Challenges in Verification of Semiconductor designs for IoT Devices

Niranjana Gurushankar

Hardware Verification Engineer at Cisco Systems

Abstract

The Internet of Things (IoT) demands semiconductor devices that are smaller, power-efficient, and increasingly complex, posing significant verification challenges. This paper explores these challenges, focusing on design complexity, ultra-low power verification, and robust security. This paper analyzes contemporary strategies like formal verification, emulation, and the rising role of machine learning in automation. Furthermore, the paper also highlights the importance of shift-left approaches and the use of virtual prototypes. Finally, we identify open research problems and future trends, including the need for improved tools and standardized verification frameworks for the evolving landscape of IoT devices.

Keywords: Semiconductor, IoT, Design Verification, Low-Power Verification, Emulation, IoT Design Challenges, Security.

Introduction

The rapid growth of the IoT presents unique verification challenges. IoT devices require complex functionality, integrating diverse components like sensors, processors, and communication interfaces, all while operating under stringent low power constraints[1]. This necessitates advanced verification techniques to ensure correct functionality and optimized power consumption across different operating modes and usage scenarios. Furthermore, security is paramount, as connected devices are vulnerable to various threats. Robust security measures must be implemented and rigorously verified to protect sensitive data and prevent malicious attacks[2]. This complexity drives the need for innovative verification methodologies to ensure the reliability and security of IoT devices.This paper explores the unique verification challenges posed by the growing complexity, low power requirements, and security concerns of IoT devices. We analyze contemporary verification strategies, including formal verification, emulation, hardware-assisted techniques, and the role of machine learning in automating verification tasks. We also discuss shift-left approaches and the use of virtual prototypes for faster design cycles. Finally, we delve into the complexities of verifying low power designs and ensuring robust security, identifying open research problems and future trends in this dynamic field

Verification Challenges in IoT Design

Complexity

IoT devices integrate a multitude of functionalities, often on a single chip [3]. Increased Interactions in the design which means more components, which results in more interactions to verify, leading to a combinatorial explosion of test cases.This interlinks with complex debug, finding the root cause of a bug

becomes harder when it could originate in hardware, software, or their interaction.Unlike other design verification methodologies and tools, traditional verification tools might struggle with the diverse nature of IoT designs [4].

Diverse Components

Imagine an IoT device like a smart thermostat. On a single chip, you might find sensors, microcontrollers, memory devices, communication interfaces, hardware accelerators [5] etc. Each of these components has its own specifications, behaviors, and potential failure modes. Verification needs to ensure they all work correctly in isolation and together as a system.

Heterogeneous Integration

This is where things get even trickier. IoT chips often combine different design styles which include analog and digital components. Analog by itself is a pretty complex chip, each chip also has third party IP [6] blocks or even its own sourced IP's. The pre-designed blocks (IP cores) for things like processors, interfaces, or security functions. These blocks may come from different vendors, use different design tools, and have varying levels of documentation.

Complex Software Stacks

It's not just about the hardware. IoT devices run complex software, which could include embedded software for controlling the devices firmware operation, real time operating system (RTOS) which manages tasks, scheduling in real-time, communication protocol[7] which implements stacks for Wi-Fi, bluetooth etc. Verification also needs to consider the interaction between hardware and software.This might involve Software-driven verification and Hardware/software co-verification where simulating both hardware and software together to find integration bugs.

In the current verification methodology, verifying these complex systems requires comprehensive testbenches, advanced debugging tools, and efficient methodologies to manage the sheer volume of verification tasks[8]. Overcoming this is discussed in the later section of this paper.

Low Power

Many IoT devices spend most of their lives in low-power states to conserve energy[9]. Think of a smart door lock that wakes up only when someone approaches or a soil moisture sensor that takes readings infrequently. Verification needs to ensure that the device operates as expected in each power mode (active, sleep, deep sleep)[10].Switching between modes doesn't cause data loss or corruption.

Clock Gating and Power Gating

These techniques shut down parts of the chip that aren't actively being used, like turning off the lights in an empty room. Clock gating stops the clock signal, while power gating cuts off the power supply. But the challenge in terms of verifying this is the Verification needs to confirm that these techniques actually reduce power consumption as intended. Switching clocks or power on and off can introduce timing violations or glitches that disrupt operation. Thorough verification is needed to prevent these issues.

Dynamic Voltage and Frequency Scaling (DVFS)

DVFS adjusts the chip's operating voltage and frequency on the fly. When the device needs more performance, it increases voltage and frequency; when idle, it lowers them to save power. Think of it like shifting gears in a car. Changing voltage and frequency can affect circuit stability, potentially leading to unexpected behavior or crashes.Verification needs to ensure the device meets performance requirements across all DVFS settings.Testing across the full range of voltage and frequency combinations is crucial to catch corner cases that might cause problems[11].

Security

Security vulnerabilities in IoT devices can have severe consequences [12], imagine your smart refrigerator suddenly running malicious code that allows hackers to access your home network. Secure boot prevents this by ensuring that only authorized software executes when the device powers on[13]. Similarly, secure firmware updates guarantee that any new software loaded onto the device is authentic and hasn't been tampered with.Verification needs to confirm that the device boots using legitimate code digitally signed by the manufacturer. This often involves cryptographic techniques like digital signatures and secure key storage.Ensuring that the firmware hasn't been modified during transmission or storage. This might involve checksums or hash functions to detect any changes.Protecting against attacks that try to roll back the firmware to an older, vulnerable version.

Data Protection

IoT devices often handle sensitive data, from personal health information to financial transactions. Protecting this data from unauthorized access is crucial. But the challenge that comes with this is verifying the correct implementation of encryption algorithms to protect data both in transit and at rest. Preventing data corruption or modification, which could have serious consequences, especially in critical applications like healthcare or industrial control.

Contemporary Verification Strategies

Formal verification

IoT devices rely heavily on communication protocols like Wi-Fi, Bluetooth, Zigbee, and MQTT to interact with each other and the cloud. Ensuring that these protocols are implemented correctly is crucial for interoperability and preventing communication errors. This methodology helps in exhaustive analysis where the formal tools can explore all possible interactions and states within a protocol, something impossible with traditional testing. This helps uncover subtle bugs or corner cases that might be missed otherwise. This technique also allows defining the properties of the protocol, such as "every message sent must be acknowledged." The tools then mathematically prove whether these properties hold true under all circumstances[14]. Protocol verification can be performed early in the design cycle, even before hardware is available, using formal models of the protocol and the device. This helps catch errors early, when they are less costly to fix. For e.g. Verifying compliance with the MQTT protocol to ensure reliable message delivery in an IoT network.

Concurrency Verification

Many IoT devices involve concurrent processes, with multiple tasks running simultaneously. This introduces the risk of concurrency bugs like deadlocks where two tasks get stuck waiting for each other and race conditions where the outcome depends on unpredictable timing. These problems can be resolved with formal verification where formal methods can accurately model concurrent behavior and analyze all possible interleavings of tasks. It can also search for potential deadlocks in the design, ensuring that the system never gets stuck. Formal verification can identify potential race conditions where the system's behavior might be unpredictable or lead to data corruption.

By leveraging the power of formal verification, developers can gain a higher level of confidence in the correctness, security, and reliability of their IoT designs. As tools and techniques continue to advance, formal methods are poised to play an even greater role in ensuring the success of the IoT revolution.

Emulation

Emulation uses specialized hardware to accelerate the execution of software on a hardware design [15].Traditionally, software development for embedded systems had to wait for the hardware to be available. This created delays and limited the ability to test software in a realistic environment. Emulation breaks this dependency. Advantage of using emulation platform is that it provides a high-speed, cycle-accurate representation of the hardware, allowing software developers to run their code on a virtual version of the target device.Software can be developed and tested in parallel with hardware design, enabling early detection of hardware/software integration issues.Emulators can run real-world operating systems, applications, and communication protocols, providing a realistic environment for software testing.

System-Level Testing

IoT devices are not just about hardware or software, they are complex systems with intricate interactions between both. Traditional verification techniques often struggle to capture these interactions effectively. Emulation allows you to run the actual software on the emulated hardware[16], enabling comprehensive testing of the entire system which helps in the case of hardware and software co-verification.Emulation provides powerful debugging capabilities to analyze the interactions between hardware and software components and identify the root cause of system-level issues.

Performance Analysis

Performance is often a critical requirement for IoT devices, especially those involved in real-time applications or data-intensive tasks. Emulation helps in accurate performance measurement since it provides cycle-accurate execution, allowing for precise measurement of software performance on the target hardware.By analyzing performance data, developers can identify bottlenecks in the system, such as slow memory access or inefficient code execution.

Emulation bridges the gap between hardware and software verification, providing a powerful platform for early software development, system-level testing, and performance analysis. As emulation technology continues to advance, it is poised to play an even greater role in accelerating the development and verification of complex IoT systems.

Shift-Left Verification and Virtual Prototypes

Shift-left verification emphasizes starting verification early in the design cycle [17]. Verification was often an afterthought, starting only after the design was mostly complete. This led to late discovery of bugs, costly rework, and project delays. Shift-left aims to prevent this.Verification planning begins in the early stages of the design cycle, alongside design specification and architecture definition.Teams define clear verification goals, identify potential risks, and develop strategies to address them proactively.

Formal Property Checking at the RTL Level

Formal verification, as we discussed earlier, uses mathematical techniques to prove design properties. Applying it at the RTL (Register Transfer Level) stage allows for early detection of bugs before they propagate to later stages.

Virtual Prototypes

Virtual prototypes are software models of the hardware design. They allow software developers to start working before the actual hardware is available, enabling parallel development and early system-level testing. Early software development helps in developing,debugging and testing the code on the virtual prototype, accelerating the software development cycle. Virtual prototypes enable early testing of the interaction between hardware and software, identifying integration issues before they become costly problems. Virtual prototypes can be used to explore different design options and evaluate their impact on software and system performance.

Shift-left verification, with its emphasis on early planning, formal methods, and virtual prototypes, is a key enabler for successful IoT development in 2021. By embracing these techniques, development teams can navigate the complexities of IoT design and deliver innovative, reliable, and secure products to market faster.

Verifying Low Power Designs

Traditional simulators often focus on functional correctness and timing, with limited capabilities for accurately modeling power consumption.Power-aware simulations incorporate detailed power models for different components and can analyze power consumption at various levels of abstraction. This allows verification engineers to Identify power-hungry components or operations, verify the effectiveness of power reduction techniques like clock gating and power gating and analyze power consumption in different operating modes and under various workloads.

Power Estimation

Estimating power consumption early in the design cycle is crucial for making informed architectural decisions and guiding design optimization.Power estimation tools provide insights into power consumption at different levels of abstraction, from the RTL to the gate level. This allows designers to explore different design options and evaluate their power implications, identify potential power hotspots early in the design process and optimize the design for low power without relying on time-consuming gate-level simulations.

Security

Security is paramount in IoT, and verification plays a crucial role in ensuring that devices are protected from various threats. Here's a deeper dive into some key security verification approaches as below,

Formal Security Verification

Traditional testing methods often struggle to cover all possible security attack scenarios.Formal methods provide a rigorous way to prove security properties and identify vulnerabilities. This can involve modeling security protocols and analyzing their behavior under different attack scenarios, Verifying the correctness of cryptographic algorithms and security mechanisms and identifying potential weaknesses in the design that could be exploited by attackers.

Side-Channel Analysis

Attackers can exploit physical characteristics of a device, such as power consumption or electromagnetic emissions, to extract sensitive information or compromise its security.Side-channel analysis techniques involve measuring and analyzing these physical characteristics to identify vulnerabilities. This allows designers to uncover weaknesses in the implementation of cryptographic algorithms, detect potential information leakage that could be exploited by attackers and develop countermeasures to mitigate side-channel vulnerabilities.

By employing these specialized techniques and tools, verification engineers can play a critical role in ensuring the low power consumption and robust security of IoT devices, paving the way for a more secure and sustainable connected world.

Future direction

As the complexity and scale of IoT systems continue to grow, the field of semiconductor verification is ripe with opportunities for future research. Prominent areas are captured below. By addressing these open research problems and pursuing these future directions, the verification community can contribute to the development of more reliable, secure, and interoperable IoT systems, unlocking the full potential of this transformative technology.

Scalability

The growing scale and complexity of IoT systems pose a significant challenge for traditional verification methodologies. Verifying systems with millions of interconnected devices requires new approaches to manage complexity and develop scalable verification tools. Future directions include leveraging cloud computing, formal methods for system-level verification, and distributed verification techniques.

Interoperability

The lack of standardization in the IoT landscape poses a challenge for ensuring interoperability between devices from different manufacturers. Developing standardized verification frameworks, effective interoperability testing methods, and benchmarks for evaluating IoT systems are key open problems. Fostering industry collaboration, promoting open-source tools, and using formal models for interoperability are crucial steps towards addressing these challenges.

Security Verification

Traditional security verification techniques struggle to keep pace with the evolving threats and dynamic nature of IoT systems. Verifying security properties in constantly changing systems and defending against

AI-driven attacks are key challenges. Future directions include runtime verification, threat modeling and simulation, and expanding the use of formal methods for security.

AI-Driven Verification

AI and ML offer the potential to automate and optimize verification tasks, addressing the complexity and time-consuming nature of traditional methods. Key challenges include using AI/ML for intelligent test generation, bug prediction, and verification data analysis. Future directions include applying reinforcement learning, deep learning, and data-driven approaches to improve verification efficiency and effectiveness.

Conclusion

The verification of semiconductor designs for IoT devices presents unique challenges due to the complexity, low power requirements, and security concerns. This paper has explored these challenges and analyzed contemporary verification strategies. By embracing advanced techniques like formal verification, emulation, hardware-assisted verification, and machine learning, and by adopting shift-left approaches and virtual prototypes, the industry can address these challenges and ensure the quality, reliability, and security of IoT devices. Continued research and development in verification methodologies are crucial to keep pace with the evolving demands of this rapidly growing field.

References

[1] A. Jerraya et al., "Internet of Things – From Hype to Reality," in *Computer*, vol. 48, no. 6, pp. 22-31, June 2015.
[2] E. Bertino, "Internet of Things Security," in *Computer*, vol. 49, no. 2, pp. 26-31, Feb. 2016.
[3] M. B. Taylor, "Is Dark Silicon Useful? Harnessing the Four Horsemen of the Coming Dark Silicon Apocalypse," in *DAC Design Automation Conference*, 2012, [1] pp. 1131-1136.
[4] J. Bergeron, *Writing Testbenches: Functional Verification of HDL Models*. Springer Science & Business Media, 2013.
[5] F. Vahid and T. Givargis, *Embedded System Design: A Unified Hardware/Software Introduction*. John Wiley & Sons, 2012.
[6] R. D. Dutton and Z. U. Rehman, "Trends in embedded systems technology," in *Embedded Systems Design with Platform FPGAs*, Springer, 2010, pp. 1-20.
[7] E. A. Lee, "Cyber physical systems: Design challenges," in *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, [2] 2008, pp. 363-369.
[8] A. C. Hsu et al., "A case study on the design and verification of a system-on-chip with an ARM core," in *IEEE Design & Test of Computers*, vol. 19, no. 5, pp. 68-77, Sept.-Oct. 2002.
[9] S. Mitra et al., "A survey of techniques for low power design," in *International Journal of Electronics & Communication Technology*, vol. 2, no. 4, pp. 64-71, 2011.
[10] S. Hashemi, D. Grover, and S. Cadambi, "Formal Verification of Power Management in Bluespec SystemVerilog," in *Formal Methods in Computer-Aided Design (FMCAD)*, 2010, pp. 139-146.
[11] D. Singh and S. Sapatnekar, "Power-Aware Verification," in *VLSI Design*, Springer, 2011, pp. 323-359.
[12] E. Roman et al., "A survey of security in the internet of things," in *Computer*, vol. 48, no. 10, pp. 50-58, Oct. 2015.
[13] Y. Shi et al., "Security Verification of Secure Boot in Embedded Systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2013, pp. 1323-1328.
[14] C. Baier and J. Katoen, *Principles of model checking*. MIT Press, 2008.

[15] F. Corno et al., "Emulation-Based Design Exploration for Multi-Processor Systems-on-Chip," in *IEEE Design & Test of Computers*, vol. 24, no. 2, pp. 146-155, March-April 2007.
[16] L. Pierre and E. G. Friedman, "Introduction to Hardware/Software Co-Verification," in *Hardware/Software Co-Design: Principles and Practice*, Springer, 2002, pp. 413-435.
[17] K. Lam et al., "Shift-Left Verification for SoC Designs," in *Design Automation Conference (DAC)*, 2017, pp. 1-6.