Educational Administration: Theory and Practice
2024, 30(6), 4388-4398
ISSN: 2148-2403
https://kuey.net/

Research Article

A Comprehensive Survey On Challenges And New Directions In Hardware Security

Ritam Rajak[1*], Shukla Banik[2]

[1*, 2]Department of Computer Science and Engineering, Brainware University, Barasat, Kolkata, India

Citation: Ritam Rajak, (2024), A Comprehensive Survey On Challenges And New Directions In Hardware Security, *Educational Administration: Theory and Practice, 30(1),* 4388-4398
Doi: 10.53555/kuey.v30i6.6358

ARTICLE INFO

ABSTRACT

In today's world obstructing hardware devices from non-authorized individuals has become rigorous particularly during the last few decades because of the internationalization of the semiconductor supply chain and widespread network connectivity of electronic equipment. In recent times computing devices have achieved a lot of success, and it has become a magnetic target surface, where the attackers hijack control flow, breach system core of trust, and steal sensitive information by fooling automated learners. To counter these unauthorized individuals from getting access to this sensitive information, security professionals are making Herculean efforts by developing various safeguarding tactics and crafting tools to detect device vulnerability and by taking multiple measures to make hardware devices more invulnerable. In this article, an overview of hardware security and trust from the perspectives of threats, countermeasures, and design tools are introduced. By, bringing forward the most up-to-date findings in hardware security research and development, the researchers strive to inspire hardware designers and electronic automation developers to contemplate the new hurdles and possibilities.

Index terms: non-authorized individuals, Design tools, hardware security, security countermeasures, security threat, survey, automated learners

I. INTRODUCTION

The challenges posed by the production and sharing of new electronic hardware devices, which are often sourced from different vendors and integrated into diverse technological environments with varying security levels. This connectivity exposes complex hardware resources to potential attackers, creating opportunities for third-party attacks without the need for their physical presence. Consequently, modern computing hardware is increasingly vulnerable to a range of security threats.

These threats can occur at various stages of the semiconductor life cycle, from initial specification to fabrication and repurposing, leading to unintended design flaws, secondary impacts, and intentional malicious alterations. Critical resources, including cryptographic functions, secure architectures, intellectual property, and machine learning models, are often targeted. Traditional hardware threats, such as reverse engineering, hardware Trojans, and covert side channels, continue to evolve, leveraging remote, cross-layer, and specification-compatible attack surfaces to bypass security measures.

Security analysts are continuously striving to enhance hardware security protections. This involves creating robust hardware security primitives as the first line of defenses, implementing Trojan detection and efficient side-channel assurance to ensure the integrity of security building blocks and trusted execution environments. Additionally, advancements in artificial intelligence and machine learning have proven valuable in optimizing detection tools.

Despite the implementation of multiple protection techniques, security considerations still do not receive sufficient attention or priority during the hardware design process. This highlights the ongoing need for robust security strategies and design tools to mitigate evolving hardware security threats effectively. The majority of vulnerabilities are only revealed after their exploitation by cybercriminals. Relying too heavily on software patches to address hardware flaws has also contributed to the abundance of zero-day exploits available to

attackers. Therefore, it is essential to design better tools to enforce security aspects of hardware to guarantee trust. Implementing techniques of proactive hardware information flow breaks down those aids in the protection of unwanted distribution of information and constant observation of hardware system. For instance, accidental hardware malfunctions and possible security breaches can be identified sooner by the advanced security-driven hardware design flow. Due to the fact the attacks and countermeasures occur often and is an infinite process, it is advisable to be in a parallel level with its latest prototype to continuously be in close proximity of the productivity gap of protected hardware blueprint. At the present level of developments in hardware design complexity, if the tool chain is unable to get regular updates with the latest prototype for trust and security verification processes, security may put an end to Moore's law before other physical boundaries.

This article thoroughly explores hardware security, focusing on threats, countermeasures, and design tools. It examines specialized areas within the field, covering both familiar and novel subjects, while staying attuned to emerging trends and technological developments. By consolidating recent progress and knowledge, the paper provides a comprehensive overview and sets the stage for future research directions. It highlights the diverse nature of hardware security and emphasizes the importance of tailored research efforts and creative solutions to tackle evolving threats.

II. HARDWARE SECURITY PROPERTIES

Hardware security properties are formalized descriptions of persistent issues concerning the security of circuit designs [15]. Typically, security threats and attacks cause breaches of security attributes, while security countermeasures offer methods for enforcing these properties. The complexities in security measures provide valuable constraints for security verification tools. The sections below briefly describe various hardware security properties.

A. Segregation

Segregation is a bidirectional feature and a common security property used in system-on-chips (SoCs), modern processors, and cloud environments. It ensures that two hardware components with different security levels are prevented from directly communicating with each other. The interaction between secure and non-secure environments is tightly controlled.

B. Statistical Security Features

Statistical security features enhance the accuracy of hardware security design by appraising the impact of vulnerabilities or determining the adequacy of security controls. Examples include leakage through side channels, unpredictable output of cryptographic functions, and the protective features of security mitigation techniques.

C. Durability

Safety is a primary and essential feature that cannot be directly measured, and catastrophic failures represent only a portion of all failures. Durability refers to the ability of a system to deliver intended outputs under normal operation, even with minimal fluctuations in the computing environment over a given period.

D. Steady Time

Breaches of the constant time property alter the timing channel through which unauthorized entities can infer private information. Such breaches can result from performance improvements like cache and branch predictors and faster methods in mathematical units. The steady time security property requires that hardware operations take a consistent amount of time to compute and produce results under various input combinations, ensuring that no information about the inputs can be derived from the computation time.

E. Discretion

Discretion is a fundamental security property that mandates sensitive information should never be obtained by monitoring public outputs or memory locations.

F. Integrity

Integrity is a component of discretion, requiring that a trusted data object should never be replaced by an unreliable entity. This type of attack often targets core complex memory locations, such as cryptographic keys, program counters, and privilege registers. These attacks typically serve as a precursor to further illicit activities, such as hijacking control flow and deceiving machine learning algorithms.

III. THREATS

A. Architectural and System Threats

1. Speculative execution threats: Typically, the initial attacks which are Spectre and Meltdown aid in caching, out-of-order execution, speculative execution, and other execution efficiency improvements to break the quarantine and safety-related policies.

Several processors perform out-of-order execution by using branch prediction. Generally, a spectre is one kind of weakness that tricks a victim's process into disclosing its data. Spectre tricks the user's system because the speculative execution of code leaves traces about its execution in the cache which can be acquired by using cache SCA i.e., (Similar to meltdown). Spectre manipulates the branch predictor so that it can make a wrong move and the code doesn't get executed in any condition. The code is executed out-of-order because the branch predictor is wrong.
Meltdown is a critical security vulnerability affecting many modern microprocessors allowing uncertified processes to analyse data from any address mapped to the current process's memory space. It exploits a race condition within the CPU's execution pipeline.

2. Cache Breach: Cache breach exploits information leakage through the state of the cache and are highly effective at extracting preserved information. As a shared source the cache retains traces of the computation performed by processes, particularly the memory addresses they accessed. Cache side-channel attacks are powerful techniques often combine with other attacks such as meltdown and spectre, to enhance their effectiveness in breaching security [11].

3. Code Reuse attacks: Code reuse attacks such as, return oriented programming (ROP), exploit existing software snippets to execute computation chosen by the attackers. In ROP the attacker sequences existing code fragments, known as gadgets to perform malicious activities without injecting new code.

4. Secure Boot attacks: Secure boot process is design to ensure the integrity and authenticity of system from the moment its powers on. It begins code loaded from an immutable boot ROM, initializing peripherals, configuring security settings, and authenticating boot images and application code, while also sanitizing data upon reset.
Moreover, the process should ensure that the hardware can fully reset all states and only load code from the boot ROM upon reset, maintaining the originality and security of the boot process.

5. Firmware breach: Firmware, the low-level software managing hardware interaction, is crucial for the security of System-on-Chip (SoC) architecture. Incorrectly configured firmware can lead to severe vulnerabilities, including data leaks, unsafe behaviour and critical flaws exploitable by attackers.
Effective access control to hardware resource further enhances secure computing underscoring the intricate and vital role of firmware SoC security.

6. Dynamic Random Access Memory threats: Dynamic Random-Access Memory (DRAM) is vulnerable to significant threats demonstrated by the Coldboot and Rowhammer attacks. Coldboot exploits the persistence of DRAM data even after power is removed.
These attacks underscore the necessity for robust protective measure in DRAM such as error-correcting codes (ECC) and enhanced memory isolation techniques, to save guard sensitive data against both physical and logical threats.

B. Covert and Side Channels

Covert and side channels represent significant security concerns in modern computing systems, leveraging microarchitectural features intended to enhance performance, such as shared caches and speculative execution. Covert channel facilitates information leakage through unconventional communication paths, often involving an insider process like a Trojan horse covertly transmitting data to an external spy process by manipulating the timing of an event or resource state. Conversely, side channels exploit physical phenomena such as variations in supply current event timing or electromagnetic emissions to extract sensitive information based on the system's intrinsic implementation flaws.
These channels underscore the need for enhanced security measures to mitigate unintentional information leakage in both covert and overt scenarios.

1)Timing channel: A timing channel is a type of stored method or component that uses an event processing system to display sensitive data. These techniques become important in multiprocessor environments due to shared hardware resources. Specifically, by monitoring the timing of certain operations, the attacker can gather information about the underlying activities.
Time channels work by exploiting resource sharing and execution mechanisms in the hardware. When software applications share hardware resources, such as cache or memory, differences in access times to these resources can cause issues. For example, storage usage in one mode can affect time in another mode, reflecting storage usage patterns.
Certain conditions must be met for the timing channels to take effect. The two processes of the victim and the attacker must run on the same processor core for a long time. This is important for cache-based attacks where the attacker controls collection.

2) Power side-channel attacks: (SCAs) exploit the power consumption patterns of electronic components to extract sensitive information. By measuring the power usage of a device during its operation, attackers can analyze these power traces to uncover internal data, such as cryptographic keys [12]. As technology nodes shrink and power density increases, the success rate of these attacks has improved.

3) Electromagnetic and Photonic Channels: Electromagnetic (EM) and photonic processes pose significant safety concerns due to unintended EM radiation emitted by electronic devices. This radiation, while a known issue that can interfere with wireless communications and pose a health hazard, can also be used to disseminate sensitive information during safety-critical operations.

4) Fault Injection: A fault injection attack is a powerful side-channel attack in which the attacker intentionally disrupts the normal functionality of the hardware to force it to reveal sensitive information such as cryptographic key bits. These attacks can severely compromise the security of systems running on processors by causing errors that use glitches or powerful lasers to exploit the exploits used to control the control of the device voltage, clock, or temperature. Studies by Rajput et al [2], as well as detailed studies by Valea et al [1], have documented the various security threats and countermeasures related to these testing standards, emphasizing the critical importance of security if severe from emphasizing the error of spinning attacks.

C. IP Theft and Counterfeiting Threats

1) Classical Digital Trojans: Classical digital hardware Trojans (HTs) often use simple but effective techniques to enable malicious behavior. Early HT systems often relied on a single trigger signal to activate the Trojan when a rare event occurred. For example, the Trust-HUB benchmarks are known for using such simple trigger mechanisms, which are more susceptible to switching probability analysis. This vulnerability, while effective in some cases, makes it easier to detect these Trojans through careful analysis of signal exchange patterns.

2) Exploitation of Don't Care Conditions: One sophisticated way to create hardware Trojans is to use don't care conditions, which refer to undefined activity in a system.
Obfuscation is a method of protecting intellectual property by making the design meaningless. In this case, the configuration functions underlying incorrect obfuscation keys are not explicitly specified, leaving the IP designer free to inject malicious circuitry without realizing it. For example, by adding a bad state to a finite-state machine (FSM), HTs can use stateless encoding to generate a floating Trojan state. This state is usually inaccessible during normal operation but can be enabled by fault attacks, forcing the FSM into the bad state. Another approach is to use satisfiability and don't care conditions for HT insertion. This method uses pairs of signals that can never be in a particular input combination due to channel connectivity.

3) Trojans-Induced Aging and Performance Degradation: In nanoscale semiconductor devices, physical processes such as hot-carrier injection, electromigration, and NBTI cause growth, affecting device performance and reliability. Third-party IPs (3PIPs) in system-on-chips (SoCs) increase vulnerability to aging attacks, as rogue vendors can introduce modifications to accelerate aging, leading to device failures early. Fast growth attacks typically use NBTI stress, where the threshold voltage of PMOS transistors increases over time due to trapped charges. This attack highlights the critical importance of robust protection against aging-induced vulnerabilities in semiconductor devices, especially in the case of third-party IP integration in SoCs.

4) Analog Trojans: Researchers have expanded hardware Trojans (HT) to include analog systems, demonstrating the ability to subtly manipulate circuitry systems for malicious purposes. Becker et al. [4] and Kumar et al. [5] achieved this by making slight modifications to the design, such as changing the dopant polarity or varying the ratio of inputs to transistors to induce short circuits at the dopant level. Another example is the A2 Trojan, a small, stealthy analog circuit that incorporates a single capacitor to siphon charge from nearby wires. When it fills up completely, it manipulates the victim's flip-flops to do bad things like increasing opportunity.

5) Trojan Insertion Through Malicious EDA Tool: The threat posed by hardware Trojans (HTs) stems primarily from the unreliability of systems and supply chains. In this case, even Electronic Design Automation (EDA) tools can inadvertently facilitate Trojan attacks. Krieg et al. [6] introduced automated HT insertion by lightweight modifications to an open-end manufacturing tool. This modified FPGA synthesis front end incorporates a main analysis table (LUT) with simulated design behavior that appears normal [14].

D. Vulnerabilities and Attacks on Deep Learning Networks

1) Model Extraction Attacks: Model extraction attacks pose a serious threat to the privacy of pre-trained deep learning models, which are often considered valuable intellectual property (IP) due to the large investments in training data and computational resources required because of the inclusion. Model extraction attacks can be broadly classified into two main types: query-based attacks and application-based attacks [13]. Extraction Attacks Role-based attacks use side-by-side information that slips during modelling. Methods such

as Differential Power Analysis (DPA) and Correlation Power Analysis (CPA) can be used to determine the number of parameters in each layer, the values of these parameters, the total number of layers, and the different activation functions used.

2) Adversarial Examples: Adversarial models represent a significant weakness in deep learning models, which have seen rapid development over the last decade. These models, especially deep neural networks (DNNs), excel in learning high-quality features from raw data, enabling them to solve complex object recognition problems with incredible and human-independent accuracy they are not slightly involved. The sensitivity of DNNs to hostile models was first highlighted by Szegedy et al. [7], who demonstrated that small-scale refined perturbations combined with image interpolation can mislead a DNN classifier in the theoretical phase.

3) Hardware-Oriented Attacks: AI of Things (AIoT) represents a major breakthrough by integrating artificial intelligence (AI) into Internet of Things (IoT) systems. Fault injection attacks pose a serious threat to edge intelligence. Techniques such as laser injection, glitch disturbance, memory collision, and row hammering are used in this attack. While such an attack can cause a denial of service (DoS) by affecting circuit functionality in the DNN, it also alerts the system to the presence of an attack.
Hardware Trojans (HTs) represent another major threat, particularly in connection with DNN integrated circuit (IC) design, manufacturing, and outsourcing testing, or when third-party intellectual property (3PIPs) are being used with DNN hardware in. HTs can surreptitiously insert triggers and payloads into the activation layer or memory controller, resulting in misallocation.

IV. PREVENTIVE MEASURES

A. Hardware Security Primitives

True random number generators (TRNGs) and physically unclonable functions (PUFs) are two important hardware-intrinsic security primitives.
1) TRNG: A random number generator (RNG) is a device or software that generates an unpredictable sequence of numbers. Traditional methods, such as rolling dice or tossing coins, are insufficient for the speed and size required by modern computing systems. Instead, pseudorandom number generators (PRNGs) use algorithms or mathematical programs to generate a sequence of random numbers. This sequence, derived from an initial seed state, is sufficiently long but ultimately finite. For cryptographic applications, PRNGs must be cryptographically secure (CSPRNGs). These CSPRNGs are bound by cryptographic primitives or complex mathematical problems designed to pass the "next-bit test". This test ensures that the (k + 1)th bit of a sequence cannot be predicted in polynomial time, due to the knowledge of the first k bits. In addition, CSPRNGs must be resistant to "state compromise extension" attacks, where an attacker uses knowledge of certain internal states to predict future outcomes or reconstruct past ones. Conventional jitter-based TRNGs use a slow jittery frequency clock to simulate a fast clock. Using clock jitters from free-running ring oscillators (ROs) as an entropy source simplifies the extractor design but requires additional power-hungry clock generators to ensure adequate jitter conversion. Yang et al. [8] proposed a TRNG resistant to structural changes by using oscillation collapse in a double-layered RO. To ensure robustness against configuration variations, 32 stages with eight selectable inverters per stage are used to provide tuning space.

2) PUF as Provenance Proof: PUF (Physically Unclonable Function) takes advantage of variations in manufacturing processes to create unique and unpredictable machine fingerprints. In the early stages of development, the reproducibility of PUF responses under different conditions presented a significant challenge, hindering widespread technical adoption. To address this, various techniques such as the number of votes, fuzzy extractor (FE), and reverse FE (RFE) have been used. Most voters choose the most difficult answer by repeating the same challenge, increasing confidence in the delay. FE enhances noise tolerance and response uniformity through the use of error correction codes (ECC) and hash functions. Currently, PUFs have gained traction in the industry, with companies such as Xilinx, NXP Semiconductor, and Qualcomm incorporating them into their products.

1) User-Device PUF: Traditional methods of deployment and device authentication typically involve a series of authentication schemes, often requiring extensive message exchange. However, the transmission of sensitive certificates presents security risks, especially due to the vulnerability of encryption keys stored in end devices to Non-Volatile Memory (NVM) key retrieval attacks. In a departure from this conventional approach, the concept of integrated automation and device (UD) Physical Unclonable Functions (PUF) was introduced. This new approach aims to distinguish between users and devices using raw biometric information, such as touch screen pressure or voice activation, with silicon sensor variations added to the.

2) Data-Device PUF: PUF-supported data-machine authentication schemes address the limitations of existing data and machine-free authentication schemes in digital forensics, especially those related to digital images and videos, which are highly important information and art but also vulnerable to fraud. These

algorithms aim to address two main issues: image distortion detection and camera recognition. Methods for image modification detection include image watermarking, digital image forensics, and mental image hashing, with the latter particularly effective due to its sensitivity to content-specific modifications and robustness for conservation management purposes.

3) Event-driven PUF: Existing PUFs, such as the CMOS image sensor PUF, typically operate based on server-provided challenges. However, since these complications are not associated with the sensors, controlling the frequency of authentication becomes a challenge, resulting in an unnecessary or insufficient safety marker. Unlike traditional sensors, the DVS responds only to temporal voltage fluctuations, recording asynchronous address events easily with accurate timing information, and with low latency, higher dynamic range, and significant data size reduction. Using these features, an event-oriented PUF scheme was developed adding only three transistors per DVS pixel to deal with the entropy from fabrication process variability. This event-driven PUF provides a discrete response triggered by locally detected asynchronous processed events, avoiding interference from the simultaneous firing of other address events.

B. System and Architectural Protection Techniques

Shared resources are inevitable in computer systems because they improve performance by allowing multiple software systems to use shared memory, datapaths, accelerators, monitors, sensors, and I/O devices. Similarly, hardware IP cores require shared access to on-chip interconnects and memories. However, these shared resources pose significant security challenges, primarily due to the risk of information leakage when processing sensitive data. The cache side channel is a prime example of this vulnerability, as it has been exploited several times for malicious purposes.

1) Cache Side-Channel Mitigations: Cache side channel mitigations aim to prevent information leaks by separating safe and unsafe paths to the cache. One important approach is cache partitioning, which requires partitioning the cache to avoid conflicts between data sets. Cache partitioning methods include static locking (PLCache), dynamic locking, page coloring, and optional cache flushing. Static locking first assigns specific cache lines to certain processes, while dynamic locking resolves these assignments based on runtime needs.

2) Control Flow Integrity: Control Flow Integrity (CFI) is an important security measure that prevents reusable attacks by ensuring that a program adheres to its intended control flow during execution. It can be implemented through software or hardware, each with distinct advantages. Software-based CFI is flexible and widely used, involving runtime checks and validations in the software. In contrast, hardware-based CFI, although infrequent due to the need for extensive microarchitectural flexibility, provides strong security with minimal operational costs through trusted hardware monitors adding to the instruction pipeline or processor debugging resources.

3) Trusted Execution Environment: The most common way to achieve software isolation is to use a Trusted Execution Environment (TEE), which relies on hardware devices to enforce isolation properties. These features typically ensure that the vulnerable computer has resources dedicated and reserved for other system members. TEEs are used differently, and each has its own way of ensuring safety. For example, Intel's Software Guard Extensions (SGX) use enclaves—protected objects containing rules and data for security-related computing. SGX achieves isolation by assigning a specific memory location to trusted computers and protecting this memory from other components including the kernel, hypervisor, and Direct Memory Access (DMA). Similarly, ARM TrustZone divides the system into two parts: a secure world for sensitive accounting and a normal world for routine operations.

C. Side-Channel Protection Techniques

1) Timing-Channel Countermeasures: Existing countermeasures against Side-Channel Attacks (SCA) encompass a broad range of software- and hardware-based strategies, aimed at addressing vulnerabilities at various stages of system development. These countermeasures focus not only on potential run-time attacks but also involve early assessments at the design stage to assess the vulnerability of such risks. Alternatively, bit slicing techniques have been explored to use time constant AES cores while improving overall performance. Compiler-based countermeasures represent an alternative defense mechanism, aimed at dealing with timing processes by introducing noise or randomization in software implementation to hide leakage-related timing.

2) Fault Attack Countermeasures: Over the years, several countermeasures have emerged to protect digital systems from terror attacks, broadly divided into infectious and detection-based methods. Detection-based countermeasures involve augmenting the design with detection circuitry such as parity or redundant copies to detect the presence of faults. Parity-based circuits and redundancy techniques have been extensively explored in various fields. Methods such as diffusion-based methods, which involve swapping redundant and original outputs, or using fixed constant matrices to swap output data serve this purpose. However, the main limitation of existing methods is the manual identification of simple fault locations by mechanical engineers, causing difficulties, especially in large-scale systems.

3) Power Side-Channel Countermeasures: Power side-channel analysis (SCA) countermeasures include algorithmic, physical, and system-level approaches. Algorithmic countermeasures involve introducing additional operations to mask or distribute sensitive computation, offering provable security benefits. Physical countermeasures rely on measurements to validate device protection, addressing side-channel leakage measurement challenges through techniques such as custom gates that consume power independently of switching. System-level countermeasures leverage the device's power supply to normalize or randomize overall power consumption. While algorithmic and system-level approaches require additional circuitry, physical countermeasures use custom logic design methods. System-level countermeasures introduce noise into the power supply to reduce the signal-to-noise ratio in side-channel leakage. However, the specialized circuitry required by these methods can significantly affect design area, power, and performance.

4) EM Side-Channel Countermeasure: Countermeasures against Electromagnetic (EM) side-channel attacks work on both the hardware and software fronts, each targeting different vulnerabilities. On the software side, techniques such as execution sequence randomization and randomization of Look-Up Tables (LUTs) serve to destroy predictable patterns in EM dumps, making it harder for attackers to extract intelligence needs out of it. In particular, a couple of instruction sequences can reveal unique features in EM signatures, which can be used to identify critical safety issues. By sequencing instructions, these distinctive characteristics can be masked, increasing security. A key to mitigating the weaknesses of the EM side-channel lies in the evaluation process for hardware systems. By systematically scanning systems for potential EM vulnerabilities, it becomes possible to identify them earlier, enabling the integration of system-time mitigation strategies.

D. IP Protection Techniques

1) Hardware Steganography: IP watermarking, although commonly used to protect intellectual property (IP) in hardware design, faces significant limitations related to optimizing design-based signatures to maximize robustness cost-effectively which is more than enough. In contrast, hardware steganography appears as a promising alternative for several reasons. First, hardware steganography offers the ability to seamlessly resolve ownership disputes and detect pirates, providing a more flexible approach compared to watermarks. Second, the secret stego-based hardware constraints are derived from the entropy threshold parameter, simplifying the process and reducing design overhead.

In a recent study, a vendor signature-free entropy-based hardware steganography method has been proposed to encrypt DSP cores, using an alternative method. The method involves inputting private information on the register allocation phase of High-Level Synthesis (HLS) through the Code Independence Graph (CIG) framework. Stego constraints are caused by edges between two nodes with the same colors, which simplifies the embedding process.

2) Logic Obfuscation: Logic obfuscation is an important technique for protecting hardware Intellectual Property (IP) from unauthorized access and reverse engineering. Common methods of logic obfuscation include XOR/XNOR and MUX-based logic locking, which change the behaviour of the internal nodes or the information that goes into the hardware. Another approach is to introduce programmable features to hide part of the logic until later programming steps. Despite these efforts, however, the mechanisms of logic confusion are not immune to sophisticated attacks. One such attack is the Functional Oracle-Guided SAT attack, which also looks for input patterns that can distinguish between valid and invalid obfuscation keys. To combat this, anti-SAT logic locking techniques have been developed.

3) Hardware Watermarking: Hardware watermarking is an important technique used to protect intellectual property (IP) from various threats such as piracy, counterfeiting, cloning, and false claims of ownership. It can be applied at different stages of the design process, including the electronic system level (ESL), high-level synthesis (HLS), or logic synthesis level. For example, ESL or HLS-based hardware watermarking involves adding the author's signature to the design during the pre-synthesis phase. In one notable example, the author's signature is binary encoded and embedded as additional design and timing constraints. These constraints are integrated into the design as binary bitstreams of ASCII characters.

E. Hardware Trojan Detection and Prevention Techniques

1) 3PIP Trojan Detection: Most hardware Trojan (HT) detection methods rely on Trojan benchmarks to assess effectiveness. Identifying unknown HTs in third-party intellectual property (3PIP) is particularly challenging due to the lack of detailed information about the Trojan's usage. Several factors complicate this assessment process: functional test coverage, the location of potential change analyzes, and noise caused by design changes.

2) Runtime Monitoring Techniques: Due to the NP-completeness of testing problems, such as controllability, observability, and automatic test pattern generation (ATPG), it is impossible to guarantee removal of hardware Trojans (HT) is completely gone before a device is used. Consequently, it is useful to use

runtime monitoring techniques to detect and prevent HT attacks in security-critical systems. These methods may include vital sign monitoring, dynamic power management, or electromagnetic (EM) radiation.

3) Design-for-Trust (DFS) Techniques: Design-for-Security (DFS) techniques add dedicated logic to aid in hardware Trojan (HT) detection. Some techniques incorporate dummy flip-flops and test points to provide more controllability and visibility of internal nodes, thus speeding up the process of activating Trojans. Another approach is to install circuit infrastructure such as ring oscillators (ROs) and current sensors that aid in production screening and on-site monitoring of all HT-infected chips. This system also supports side-channel analysis (SCA)-based HT detection.

4) HT Prevention Techniques: Hardware Trojan (HT) prevention methods are designed to make HT installation extremely difficult or, at best, impossible. The three most common strategies for preventing HT interference are rationalization, decoupling, and scheduling.
Logic Obfuscation: Originally designed for intellectual property (IP) protection, logic obfuscation can also be used to prevent HT from being inserted. This method locks into the operation of the circuit a key other than that of the untrusted foundry. Without this key, the circuit remains locked, making it difficult for an adversary to successfully inject a Trojan.
Split Manufacturing: This method involves splitting the manufacturing process into two parts: the front end of the line (FEOL) and the back end of the line (BEOL). The FEOL part is done by a trusted agency, while the BEOL part is handled by an untrusted agency. Since the unreliable foundry lacks general design information, especially the BEOL part, the installation of a functional HT becomes more difficult.

5) Presilicon Countermeasures: Presilicon hardware Trojan (HT) detection methods are designed to detect HTs in the initial configuration. These methods include switching probability analysis, structural checking, and security verification. HT detection methods based on configuration analysis focus on extracting specific configuration features related to HT systems, such as gate type, number of gates, and communication systems. Detection is done using techniques such as pattern matching, which involves scoring algorithms to match these objects with the circuit structures being tested to detect Trojan circuitry.

6) Postsilicon Countermeasure: Presilicon hardware Trojan (HT) detection methods aim to detect malicious modifications early in the design phase, while postsilicon detection looks for such modifications after chip manufacture. A common postsilicon approach is destructive reverse engineering (RE), which involves depackaging and delayering integrated circuits (ICs) to extract the circuit structure from layout diagrams. Although effective, this method is costly, time-consuming, and subject to failure if the HT is confined to only a few chips.

F. ML-Assisted Solutions

1) ML for Detection: Machine learning (ML) techniques are increasingly being used to detect integrated circuit (IC) counterfeits and hardware Trojans (HTs), which pose a serious threat to IC facilities. Traditional analytical methods for identifying these threats can be too time-consuming or ineffective. To address this, ML models automate the analysis process by analyzing and classifying parametric measurements collected from on-chip sensors. For example, support vector machines (SVMs) can be used to detect recycled ICs by detecting anomalies in these measurements. SVMs are also effective for real-time HT detection, providing fast and proactive responses to potential threats. Besides SVMs, other ML models such as random forests and multilayer perceptrons (MLPs) have been used to combat microarchitectural side-channel attacks (SCAs).

2) ML for Robust Architecture Design: Highly robust systems are set to stay ahead of evolving security threats. Researchers have explored new ways to combine machine learning (ML) techniques with unique system characteristics to enhance security. For example, Yang et al [9]. exploited the obsolescence effect of the memristor to design a secure neuromorphic computer system, using the availability of memristors to enhance security measures. Similarly, Shan et al. introduced ML-assisted compensatory capabilities aimed at improving resilience against flanking attack (SCA).

G. Countermeasures Against DNN Attacks

Most countermeasures against anti-DNN attacks can be divided into proactive and reactive categories. The former aims to improve model robustness while the latter aims to detect adversarial inputs.

1) Reactive Measures: Reactive methods for combating adversary attacks on deep neural networks (DNNs) provide a proactive method of defenses by focusing on detecting adversary inputs and taking appropriate actions on. These methods derive from passive regularization techniques, which modify model parameters to improve robustness without directly addressing the presence of adversarial perturbations. Instead, tactics aim to detect and respond to enemy input in real time. One common methodological component is Sample Statistics, which uses statistical algorithms derived from DNN applications to distinguish between natural and adversary inputs.

Furthermore, Xu et al. [3] propose a method to determine the quality of inputs by comparing the classification results of initial and squeezed inputs with a predetermined threshold. This approach aims to minimize adversary abuse in high-dimensional feature spaces by distinguishing between benign and adversarial inputs based on their classification results.

2) Proactive Measures: Measures taken to counter adversary attacks are commonly implemented on the internet and can be divided into three main approaches. Adversarial training involves retraining the model by adding off-the-shelf adversarial examples to the original training dataset. While effective, this approach is hindered by the cost of creating a bad copy and by the assumption that the defender knows the attacker's route. This method controls the Lipschitz constant of the network during quantization, effectively protecting neural networks from adversarial attacks while preserving hardware efficiency for small bitwidth data. All these proactive measures help strengthen the resilience of DNN models against adversary threats.

V. DIFFERENT HARDWARE SECURITY TOOLS

A) Security-Centric Hardware Design Tools:

Takarabt et al. propose a presilicon analytical method and tool to enable safety verification along with operational verification. This tool identifies vulnerabilities and identifies specific code lines where vulnerabilities lie, providing additional attributes such as severity. Recent developments also include security-driven metrics, models, and Computer-Aided Design (CAD) flows that integrate logic encryption, split manufacturing, and camouflaging to enhance secure hardware design. Knechtel et al [10]. provides a comprehensive survey of the role of Electronic Design Automation (EDA) in hardware security, highlighting the challenges of efficiently assembling security considerations and constraints at different levels of abstraction, modeling and testing hardware security metrics, and general integration of safety countermeasures to avoid unintended side effects.

B. Security Verification Tools

The researchers have summarized a few projects from the hardware side and also present some recent developments.

1) Academia Tools: One of the first hardware security verification tools developed by the gate-level information flow tracking (GLIFT) project laid the foundation for hardware information flow tracking (IFT). GLIFT established the basic concepts of IFT by formalizing tracking logic and complexity theories. It has been used to demonstrate tight isolation in computer systems, detect time channels, and detect Hardware Trojans (HTs). As the hardware design evolved, high-level IFT techniques such as register-transfer-level IFT (RTLIFT) emerged to address verification performance challenges at the gate level. GLIFT, RTLIFT, Clepsydra, and VeriSketch are secure hardware design tools targeted at RTL Verilog designs. RTLIFT has shown significant improvement in verification performance compared to GLIFT. Clepsydra focuses on time-only and full-time property authentication, while VeriSketch integrates hardware policies that conform to desired security properties such as confidentiality, integrity, and consistency.

2) Commercial Tools: Many EDA and hardware security companies have introduced a variety of secure hardware design tools. Mentor Graphics SecureCheck is a secure format authentication tool built on top of the Questa Formal authentication engine. It uses assertion-based formal verification to establish privacy and integrity properties, and identifies dangerous paths that are vulnerable to security breaches. Tortuga Logic provides Prospect, a hardware security formal verification tool that uses GLIFT to generate logic for information flow tracking across circuits. This logic is used for design-time verification without providing additional circuitry. Synopsys emphasizes reliability and functional safety certification. CustomSim provides tools for device-level and network reliability analysis, covering aspects such as infrared radiation, current density, electromigration, and device aging.

VI. POTENTIAL RESEARCH DIRECTIONS

A. IoT and Cyber–Physical Security

The importance of ensuring security in IoT and CPS is growing rapidly for several reasons. First, these systems interact directly with the physical world, making any security vulnerability a potential security threat. Second, they are typically developed and implemented under severe cost and time constraints, which limit a comprehensive security system and authentication system.

B. Security-Driven EDA

The current state-of-the-art EDA streams prioritize functional discipline and performance within specified budgets as key design constraints. However, there is increasing recognition of the importance of incorporating security considerations into the hardware design process. This represents a promising but challenging research direction for both the hardware security and EDA communities.

C. System and Architecture Security

System designers are constantly trying to balance performance, energy consumption, and community in hardware design. Nowadays, with the increasing importance of security, it has become necessary to think about security as an innovation for effective implementation. However, quantifying "safety" poses significant challenges in hardware design. Security metrics play an important role in the vulnerability assessment, threat mitigation, and security certification process. Ideally, an effective security metric should provide an accurate assessment of the severity of the threat. Because security includes multiple threat models, no single metric can adequately cover all aspects.

D. ML for Hardware Security and Security of ML

Machine learning (ML) models have the dual ability to initiate or defend against attacks on hardware companies. Current ML-assisted countermeasures are mainly based on simple models such as Support Vector Machines (SVMs), which may be due to the size of the problem and the limited availability of training data. However, as attacks progress, sophisticated ML models such as Deep Neural Networks (DNNs) may emerge due to their skills in data processing, and large amounts of training data will be required. Thus, unsupervised learning can be a solution, as labeled articles are generally more valuable than unlabeled articles. Furthermore, it is important to direct ML-enabled methods towards robust architecture design rather than mere anomaly detection, as the latter is often too late to prevent damage.

VII. CONCLUSION

Hardware security is a complex and multifaceted field that spans multiple abstractions in the computer systems stack. Because of its broad and interdisciplinary nature, it is impractical to fully cover all aspects in a single document. In this article, we have focused on selected subfields of hardware security and discussed recent advances in them. In particular, we have explored attacks and countermeasures on security systems, IP components, and Deep Neural Network (DNN) models, as well as the design and implementation of hardware-intrinsic security primitives. In addition, we have discussed recent developments in security-driven hardware design tools. The landscape of hardware attacks and countermeasures is constantly evolving, often changing with major changes in processor architectures and computing technologies. This complex interplay between attack and defense in hardware security is expected to continue for the foreseeable future. Our goal in this review is to enable hardware designers and tool manufacturers to understand significant security gaps that cannot be adequately addressed by traditional hardware design and authentication methods. By doing so, we hope to stimulate further research and innovation in the field to strengthen the security of hardware systems.

REFERENCES

1. E. Valea, M. Da Silva, G. Di Natale, M. -L. Flottes and B. Rouzeyre, "A Survey on Security Threats and Countermeasures in IEEE Test Standards," in *IEEE Design & Test*, vol. 36, no. 3, pp. 95-116, June 2019, doi: 10.1109/MDAT.2019.2899064.
2. P. H. N. Rajput and M. Maniatakos, "JTAG: A Multifaceted Tool for Cyber Security," *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, Rhodes, Greece, 2019, pp. 155-158, doi: 10.1109/IOLTS.2019.8854430.
3. Y. Liu, L. Wei, B. Luo and Q. Xu, "Fault injection attack on deep neural network," *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Irvine, CA, USA, 2017, pp. 131-138, doi: 10.1109/ICCAD.2017.8203770.
4. Becker, G.T., Regazzoni, F., Paar, C., Burleson, W.P. Stealthy Dopant-Level Hardware Trojans. In: Bertoni, G., Coron, JS. (eds) Cryptographic Hardware and Embedded Systems - CHES 2013. CHES 2013. Lecture Notes in Computer Science, vol 8086. Springer, Berlin, Heidelberg.
5. R. Kumar, P. Jovanovic, W. Burleson and I. Polian, "Parametric Trojans for Fault-Injection Attacks on Cryptographic Hardware," *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, Busan, Korea (South), 2014, pp. 18-28, doi: 10.1109/FDTC.2014.12.
6. C. Krieg, C. Wolf and A. Jantsch, "Malicious LUT: A stealthy FPGA Trojan injected and triggered by the design flow," *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Austin, TX, USA, 2016, pp. 1-8, doi: 10.1145/2966986.2967054.
7. Goodfellow, Ian J and Shlens, Jonathon and Szegedy, Christian. In Explaining and harnessing adversarial examples, 2014, arXiv preprint arXiv:1412.6572.
8. Breier, Jakub and Hou, Xiaolu and Jap, Dirmanto and Ma, Lei and Bhasin, Shivam and Liu, Yang, Practical fault attack on deep neural networks, 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 2204—2206.
9. K. Yang, M. Hicks, Q. Dong, T. Austin and D. Sylvester, "A2: Analog Malicious Hardware," *2016 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, 2016, pp. 18-37, doi: 10.1109/SP.2016.10.
10. Knechtel, Johann and Kavun, Elif Bilge and Regazzoni, Francesco and Heuser, Annelie and Chattopadhyay, Anupam and Mukhopadhyay, Debdeep and Dey, Soumyajit and Fei, Yunsi and Belenky, Yaacov and Levi, Itamar and others, "Towards Secure Composition of Integrated Circuits and Electronic Systems: On the

Role of EDA," *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, 2020, pp. 508-513, doi: 10.23919/DATE48585.2020.9116483.
11. Lyu, Y., Mishra, P. A Survey of Side-Channel Attacks on Caches and Countermeasures. J Hardw Syst Secur 2, 33–50 (2018).
12. Asanka Sayakkara, Nhien-An Le-Khac, Mark Scanlon. A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics, Digital Investigation, Volume 29, 2019,pp 43-54, ISSN 1742-2876.
13. P. Kocher *et al*., "Spectre Attacks: Exploiting Speculative Execution," *2019 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 2019, pp. 1-19.
14. F. Schellenberg, D. R. E. Gnad, A. Moradi and M. B. Tahoori, "An inside job: Remote power analysis attacks on FPGAs," *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, 2018, pp. 1111-1116.
15. Prasanna Ravi, Zakaria Najm, Shivam Bhasin, Mustafa Khairallah, Sourav Sen Gupta, Anupam Chattopadhyay, Security is an architectural design constraint, Microprocessors and Microsystems, Volume 68, 2019, pp 17-27.