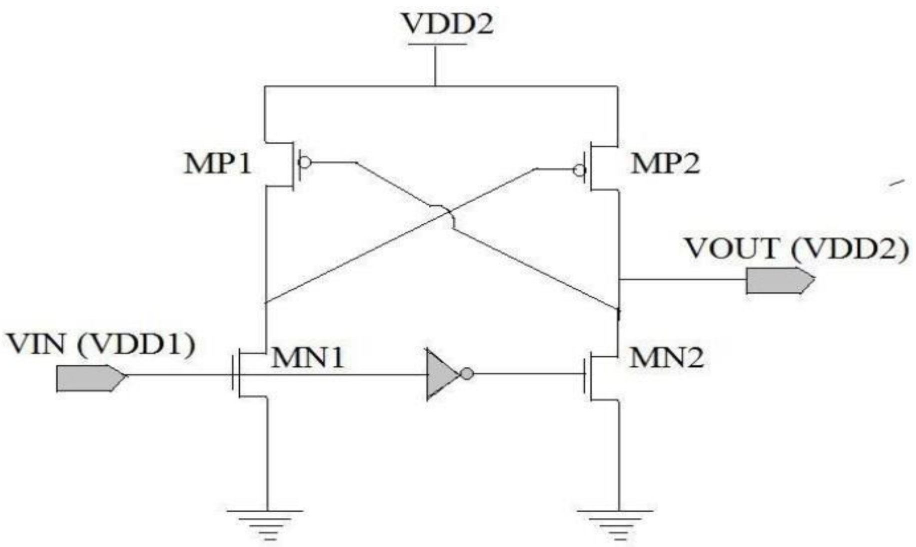
This module is designed with the tunable threshold. Using the auto-tunable threshold a Regulator is capable of generating digital signal from analog input which can achieve good tolerance and efficiency. This module helps to avoid the kick-back noise and Mis-match in the existing double tail Regulator design using auto-tunable threshold.

Module 4: Design and Analysis of the Integration Module

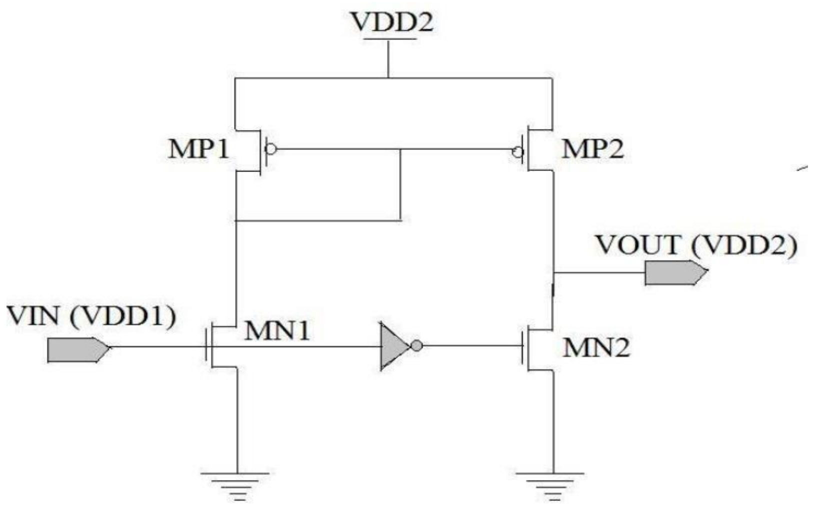
We integrate all the sub-modules and their analysis and performance are implemented with the Model Sim software simulation results.

Methods

1. Cross coupled level shifters



2. Current Mirror level shifter



3. Software Tools

Xilinx software:

Xilinx designs, develops and markets programmable logic products, including integrated cir- cuits (ICs), software design tools, predefined system functions delivered as intellectual property (IP) cores, design services, customer training, field engineering and technical support. Xilinx sells both FPGAs and CPLDs for electronic equipment manufacturers in end markets such as communications, industrial, consumer, automotive and data processing. Xilinx introduced new high capacity 3D FPGAs, including Virtex-7 2000T and Virtex-7 H580T products, these devices began to outpace the capacity of Xilinx's design software, which led the company to completely redesign its tool set. The result was the introduction of the Viv ado Design Suite, which reduces the time needed for programmable logic and I/O design, and speeds systems integration and implementation compared to the previous software.

Kintex:

The Kintex-7 family is the first Xilinx mid-range FPGA family that the company claims delivers Virtex- 6 family performance at less than half the price while consuming 50 percent less power. The Kintex family includes high-performance 12.5 Gbit/s or lower-cost optimized 6.5 Gbit/s serial connectivity, memory, and logic performance required for applications such as high volume 10G optical wired communication equipment, and provides a balance of signal processing performance, power consumption and cost to support the deployment of Long Term Evolution (LTE) wireless networks.

LTSPICE:

LTSpice is a powerful, widely-used circuit simulation tool developed by Analog Devices (formerly by Linear Technology). It is primarily used for simulating analog circuits but is also capable of digital and mixed-signal simulations. LTSpice is favored by engineers for its speed, accuracy, and its ability to simulate large-scale circuits with minimal computational overhead.

LTspice: An Overview of the Simulation Tool for Electronic Circuits

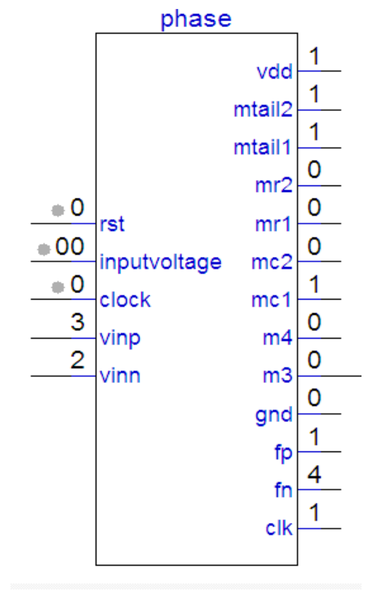
LTspice is a high-performance, widely used circuit simulation tool developed by Linear Technology (now part of Analog Devices). It is primarily used for simulating analog circuits, including both linear and non-linear components, and is especially popular among electrical engineers and circuit designers due to its speed, accuracy, and extensive feature set. LTspice allows for the analysis of complex circuits and systems by providing a powerful platform to simulate the behavior of components before actual hardware implementation, thus helping in optimizing designs and identifying potential issues early in the design process.

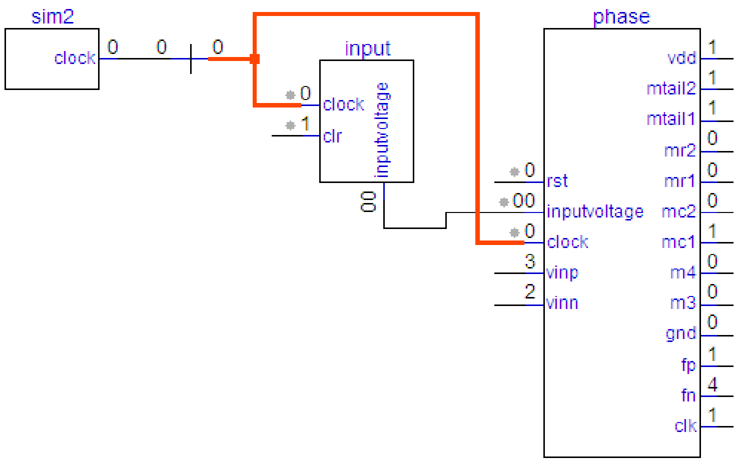
Key Features of LTspice:

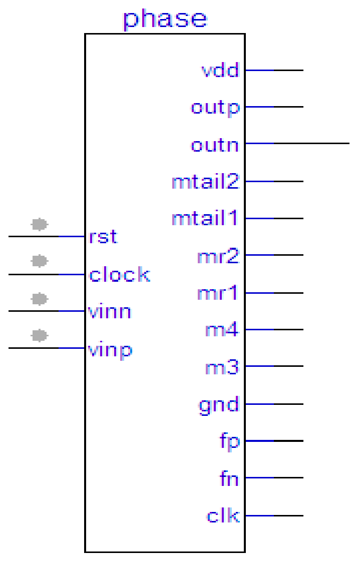
Simulation Capabilities: LTspice offers comprehensive support for various types of simulations. These include Transient Analysis, AC Analyzing, DC Operating Point (Bias Point),Noise Analysis, Component Library and Models, Fast Simulation Speed, User Interface

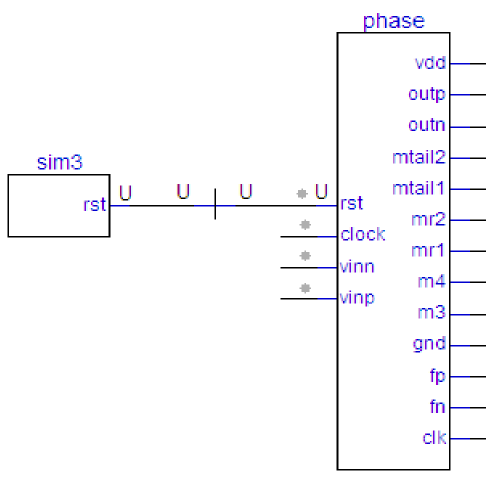
4. SIMULATION RESULTS

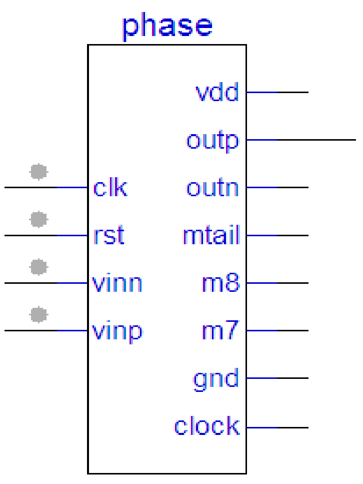
4.1 Data Flow Results

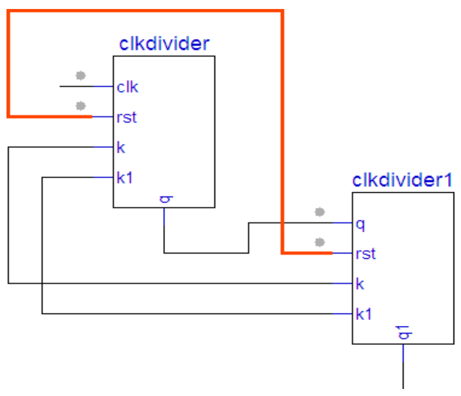


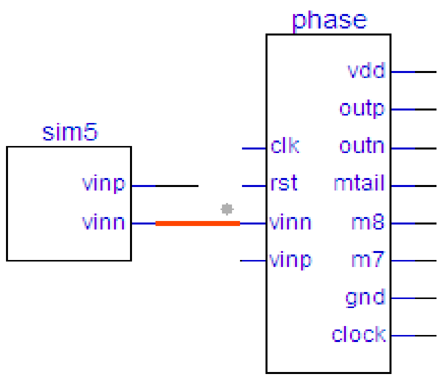












4.2 Software coding testing

Timing Simulation

Timing simulation is a crucial step in the verification process of digital circuits, especially as designs become more complex and operate at higher speeds. It focuses on assessing whether the design meets specific timing requirements and constraints, such as setup time, hold time, and clock frequency. This simulation evaluates the behavior of the design while taking into account the propagation delays of individual gates and interconnects.

Key Components and Processes

Propagation Delays: Timing simulation considers the delays incurred by signals as they propagate through various gates and interconnects within the design. Each gate has its delay characteristics, which are typically specified by the technology library used in the design.
Setup and Hold Time: Timing simulation verifies that signals meet the setup and hold time requirements at each flip-flop or latch in the design. Setup time refers to the minimum time before the clock edge that the input signal must be stable for reliable capture, while hold time specifies the minimum time after the clock edge that the input signal must remain stable.
lock Frequency: Another aspect of timing simulation involves verifying that the design operates within the specified clock frequency. This ensures that the design meets performance requirements and can operate reliably within the desired speed range.
Clock Domain Crossing (CDC) Analysis: Timing simulation may also involve checking for issues related to signals crossing between different clock domains within the design. CDC analysis helps identify potential problems such as metastability and data loss due to improper synchronization between clock domains.

Verification Challenges

Complexity: As designs become more complex and integrate a larger number of components, timing simulation becomes increasingly challenging due to the sheer number of interactions and dependencies to consider.
Resource Intensive: Timing simulation can be resource-intensive in terms of computational resources and simulation time, particularly for large-scale designs with high gate counts.
Corner Cases: Ensuring coverage of all possible corner cases and timing scenarios can be difficult, requiring comprehensive testbench setups and simulation scenarios.
Accuracy: Achieving accurate timing simulation results requires precise modeling of gate delays, interconnect delays, and other timing-related parameters, which may vary based on technology libraries and manufacturing processes.

Tools and Techniques

Timing Analysis Tools: Various EDA (Electronic Design Automation) tools provide capabilities for timing analysis and simulation, including tools for static timing analysis (STA) and dynamic timing analysis.

Constraint driven Design: Using timing constraints in the design process helps guide synthesis and optimization efforts towards meeting specific timing requirements. Constraints specify timing parameters such as clock frequency, setup time, hold time, etc.
Advanced Simulation Techniques: Techniques such as advanced event-driven simulation algorithms and parallel simulation can help improve the efficiency and scalability of timing simulations for large designs.
Clock Domain Crossing (CDC) Tools: Specialized tools for CDC analysis help identify and mitigate potential timing issues related to signals crossing between different clock domains.

Model Checking

Model checking is a formal verification method used to analyze the behavior of a design exhaustively against specified properties expressed in temporal logic. The primary goal is to verify whether a system (or model) satisfies certain behavioral specifications under all possible inputs and conditions.

Key Components and Processes

Temporal Logic: Properties to be verified are typically expressed using temporal logic, such as Linear Temporal Logic (LTL) or Computation Tree Logic (CTL). These logics allow the specification of temporal properties, such as "eventually," "always," "until," etc., enabling precise descriptions of desired system behaviors.
State Space Exploration: Model checking tools systematically explore all possible states and transitions of the design's state space to verify whether the specified properties hold true. This exhaustive analysis ensures that no corner case or potential issue is overlooked.
Counterexample Generation: If a property is violated, model checking tools can provide counterexamples, i.e., specific sequences of states and transitions that lead to the violation. These counterexamples help designers understand why the property fails and facilitate debugging and refinement of the design.
Formal Proof: In cases where properties are verified successfully, model checking provides a formal proof of correctness, demonstrating that the design satisfies the specified requirements under all possible scenarios.

Advantages and Challenges

Exhaustive Coverage: Model checking provides exhaustive coverage of the design's behavior, ensuring that all possible scenarios are considered.

Automatic Analysis: Model checking tools automate the verification process, reducing the need for manual effort and potential human errors.
Formal Guarantees: Successful verification provides formal guarantees of correctness, enhancing confidence in the design's reliability.

Challenges

State Space Explosion: For complex designs, the state space can grow exponentially, leading to scalability issues and longer verification times.

Property Specification: Formulating precise and relevant properties in temporal logic can be challenging, requiring a deep understanding of both the design and the verification objectives.
Tool Limitations: The effectiveness of model checking heavily relies on the capabilities and limitations of the underlying verification tools and algorithms.

Equivalence Checking

Equivalence checking is a formal verification technique used to compare two designs or versions of a design to ensure functional equivalence. It verifies that modifications made during synthesis, optimization, or physical design stages do not alter the design's functionality compared to a golden reference model.

Key Components and Processes:

Golden Reference Model: Equivalence checking typically involves comparing a modified design (e.g., post-synthesis or post-layout) with a golden reference model, which represents the original, unmodified design.
Logical Equivalence: Equivalence checking verifies whether the logic behavior of the two designs is identical, regardless of differences in implementation details such as gate-level structures, netlists, or layout.
Verification Flow: The equivalence checking flow involves several steps, including logic synthesis, formal verification, and comparison of logic representations to identify any discrepancies between the designs.
Optimization Validation: Equivalence checking ensures that optimization techniques applied during synthesis or physical design stages, such as logic restructuring or technology mapping, preserve the design's functional behavior.

Advantages and Challenges:

Advantages:

Design Confidence: Equivalence checking provides confidence that design modifications or optimizations do not introduce unintended changes in functionality.

Automation: Equivalence checking tools automate the comparison process, reducing the need for manual effort and extensive testing.
Scalability: Equivalence checking techniques are scalable and can handle designs of varying complexity, including large-scale integrated circuits.
Challenges:

Complexity: Analyzing the equivalence of large designs with millions of gates can be computationally intensive and may require efficient algorithms and hardware resources.
False Positives/Negatives: Equivalence checking tools may produce false positives or negatives due to limitations in abstraction, modeling, or corner case handling.
Corner Cases: Ensuring coverage of all relevant corner cases and scenarios during equivalence checking can be challenging, requiring comprehensive testing strategies.

Formal verification techniques, including model checking and equivalence checking, are essential components of modern digital design verification methodologies. Model checking enables exhaustive analysis of design behavior against specified properties, providing formal guarantees of correctness. Equivalence checking ensures that design modifications maintain functional equivalence with respect to a golden reference model, enhancing confidence in the design's integrity and reliability. Despite challenges such as scalability and complexity, these formal verification techniques play critical roles in ensuring the correctness and quality of digital designs in various application domains.

Static Timing Analysis (STA) is a crucial step in the design and verification process of digital integrated circuits. It ensures that the designed digital circuit operates correctly within specified timing constraints. Here's an exploration of various aspects of STA:

Timing Paths: STA analyzes timing paths within the digital circuit. A timing path is a sequence of logical elements (like gates, flip-flops, etc.) and interconnecting wires that a signal traverses from its source to its destination. These paths determine the maximum delay that a signal experiences from one point to another.

Clock Constraints: STA considers the timing relationships between the clock signal and other signals in the circuit. Clock constraints define the timing requirements for clock signals, such as clock period, setup time, hold time, and clock skew. These constraints ensure proper synchronization of signals in synchronous digital circuits.

Signal Arrival Times: STA calculates the arrival times of signals at various points in the circuit. It considers factors such as propagation delays through logic gates, interconnect delays, and clock uncertainties. By analyzing signal arrival times, STA determines whether signals meet setup and hold time requirements at their respective destinations.

Critical Paths: STA identifies critical paths within the circuit. A critical path is a path with the longest delay in the circuit, and it determines the maximum achievable operating frequency. Analyzing critical paths helps designers optimize circuit performance by focusing on improving the timing characteristics of these paths.

Violation Detection: STA detects timing violations that occur when signals fail to meet specified timing requirements. Violations may include setup time violations, hold time violations, clock skew violations, and max delay violations. Identifying and resolving these violations are essential to ensure reliable and robust operation of the designed circuit.

Margin Analysis: STA provides insights into timing margins, which represent the difference between the actual timing behavior of the circuit and the specified timing requirements. Positive timing margins indicate that the design meets timing requirements with some degree of slack, while negative margins indicate potential timing violations.

Path Optimization: Based on STA results, designers perform path optimization techniques to improve timing characteristics, such as restructuring logic, resizing gates, adding pipeline stages, or adjusting clock distribution strategies. These optimizations aim to reduce critical path delays and enhance overall circuit performance.

Static vs. Dynamic Timing Analysis: STA primarily focuses on analyzing timing behavior under static conditions, considering factors like gate delays, interconnect delays, and clock timing. In contrast, dynamic timing analysis considers timing variations due to environmental factors such as temperature, voltage fluctuations, and process variations.

Design Rule Checking (DRC):

Minimum Feature Size: This rule ensures that the features (transistors, interconnects, etc.) in the layout are not smaller than the minimum size allowed by the manufacturing process. Violations can lead to manufacturing defects.
Metal Spacing: It checks for proper spacing between metal layers to prevent shorts or other electrical issues.
Overlap Violations: This involves checking for any unintended overlaps between different layers, which can cause shorts or other connectivity issues.
Layout vs. Schematic (LVS) Check:

Consistency Verification: LVS ensures that the physical layout matches the intended schematic design. It verifies that all connections in the layout match those specified in the schematic, detecting missing connections, shorts, opens, and other discrepancies.
Hierarchy Handling: In large designs, LVS tools handle hierarchical designs efficiently, comparing the layout netlist with the hierarchical schematic netlist.

Electromigration (EM) and IR Drop Analysis:

Electromigration (EM): This analysis predicts the movement of atoms in conductive materials due to the high density of electrons. Excessive current densities can cause atoms to migrate, leading to voids or hillocks that affect the reliability of the design.

IR Drop Analysis: It assesses voltage drops across the power distribution network caused by resistive losses. Excessive voltage drops can result in timing violations or even failures in functional operation due to inadequate voltage supply at certain points in the chip.
Additional aspects to explore in physical design verification include:

Timing Verification: Checking timing constraints to ensure that the design meets its performance targets in terms of clock frequency, setup and hold times, and other timing parameters.

Power Integrity Analysis: Evaluating the integrity of power distribution networks to ensure that the design can adequately supply power to all components without excessive voltage drops or noise.
Signal Integrity Analysis: Assessing the quality of signals on interconnects to prevent issues such as crosstalk, ringing, or signal distortion.
Design for Manufacturability (DFM): Implementing techniques during design to enhance manufacturability, such as optimizing layout for yield, reducing variability, and improving reliability.

Physical Verification Automation: Utilizing scripting and automation tools to streamline the verification process and minimize manual effort. These aspects collectively ensure that the physical design meets the required specifications, reliability standards, and manufacturing constraints before proceeding to fabrication.

5. Acknowledgement

We would like to express my special thanks to our guide **Er.S.Madhan M.E,** Assistant Professor, Department of ECE, for her time and efforts she provided throughout the year. Your useful advice and suggestions were really helpful to me during the project's completion. In this aspect, We are eternally grateful to you.

6. Conclusion

The proposed reconfigurable and delay reduced level shifter arout for multi process FPGAs provides a flexible, efficient, and high performance solution to address the challenges of interfacing different voltage domains in modem FPGA architectures. By antegrating dynamic reconfiguration, delay minamization techniques, and power optimization strategies, the system enables seamless and rellable communication between logic blocks operating at varying voltage levels, without compromising performance.

7. References

1. V. D. V and P. K. N, "Design and Optimization of a Low Power and High-Speed Voltage Level Shifter Circuit using 45nm and 180nm MOSFETs with Variable Voltage Source," 2024 IEEE North Karnataka Subsection Flagship International Conference (NKCon), Bagalkote, India, 2024, pp. 1-7, doi: 10.1109/NKCon62728.2024.10774706

2. Li, Y., Zhao, J., Liu, Y., & Wang, G. (2022). A comprehensive study on the design methodology of level shifter circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *70*(1), 302-314.

3. Saleh, R. System-on-chip: Reuse and integration. Proc. IEEE 2020, 94, 1050-1069.

4. Lanuzza, M.; Perri, S. Fast and wide range voltage conversion in multisupply voltage designs. IEEE Trans. Very Large Scale Integr. Syst. 2020, 23, 388-391.

5. Zhao, W. Alvarez, A.B.; Ha, Y. A 65-rm 25.1-ns 30.7-41 robust subthreshold level shifter with wide conversion range. IEEE Trans. Circuits Syst. II Express Briefs 2019 62, 671-675.

6. Lanuzza, M.; Crupi, F.; Rao, S; De Rose, R.; Strangio, S.; lannaccone, G. An ultralow-voltage energy- efficient level shifter IEEE Trans Circuits Syst. II Express Briefs Briel 2020, 64, 61-65

7. Mocan, B.; Osan, C.; Fulen, M.; Chia, LA; Timoftei, S.; Sarb, A. An integrated Model for Solving Cell Formation Problem and Robot Scheduling using Timed Petri Nets. In Proceedings of the 2016 In-ternational Conference on Production Research Africa, Europe and the Middle East and 4th International Conference Quality and Innovation in Engineering and Management, Cluj-Napoca, Romanis, 25-30 July 2021; pp. 91-96

8. Fassio, L., Settino, F., Lin, L.; De Rose, R; Lanuzan, M., Crupi, F., Alioto, M. A robust, high speed and energy efficient ultralow voltage level shifter. IEEE Trans. Circuits Syat. II Express Briefs 2021, 68, 1393-1397.

9. Låte, E.; Ytterdal, T.; Aumet, S. An energy efficient level shifter capable of logic conversion from sub-15 mV to 1.2 V.1 IEEE Trans. Circuits Syst. II Express Briefs 2020, 67, 2687-2691. CrossRef 8.

Cao, L.; Bale, SJ.; Trefzer, M.A. Multi-objective digital design optimization via improved drive granularity standard cells. IEEE Trans. Circuits Syst. I Regul. Pap. 2021, 68, 4660-4671