Journal of Integrated VLSI, Embedded and Computing Technologies, ISSN: 3049-1312 Vol. 2, No. 2, 2025 (pp. 52-60)
RESEARCH ARTICLE ECEJOURNALS.IN

Verification and Testing Techniques for Reliable System on Chip Solutions

M.E. Lonescu[1], F.A. Stoica[2]*

[1,2]Applied Electronics Department, Technical University of Cluj-Napoca, Baritiu Street, 400027 Cluj-Napoca, Romania

Keywords:
Built-in Self-Test (BIST);
Emulation;
Formal Verification;
Reliability;
Scan-based Testing;
System on Chip (SoC)

Corresponding Author Email:
Fastoica4@ael.utcluj.ro

DOI: 10.31838/JIVCT/02.02.07

Received : 25.12.2024
Revised : 23.01.2025
Accepted : 25.03.2025

Abstract

Verification and Testing Techniques for Reliable System on Chip (SoC) Solutions is a class of research article that investigates the difficulties and advances in making certain that present day SoC outlines are dependable and accessible. With SoCs integrating complex components such as processors, memory and peripheral interface into a single chip, it becomes increasingly important to have efficient verification and testing methodologies. Simulation based verification, formal verification and emulation are some of the kinds of various form of verification that this article describes, along with various things it does to point out design flaw so that they can be identified early in the development process. Furthermore, testing strategies including scan based testing, built-in self test (BIST) and multi-level testing techniques, are also addressed for detecting faults and to achieve the eventual long term reliability of the SoC solutions. The paper examines emerging trends, including machine learning assisted verification, that holds potential to improve the efficiency of classic verification techniques. The article ultimately offers a comprehensive summary of strategies applicable for achieving robust and reliable SoC designs in a rapidly growing number of potential contaminates in the electronic landscape.

How to cite this article: Lonescu ME, Stoica FA (2025). Verification and Testing Techniques for Reliable System on Chip Solutions. Journal of Integrated VLSI, Embedded and Computing Technologies, Vol. 2, No. 2, 2025, 52-60

Introduction

A holistic verification and testing viewpoint is needed to cope with the challenges of providing a robust and reliable System on Chip (SoC) solution. As complexity of these integrated circuits grows, the engineer will sooner or later require more sophisticated design techniques for functional, performance and reliability. This article provides an introduction into the world of SoC verification and testing and how this can help deliver reliable semiconductor solutions using methods, tools and best practices.[1-5]

System on Chip (SoC) Architecture

In system on chip (SoC), all the essential components of a computing system are put together in a single chip. An SoC is an efficient solution when a modern electronic device requires the integrations of a processor, a memory, I/O interfaces, and several peripheral units within one chip. SoC architecture tries to achieve best performance with least size, least power consumption and least cost, which is very important for applications like mobile devices, embedded systems, automotive electronics and IoT. High performance is the major goal of designing SoC architecture for a given system and its major components are tightly coupled, which mean there is a single interface for each component. A microprocessor (CPU) is included in most SoCs as the central unit that can be from a simple core to a complex multi core CPU that can handle multiple tasks simultaneously. The SoC includes memory components such as RAM and ROM along with cache memory that helps the data to access and processing data in time (Figure 1).[6]

Another important characteristic for SoC architecture is special processing units integration.

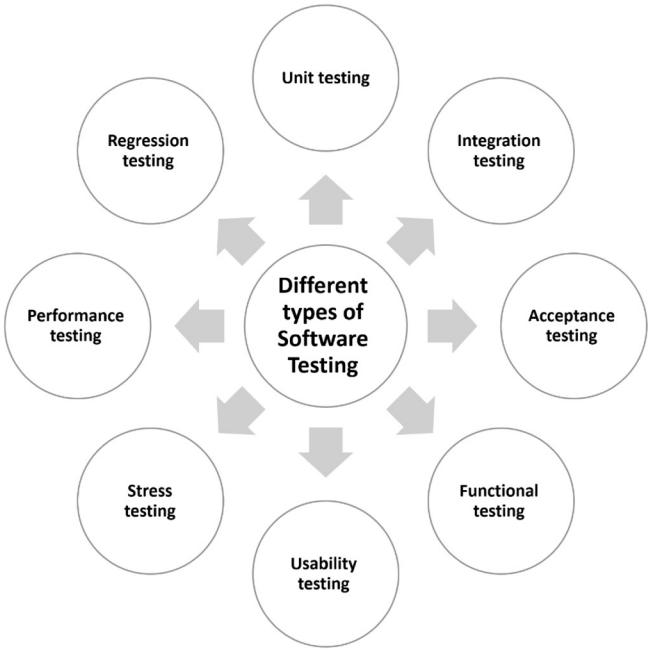

Fig. 1: System on Chip (SoC) Architecture

Specially designed to handle audio and video data and complex graphics, such as Graphics Processing Units (GPUs), the Digital Signal Processors (DSPs), co-processors for some specialized functions including encryption and machine learning. Specialized units offloaded specific tasks of the CPU and increases overall system performance and efficiency.

Also, SoC has communication and I/O interfaces like UART, SPI, I2C, etc. that helps the chip to talk to external devices like sensors, displays, and storage modules. Within an SoC, the interconnect architecture is considered the critical part to ensure the communication between various subsystems in an optimized way. Other high speed protocols that link different components within the chip are the popular interconnect standards like ARM's AMBA and others. Power management is one of the hardest problems in SoC architecture. With larger and more complex SoCs, the problem of power consumption management and performance is becoming more critical as additional components are integrated. To enhance power efficiency, techniques such DVFS or dynamic voltage and frequency scaling can be implemented.

Overall, SoC architecture is one of the key components of modern electronic systems, particularly in terms of high integration, efficiency, and performance of the system for a broad spectrum of applications. As a consequence of ever-increasing numbers in the field of embedded systems, the need for more powerful and less costly, less power hungry embedded systems requires them to continue being continuously evolved.[7-9]

Core Components of SoC Design

System on Chip (SoC) design is developed in such a way that important components of a computing platform is integrated in single chip. An SoC designed to operate at highest performance at reduced power consumption and in smallest space terms, is composed of the main SoC components, which are designed to work together to be run in efficient way. The SoC's Central Processing Unit (CPU) forms the heart of the whole SoC, which is responsible for executing the instruction and, as such, control the operation of the whole system. However, today's modern SoCs have a single core or multi cores processors to perform parallel processing to enhance performance. As a general purpose CPU, it optimizes for handling generic tasks and can be coupled to specialty cores to make it more competent at other operations.

Random Access Memory (RAM) is one of the types of Memory; there is Read-Only Memory (ROM) and cache memory. ROM uses permanent data storage used for boot system firmware and software, while RAM is used for temporary data storage taken while processing. Cache memory is a small and fast memory, when compared with main memory, that is close to the CPU to speed up access to data by storing frequently used data. Graphics Processing Unit (GPU) of an SoC helps graphical data and rendering images in gaming and multimedia and in all the applications that deal with user interfaces. GPU offload removes graphics computation from CPU thereby improving overall system performance especially in the task involving graphics.

A Digital Signal Processor (DSP) is a dedicated processor made for the signal processing tasks like audio, video, and communication processing. As an embedded system, it is used to perform operations such as filtering, modulation and demodulation. The components of the SoC are connected through interconnects to communicate efficiently with each other. Buses, switches, and data transfer protocols likewise form a part of these: connecting the CPU, memory, I/O devices, and other subsystems. ARM's AMBA and AXI (Advanced eXtensible Interface) are just a few common interconnects standard that will keep the parts of the SoC well and effectively communicate (Figure 2).[10-12]

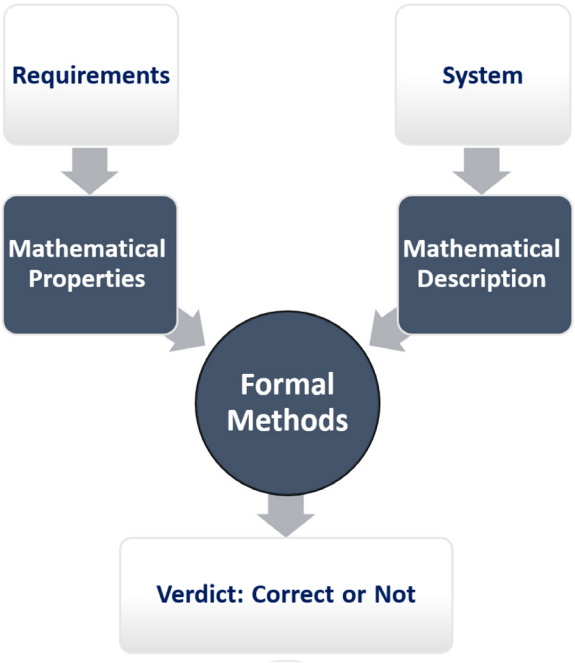

Fig. 2: DSP)dedicated processor

In an SoC, the I/O interfaces are important for communication with external peripherals. The SoC has these interfaces, such as Universal Serial Bus (USB), Ethernet, Bluetooth, and Wi-Fi and others that provide interfaces for the peripherals like sensors, displays, storage devices. The SoC comes with a Power Management Unit (PMU) that manages the power consumption of the SoC, regulating voltage and frequency to use the energy in an efficient way. It is especially important in mobile and embedded systems where battery life is important. Power consumption is optimized by techniques such as dynamic voltage and frequency scaling (DVFS).

Many SoCs integrate security and encryption modules for security during data processing. Ideally, these modules that protects us from cyberattacks such as Trusted Execution Environments (TEE), Secure boot, and encryption decryption units are necessary for applying in IoT, in mobile devices, in financial systems. The ability to contact the real world is provided by analog to digital (ADC) and digital to analog (DAC) converters in the SoC. Prominent application of ADCs and DACs is in sensor systems and communication devices that convert an analog signal like sensor data or Audio input into a digital signal that consists of bits and then convert the digital signal into an analog signal for an output.[13-15]

All SoC components need to operate synchronously, and thus have important role in clock and time circuits. The speed of operation of these circuits also depends on them, and they make sure that the tasks on the different parts of the chip will work well together. It coordinates the operations so that the whole system operates in harmony. Finally, the core components of SoC design allow for efficient and power efficient chip which requires minimum physical space. Aside from feature integration, SoCs are also increasing the amount of specialized component integration for meeting the needs of applications such as artificial intelligence (AI), IoT and 5G communication due to the rising complexity of these applications.

A significant number of SoC solutions developed are robust and reliable, which are necessitated by a comprehensive verification and testing effort. With the ongoing increase in complexity of integrated circuits, engineers are required to use sophisticated techniques in order to guarantee functionality, performance, and reliability. The aim of this article is to understand the complex nature of SoC verification and testing, what the methodologies, tools, and best practices are that lead to such semiconductor solutions.[16-17]

Understanding System-on-Chip Architecture

System on Chip is a paradigm shift in the Integrated Circuit design, where several components are put into one die. It has many advantages such as consuming less power, increased performance and decreased form factor. For example, integration of multiple components also brings about difficulties pertaining to design verification and testing. InCCC technology consolidates multiple functions on a single silicon die in System on Chip (SoC). This approach provides a several advantages such as lower power consumption, better performance, smaller package. Nevertheless, integration of various elements also brings second set of problems in the design verification and testing.

Modern electronic system is based on the System on Chip (SoC) architecture wherein all the major components required for a computing is integrated onto a single chip. They consist of processors, memory, input/output (I/O) interface, etc., plus additional specialized units. Because of the compact size, energy efficiency, cost effectiveness, SoCs are prevalent in the mobile devices, embedded system, consumer electronics and Internet of things (IoT) devices. Based on the history and architecture development, the microprocessor (CPU) is often the central component of SoC architecture for most of computing computation,

Table 1. System-on-Chip Architecture

Verification Technique	Description	Benefit
Simulation-Based Verification	Involves creating a testbench to simulate the SoC design and check if outputs match expected results.	Helps catch logical errors and timing issues early in the design cycle.
Formal Verification	Uses mathematical methods to prove that the design meets specifications without exhaustive testing.	Proves functional correctness, particularly for critical and safety-related components.
Assertion-Based Verification	Incorporates assertions in the RTL code to monitor specific design conditions during simulation.	Automatically detects violations of expected behavior during simulation, reducing debugging time.
Static Timing Analysis (STA)	Checks the timing constraints of the design to ensure that signals propagate within the required time frame.	Ensures the design meets performance specifications, preventing timing-related failures.
Functional Verification	Verifies that the SoC performs the intended functions under different test conditions.	Verifies that the SoC behaves correctly and satisfies its functional requirements.

including data processes and decision making. The CPU can be single core or multi core and more to increase the cores improves performance and parallel processing. SoCs often also contain memory portions like memory components (RAM and ROM) with storage of the firmware necessary for booting and temporary data (Table1).[18-19]

Specialized processors such as Digital Signal Processors (DSPs) for audio or video processing, as well as Graphics Processing Units (GPU) for image and video rendering, can also be integrated into a SoC. Specifically, these specialized processors offload tasks away from the main CPU in order to increase the overall efficiency of the system as these tasks are performed more likely with these processors than with the main CPU. The capacity to support many communication protocols and I/O interfaces such as USB, Bluetooth, Wi-Fi and Ethernet is another biggest feature of SoC architecture. On the other hand, these interfaces combine in order to allow the SoC to communicate with external devices, sensors, displays, and system storage; the variety of such applications ranges from communication and control to data collection and processing.

The interconnect architecture within an SoC facilitates communication between the different components of the chip. Data can be moved quickly between the CPU and its memory, other peripherals, and other specialized processors using high speed buses and switching networks. Power management units (PMUs) that are also part of the soC architecture will control the voltage and frequency to minimize energy use. This is because mobile and embedded systems have become increasingly power conscious, with energy efficiency now becoming a design consideration. To manage the power use of SoCs, based on system activity, various techniques such as dynamic voltage and frequency scaling (DVFS) are used.

Another compelling reason for SoC design is security. Since SoCs are a common inclusion on IoT and mobile devices, therefore they usually have hardware based security such as encryption units, secure boot mechanisms and TEEs to ensure the data is kept secure as well as the communication among devices. An SoC is intrinsically set to scale for application needs in terms of performance or functionality. For example, IoT device may require a simpler architecture with lower power consumption, mobile devices and high performance computing applications may demand more processing power, higher memory capacity and faster I/O interfaces.

First, the integration of many functions in SoCs gives the opportunity to develop a variety of electronic systems more compactly, efficiently, and less expensively. The demand continues to grow for more advanced, reliable, and energy efficient systems where SoC architecture is evolving to feature newer technologies like artificial intelligence (AI) processors advanced communication protocols and the latest security measures. The fact this is going on shows how alive the industry is and the demand for smarter and more powerful systems in the modern electronic device.[20-22]

Reliability and Fault Tolerance Verification

Reliability and fault tolerance are two vital aspects in system design, especially in high performance, safety critical and mission critical applications such as aerospace, automobile, healthcare or telecommunications. Rigorous verification process is needed to ensure that the system can keep running in the presence of faults or failures, and consistently over time.

Reliability Verification is the evaluation of the system's ability to perform assigned function under a wide range of environmental stresses such as constant operation without scheduled maintenance, constant use despite component degradation or wear and tear, etc. It opts to ensure that the system can accomplish its performance goals under normal operating circumstances. Reliability verification is usually performed through the means of accelerated life testing, stress testing and environmental testing. These methods then simulate the so-called wear-out phase of the components to validate the durability and lifespan the system. Expected life time is quantified using reliability models (such as for example Mean Time Between Failures, MTBF or Failure Mode and Effect Analysis, FMEA) in order to find the weak points in the design which may trigger the failure.[23-25]

Fault Tolerance Verification is verifying that a system can bear and heal from faults without losing function. In systems where failure is not acceptable, it is a basic design requirements. Usually, fault tolerance verification simulates different types of failures, such as hardware faults, software errors and communication failures. Fault tolerance verification has traditionally used common techniques such as:

1. **Back up components:** Redundancy utilizing multiple (for example, backup) processor cores, memory modules or power supplies should be used for ensuring that these components are available to take over in the event of a failure.
2. **Error Detection and Correction (EDAC):** including mechanisms that perform error detection and correction, which checks and corrects errors in the data.
3. **Redundancy:** The duplication or replication of components to provide some measure of protection against failure of an individual component.
4. **Self Healing:** These are very advance technologies whereby the system generate such a fault and automatically configures itself or operates using alternative components to continue to perform the function.

Typically, the process of fault tolerance is done by stress testing, fault injecting, and scenario based testing to simulate the real world faults and check how system react to recover from errors. Fault Injection Testing is an important tool that injects errors like power fluctuations, corrupted data, network delays, etc deliberately to find out the system's tolerances in integrating these errors (Figure 3).

Formal verification, model checking, and simulation-based testing are all techniques and tools used in their increasing verification both reliability and fault tolerance. The formal verification deals with prove mathematically that system always work as expected for any possible condition. In particular it's useful in critical systems where human error or unexpected failures can have large consequences. Reliability and fault tolerance verification is aimed at system robustness improvement, so that the system will function as reliably as possible within real world conditions and maintain high availability in face of

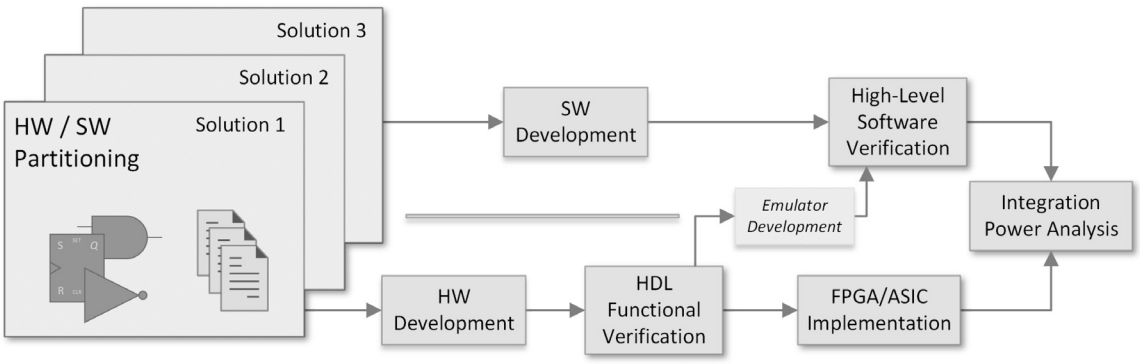

Fig. 3. Reliability and Fault Tolerance Verification

impious disruptions. This will help system's designers to protect their products against reliability failure by guaranteeing product performance without downtime, or the failure related to faults.[26-27]

Fault Injection and Simulation

Fault injection and simulation are important tools for assessing a system's robustness and reliability as it can represent mission critical applications in aerospace, automotive, telecommunications and medical devices. Such methods allow the evaluation of fault tolerance and system recovery mechanism.

In fault injection we deliberately inject faults in a system to model potential failure conditions. This method evaluates the how well the system detects, handles, and recovers from faults. Faults can be injected at different levels including hardware level, software level, network level and environmental level. Hardware fault injection is the process of simulating them issues like, bit flips in memory, defective components or timing errors to test the system's resistance to physical failures. Software fault injection may introduce errors like buffer overflows, logic errors or memory access faults in order to test software's response to faulty conditions. Network fault injection fakes the existence of such problems like packet loss, corruption or delay to test the system against communication failures. Moreover, power and environmental fault injection generates as well as assesses power and environmental stressors such as voltage fluctuations or extreme temperatures (Table 2).[28-29]

Table 2: Fault Injection and Simulation metrics

Testing Technique	Description
Boundary Scan Testing (JTAG)	Uses JTAG standards to test the connections and interconnects between SoC components.
Power Integrity Testing	Verifies that the SoC operates efficiently under different voltage and current conditions.
Thermal Testing	Tests the SoC,Äôs resilience to temperature fluctuations and ensures it operates within thermal limits.
Failure Mode and Effect Analysis (FMEA)	Systematically evaluates potential failure modes in the design to assess their impact.
Manufacturing Testing	Performs electrical and defect tests on fabricated SoCs to identify any manufacturing issues.

However, simulation is a computer based process where the system behavior is replicated using computational models under different conditions including fault scenarios. It is a good technique because it does not require physical hardware, and is very useful for a large or complex system. Defining a fault condition that the system might encounter may be difficult to replicate physically, but simulations can help predict how the system would respond during such conditions, including rare or worst case scenarios. However, by performing these simulations early in the design phase, engineers can discover vulnerabilities and weaknesses in the system with minimal expenses before physical prototypes are produced, as these earlier models help them to achieve their desired goals in minimal time and without consuming a lot of resources (Figure 4).

Finally, since simulation allows for quick, cost effective testing of a broad range of fault scenarios, it is also a valuable tool for simulation. With fault injection and simulation virtual, engineers have an easy opportunity to run multiple configurations and fault conditions without an additional physical hardware. This scalability helps to explore a plethora of failure modes and understand how the system will respond to different situations. Moreover, simulation enables us to continuously monitor the system and learn how it behaves with faults, en route for improving the design of the system. When integrated with model checking, formal verification, and runtime monitoring, fault injection and simulation constitutes a more complete method in ensuring the reliability of a system. On the other hand, formal methods mathematically prove that a system will run correctly in spite of faults and provides high assurances of correctness. Consequently, engineers using these methods combined with fault injection can achieve stronger guarantees about the system fault tolerance and performance.[30-32]

In industries where system failure leads to serious consequences fault and simulation of faults is well used. They are used in aerospace to test avionics systems under different failure condition, to verifies that they will work in case of a failure. Likewise, in the automotive control systems, these are applied to guarantee the reliability of systems as brake controls and airbags, which are a shame of failure. In the telecommunications sector, testing network protocols in a network failure is important, and the fault injection and simulation help to test network protocols so that network is uninterrupted in the case

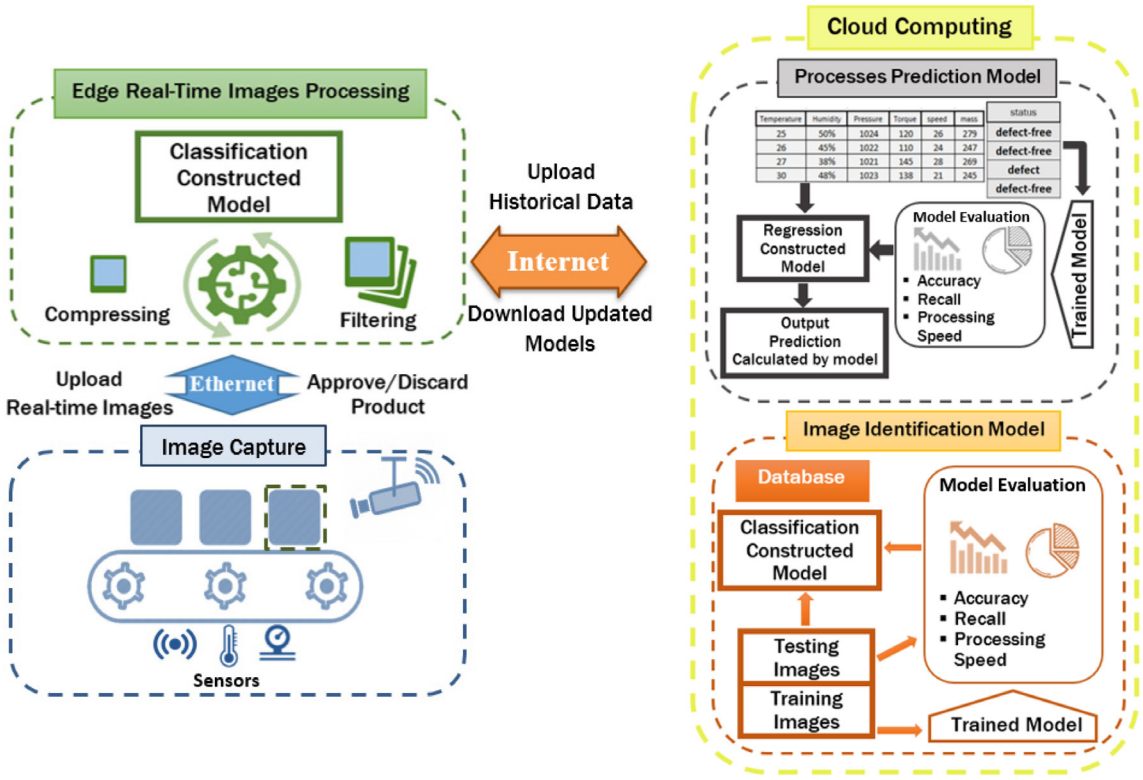

Fig. 4: Fault Injection and Simulation

of a network failure. There are also these techniques used to validate embedded devices, medical devices, and distributed systems, where system will undergo failure which could result in public safety risks. This concludes that fault injection and simulation are very useful techniques in verifying the reliability and fault tolerance of large, complex systems. They offer engineers a means to simulate different fault conditions, analyze the system performance before deployment, and identify weaknesses. These are a set of methods that improve system robustness and resilience that enables systems to continue to operate reliably in the presence of failures or unexpected disruptions.

Conclusion

It is pretty clear that System-on-Chip solutions verification and validation can very easily be summarised in a set of distinct phases, that are different but interlinking, and requiring specific techniques and methodologies to obtain the right characteristics of the last product. So, while SoC complexity is going, it is more and more evident that arbitrary ADAS verification and testing procedures are imperative. Using simulation, formal, emulation, and post silicon validation, engineering teams can develop complex semiconductor solution to the standards for todays electronic systems. More possibilities for future of SoC verification and testing are expected: machine learning enable of verification, cloud emulation, debug automation, etc. But as the industry evolves and attempts to become more integrated and more optimized in SoC designs, an effective backend of line verification and testing solution is going to become ever more important in SoC projects.

References:

1. Shalan, M., & Edwards, T. (2020). Building OpenLANE: A 130 nm OpenROAD-based tapeout-proven flow: Invited paper. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (pp. 1–6). San Diego, CA, USA.
2. Lopera, D. S., Servadei, L., Kasi, V. P., Prebeck, S., & Ecker, W. (2021). RTL delay prediction using neural networks. In *Proceedings of the IEEE Nordic Circuits and Systems Conference (NorCAS)* (pp. 1–7). Oslo, Norway.
3. Colombo, A. W., Karnouskos, S., Yu, X., Kaynak, O., Luo, R. C., Shi, Y., Leitao, P., Ribeiro, L., & Haase, J. (2021). A 70-year Industrial Electronics Society evolution through industrial revolutions: The rise and flourishing of information and communication technologies. *IEEE Industrial Electronics Magazine, 15*, 115–126.

4. Bokor, J., Anderson, E., Kuo, C., Asano, K., Takeuchi, H., Kedzierski, J., & Hisamoto, D. (2000). FinFET-a self-aligned double-gate MOSFET scalable to 20 nm. *IEEE Transactions on Electron Devices, 47*(12), 2320–2325.
5. Borkar, S. (2007). Thousand core chips: A technology perspective. In *Proceedings of the 44th Annual Design Automation Conference* (pp. 746–749). ACM.
6. Burgess, S., Ahonen, T., & Nurmi, J. (2010). Software-based approach to fault diagnosis for multi-die networks-on-chip. In *2010 Conference on System, Software, SoC and Silicon Debug (S4D 10)*.
7. Koo, H., & Mishra, P. (2008). Specification-based compaction of directed tests for functional validation of pipelined processors. In *Proceedings of the International Symposium on Hardware/Software Codesign and System Synthesis (CODES+ISSS)* (pp. 137–142).
8. Vijay, Vallabhuni, et al. (2022). A review on N-bit ripple-carry adder, carry-select adder, and carry-skip adder. *Journal of VLSI Circuits and Systems, 4*(1), 27–32.
9. Mishra, P., & Chen, M. (2009). Efficient techniques for directed test generation using incremental satisfiability. In *Proceedings of the International Conference on VLSI Design* (pp. 65–70).
10. Chen, M., & Mishra, P. (2010). Functional test generation using efficient property clustering and learning techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 29*(3), 396–404.
11. Radulescu, A., Dielissen, J., Pestana, S. G., Gangwal, O. P., Rijpkema, E., Wielage, P., & Goossens, K. (2005). An efficient on-chip NI offering guaranteed services, shared memory abstraction, and flexible network configuration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 24*(1), 4–17. https://doi.org/10.1109/TCAD.2005.869479
12. Pande, P. P., Grecu, C., Jones, M., Ivanov, A., & Saleh, R. (2005). Performance evaluation and design trade-offs for network on chip interconnect architectures. *IEEE Transactions on Computers, 54*(8), 1025–1040. https://doi.org/10.1109/TC.2005.136
13. Duato, J., Yalamanchili, S., & Ni, L. (2002). *Interconnection networks: An engineering approach*. Morgan Kaufmann.
14. Babu, P. A., et al. (2021). Speech emotion recognition system with Librosa. In *Proceedings of the 2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT)* (pp. 421–424). IEEE.
15. Dahlgren, P., Dickinson, P., & Parulkar, I. (2003). Latch divergency in microprocessor failure analysis. In *Proceedings of the International Test Conference (ITC 03)* (pp. 755–763). IEEE.
16. Reddy, V. K., Al-Zawawi, A. S., & Rotenberg, E. (2006). Assertion-based microarchitecture design for improved fault tolerance. In *Proceedings of the 24th International Conference on Computer Design (ICCD 06)*. IEEE.
17. Boule, M., Chenard, J.-S., & Zilic, Z. (2007). Assertion checkers in verification, silicon debug, and in-field diagnosis. In *Proceedings of the 8th International Symposium on Quality Electronic Design (ISQED 07)* (pp. 613–620). IEEE.
18. Basak, A., Bhunia, S., Tkacik, T., & Ray, S. (2017). Security assurance for system-on-chip designs with untrusted IPs. *IEEE Transactions on Information Forensics and Security, 12*(7), 1515–1528. https://doi.org/10.1109/TIFS.2017.2676344
19. Subramanyan, P., & Arora, D. (2014). Formal verification of taint-propagation security properties in a commercial SoC design. In *Proceedings of the 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 1–2). IEEE.
20. Laeufer, K., Koenig, J., Kim, D., Bachrach, J., & Sen, K. (2018). RFuzz: Coverage-directed fuzz testing of RTL on FPGAs. In *Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (pp. 1–8). IEEE.
21. Patel, R., et al. (2017). Advanced formal verification techniques in FPGA-based SoC verification. *ACM Transactions on Design Automation of Electronic Systems, 22*(3). https://doi.org/10.1145/3078832
22. Smith, K., et al. (2018). Verification methodologies for FPGA-based SoC: A comparative study. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays (FPGA)*.
23. Kumar, G., et al. (2020). Formal verification techniques for complex SoC designs implemented on FPGAs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 28*(8), 1–10.
24. Vallabhuni, R. R., et al. (2020). 6Transistor SRAM cell designed using 18nm FinFET technology. In *Proceedings of the 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)* (pp. 1584–1589). IEEE. https://doi.org/10.1109/ICISS49785.2020.9315929
25. Sharshunov, S. G. (1985). Construction of microprocessor tests: The general model. *Data Processing Check, Automation and Telemechanics, 11*, 145–155.
26. Zorian, Y. (2002). What is infrastructure IP? *IEEE Design & Test of Computers*, **19**(5), 5–7. https://doi.org/10.1109/MDT.2002.1044108
27. Densmore, D., Passerone, R., & Sangiovanni-Vincentelli, A. (2006). A platform-based taxonomy for ESL design. *IEEE Design & Test of Computers, 23*(5), 359–373. https://doi.org/10.1109/MDT.2006.129
28. Yang, B., Wu, K., & Karri, R. (2004). Scan-based side-channel attack on dedicated hardware implementations of data encryption standard. In *Proceedings of the 2004 International Conference on Test* (pp. 339–344). IEEE. https://doi.org/10.1109/TEST.2004.1387395

29. Ali, S., Sinanoglu, O., Saeed, S., & Karri, R. (2014). New scan attacks against state-of-the-art countermeasures and DFT. In *Proceedings of the 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)* (pp. 142–147). IEEE. https://doi.org/10.1109/HST.2014.6855585
30. Ege, B., Das, A., Ghosh, S., & Verbauwhede, I. (2012). Differential scan attack on AES with X-tolerant and X-masked test response compactor. In *Proceedings of the 2012 15th Euromicro Conference on Digital System Design* (pp. 545–552). IEEE. https://doi.org/10.1109/DSD.2012.76
31. Chakraborty, R. S., & Bhunia, S. (2011). Security against hardware Trojan attacks using key-based design obfuscation. *Journal of Electronic Testing: Theory and Applications (JETTA), 27*(6), 767–785. https://doi.org/10.1007/s10836-011-5245-9
32. Narasimhan, S., Wang, X., Du, D., Chakraborty, R. S., & Bhunia, S. (2011). TeSR: A robust temporal self-referencing approach for hardware Trojan detection. In *Proceedings of the 2011 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)* (pp. 71–74). IEEE. https://doi.org/10.1109/HST.2011.5954993
33. Alnumay, W.S. (2024). The past and future trends in IoT research. *National Journal of Antennas and Propagation, 6*(1), 13–22.
34. Dorofte, M., & Krein, K. (2024). Novel approaches in AI processing systems for their better reliability and function. *International Journal of Communication and Computer Technologies, 12*(2), 21-30. https://doi.org/10.31838/IJCCTS/12.02.03
35. Rahim, R. (2023). Effective 60 GHz signal propagation in complex indoor settings. *National Journal of RF Engineering and Wireless Communication, 1*(1), 23-29. https://doi.org/10.31838/RFMW/01.01.03
36. Prasath, C. A. (2024). Optimization of FPGA architectures for real-time signal processing in medical devices. *Journal of Integrated VLSI, Embedded and Computing Technologies, 1*(1), 11-15. https://doi.org/10.31838/JIVCT/01.01.03
37. Rahim, R. (2024). Optimizing reconfigurable architectures for enhanced performance in computing. *SCCTS Transactions on Reconfigurable Computing, 1*(1), 11-15. https://doi.org/10.31838/RCC/01.01.03
38. Hoa, N. T., & Voznak, M. (2025). Critical review on understanding cyber security threats. *Innovative Reviews in Engineering and Science, 2*(2), 17-24. https://doi.org/10.31838/INES/02.02.03
39. Muralidharan, J. (2024). Machine learning techniques for anomaly detection in smart IoT sensor networks. *Journal of Wireless Sensor Networks and IoT, 1*(1), 15-22. https://doi.org/10.31838/WSNIOT/01.01.03
40. Sadulla, S. (2024). A comparative study of antenna design strategies for millimeter-wave wireless communication. *SCCTS Journal of Embedded Systems Design and Applications, 1*(1), 13-18. https://doi.org/10.31838/ESA/01.01.03
41. Sadulla, S. (2024). Next-generation semiconductor devices: Breakthroughs in materials and applications. *Progress in Electronics and Communication Engineering, 1*(1), 13–18. https://doi.org/10.31838/PECE/01.01.03