Technical Report

ISO/IEC TR 5891

First edition
2024-04

Information security, cybersecurity and privacy protection — Hardware monitoring technology for hardware security assessment

Sécurité de l’information, cybersécurité et protection de la vie privée — Technologie de surveillance des matériels pour l'évaluation de leur sécurité

Reference number
ISO/IEC TR 5891:2024(en)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *Information security, cybersecurity and privacy protection*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

Hardware components and the computing ecosystem are becoming increasingly complex. As a result, it becomes increasingly difficult to evaluate the security of hardware. Even in the design stage, it is quite difficult to identify abnormal parts that can cause flaws from among millions of source code lines or billions of transistors, as well as the physical connections between them. Other areas of technology use monitoring to assist with the evaluation aiming to mitigate such difficulties. In those technologies, runtime activities such as changes in internal or external status can be monitored to identify deviations from normal behaviour patterns, and by these means, the evaluation can focus on a small set of patterns that the monitored subject typically works with. This method now becomes an available option to assist in hardware security assessment. In such cases, either the target of security assessment is supposed to be "runtime hardware-behaviour-based security", or introduced as a proactive approach to security.

Many evaluation and assessment standards, such as ISO/IEC TS 30104, ISO/IEC 19790 and ISO/IEC 17825, focus on physical security (invasive/nonintrusive) at the hardware boundary. However, they do not focus on the monitoring data, either offline or in real time.

Information security, cybersecurity and privacy protection — Hardware monitoring technology for hardware security assessment

1 Scope

This document surveys and summarizes the existing hardware monitoring methods, including research efforts and industrial applications. The explored monitoring technologies are classified by applied area, carrier type, target entity, objective pattern, and method of deployment. Moreover, this document summarizes the possible ways of utilizing monitoring technologies for hardware security assessment with some existing state-of-the-art security assessment approaches.

The hardware mentioned in this document refers only to the core processing hardware, such as the central processing unit (CPU), microcontroller unit (MCU), and system on a chip (SoC), in the von Neumann system and does not include single-input or single-output devices such as memory or displays.

The hardware monitoring technology discussed in this document has the following considerations and restrictions:

— the monitored target is for the post-silicon phase, not for the design-house phase (e.g. an RTL or netlist design);

— monitoring is only applied to the runtime system.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 15408-1, *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 1: Introduction and general model*

ISO/IEC/TS 30104:2015, *Information Technology — Security Techniques — Physical Security Attacks, Mitigation Techniques and Security Requirements*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 15408-1, ISO/IEC TS 30104 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
hardware monitoring
hardware or software component allowing an individual to monitor devices connected to a computer

3.2
runtime hardware-behaviour-based security
function of a hardware that protects running physical devices from harm caused by abnormal or unexpected state transitions

Note 1 to entry: Such transitions come from vulnerability, non-declarations, or malicious logic.

4 Abbreviated terms

CPU	central processing unit
DRAM	dynamic random-access memory
EDA	electronic design automation
FSM	finite state machine
I/O	input/output
MCU	microcontroller unit
NIC	network interface controller
RAS	reliability availability and serviceability
RTL	register transfer level
SoC	system on a chip
JTAG	Joint Test Action Group
IP	intellectual property
CISC	complex instruction set computer
QoS	Quality of Service
ISA	instruction set architecture
ECC	error checking and correction
ROM	read-only memory
EEPROM	electrically erasable programmable ROM
VMM	virtual machine manager
FPGA	field programmable gate array

5 Relationship to existing standards

5.1 Standards of security assessment

Existing security assessments and technical standards face challenges in addressing hardware uncontrollability. ISO/IEC TS 30104, ISO/IEC 19790, and ISO/IEC 17825 focus on (invasive/nonintrusive) physical security at the hardware boundary, but not over the boundary. ISO/IEC TR 20004 complements the vulnerability analysis of ISO/IEC 15408-3 from the perspective of software. Among these, there are no relevant hardware standards.

5.2 Relationship to ISO/IEC 15408-3

In ISO/IEC 15408-3, vulnerability assessment is defined. This document aims to survey technologies to support hardware vulnerability assessment that can be done at runtime.

5.3 Relationship to ISO/IEC TS 30104

This document aims to supplement ISO/IEC TS 30104:2015, 7.2 and 7.3.

6 Background

6.1 Complexity and security

Modern circuits are very complex, and their complexity, amplified by time-to-market pressure, is increasing rapidly in modern computing environments. Consequently, design houses frequently use external IPs, and most IC design enterprises are fabless.

The complexity of modern systems increases the attack surface. Because the semiconductor industry has shifted to a horizontal business model for the integrated circuit supply chain, malicious hardware (hardware Trojans) can be implanted in untrusted phases or components, e.g. commercial IP cores, EDA tools, fabrication, and assembly services. Such malicious modifications to the original circuitry are inserted by adversaries to exploit hardware or to use hardware mechanisms to create backdoors in the design.

6.2 Challenges in defining hardware security assessment techniques

It is difficult to address all such security risks because of the complexity of processes and components, outsourcing of design and fabrication, and the increase in the sophistication of potential attacks.

For a given piece of hardware, especially a complex system such as a modern SoC, it is also difficult to identify small malicious modifications to the original design (e.g. at the gate-level netlist). Even with trusted source code, a piece of hardware cannot be guaranteed to be Trojan-free in the post-silicon phase. Traditional tests (e.g. function coverage tests, fault tests, and random case tests) are less likely to help with such detection because hardware Trojans are typically activated, i.e. designed to be activated, by specified conditions, such as specific sequences of instructions, particular combinations of external signals, a timer, or a temperature threshold. Adversaries can make the Trojan active and launch attacks, then switch it off during hardware runtime. In other words, traditional methods of auditing the source code or performing manufacturing fault detection are ineffective for hardware security assessment.

A hardware Trojan is a malicious inclusion or modification of hardware. A hardware Trojan consists of a trigger circuit and a payload circuit. The trigger circuit activates the payload circuit under a specific condition, and the payload circuit implements the malicious behaviour of the hardware Trojan. The hardware Trojan can leak secret information in the hardware or bypass or disable the security functions of the hardware.

Hardware Trojan detection is applicable in multiple phases of hardware production and distribution. For example, detection based on the netlist can be applied in the designing phase. Side-channel and logic approaches can be applied during the manufacturing or in-use phase. Focusing on the fact that a hardware Trojan alters the behaviour of the circuit, the side-channel approach detects abnormal behaviour from the side-channel information. The logic-test-based approach generates test patterns to detect hardware Trojans via the output. Some state-of-the-art approaches use machine learning technologies to detect hardware Trojans from the netlist or side-channel information of the hardware.

Hardware flaws, such as Meltdown, Spectre and a series of newly revealed flaws,[99-101] are a result of pursuing performance, for instance, parallelism, during microarchitecture development. First, they are difficult to fix, and the fixes can cost more than the gains from hardware optimization. Second, some of these flaws exist for approximately 10 to 15 years before they are revealed. Traditional detection in the pre-silicon phase would be unlikely to help since flaws are not malicious modifications. It is claimed that some advanced security verification techniques are able to find such flaws by chance early in the design lifecycle.[102] However, rather than being used in an evaluation approach, such techniques are more likely to assist in design and are probably not available to third parties. For hardware products with commercial IPs, it is extremely difficult to apply security assessment to the microarchitecture because of the need to preserve commercial secrets.

7 Hardware monitoring technologies

7.1 Overview

Hardware monitoring technology began in the computer boom period in the 1980s. It was first used to assist with debugging and later developed into applications in various fields. However, with the rapid development of computing hardware and networks, especially the emergence of multicore processors and complex systems, including multicore processor systems, hardware monitoring technology has also undergone tremendous changes. While the fast-developing software and hardware environment has led to complex application functions, it also faces increasingly complex security challenges. In cloud computing-based systems, runtime stability and security are highly important, as they provide online services nonstop. The unique runtime characteristics of hardware monitoring technology give it a natural advantage in coping with these scenarios.

Hardware monitoring has made considerable progress in academic fields and industrial applications. It is widely used in/for the following areas/purposes:

— security

— debugging and testing

— performance analysis, evaluation, and optimization

— system fault tolerance and reliability

— physical parameter measurement and early warning

Although the focus of this document is hardware security assessment, monitoring techniques used for other purposes have implications for building assessment models. For example, the technology used for commissioning and reliability analysis is similar to the technology used for replay in the safety assessment process; the technology used for performance analysis has some consistency with runtime Trojan-based hardware detection technology. Therefore, in this document, keywords such as “debug” and “performance” are widely used in literature searches.

7.2 Research in academic areas

On the academic side, different studies have reviewed monitoring technologies from different perspectives.

Ian Cassar et al.[1] divided runtime monitoring instrumentation techniques into offline and online categories. Detailed online segmentation ranges from tightly coupled completely synchronous (CS) monitoring instrumentation approaches, to loosely coupled completely asynchronous (CA) monitoring approaches.

Heidar Pirzadeh et al.[2] divided monitoring technologies into software and hardware monitoring technologies and subdivided software monitoring technologies into add-on monitoring, manual instrumentation, online instrumentation, instrumenting compilers, interpreter instrumentation and OS instrumentation. In terms of security, cost, flexibility intrusiveness, performance and broadness, various monitors were compared horizontally.

Lihua Gao et al.[3] and Frank Cornelis et al.[4] separately subdivided software runtime monitoring technology. Reference [3] classifies and discusses the different levels of monitoring objectives (functional, module, architecture, and subsystem), while Reference [4] focuses on non-deterministic events and addresses the needs of various technologies for external resources (time, order, language, etc.) for subdivision comparison.

Georgios Kornaros et al.[5] focused on on-chip monitoring technology in a multicore SoC system and discussed the monitoring technology in detail from the perspective of function and methodology.

7.3 Industrial cases

In terms of industrialization, mainstream chip manufacturers and IT service providers also use hardware monitoring technology in their products.

Since 2013, Intel®[1] has introduced technology in commercial processors that can be enormously helpful in debugging because it exposes an accurate and detailed trace of activity and has triggering and filtering capabilities to help with isolating the important traces.

AMD®[2] has provided a similar technology for monitoring and controlling processors. This custom-built tool is designed specifically for a proprietary line of devices, thus indicating that the hardware and utility designers work together to provide the best service for their products.

ARM®[3] designed a set of utilities, including various trace macrocells, system and software measurements for the ARM processor, and a complete set of IP blocks, to debug and trace the most complex multicore SoC.

There are also hardware monitoring programs that are used to read the main health sensors of PC systems: voltages, temperatures, powers, currents, fan speed, utilization, and clock speeds during runtime.

Hardware security is a complex concept. The types of hardware are very complex. Figure 1 shows a comprehensive description of hardware monitoring technologies from five perspectives: target entity, purpose, carrier, objective patterns and deployment.

1) This tradename is provided for reasons of public interest or public safety. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC.

2) This tradename is provided for reasons of public interest or public safety. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC.

3) This tradename is provided for reasons of public interest or public safety. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC.

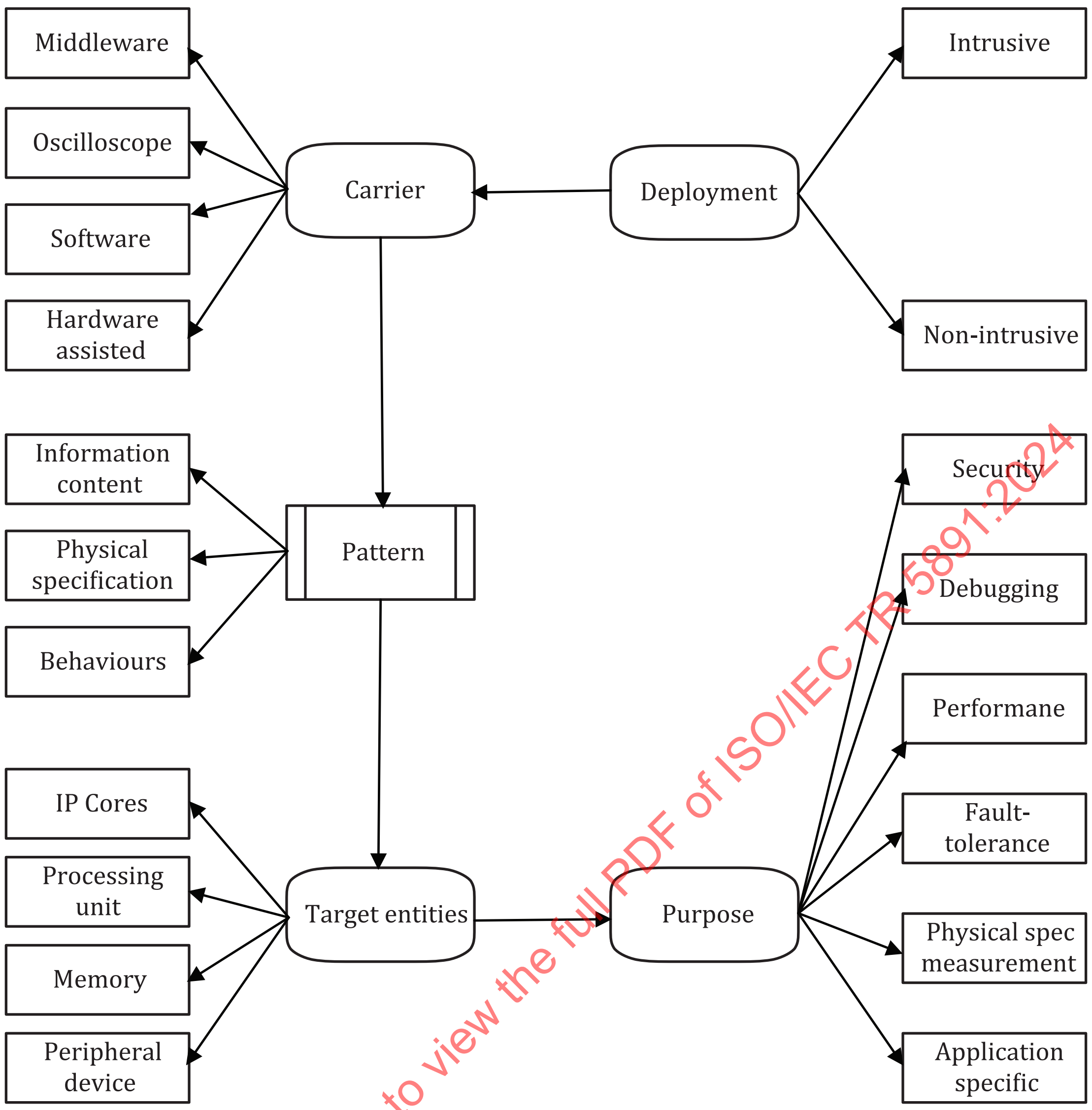

Figure 1 — Taxonomy of hardware monitoring technologies

7.4 Purpose

7.4.1 Security

The application of hardware monitoring technology for security purposes mainly aims to compare whether the running behaviour matches expectations. Then, security control is performed based on the matching results. Security control can terminate the operation of the system, re-execute the target module, or only record the current events and status to provide offline analysis or security audits.

According to the various monitoring objectives, different monitoring methods are adopted. From the scope of division, these methods can be divided into those that monitor a certain key hardware in the system and those that help ensure the safe operation of the entire system. The security of critical hardware can be divided into the implementation monitoring of malicious Trojan horses and the vulnerability security protection of hardware operation mechanisms.

Architecture monitoring support for security usually focuses on achieving tamper resistance and encryption. Some built-in on-chip technologies, such as control-flow integrity (CFI),[6] can assist users with high-performance integrity verification. They focus on the anti-attack capability of the core hardware itself. Some technology[7] can help defend against control-flow hijacking malware, for example, indirect branch tracking and shadow stacks. The integrity and timeliness of the CFI shadow stack itself is also a key focus of attention.[8]

For detecting Trojans, some approaches rely on a golden chip through a built-in sensor[9] to monitor the physical characteristics of some key circuits. If an abnormal voltage or current is found, it is identified as Trojan horse behaviour. In contrast, some approaches do not require a golden chip. Relying on existing knowledge, these approaches extract the typical characteristics of hardware Trojans and perform pattern recognition by using RTL or other hardware descriptions such as netlist.[10][11] Some state-of-the-art approaches use machine learning technologies to detect hardware Trojans from the Netlist or side-channel information of the hardware.[12-15]

The monitoring of the overall system focuses on the upper-level data and event logic and provides security monitoring in a way that is easier for users to understand. This monitoring approach is more conducive to leading a wider range of non-professional users to take effective security measures. Sentry[16] is a system which collects data through an internal bus connected to the system to determine attack behaviour and acts as a data filter between the internal system and the external system, providing effective security control.

Hardware monitoring technologies can also leverage real-time anomaly detection algorithms. These sophisticated tools are capable of tracking deviations from expected behaviour using statistical models, machine learning, or artificial intelligence to provide immediate alerts on potential security threats.

The following points are also important to consider for the security of hardware monitoring:

a) The role of hardware monitoring in the era of quantum computing also warrants attention. As quantum computers pose a threat to existing encryption methods, hardware monitoring can play a critical role in implementing and ensuring the effectiveness of post-quantum cryptography methods.

b) The Hardware Root of Trust (HRoT) is an essential element of hardware security. HRoT serves as a starting point for trust, ensuring the integrity and confidentiality of subsequent stages in a system's operation.

c) Monitoring Hardware Performance Counters (HPCs) can also provide valuable insight into the behaviour of the system and potential security breaches. Analysis of these counters can reveal abnormal patterns indicative of malware or hardware Trojans.

d) Use of best practices for hardware monitoring would ensure consistent measurement and comparison of hardware security across different systems, promoting broader and more effective security practices industry wide.

7.4.2 Debugging

Hardware debugging, which can monitor the runtime behaviour of hardware, is a hardware implementation monitoring method. Monitoring and debugging computing, especially of a single processor, has become a mature field of development tools and technologies.[17][18][19] The JTAG (Joint Test Action Group) interface is a common standard approach that is used to verify the design and test the functions of an integrated circuit. Some well-known processor providers have developed their own tracking functions. JTAG uses boundary-scan technology, which enables engineers to perform extensive debugging and diagnostics on a system through a small number of dedicated test pins. Signals are scanned into and out of the I/O cells of a device serially to control its inputs and test the outputs under various conditions.

In some Intel processors, breakpoints can be set on branches, interrupts and exceptions, and single-step debugging and analysis can be performed from one branch to the next. The enhancements of such processors make it possible to create last branch record (LBR) and branch tracking storage (BTS) mechanisms. Processors based on the Intel Core™4) microarchitecture also support precise event-based sampling (PEBS), which stores a set of additional status information for precise event monitoring.

4) Core is the trademark of a product supplied by Intel. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

There are several hardware approaches for debugging. If the erroneous behaviour is investigated during the development of a device, then searching for bugs in silicon is also referred to as post-silicon validation and debug, or just post-silicon validation. Design-for-debug (DFD) techniques are used to localize functional bugs through logic probing, scan chains, and real-time trace collection and are commonly referred to as embedded logic analysis methods. Various approaches address debugging support frameworks and interface standardization for SoCs[20][21] and debug structures for instrumentation, such as assertion-based on-chip debug checkers, triggers, and event counters.[22][19] Typical post-silicon validation techniques are applied to record the values of the circuit's internal signals into an on-chip trace buffer and feed the obtained values into a simulation framework to reproduce the erroneous behaviour.[23] Due to the vast amount of debug data, it is more efficient to record higher-level information such as instruction footprints for processors.[24]

Even though the power of simulators continues to improve linearly, the inability to adequately simulate all the possible permutations in the pre-silicon stage has led to alternatives, such as emulation and FPGA-based prototyping before design completion. Chip instrumentation is the key innovation that has enabled more efficient observation and verification of sustained functional integration combined with faster internal clock speeds and complex, high-speed I/O. Moreover, runtime debugging is also becoming increasingly important for multicore systems, especially for detecting race conditions or deadlocks, requiring architectural enhancements and tool support.

7.4.3 Tuning performance

Performance monitoring is one of the key goals of computing system design. Performance monitoring objects include execution time, cache hit/miss, CPU and memory occupancy, and the read and write performance of key storage devices. Reference [25] divided performance monitoring into four categories: software, hardware, hybrid and built-in on-chip performance counters. The comparison proved that the on-chip counter is the most efficient. Currently, all mainstream processors provide performance counters. However, although the hardware is updated very quickly, the complexity of the system itself also increases geometrically. The performance counters cannot be used to record all events. Therefore, filtering core data and events becomes key. Reference [26] first proposed a novel hybrid counter array architecture based on SRAM arrays and discrete counters, which can track many events concurrently.

Reference [27] divided performance profiles into two categories: time-based profiles and event-based profiles. A time-based profile relies upon interrupting an application's execution at regular time intervals. To support event-based sampling (EBS), performance-monitoring hardware typically generates a performance monitor interrupt when a performance event counter overflows. Alan Mink et al.[28] proposed a hybrid performance measurement system that collects data and event samples by embedding hardware on the motherboard; this system can be used for monitoring multiprocessor performance.

For the multicore SoC platform, Reference [29] presented a performance monitoring unit (PMU) for the SoC on-chip bus. The PMU can measure major performance metrics, such as bus latency for specific leader requests and the amount of memory traffic in specific durations. The PMU can also measure the contention of the bus leaders and followers in the SoC. In addition to studies on performance monitoring for the key hardware in a system, there have been studies on performance monitoring of an overall system. Reference [30] divided system monitoring into three steps: system monitoring, system modeling and system improvement. The authors implemented a tool called Charmonto to monitor software and hardware characteristics and used it for long-term monitoring. Charmonto can run continuously while sampling various types of hardware metrics through performance monitor counters (PMCs)[31] with certain frequencies.

7.4.4 Fault tolerance and QoS

Ensuring system reliability involves multiple abstraction layers, from circuits and microarchitectures to algorithms and application layers. For this reason, various monitoring methodologies have been developed across these layers to provide runtime self-diagnosis, adaptivity, and self-healing. The majority of all mechanisms assume asymmetric reliability, taking the conservative view that some components are almost fault-free, and thus, these mechanisms are expected to protect individual system components such as buses, network-on-chip, memories and datapaths against transient or permanent errors. Additionally, monitoring and diagnostic units have modest performance requirements, so they can operate at low voltage and frequency, and they usually use aggressive built-in redundancy, making them effectively immune to failure.

[32] However, centralized monitoring schemes represent dependability exposure due to their single points of failure.

Online monitors aim to extract metrics for accurate determination of ageing models, performing complete reliability analysis[33] or providing the reliability characteristics of components. Future architectures contain components that exhibit varying dependability characteristics, such as processor cores with different arithmetic precisions and memories with different reliability guarantees. Applications will be allowed to trade off high hardware reliability for high performance through switching modes in mixed-mode multicore systems with reconfigurable dual-modular redundancy.[34] Moreover, by using reliability annotations, compilers can create a mapping that ensures the assignment of reliability-critical code and data objects to the most reliable components of a system, or operating systems can perform proactive, reliability-driven thread migration and shadowing (e.g. shadow threads can be assigned to dependable cores with low thermal stress).

Many modern embedded and distributed systems (including real-time and non-real-time systems) employ utilization monitors, rate modulators, or model predictive or workload controllers with feedback control techniques to ensure quality of service.[35][36] In response to workload variations in unpredictable environments, dynamic resource allocation can be used to achieve high processor utilization while still meeting real-time constraints, to enforce appropriate schedulable utilization bounds, or to handle the system dynamics caused by load balancing for large-scale server clusters. From a different angle, monitors help to avoid saturation of processors, which can cause a system crash or severe service degradation. However, these management approaches based on feedback control require an accurate system model, which can be difficult to obtain for realistic distributed or multicore embedded systems.

Monitoring for sustaining QoS can be crucial for particular application domains. For instance, servers providing adaptive video streaming services exploit the inherent adaptiveness of video applications to perform controlled and graceful adjustments to the perceptual quality of the displayed MPEG video stream in response to fluctuations in the QoS delivered by the underlying infrastructure. Increased efficiency is attained when the monitoring service is not platform agnostic. In embedded system design, where the platform cost and its resources are bounded, quality monitoring and control assisted by the system and in synergy with the applications is inevitable.[37]

An individual monitoring component could provide metrics related to QoS to the higher abstract layers of the system. This information is utilized by the higher layers to analyse and react to QoS variations, for example, switching to the redundant or secondary functional block or running in backup mode. For additional safety critical applications, individual models can make decision based on their QoS Tolerances.[41]

7.4.5 Physical specification measurement

As power densities increase with continuously shrinking geometries and large temperature variations can occur on the chip and are expected to be avoided or at least mitigated. Monitoring mechanisms play a fundamental role in building adaptive chip architectures that struggle to achieve the theoretical highest performance within a thermally safe dynamic voltage and frequency scaling (DVFS) configuration for specific or varying workloads.

Modern microprocessors always have built-in digital temperature sensors that allow for real-time temperature monitoring. These sensors are in each individual processing core, near the hottest part. The data collected from these sensors is very accurate as it does not rely on an external circuit located on the motherboard to report temperature. For example, recent Intel microprocessors contain at least one per-core temperature sensor, often many additional ones as well. More than 16 temperatures have been reported across the die.

There are also many research efforts in this domain, since the design space exploration for runtime power and temperature management involves a very large number of parameters. There are several approaches for a single processor and some for the multicore era.[38][39][40][41] The main goal is to fit the best monitoring strategy in terms of efficiency to the right chip, system, and environment.

7.4.6 Application-specific monitoring

Application-specific monitoring involves mechanisms that are developed to satisfy any of the objectives discussed thus far but are further customized to consider the behaviour of a particular application. For instance, this category includes monitoring techniques for online exploration of the best performance/energy parameter values that can be customized related to an application's memory usage characteristics. Runtime monitors and managers have proposed extensible processors with customized instruction sets that utilize the most appropriate special instruction for a multi-media video encoder[44]. The main idea of this online monitoring approach is to estimate the quality of usage of the special instructions that are reconfigured in the processor datapath. The method employs counters to reflect the execution counts of these custom instructions for each iteration of a computationally intensive loop.

Moreover, in embedded system development, application-specific instruction-set processors (ASIPs) have gained popularity because they combine the flexibility of software with the energy-efficiency and scalable computational performance of dedicated hardware implementations. By modifying the development methodology for ASIPs, monitoring can enhance the observability and adaptation capabilities of the system. For instance, Reference [45] describes a methodology for monitoring routines to check insecure operations through the addition of extra microinstructions within vulnerable machine instructions. ASIPs can be synthesized and targeted for different applications, which can employ different security monitoring methods (instruction memory data bus monitoring, data path monitoring, return address stack monitoring and branch instruction monitoring).

For performance critical applications, customized objectives for application or application-specific sub-modules can leverage one or more of the existing monitoring metrics. For instance, QoS can be a metric of natural robustness and the performance of the system. A drop in QoS can be a direct impact of a security breach. Such composite objectives can be considered application specific and not generalized.

7.5 Carrier type

7.5.1 Middleware

7.5.1.1 In cells

Among the strategies to detect transient faults caused by radiation or optical sources, bulk built-in current sensors (BBICSs) offer a solution that is suitable for system design flows based on the CMOS standard cells of commercial libraries. BBICSs have the high detection efficiency of costly fault-tolerance schemes (e.g. duplication with comparison) with the low area and power overheads of less efficient mitigation techniques, such as time redundancy approaches.[46][47][48][49] Without impacting the system operating frequency, BBICSs address transient faults of short and long durations and multiple faults. Furthermore, as BBICSs closely monitor the zones where faults arise, early detection is possible immediately after fault occurrence, thus preventing the induction and propagation of errors to other clock cycles or system blocks.[50]

7.5.1.2 Attached hardware components inside

Reference [51] describes MAMon, a monitoring system that can monitor both the logic level and the system level in single-/multiprocessor SoCs. A small hardware probe unit is integrated into the SoC design and connected via a parallel port link to a host-based monitoring tool environment.

Reference [52] describes RG-Secure, which combines the third-party intellectual property trusted design strategy with scan-chain netlist feature analysis technology, and is equipped with a distributed, lightweight gradient lifting algorithm called lightGBM.

To enable full-system deterministic replay of multiprocessor executions, Flight Data Recorder (FDR)[53] adds modest hardware to a directory-based sequentially consistent multiprocessor.

7.5.1.3 Filter hardware outside

In contrast to the hardware embedded in the system, a monitor can be located outside the system and is used to filter and monitor whether the system communicates with the outside world or whether the interactive behaviour is compliant. The advantage of the monitor being external is that it does not require large-scale transformation of the original system. It is convenient to evaluate and monitor the high-level behaviour of the system and to report security issues in a way that users can better understand.

Typical methods include TrustGuard.[54] Security and privacy assurances in TrustGuard are founded on a pluggable and simple hardware element, called the Sentry. With an improved Sentry model,[16] TrustGuard now allows output only from the correct execution of signed software. The Sentry in TrustGuard checks:

a) the correctness of instruction execution with respect to the ISA (including determining the next instruction in program order); and

b) whether each instruction is part of a signed program.

Policy dictates what to do upon the detection of an incorrect execution. TrustGuard supports many different incorrect execution policies, including halting execution, continuing execution while preventing erroneous output, alerting the user, and reporting the incorrect instruction. This proof-of-concept supports a system with a single-core processor running software, including an OS, signed by a trusted authority. Additionally, TrustGuard requires that all I/O originates from explicit I/O instructions.

7.5.2 Software

There are an increasing number of types of hardware, and hardware structures are becoming increasingly complex. A hardware monitor can be combined with a software monitor to achieve better evaluation results. Therefore, this subclause provides a brief summary of software monitoring technology.

Software monitoring technology is divided into levels and can be sorted into the following categories:

a) Operating system layer, which provides monitoring APIs or system drivers.

Due to the permission restrictions of the operating system, upper-layer applications cannot directly access sensitive resources. The monitoring program calls the operating system-specific monitoring API, such as the method of Instant-Replay,[55] to obtain relevant operating information.

b) Application layer, which adds monitoring functions or embeds monitoring code.

Code is added to a normally executed function, or parallel processes or threads are started to perform interesting monitoring. For example, observation variables or path information recording, such as jumps, can be added. Software monitors used on the application layer include JaRec,[56] which is a platform independent record/replay environment for multi-threaded Java™5) applications implemented using Java Virtual Machine Profiler Interface (JVMPI).

c) Architecture layer, which builds a system architecture with monitoring capabilities.

Monitoring is considered during the construction of the system framework. Through the invocation process and usage specifications, the implementation of subsequent monitoring requirements is guided. Software monitors used on the architecture layer include ReVirt.[57]

The related documents are summarized in Table 1.

5) Java ™ is the trademark of a product supplied by Oracle. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named.

Table 1 — Classification of software monitors

Approach	Level	Nonintrusive	Single- or multicore processors	Description
Halsall-Hui[58]	Architecture	No	Multicore	This monitor is designed to gather the data from each processing node of a real-time embedded system that is based on a distributed architecture.
ReVirt[57]	Architecture	Yes	Multicore	ReVirt can replay the complete, instruction-by-instruction execution of a virtual machine, even if that execution depends on non-deterministic events, such as interrupts and user input. An administrator can use this type of replay to answer arbitrarily detailed questions about what occurred before, during, and after an attack.
BugNet[59]	Architecture	Yes	Multicore	The BugNet architecture is focused on continuously tracing program execution. BugNet focuses on deterministically replaying the instructions executed in user code and shared libraries.
Instant Replay[55]	OS	Yes	Multicore	Instant Replay is a technique to replay shared memory accesses by using an ordering-based approach. This technique restricts the program to accessing shared memory objects only through well-defined CREW (concurrent-reader-exclusive-writer) protocol primitives, which Instant Replay uses for the execution replay.
Interrupt Replay[60]	OS	No	Multicore	Audenaert and Levrouw described a method for replaying parallel programs on bus-based shared-memory multiprocessor systems that use interrupts. This monitor is used as an extension to Instant Replay.
HMON[61]	OS	Yes	Multicore	HMON was developed to monitor the performance of the Hexagonal Architecture for Real-Time Systems (HARTS), a distributed real-time system.
IGOR[62]	Application	No	Single-core	Based on checkpointing techniques, IGOR is a system for replaying programs. Given a checkpoint, IGOR reconstructs the state of the program at that point and allows the program to continue its execution. IGOR does not record external I/O, so the replayed execution can differ from the original execution if the environment has changed. Additionally, IGOR does not handle non-determinism caused by multi-threaded programs.
JaRec[56]	Application	No	Multicore	JaRec operates entirely on the Java-bytecode level. Each class that is loaded by the JVM (Java Virtual Machine) is transferred through the JVMPI (JVM Profiler Interface) to an instructor, after which the instrumented class is loaded.

[a] This trademark is provided for reasons of public interest or public safety. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC.

Table 1 *(continued)*

Approach	Level	Nonintrusive	Single- or multicore processors	Description
jRapture[63]	Application	Yes	Single-core	jRapture [SCFP00] is a tool designed to capture interactions between a Java application and the underlying system, i.e. interactions with the user interface, files, keyboard. During the replay phase, jRapture presents the executing threads with the same input sequence they received during the record phase.
Recap[64]	Application	Yes	Multicore	Recap is a tool that provides the illusion of reverse execution for parallel programs. Recap uses a combination of checkpointing and replay to provide the user with an illusion of reverse execution.
DejaVu[65]	Architecture	No	Single-core	DejaVu is an execution replay infrastructure designed at IBM®[a] for addressing the non-determinism caused by threads and other related concurrent constructs in Java applications.

[a] This trademark is provided for reasons of public interest or public safety. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC.

7.5.3 Hardware-assisted monitors

Hardware-assisted monitors are technologies or systems for debugging, performance tuning, fault tolerance, or meriting power, energy and temperature. They are not designed on purpose of security assessment.

Table 2 lists the main literature related to hardware-assisted monitors.

Table 2 — Classification of hardware-assisted monitors

Approach	Purpose	Carrier	Target Entity	Objective Patterns	Deployment Method	Description
FDR[53]	Debugging	Middleware	Multicore	Behaviours	Nonintrusive	Similar to an aircraft flight data recorder, FDR continuously records the execution in anticipation of a trigger, which can be a fatal crash or a software assertion violation.
SafetyNet[66]	Debugging	Middleware	Multicore	Behaviours	Nonintrusive	SafetyNet periodically checkpoints the system state to allow the system to recover its state to a consistent previous checkpoint. If a fault is detected, SafetyNet recovers the state to the recovery point, which is the old checkpoint most recently validated as fault-free.

[a] Jintide® is the trademark of a product supplied by Montage. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

Table 2 *(continued)*

Approach	Purpose	Carrier	Target Entity	Objective Patterns	Deployment Method	Description
RnR-Safe[67]	Security	Middleware	Single-core	Behaviours	Nonintrusive	RnR-Safe complements hardware security features to offload intrusion checks and/or to eliminate check imprecision. RnR-Safe reduces the cost of security hardware by allowing the hardware to be less precise at detecting attacks, thus potentially reporting false positives65.
PASM[68]	Debugging	Middleware	Multicore	Information and Behaviours	Intrusive	PASM is a programmable hardware monitor that provides a flexible set of tools for the user to specify events for a wide variety of monitoring applications. The user can include with the application a monitoring section that defines events of interest, actions to be executed upon detection of these events, and the binding of events to actions.
ART[69]	Debugging	Middleware	Multicore	Behaviours	Intrusive	This approach focuses on visualizing the timing behaviour of the system processes. Rate monotonic and deferrable server algorithms are supported by this monitor, and the monitoring task is performed as part of the target system.
ZM4[70]	Debugging and Performance	Middleware	Multicore	Behaviours	Nonintrusive	In this approach, a hardware system called ZM4 and an event trace processing software called SIMPLE, which works independently from the monitor, are developed. The connection between the hardware and the monitored system is a local area network type.
Trams[28]	Debugging	Middleware	Multicore	Behaviours	Intrusive	The architecture of this system consists of software for event triggering that inserts write commands in the code and a hardware subsystem used to sample the time and identity of the CPU.
SoC-based[71]	Debugging, Performance, Fault-tolerance and Security	Middleware	Multicore	Behaviours	Nonintrusive	This system-on-a-chip-based monitor uses a hybrid method for run-time verification of embedded systems. The monitor consists of an event recognizer, a verification tool, and the monitor output. The event recognizer decides whether the collected data are relevant to the event definition. After passing this step, the event data are sent to the verification section, where they are compared with the requirement constraints.

[a] Jintide® is the trademark of a product supplied by Montage. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

Table 2 *(continued)*

Approach	Purpose	Carrier	Target Entity	Objective Patterns	Deployment Method	Description
MAMon [51]	Debugging	Middleware	Multicore	Behaviours	Nonintrusive	MAMon is a monitoring system for hardware-accelerated real-time operating systems.
Roadnoc[72]	Observability, Buffer utilization of NoC routers	Middleware	Multicore	Behaviours	Nonintrusive	Roadnoc is a counter-based monitoring component attached to a router.
ADAM[73]	Optimization: Task Mapping on an NoC	Middleware	Multicore	Behaviours	Nonintrusive	ADAM performs runtime application mapping on an NoC in a distributed manner by using an agent-based approach.
Vicis[74]	Reliability for NoC	Middleware	Single-core	Behaviours	Intrusive	Built-in self-testing at each router diagnoses the number and locations of hard faults, and this method includes ECC, a crossbar bypass bus, and port swapping.
NUDA[75]	Debugging (race detection)	Middleware	Multicore	Behaviours	Nonintrusive	NUDA is a nonintrusive debugging framework for many-core systems that operates in parallel to the original data interconnection, thus enabling “non-intrusive” debugging methods.
HARD[76]	Debugging (race detection)	Middleware	Multicore	Behaviours	Nonintrusive	The lockset algorithm is used in hardware to exploit the race detection capability of this algorithm by using bloom filters.
Non-INTER[77]	Debugging and Performance	Middleware	Multicore	Behaviours	Nonintrusive	Based on a non-interference monitoring architecture, non-INTER monitors the execution of distributed real-time systems without interfering with their execution by using additional hardware to collect state information and assist in monitoring.
TrustGuard [54]	Security	Middleware	Multicore	Behaviours	Nonintrusive	TrustGuard makes the Sentry compatible with a wide range of devices without requiring hardware modifications to the host system and gives developers the option to use trusted code to verify the execution of untrusted code, thus reducing the size of the trusted code base.
Jintide®[a] [78]	Security	Middleware	Multicore	Behaviours	Nonintrusive	Jintide® monitors Xeon cores at runtime with a reconfigurable computing processor (RCP) chip. Jintide® captures particular features by recording and analysing the microarchitecture-level behaviours that are software transparent.

[a] Jintide® is the trademark of a product supplied by Montage. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

Table 2 *(continued)*

Approach	Purpose	Carrier	Target Entity	Objective Patterns	Deployment Method	Description
Side-channel based approaches[14,15]	Security	Hardware and Software (Optional)	Single-/Multicore	Behaviours	Nonintrusive	These approaches monitor hardware behaviours by using side-channel information to detect hardware Trojans.
[a] Jintide® is the trademark of a product supplied by Montage. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.						

7.5.4 Software vs. hardware-assisted solutions

Detecting hardware behaviour by using software has many limitations:

a) The acquisition software cannot record every memory access of the CPU and can only use page granularity, thereby incurring a large performance overhead and memory space overhead, greatly limiting the use range.

b) The acquisition software also cannot record the DMA operation of a transparent transmission device and can perform accurate event recording only on an analogue device.

c) On the playback side, using different architectures causes false positives, and using the same architecture leads to the problem of insufficient confidence.

Software can be combined with hardware modules to achieve higher confidence with higher performance. That is, the software collects the CPU status, the hardware records memory and IO access, and then implements a hardware monitoring framework through other hardware cores running instruction simulators.

7.6 Target entity

7.6.1 IP cores

7.6.1.1 General

The security of IP cores comes from two aspects:

— To maintain their commercial credit, IP vendors provide IP anti-counterfeiting solutions to verify their legitimacy.

— Users of IP cores ensure that the embedded third-party IPs will not bring safety risks.

7.6.1.2 IP anti-piracy

IP anti-counterfeiting solutions can be roughly divided into four strategies: obfuscation, watermarking, fingerprinting and metering. IP anti-counterfeiting can be used only for IPs originating from legitimate suppliers, and it is based on the supplier's own business reputation and verification methods. Therefore, as a user, in the case of high security requirements, a hardware monitor can be added to verify the compliance of the third-party IP.

7.6.1.3 IP compliance monitoring

Third-party IPs are often integrated into the SoC as a core security function, such as an encryption engine. Their own security directly determines the security level of the overall system. There are also potential risks that malicious functions, e.g. hardware Trojans, can be inserted into IP cores. In particular, due to cost and time constraints, IP cores without a complete authentication chain are used in many scenarios. Therefore, ensuring the safe operation of IP cores, the absence of a security backdoor and hardware monitoring technologies that guarantee the absence of malicious functions are very important.

7.6.1.4 Hardware integrity assurance

In the scope of post-silicon hardware operation, methods such as runtime attestation become crucial to ascertain that the hardware components have not been illicitly altered during their execution cycle. This approach forms a part of the hardware monitoring strategies which ensure the integrity of third-party IPs, effectively supplementing the anti-counterfeiting solutions provided by IP suppliers. Such a verification approach becomes paramount in high-security applications where any tampering can lead to a catastrophic failure of the overall system.

7.6.1.5 IP core verification and validation

IP core verification and validation mechanisms take centre stage in establishing the functional correctness, performance, and security characteristics of third-party IPs. These mechanisms can include formal verification methods, simulation, and hardware emulation. Much like the anti-counterfeiting solutions, these procedures depend heavily on the verification methodologies provided by the legitimate suppliers. They ensure the operational safety of the IP cores, preventing any potential security backdoors that can be present.

7.6.1.6 Runtime monitoring of IP cores

Real-time hardware monitoring provides a continuous assurance of the secure operation of IP cores. Technologies capable of capturing and analysing real-time data are particularly valuable in this context. They can observe variations in power consumption, timing analysis, or performance metrics to identify any unusual activity that can signify a security breach. Given that IP cores often fulfill core security functions within SoCs, such as encryption engines, runtime monitoring can be critical in maintaining the security of the overall system, particularly when the authentication chain of the IP cores is incomplete.

7.6.2 Processing units

Processing units can be divided into single processors, multiprocessors and networks on a chip (NoCs) according to the number of cores. Single-processor monitoring is found mainly in the early literature.[67] With the rapid development of chip manufacturing and the rise of cloud computing, the Internet of Things and other fields, many studies have concentrated mainly on multiprocessors[18][29] and NoCs[79] to solve debugging, fault tolerance and security issues in multicore scenarios.

7.6.3 Memory

Memory includes two types: volatile and non-volatile. A representative of volatile memory is RAM. Due to the read-only nature of ROM, non-volatile memory is discussed only in terms of SoC components (Flash/EEPROM, etc.). Non-volatile memory (such as a hard disk or USB) outside of the system is discussed in terms of peripheral devices.

Random-access memory (RAM) is volatile random-access memory with high-speed access and equal read and write times. It is not related to addresses. RAM can be divided into static RAM (SRAM) and dynamic RAM (DRAM). SRAM stores the temporary data generated during the operation of a single-chip microcomputer. Its speed is very high. SRAM is currently the fastest storage device for reading and writing. SRAM is often used for high-speed buffer memory, such as the primary and secondary buffers of the CPU; DRAM retains data for a short time. DRAM is slower, cheaper and has more computer memory than SRAM.

Monitoring SRAM involves focusing on two main areas: the CPU cache and the SRAM used in the SoC. RG-Secure obtains cache information by monitoring the data of the bus controller so that it can process the dynamic data of SRAM in multicore scenarios. FDR[53] identifies memory race by embedding a special field (VIC/CIC) in the cache.

DRAM monitors are divided into two realizations: software and hardware. SMM[80] is a software implementation that provides detailed information and a graphical view of the running programs inside the computer. Through hardware implementation, SMM can provide better performance and safety. Jintide [78] uses special hardware, namely, MTR, to efficiently collect DRAM data. Jintide can collect only the memory that changes between checkpoints as needed, thereby greatly reducing the log size.

7.6.4 Peripheral devices

Peripheral equipment is divided into two categories: input and output. These devices handle data or signal input/output and include non-volatile storage devices (such as data carriers) and input/output devices for non-deterministic events (such as user keyboard input and interrupt/abnormal signals).

The monitoring of peripheral devices is performed directly or indirectly by relying on the I/O interface (I2C/SPI/PCIe, etc.) provided by the system boundary. Direct acquisition methods include collecting peripheral bus data. Indirect methods include using the monitor function provided by the hypervisor technology to obtain interrupt and data information.

FDR[53] logs DMA by modeling DMA interfaces as pseudo processors and applying memory-race logging techniques. Jintide [78] uses dedicated hardware ITRs to collect I/O information.

A hypervisor is also known as a hosted VMM, since a hypervisor runs as an application in a normal operating system known as a host operating system. The host OS does not have any knowledge about the VMM and treats it like any other process. The host OS typically performs I/O on behalf of a guest OS. The guest OS issues the I/O request, which is a trap by the host OS. The host OS, in turn, sends the I/O request to the device driver that performs I/O. The completed I/O request is again routed back to the guest OS via the host OS.

7.7 Objective patterns

7.7.1 Information content

Information content-based monitoring focuses on the information flow between the internal components of the system or between the system and the outside, including communication-related timing and other protocol data and audit logs related to upper-level business logic. This type of monitor pays attention to the information itself and feeds back the operation of the system by monitoring the content.

The information related to the communication protocol can be obtained by using a logic analyser or oscilloscope, or directly using middleware by intercepting the bus. For business-related information, software can be used to record log information.

7.7.2 Physical specification

Monitoring the physical characteristics of the hardware involves obtaining the hardware's physical data, such as current, voltage, power consumption, electromagnetic emissions, computation time, and heat, in real time through a sensor, probe, or intermediate hardware. The main purposes of physical feature monitoring include threshold control, performance optimization, fault tolerance, QoS, and security monitoring (to detect hardware Trojans, etc.).

From the perspective of the supply chain, monitoring physical characteristics play an important role in every link. In the in-house design stage, monitoring physical characteristics can help determine the legitimacy of the third-party IP. In the test stage, monitoring physical characteristics can help verify that the key modules are slightly resistant to side channel analysis. In the post-silicon stage, monitoring physical characteristics can ensure a stable operating environment and excellent performance.

Physical features can be derived from golden values or can be dynamically adjusted by using machine learning and artificial intelligence algorithms.

7.7.3 Behaviours

Behaviour-based monitors emphasize logical compliance at a high level. For example, hardware Trojans can be detected to analyse the behaviour of the hardware. Unlike information content-based and physical specification-based monitors, behaviour-based monitors have an important runtime feature.

Hardware behaviour is restricted by the system environment in which the entity is located. Behaviour monitoring often involves an overall analysis of the entire system and does not limit important security components.

7.8 Deployment method

7.8.1 General

The deployment of monitoring involves consideration of:

a) intrusiveness;

b) whether the deployment is offline or online;

c) whether the deployment is synchronous or asynchronous; and

d) whether there is one or multiple monitors.

7.8.2 Intrusiveness

Intrusiveness is reflected in the impact on the time and space of the running target system. In some complex situations, poorly designed monitors can even bring operational errors and security risks. Considering the cost, lower intrusiveness is one of the main goals of monitor design. The advantage of a software-based monitor is that it can provide good flexibility. Where there are sufficient resources, a monitor parallel to the target system can show good non-intrusiveness. However, in the case of embedded systems, pure software often results in considerable overhead. The advantage of hardware-based monitors is that with the good performance of parallel monitor hardware, they can provide lower intrusiveness. However, compared with software solutions, hardware solutions involve more development funds and cycle investment, and flexibility is also limited. Configurable hardware-assisted solutions based on FPGAs consider the advantages of flexibility and low overhead to a certain extent. Balancing cost, flexibility, and low intrusion can be a problem at the monitor deployment phase.

7.8.3 Offline or online

Offline methods can result in lower intrusiveness than online methods because the processing time can be delayed. However, unlike online solutions, offline solutions involve store sampling or tracing records, thereby introducing additional space overhead. Online deployment methods are more suitable for real-time systems. When monitoring abnormal behaviours, this method can provide timely security control. However, the online approach either increases costs or brings greater overhead.

7.8.4 Synchronous or asynchronous

To balance the advantages and disadvantages of offline and online methods, an asynchronous monitor can be used to achieve a certain balance between time lag and detection timeliness. Compared with a synchronous solution, an asynchronous solution brings some delay, but as long as the delay is within the acceptable range of the system, the asynchronous monitor solution is a better solution with lower intrusiveness.

7.8.5 Single or multiple monitors

To date, the strategy of centralized processing with a single monitor[80] is more common, but with the increasing complexity of SoC and NoC systems, distributed multimonitor solutions have gradually emerged. Reference [81] divides multimonitor solutions into two categories: orchestrated and choreographed monitors. Orchestration relies on a single coordinating entity to gather, order and analyse events, whereas a choreographed approach disseminates these tasks across multiple monitors.[82][83] While orchestration is typically simpler to synthesize and thus easier to obtain correctly, choreography is more attuned to the characteristics of distributed computing, thus leading to lower network traffic and a higher degree of fault tolerance.[84]

7.8.6 Scalability

Scalability is a critical consideration for hardware monitoring systems as the complexity and monitoring requirements of modern systems continue to grow. A monitoring solution that lacks scalability can become inadequate to handle the increasing number of components, devices, or data points that need to be monitored. To maintain scalability, it is vital to design hardware monitoring systems with modular architectures, as this will facilitate easy expansion and adaptation. For instance, Reference [42] presents a scalable hardware monitoring framework that leverages distributed data collection and processing to accommodate monitoring requirements across large-scale distributed systems.

7.8.7 Resilience and redundancy

The design of hardware monitoring systems prioritizes resilience and redundancy to provide continuous monitoring, even when faced with failures or disruptions. Downtime or malfunctions in the monitoring system can result in the loss of critical data, which can have serious repercussions. A viable approach for this situation involves implementing redundant data storage and failover mechanisms. As exemplified in Reference [43], this method utilizes a redundant monitoring architecture to maintain uninterrupted data collection and analysis.

7.8.8 Compatibility

Seamless integration of monitoring solutions often rests on their compatibility with existing hardware and software systems. The design of these systems ideally facilitates interactions with a wide array of hardware configurations and software environments, thus avoiding any conflicts or disruptions. An instance of this can be found in Reference [95], which elaborates on a System-on-Chip (SoC) architecture integrating multiple processors with other memory blocks. Such a design enhances the system's adaptability to various hardware configurations, promoting effortless deployment in a variety of systems.

7.8.9 Impact on performance

The effect of hardware monitoring on the performance of the observed system is ideally kept as low as possible to prevent interference with its regular functionality. Monitoring systems are best designed with great precision to exert as little burden as possible on the resources of the targeted system. For instance, Reference [112] investigates methods for lightweight hardware monitoring which result in an almost non-existent impact on the CPU's performance, thus allowing for continuous monitoring without diminishing the responsiveness of the system.

7.8.10 Lawful and ethical data handling regulations and requirements

Hardware monitoring systems complying with legal regulations and ethical guidelines allows for lawful and ethical data handling. As data privacy and security become more prominent concerns, prioritizing the protection of collected data and its responsible usage becomes paramount. For instance, Reference [113] emphasizes the value of compliance with data protection regulations and the implementation of privacy-preserving techniques in hardware monitoring systems, which helps in maintaining the trust and confidence of users.

8 Utilizing monitoring technologies for hardware security assessment

8.1 Existing state-of-the-art security assessment approaches

The stages of existing hardware security assessment approaches can be classified into pre-silicon and post-silicon stages. The former stage is used to guide the fulfilment of specific safety requirements in the design phase, and the latter is used mainly to evaluate product safety risks.

The methods used in the pre-silicon stage can be divided into top-down and bottom-up methods that guide the hardware design phase to meet safety requirements. R. Huang et al.[85] proposed a hardware security assessment scheme that provides a systematic way of measuring and categorizing the security concerns of hardware features. The proposed scheme focuses on security measurements in the early design stage by utilizing the experiences of security experts and the design knowledge of feature owners. The assessment scheme employs a two-level questionnaire format that scores and categorizes security measurements.

B. Sherman et al.[86] proposed a bottom-up design approach, transitioning from a product-centric to an IP-based approach, thereby effectively improving the reuse of public components of various products. The outcome was the Security Risk Assessment (SRA) tool, which guides an IP working group in conducting security assessments.[86] The SRA tool divides the security assessment process into two layers: a high-level filter and a low-level assessment. The intent of the high-level filter is to quickly identify the IPs, and if there are no significant security concerns, they are dismissed as having minimal risk. The low-level assessment comprises a set of questions organized into focused topic categories, where each question is assigned a risk rating and weighted score.

Hasegawa et al.[12,13] proposed a machine-learning-based hardware Trojan detection method for gate-level netlists by using multilayer neural networks. The authors extracted 11 features for each net in a netlist. Then, the authors classified the nets in an unknown netlist into sets of Trojan nets and normal nets by using multilayer neural networks.

The post-silicon phase is the security assessment of the finished product. CC EAL (Common Criteria, Evaluation Assurance Level) relies on the CC specification to test the specific product level and establishes a mutual recognition mechanism as a safety guarantee for the passage of products. The CC series specifications define the overall information security framework with the two levels of security function and security assurance.

Different industries have different security assessment requirements for hardware security assessment according to their own characteristics. For example, the Society of Automotive Engineers (SAE) International is actively involved in developing series of standards, such as AS6171A.[104] The SAE has introduced a set of test methods for detecting counterfeit parts, by testing physical properties and electrical properties of different parts.[87] The SAE uses image tools and physical measurement tools to evaluate hardware security.

Because of the sensitivity of certain fields (aviation, military, civil livelihood, etc.), different countries have introduced native hardware assessment and certification requirements, such as NIST FIPS 140-2 (USA).[105] In contrast, ISO/IEC 19790 states that certified products can be used in specific departments. These certifications are tested and verified by authorized laboratories on samples submitted for inspection to match specific safety requirements one by one and yield a safety assessment report to be submitted to the authority for review.

8.2 How hardware monitoring can help

The principal target of a security assessment methodology analysing the sufficiency of countermeasures for security problems such as hardware Trojans or hardware flaws is to evaluate runtime hardware behaviour-based security. The evaluation is performed in discrete periods called monitored intervals.

For this purpose, the concept of runtime hardware-behaviour-based security is defined as a proactive approach to security for hardware in the post-silicon phase, in which relevant hardware runtime activity is monitored so that deviations from patterns of normal behaviour can be identified and addressed quickly.

Referring to Figure 2, the evaluation of runtime hardware security checks runtime behaviours to distinguish unauthorized or unexpected activities on running post-silicon hardware. Since runtime behaviours are a subset of all possible behaviours, and the behaviours activated at runtime are a subset of the behaviours activated at any moment, the evaluation of runtime security on the target post-silicon hardware can potentially enhance the confidence that risks from threats have been minimized, such as hardware Trojans or hardware flaws.

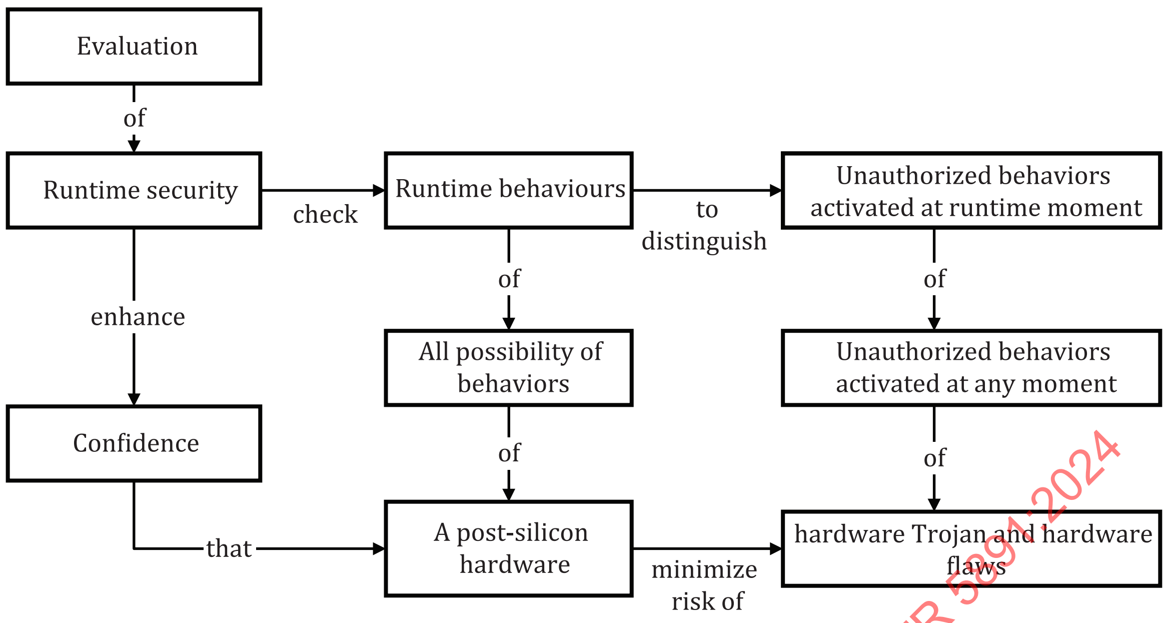
Evaluation
of
Runtime security
check
Runtime behaviours
to distinguish
Unauthorized behaviors activated at runtime moment
enhance
of
All possibility of behaviors
of
Unauthorized behaviors activated at any moment
Confidence
of
of
that
A post-silicon hardware
minimize risk of
hardware Trojan and hardware flaws

Figure 2 — Runtime hardware-behaviour-based security — concepts and relationships

An explicit characteristic of a hardware Trojan is that host hardware operations (behaviours) can result in an unexpected status against the original specifications when the Trojan is activated. For example, the operation can result in registers that violate the instruction manual or the protocol to generate a catastrophic-mistake attack, or an internal secret can be smuggled outside to pads with a bypass. However, the activity conditions and corresponding behaviours of most flaws referring to functional bugs and side-channel bugs can be consulted to obtain prior knowledge. For example, if a program is detected which is frequently measuring the latency of a memory address that would have only been visited by a CPU on speculative execution, it can be concluded that this is part of a CPU cache side channel attack.

At present, hardware monitoring technologies have been developed for many purposes. These technologies are widely used to collect dynamic information that presents the runtime changes of the attributes, content, internal states, or surface looks of the target. By further analysis of such information, meaningful runtime target activities can be extracted, which can be identified as behaviours.

8.3 Challenges

Challenges also exist when employing monitoring technologies. Examples of such challenges are listed below.

— Runtime monitoring can bring extra overhead to the original system, and the security of the monitoring is important.

  — An internal module or an external agent that performs monitoring, as an additional functional part of the target, will probably bring extra overhead, thus likely decreasing the overall performance.

  — Certification for monitoring hardware is an essential requirement for operating monitoring with trust. Monitoring hardware can coordinate with external devices to achieve better performance. Potential threats to monitoring hardware exist in both cases. For example, monitoring data can be revealed to the outside. Thus, certification based on security requirements to address these threats can be considered in applying the monitoring hardware.

— Most existing hardware monitoring technologies are ad hoc. Extra optimization or redesign of the hardware target can be required for runtime evaluation tasks.

  — Among existing cases, the technologies that monitor hardware states most likely play the role of design assistance rather than assessment assistance. These technologies are designed to serve different purposes (see 6.2) with various features, some of which are possibly not convenient for evaluation. A common challenge is that some of these technologies can be realized only by an intrusive approach to the circuit, while reusability, configurability and non-intrusiveness are more desirable for assessment.

— Various hardware components present quite different attributes, features and behaviours.

  — Third-party agencies or users can be faced with different kinds of hardware components. Currently, third-party agencies or users who want to monitor hardware components to evaluate them, must do so case by case. Therefore, a general methodology of runtime security evaluation is needed.

— False positives and negatives impact hardware monitoring effectiveness, wasting resources or leaving the system vulnerable to undetected threats.

  — False positives and negatives are critical concerns in hardware monitoring systems. False positives refer to instances where the monitoring system erroneously identifies a normal activity as a threat, leading to unnecessary alerts and resource consumption. On the other hand, false negatives occur when the system fails to detect actual threats, leaving the system exposed to potential harm. Both situations can have detrimental effects on the overall effectiveness of the monitoring system.

  — These issues often arise due to the complexity of the algorithms used for threat detection and the inherent uncertainty in monitoring large-scale hardware systems. For example, in intrusion detection systems (IDS), an incorrect signature match or anomaly detection can trigger a false positive. Similarly, sophisticated malware can evade detection mechanisms, leading to false negatives.

  — Mitigating false positives and negatives involves a delicate equilibrium of diminishing inaccurate alarms while also maintaining robust threat coverage. By incorporating advanced machine learning techniques and perpetual real-world data training, the precision of threat detection can be significantly improved. The incorporation of ensemble methods, which merge multiple detection algorithms, assists in minimizing false positives and negatives.

— Inefficient resource allocation for processing power and memory in hardware monitoring systems can impact overall system performance.

  — Resource allocation is a critical aspect of hardware monitoring systems, especially in resource-constrained environments. Inefficient allocation of processing power and memory can lead to suboptimal system performance, affecting the overall functionality and responsiveness of the monitored system.

  — In resource-constrained embedded systems or Internet of Things (IoT) devices, the operation of the hardware monitoring system can be challenging due to the limited resources. For example, an embedded system can find it difficult to accommodate the high computational demands of real-time monitoring and analysis given its restricted processing power.

  — Addressing this challenge can involve the implementation of lightweight and efficient algorithms in hardware monitoring systems. One way to do this can be leveraging hardware-accelerated processing units like FPGA or ASIC. This technique can lessen the computational load of the main CPU. Also, focusing on the optimization of data storage and transmission has the potential to conserve memory resources.

— Scarcity of technical expertise hinders effective design, deployment, and maintenance of hardware monitoring systems.

  — The complexity of hardware monitoring systems demands specialized technical expertise. However, a scarcity of skilled professionals with in-depth knowledge of hardware, cybersecurity, and system architecture can hinder the successful implementation and maintenance of such systems.

  — Hardware monitoring systems require a deep understanding of various disciplines, including hardware design, operating systems, networking, cryptography, and cybersecurity. For instance, implementing secure hardware monitoring requires knowledge of threat modeling, risk assessment, and security protocols.

  — To overcome the scarcity of technical expertise, organizations can invest in training and education programs for their personnel. Additionally, collaborations with academic institutions and cybersecurity training centres can help fill the skill gap and create a pipeline of qualified professionals.

— Managing the lifecycle, especially in embedded systems, poses complexities and costs when updating or replacing components.

  — Hardware monitoring systems, especially in embedded systems, face unique challenges in managing the lifecycle of their components. The process of updating or replacing hardware components can be intricate, costly, and time-consuming.

  — Embedded systems often have limited modularity, and hardware components can be tightly integrated with the overall system architecture. For example, replacing a critical component in an embedded system can require significant redesign and revalidation of the entire system.

  — To address these challenges, careful planning and foresight during the initial design phase are crucial. Adopting modular hardware architectures and ensuring component compatibility with future updates can ease the lifecycle management process. Additionally, remote firmware updates and over-the-air (OTA) technologies can facilitate more efficient and cost-effective component updates in embedded systems.

Moreover, with reference to Reference [88], divide hardware security threats are divided into five categories: hardware Trojans, side channel attacks, IP piracy, invasion attacks, and reverse engineering. Since the security of the hardware monitor itself is also very important, the monitor in the last row of Table 3 has been included. New threats can also arise from the evolving threat landscape, therefore appropriate and specialized hardware monitoring techniques are useful for detecting such threats.

Table 3 — Values and limitations of hardware monitoring assessment technology

Threats [88]	**Detections or countermeasures**	**Hardware monitoring assessment technology values and limitations**
Hardware Trojan	Functional verification[89] Hardware Trojan triggering[90] Side-channel analysis[91] Security design[92]	Redundant checks and functional verification based on trusted hardware. Illegal instruction identification.
Side channel attack	Hiding and masking[93]	Effectively identifies attack patterns for frequent sampling events. Updates matching patterns quickly and diagnoses cache vulnerabilities such as Meltdown and the Spector series.
IP piracy	Trust verification[94]	Inspection of the calculation results of key assets (such as Keys). Identification of illegal instructions.
Reverse engineering	Hardware obfuscation[92]	Not available.
Attacks against monitoring hardware	Encryption, access control, entity authentication, physical protection of the hardware, and runtime self-testing.	Certification for monitoring hardware.

9 Certification for monitoring hardware

A certification for monitoring hardware is essential for using hardware monitoring technology with trust. Two use cases are considered. In the first use case, the monitoring hardware monitors the target hardware and detects malicious hardware. In the second case, the monitoring hardware coordinates with external devices to achieve a better performance.

In the first use case, the monitoring hardware obtains side-channel information (e.g. the power consumption, electromagnetic emissions, and computation time) of the target hardware to check the status of the hardware. A detection algorithm executed on the monitoring device can detect malicious behaviours by using the side-channel information. However, detection using only monitoring hardware can cause false detections due to the limited computational resources.

In the second use case, the monitoring hardware provides side-channel information to an external device. The detection algorithm can be executed partially or entirely on external resources (e.g. devices or PCs) to improve the detection accuracy.

Example threats against the monitoring hardware include the following:

— Unauthorized access

  An attacker can access (read, modify, or delete) the data (e.g. side-channel traces and cryptographic keys of the monitored target) in the monitoring hardware.

  An attacker also can access transmitted data or compromise the security of the monitoring hardware by monitoring or manipulating communication between the monitoring hardware and external resources. This also includes fault injection attacks.

— Unauthorized update

  An attacker can install unauthorized software (e.g. firmware) on the monitoring hardware.

— Malfunction

  A monitoring malfunction can reduce the security of the monitoring hardware.

To mitigate the risks of the above threats, the following countermeasures can be used:

— Access control

  Access control in accordance with security policies can improve the security of the monitoring hardware against unauthorized access.

— Entity authentication

  Authenticating an external entity (e.g. user or external resource) that requests access to the monitoring hardware also improves the security of the monitoring hardware against unauthorized access.

— Data integrity and authentication

  Authenticating the source of data and confirming the integrity of data can improve the security of the monitoring hardware against unauthorized updates.

— Confidentiality

  Encrypting data that are stored in, sent from or sent to the monitoring hardware can ensure confidentiality and improve the security of the monitoring hardware against unauthorized access. Hardware security modules (HSMs) and trusted platform modules (TPMs) can be used to protect encryption keys.

— Software update verification

  Verifying software updates can mitigate the risk of malicious software being installed and can improve the security of the monitoring hardware against unauthorized updates and malfunctions.

— Physical protection

  Locating the monitoring hardware in a controlled or monitored physical environment can improve the security of the monitoring hardware against unauthorized access.

— Self-testing

  Self-testing of the functionalities of monitoring hardware can improve its reliability and the security of the monitoring hardware against malfunctions.

Certification for monitoring hardware guarantees that the monitoring hardware meets these security targets.

ISO/IEC 15408-2 can be applied to the certification of monitoring hardware. The related security functional components are listed in Table 4.

Table 4 — Related security functional components in ISO/IEC 15408-2

Security targets	Related security functional components in ISO/IEC 15408-2
Access control	FDP_ACF (access control)
Entity authentication	FIA_UAU (user authentication), FDP_DAU (data authentication)
Data integrity and authentication	FDP_UIT (user data integrity transfer protection), FDP_DAU (data authentication)
Confidentiality	FDP_UCT (inter-TSF user data confidentiality transfer protection), FCS_COP (cryptographic operation)
Software update verification	FDP_UIT (user data integrity transfer protection), FDP_DAU (data authentication)
Physical protection	FPT_PHP (TSF physical protection)
Self-test	FPT_TST (TSF self-test)

The above relations can be used to describe PPs/STs for the monitoring hardware.