Pro-Active Policing and Policy Enforcement Architecture for Securing MPSoCs

Pro-active Policing and Policy Enforcement Architecture for Securing MPSoCs

Fahad Siddiqui, Matthew Hagan, Sakir Sezer,
The Centre for Secure Information Systems (CSIT), Queen's University Belfast
Belfast, United Kingdom
f.siddiqui, m.hagan, s.sezer@qub.ac.uk

***Abstract*—Embedded multiprocessor system-on-chip (MPSoC) architectures allow implementation of mixed critical applications and provide security mechanisms to segregate and protect system resources such as ARM TrustZone. These architectures enforce strict security measures right from the powering on of the system, to prevent misuse and compromise. However, such security measures have been found vulnerable where security design practices are not considered or are poorly implemented, particularly at software and hardware stack boundaries. Also, the embedded solutions developed using these MPSoC platforms are vulnerable to single points of failure and do not contain active response or mitigations for circumstances where a compromise occurs.**

This paper proposes pro-active hardware based policing and policy enforcement approach, along with system architecture and its hardware components, to this research problem. The architecture is physically isolated from the rich computing resources which actively monitors communications of system resources on the ARM AMBA-AXI4 bus. It detects anomalous system behaviours such as policy violation or compromised bus communication responses, and responds with predefined active countermeasures, such as deletion of secret data or disabling of the device to tackle security vulnerabilities and attacks at runtime. This proposed solution complements existing embedded hardware and software security technologies and provides an additional layer of hardware security when a vulnerability is found and exploited. This contribution lends itself to the principle of least privilege, implemented in software-based access control solutions like SELinux to mitigate when other protections have failed. This paper presents a proof-of-concept work supported by preliminary synthesis results on Xilinx Zynq-7000 and UltraScale+ MPSoC chips.

***Index Terms*—FPGA MPSoC, Zynq, ARM AMBA AXI4, ARM TrustZone, Hardware Trojan, Active policing**

I. Introduction

The *Field programmable gate array* (FPGA) market is estimated to experience an incremental growth of ≈ $2.70b in 2021, according to global technology research [1]. This trend will be driven by an increased proliferation of *Internet-of-things* (IoT) products, industrial automation and development of critical infrastructure for next-generation smart technologies [2], [3]. Embedded architectures are becoming complex as designers solidify different functionalities into a single application including mixing of sensitive data with non-sensitive data and physical actuation. These complex embedded architectures feature various timing requirements as well as differing levels of security [4], [5], [6], [7]. These design requirements lead to a trend of applying a combination of different processors, hardware *Intellectual Property* (IP) blocks and shared memory resources into a heterogeneous multiprocessor *system-on-chip* (MPSoC) to accommodate the increasing complexity of the applications [8], [9], [10]. These flexible systems, which provide hardware and software programmability, are adaptable, reusable, easy to upgrade and get to market faster, thus maximising the value of the end-product.

Inter-connectivity has exposed embedded architectures to a wide range of hardware and software attacks, often derived from methods used to attack conventional IT systems. It includes the devices used in several critical applications, such as Industrial Internet-of-things (IIoT), transportation, SCADA, as well as consumer IoT. Attacks may target a wide range of aspects, from the protocol stack to the systems applications, bootloader and hardware peripherals, including even the security technologies designed to protect them. The development of complex MPSoC based embedded solutions often involves the integration of third-party *intellectual property* (IP) to reduce time-to-market. The use of vulnerable third-party IP can open the door to attacks such as *Hardware Trojans* and malware, that can be launched within any device using the compromised IP [11], [12], [13]. Besides, the adaptation of third-party software components for ease of integration of network communication stacks, software libraries and services that were not designed primarily for *machine-to-machine* (M2M) devices pose serious security risks to the device when they are connected to a network or exposed to the Internet.

II. FPGA MPSoC: A flexible embedded security platform

The increasing demand for embedded security has driven FPGA vendors to integrate multiple levels of security, increased safety, and advanced power management mechanisms into newer generation of FPGA MPSoCs such as Xilinx Zynq UltraScale+ and Intel Stratix 10. These MPSoCs offer improved built-in hardware security features to harness and deploy system-level security by protecting the system assets [14], [15], [16]. Table I presents a detailed comparison of the on-chip hardware security features supported by market leading MPSoCs from Xilinx and Intel (Altera). They provide hardware components for building a system security foundation, based on the principles of *information assurance*, *trust* and *security*. They allow partitioning of system resources by dividing a system into subsystems and isolating the memory-space of these subsystem. To support flexibility of

978-1-5386-1491-4/18/$31.00 ©2018 IEEE

TABLE I
COMPARISON OF ON-CHIP SECURITY FEATURES SUPPORTED BY LEADING FPGA MPSOCS [14], [15].

FPGA Vendors	**Xilinx**		**Intel (Altera)**	
MPSoC Chips (technology)	**Zynq-7000** (28nm)	**Zynq UltraScale+** (16nm)	**Arria 10** (20nm)	**Stratix 10** (14nm)
Symmetric Encryption	AES-CBC 256	AES-GCM 256	AES-CTR 256	AES-GCM 256
Asymmetric Encryption	RSA	RSA-4096	RSA, ECC	RSA, ECC
Integrity check	SHA-256	SHA-3/384	SHA-256/384 ECDSA	HMAC-SHA-256, SHA-256/384, ECDSA
Key Storage	BBRAM, eFUSE	BBRAM, eFUSE, PUF KEK	BBRAM, eFUSE	BBRAM, eFUSE
Hardware root-of-trust	No	PUF (Ring Oscillator)	No	PUF (SRAM pattern)
Unique Identifier (OEM)	eFUSE	eFUSE(plain, obfuscated, and encrypted)	eFUSE	eFUSE
Key Agility (update)	No	Yes	No	Yes
Chain-of-Trust	Secure boot, ARM TrustZone	Secure boot, ARM TrustZone	-	Secure boot, ARM TrustZone
Side-channel protection	No	DPA resistant	Yes	Yes
Anti-tamper detection	No	Yes	No	Yes
Tamper logging	No	Yes	No	No
Hardened read-back	Yes	Yes	No	Yes
Debug Enable/Disable	Yes	Yes	Yes	Yes

software-hardware co-design practices, these security services are extended beyond the general purpose multiprocessing cores (ARM Cortex-A) down to the FPGA glue logic, to address system security issues at the application and user level. It facilitates a software controlled and hardware-enforced approach towards system security. The system software could execute as a standalone bare-metal application, real-time operating system, or full fledge rich-operating system (Embedded Linux, PetaLinux etc.) or combination thereof. The ARM TrustZone is an example that virtually segregates the hardware resources into secure and non-secure worlds, inspired by the concept of privileged and unprivileged access, supported by the operating systems [17], [18].

III. RELATED-WORK

This section will survey security vulnerabilities reported in the open literature on embedded MPSoC architectures and their supported security features.

Chen *et al.* exploited a lack of proper rollback protection within numerous devices' *secure boot* process, breaching the loading verification process used by ARM's TrustZone technology. An old key was reused to load a malicious application that could run in the secure-world, providing access to device's secret data [19]. Jacob *et al.* described an attack against the *secure boot* mechanism on FPGA MPSoC, which ensures secure system start-up and building chain-of-trust in embedded devices [20]. This attack allowed modification of boot parameters to load an unauthorised kernel image from a remote server over the network, instead of booting from the verified source. Though, mitigation of these attacks by formulating a well-defined policy for distributing patches and updating configurations to revoke older software versions, across a wide range of devices, is difficult from a practical perspective. Lipp *et al.* further attacked the ARM TrustZone security extension using side-channel analysis. By monitoring cache activity in the special area of the Android operating system, they were able to infer and extract details of the last running process [21]. A potential mitigation of cache attacks is deploying separate caches for each virtual context, as introduced in the latest ARM Cortex-M processors.

Cotret *et al.* proposed a *hardware firewall* architecture that enforces programmable security policies to protect external and internal traffic in FPGA based multiprocessor architectures, serving as a security bridge between the system resources and the shared system communication bus to monitor control activities. Their proposed work reduces the performance penalty compared to a fully encrypted approach, however it is limited to the detection of attacks and does not provide countermeasures, other than freezing the system communication bus [22]. Jacob *et al.* presented a detailed study of FPGA MPSoC security mechanisms with the focus on *Hardware Trojans* [13]. They proposed the concept of a light-weight AXI-wrapper as one of the prevention methods, similar to one proposed by [22], to encapsulate system assets. Following this work, Benhani *et al.* presented a security evaluation of the ARM TrustZone technology available on FPGA MPSoC. They presented some possible hardware attack scenarios caused by the presence of *Hardware Trojans* residing within an MPSoC's FPGA logic [23]. They demonstrated the modification of the hardware security attributes (set by ARM TrustZone) and propagation to the host processor by targeting AMBA-AXI4 communication system bus. These *Hardware Trojans* can be introduced during chip manufacturing process or at different stages of the design and development life-cycle. To mitigate these vulnerabilities, they proposed dividing the system into secure and non-secure subsystems. However, this strategy is not viable for systems with mixed criticality [5], [6].

A. Shortcomings of existing security approaches

After reviewing different aspects of security vulnerabilities and state-of-art of secure architectures and their methodologies to protect system's critical assets, there is clear evidence that existing approaches fall short due to:

- Security architectures rely on building and maintaining a robust *chain-of-trust*. This *chain-of-trust* is based on a nested series of assumptions and as vulnerable as its weakest link which, once broken, compromises the security of the complete system.

- A lack of security aware design and development practises, leading to the development of inconsistent and vulnerable software applications by the developers.
- Complex hardware-software co-design and integration practices, giving rise to vulnerabilities in hardware and software protocol stack boundaries, allowing to launch insider and outsider attacks.
- Lack of availability of independent run-time security mechanisms that can detect, curtail and secure system assets in hardware if the existing system security mechanisms are compromised. It includes systems with compromised *chain-of-trust*.

IV. Pro-active hardware policing and security policy enforcement

In order to address these shortfalls, we propose a pro-active policing and policy enforcement architecture to enhance system security, scalability and provisioning during operational life-cycle by complementing existing security mechanisms available in MPSoC based embedded systems. The concept of the policy-based approach is inspired by the fundamental concepts of *Least privilege* [24], which is enforced in computer systems through methods such as access control lists [25] and widely implemented in operating system solutions such as SELinux [26]. This policing approach shall monitor the undergoing communication of system resources, comparing it against expected behaviour. In case of unexpected behaviour, for example a malicious action initiated by a *Hardware Trojan* or launching of a compromised application by an adversary, the system can take both pro-active countermeasures to mitigate the attack and ensure the protection of system assets. Expected behaviours shall be defined within the system *security policy*, which can be updated at any stage of the device life-cycle. The policing approach, deployed at system communication layer, will be transparent and independent to existing embedded system architectures, complementing their security extensions (such as ARM TrustZone) that use the de-facto industry standard ARM AMBA-AXI4 SoC bus communication protocol. Adoption of the proposed approach will be easy to adopt in embedded architectures build upon different *software stacks* such as bare-metal, embedded Linux, RTOS, hypervisor, device-drivers etc. The proposed system architecture will not only enhance the security of the device but will also improve flexibility to meet changing security requirements of next-generation embedded architectures, as well as the next-generation of threats.

V. System architecture

To realise the proposed pro-active policing and policy enforcement approach, Fig. 1 presents a system architecture, using the Xilinx Zynq UltraScale+ MPSoC platform. This platform has been chosen due to the availability of widely adopted on-chip *cryptographic* and *hardware security* features, including *Physically Unclonable Function* (PUF) and an ARM TrustZone security module that extends and propagates the secure attributes of software applications down to the shared ARM AMBA-AXI4 system-bus and to the end-point slave. Besides the MPSoC features versatile processing capability providing ARM Cortex-A53 *Application Processing Unit* (APU) for general purpose computing and a Dual-core ARM Cortex-R5 *Real-time Processing Unit* (RPU) for deterministic time-critical computing [15]. Though, the proposed approach can be integrated to any AMBA-AXI4 compliant ASIC or SoC architecture. It can be adopted into existing and new embedded architectures at any stage of design and development process, to establish or enhance the hardware security into new or existing embedded architectures.

For the proposed system architecture, we assume that *untrusted software stack* can execute on the APU and can access connected resources. We also assume that all security related tasks, including configuration, provisioning and management of the policing components, are executed on the RPU and are treated as a *trusted application*. The RPU has a dual 64 KB *tightly-coupled memory* (TCM) that is physically isolated from L1 and L2 cache of the APU. This physical isolation and segregation ensure that an *untrusted application* cannot read, write or modify any part of the *trusted application*. This system security boundary has been highlighted in Fig. 1 which can overcome shortcomings of the virtualised security architectures. Following hardware components are central to our proposed approach:

1) Security policy engine (SPE)
2) AXI4-Bus sanity checker (SCK)
3) Security response engine (SRE)
4) Anti-tamper engine (ATE)

The SPE and SCK are implemented on FPGA glue logic that is portable and can be integrated at any development stage to incorporate hardware security. On the other hand, SRE and ATE utilise the built-in hardware security features of the Xilinx Zynq UltraScale+ MPSoC. They allow:

- Monitoring system-level communication between master and slave data and memory peripherals, enforcing the defined security policies for each slave.
- Checking the sanity of system bus signals at the interface between the host processor and hardware IP's to detect and curtail *insider* and *Hardware Trojan* attacks.
- Initiate pro-active response against vulnerabilities and attacks caused either by *insider* or *outsider* attacks, to protect the system's critical assets.

Each hardware component has an AXI4-Lite interface that allows a *trusted application* to configure and manage their supported features. It can enable third-party vendors and security architects to independently provision embedded devices by defining policies during the life-cycle of the device [27].

The AXI4 establishes a communication between a master and a slave using five separate channels; *address write, write data, write response, address read* and *read data* [15]. Each channel has a VALID and READY handshake signal that represents the validity of data available on the bus and indicates that the receiver is ready to receive the data available on the bus respectively. The exchange of data happens when both

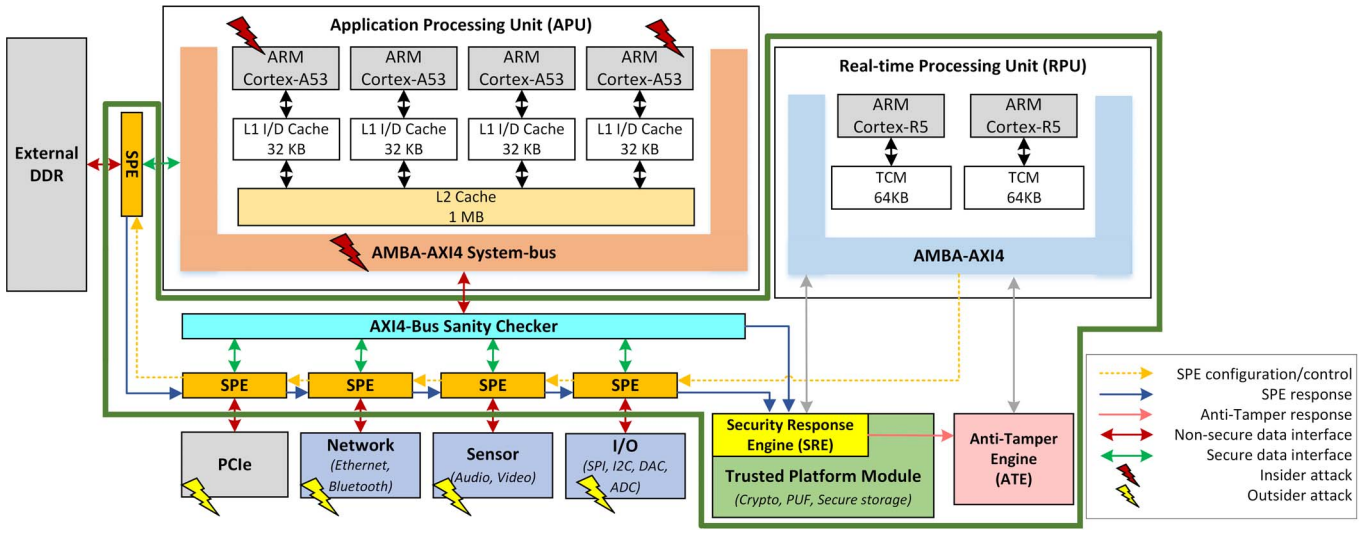

Fig. 1. Block diagram of the FPGA MPSoC system architecture to implement the proposed pro-active policing and policy enforcement approach. The architecture allows for the harnessing and enforcement of the dynamic security requirements defined as security policies, to curtail insider and outsider attacks.

VALID and READY signals are high. The AXI4 specification supports optional control signals to incorporate system-level security [15]. Both address channels have 3-bit AWPROT and ARPROT signal, the second bit of which is *Non-Secure* (NS)-bit that defines the security attribute of the transactions. During system execution, they propagate down to the FPGA glue logic through the system interconnect (AXI4-Interconnect) as shown in Fig. 2. The value of NS-bit virtually segregates system resources into *secure* and *non-secure* world. Besides, 2-bit RRESP and BRESP signal are used to acknowledge read and write transfers back to the host processor as shown in Fig. 3. Their status (OKAY, SLVERR, DECERR) informs the master whether the transfer is successful or completed with error. But, if either NS-bit and xRESP signals are exposed to insider attacks or *Hardware Trojan* that can lead to the compromised resources, *denial-of-service*, *slave impersonation* and other range of man-in-the-middle attacks [23] as illustrated in Fig. 2 and Fig. 3.

A. Security Policy Engine (SPE)

The role of the SPE is to check issued AXI4 transactions, compare them against an approved list of masters and their defined policies, and either grant or restrict access to an attached slave. Fig. 1 shows a distributed deployment of SPEs to enforce resource specific security measures by actively monitoring communications between the system-bus and the slave peripheral such as memory, sensor or actuator etc. It is a four-stage pipelined architecture as shown in Fig 4 comprised of the following five hardware blocks.

1) Engine configuration block: It enables secure programming and configuration of security parameters of the SPE, by the *secure application* using the AXI4-Lite interface as shown in Fig. 4. It facilitates changing policies and updating system-level security if compromised.

2) AXI4-Sniffer: This element monitors and maintains the mandatory handshake signals that ensure correct AXI4 bus operations. It also retains physical isolation between master and slave until the *decision block* either grants or block access. It can sample the *master* and *slave* addresses using AxADDR, and uses AxVALID to distinguish between read and write transactions. Though, this element uses the AxUSER signal to identify master and slave addresses and does not rely on system generated tags (AxID, xID). Once sampled, the address is passed to the *device table* as shown in Fig. 4. During this process, it maintains physical isolation between the master and slave by de-asserting AxVALID signals to slave and AxREADY control signals to the master.

3) Device table: The device table holds a list of *trusted masters* with permissions to access the slave. Each *trusted master* entry holds corresponding base-address of the assigned policies in the *Policy table* as shown in Fig. 4. It filters-out the *trusted* and *un-trusted* master requests which are useful for *intrusion detection* and helps to actively curtail or suppress the

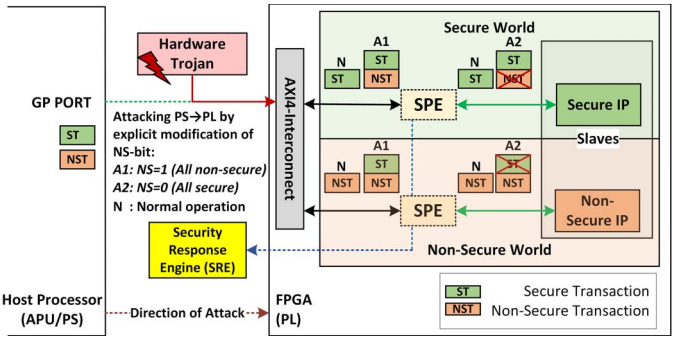

Fig. 2. Virtual segregation of slaves into secure and non-secure worlds using the AXI4 security attribute (NS-bit). This shows the possible insider attack (from PS to PL) scenarios and how they can be detected and mitigated by the proposed SPE.

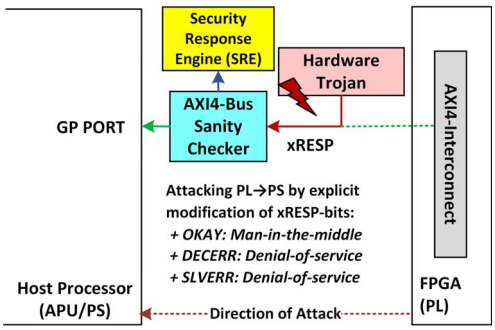

Fig. 3. A possible insider attack scenarios (PL to PS) by modification of the response channel signals (xRESP). This shows how they can be detected by the SCK and curtailed by the SRE.

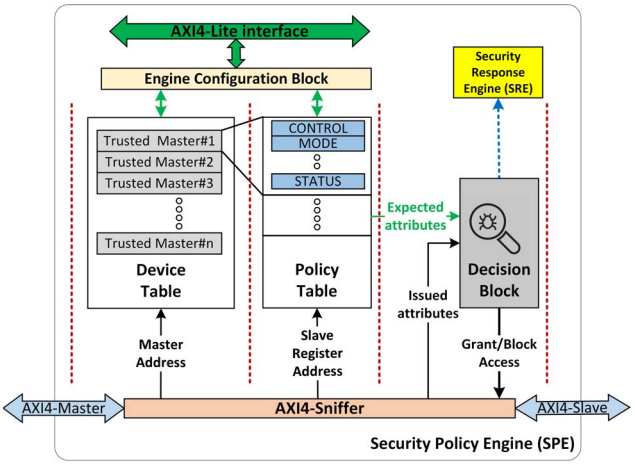

Fig. 4. Block diagram of the SPE, deployed as bridge between master and slave, illustrating the interaction among its hardware components and the SRE.

malicious activities or intrusions by SRE.

4) Policy table: The policy table contains a list of fine-grained (register-specific) security policies of each *trusted* master as shown in Fig. 4. These policies can be simple as read and write permissions, or security tailored attributes (AxPROT). Moreover, it facilitates dynamic device provisioning by defining policies and enforcing them independent of silicon manufacturers and third-party vendors.

5) Decision block: The decision block is designed to detect and suppress fine-grained intrusion or malicious activities such as explicit modification of security attributes illustrated in Fig. 2 and Fig. 3. The decision block references *expected security attributes* from the *Policy table*, compares them against the *issued security attributes* and generates a grant or block decision as shown in Fig. 4. This result is passed to the *AXI4-Sniffer* which physically grants or block access to the slave.

B. AXI4-Bus sanity checker (SCK)

The SCK can be deployed at the interface between AXI4 system-bus and slaves as shown in Fig. 1. It ensures the integrity of AXI4 response channel signals issued by the slaves to detect and mitigate insider attacks [23] launched by *Hardware Trojans* as illustrated in Fig. 3. It is composed of a three state *finite-state machine* (FSM) and a configurable 32-bit timer through a *secure application*. It has a CONTROL and STATUS registers to enable/disable and read the real-time system status reported by SRE. In case of an attack (SLVERR or DECERR), the FSM moves from NORMAL to DETECTION state and starts the free running timer. The timer decrements at each clock cycle from the set value. If de-assertion of SLVERR or DECERR has detected before the timer time-out, FSM returns to NORMAL state else moves to the ATTACK state and reports it to SRE to take defined pro-active countermeasures. The OKAY attack can be detected by comparing the system status reported by SRE and SCK.

C. Security Response Engine (SRE)

The SRE manages and triggers pro-active responses against malicious activities or hardware attacks reported by the SPE or SCK as shown in Fig. 1. It interrupts the execution of the *trusted application* running on the RPU and executes a priority interrupt service routine that initiates a programmed pro-active response, enforced by ATE, which may consist of:

- Erasure of secret keys stored in the device.
- Permanent disabling of system cryptographic functions.
- Disable system read-back interfaces.
- Secure system lock-down or reset.

D. Anti-tamper Engine (ATE)

The ATE enforces active countermeasures initiated by the SRE, as well as passive countermeasures against physical attacks. It independently monitors the device's physical parameters such as voltage and temperature. If an irregularity occurs, an alarm are raised in the form of system-level interrupt.

VI. Prototyping and Validation

The proposed SPE are coded in Verilog HDL. The architecture is simulated, synthesised and implemented using Xilinx Vivado v2017.1 on the high-end chips of both Zynq-7000 and Zynq UltraScale+ MPSoC. Table II reports the synthesis results of proposed SPE architecture. The design is compact and consumes less than 1% of FPGA programmable logic resources. The device table implemented using Distributed RAM utilises 10 LUTRAM, while the policy table has used a BRAM which can store up to 1024 policies. The reported timing results show that SPE can operate at maximum 250 MHz and 300 MHz on Artix-7 and Kintex UltraScale+ chips respectively. The critical path limited by AXI-interconnect that works as a bridge to connect the ARM Cortex-A processor to the FPGA glue logic. Similarly, the proposed SCK consumes a fraction of FPGA resources.

A bare-metal software application has written to validate the proposed architecture. It executes on the APU and emulates both normal and malicious activities by generating benign AXI transactions. In case of malicious and unexpected behavioural activities, SPE detected by comparing the AXI security attributes against the stored security policies inside the policy table and triggers the FSM which validates the correct operation of the proposed architecture.

TABLE II
SYNTHESIS RESULTS OF PROPOSED HARDWARE COMPONENTS TO REALISE POLICING APPROACH ON ZYNQ-7000 AND ZYNQ ULTRASCALE+ MPSOC.

Hardware Component	MPSoC	FPGA fabric (technology)	Speed Grade	Area				Frequency (MHz)
				LUT	LUTRAM	FF	BRAM	
SPE	**Zynq-7000**	Kintex-7 (28nm)	-3	156	10	464	1	250
	Zynq UltraScale+ EG	Kintex UltraScale+ (16nm)	-3	185	10	464	1	300
SCK	**Zynq-7000**	Kintex-7 (28nm)	-3	48	0	39	0	250
	Zynq UltraScale+ EG	Kintex UltraScale+ (16nm)	-3	56	0	39	0	333

VII. CONCLUSION AND FUTURE WORK

This paper has outlined some of the shortcomings of existing embedded security approaches and architectures, proposing a hardware-based pro-active policing and policy enforcement proof-of-concept system demonstrator. The architecture provides an additional layer of hardware security, independently complementing existing security mechanisms. It provides active countermeasures that respond to detected attack behaviour to secure the system's critical assets when a vulnerability is found and exploited. The proposed SPE and SCK are light-weight security components that enable software controlled and hardware enforced security policies, alongside security against system bus level insider attacks, to achieve near real-time policing to fulfil the changing security requirements of different applications and security levels.

Future work will target effective hardware and software isolation of the SPE introducing anti-tamper mechanism, eliminating vulnerabilities introduced by traditional debug interfaces. Policy features will be extended to support HW and SW debugging. Countermeasures are usually use-case specific. Further research will explore various base-line countermeasure circuit architectures for various use-cases, including home gateways and critical industrial control applications.

REFERENCES

[1] (2017) Global Field-Programmable Gate Array Market 2017-2021. [Online]. Available: https://www.technavio.com/report/global-semiconductor-equipment-global-field-programmable-gate-array-market-2017-2021

[2] M. D. V. Pena, J. J. Rodriguez-Andina, and M. Manic, "The Internet of Things: The Role of Reconfigurable Platforms," *IEEE Industrial Electronics Magazine (IEM)*, vol. 11, no. 3, pp. 6 – 19, Sep. 2017.

[3] J. J. Rodrguez-Andina, M. D. Valds-Pea, and M. J. Moure, "Advanced Features and Industrial Applications of FPGAs - A Review," *IEEE Trans. Ind. Informat.*, vol. 11, no. 4, pp. 853 – 864, Aug. 2015.

[4] M. Hassan, "Heterogeneous MPSoCs for Mixed Criticality Systems: Challenges and Opportunities," *IEEE Design & Test*, vol. PP, no. 99, pp. 1 – 1, Nov. 2017.

[5] B. Tan, M. Biglari-Abhari, and Z. Salcic, "A system-level security approach for heterogeneous MPSoCs," in *Proc. IEEE Conference on Design and Architectures for Signal and Image Processing (DASIP)*, Rennes, France, Oct. 2016, pp. 74 – 81.

[6] T. Benjamin, B.-A. Morteza, and S. Zoran, "Towards decentralized system-level security for MPSoC-based embedded applications," *Journal of Systems Architecture*, vol. 80, pp. 41 – 55, Oct. 2017.

[7] S. M. Trimberger and J. J. Moore, "FPGA Security: Motivations, Features, and Applications," *Proc. IEEE*, vol. 102, no. 8, pp. 1248 – 1265, Aug. 2014.

[8] T. Dorta, J. Jimnez, J. L. Martn, U. Bidarte, and A. Astarloa, "Overview of fpga-based multiprocessor systems," in *Proc. IEEE Conference on Reconfigurable Computing and FPGAs (ReConFig'09)*, Quintana Roo, Mexico, Dec. 2009, pp. 273 – 278.

[9] C. Wang *et al.*, "Architecture Support for Task Out-of-Order Execution in MPSoCs," *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1296 – 1310, May 2015.

[10] F. Siddiqui, M. Hagan, and S. Sezer, "Embedded policing and policy enforcement approach for future secure IoT technologies," in *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, Mar. 2018, pp. 1–10.

[11] S. Mal-Sarkar, A. Krishna, A. Ghosh, and S. Bhunia, "Hardware trojan attacks in fpga devices: Threat analysis and effective counter measures," in *ACM Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI (GLSVLSI'14)*, Houston, USA, May 2014, pp. 287 – 292.

[12] M. Tehranipoor *et al.*, "Trustworthy Hardware: Trojan Detection and Design-for-Trust Challenges," *IEEE Computer*, vol. 44, no. 7, pp. 66 – 74, Jul. 2011.

[13] N. Jacob, C. Rolfes, A. Zankl, J. Heyszl, and G. Sigl, "Compromising FPGA SoCs using malicious hardware blocks," in *Proc. IEEE Design, Automation Test in Europe Conference Exhibition (DATE)*, Lausanne, Switzerland, Mar. 2017, pp. 1122 – 1127.

[14] S. Lester and P. Ed, "A FIPS 140-2 Primer for the Zynq-7000 All Programmable SoC," Xilinx Inc., White Paper: WP467 (v1.1), 2016.

[15] Xilinx, "Zynq UltraScale+ MPSoC Embedded Design Methodology Guide," Xilinx Inc., User Guide: UG1228 (v1.0), 2017.

[16] I. Polian, F. Regazzoni, and J. Sepulveda, "Introduction to hardware-oriented security for MPSoCs," in *Proc. 30th IEEE International System-on-Chip Conference (SOCC)*, Munich, Germany, Sep. 2017, pp. 102 – 107.

[17] J. Winter, "Trusted Computing Building Blocks for Embedded Linux-based ARM Trustzone Platforms," in *Proc. 3rd ACM Workshop on Scalable Trusted Computing (STC'08)*, Alexandria, Virginia, USA, Oct. 2008, pp. 21 – 30.

[18] N. Santos, H. Raj, S. Saroiu, and A. Wolman, "Using ARM Trustzone to Build a Trusted Language Runtime for Mobile Applications," *ACM SIGARCH Comput. Archit. News*, vol. 42, no. 1, pp. 67 – 80, Mar. 2014.

[19] C. Yue, Z. Yulong, W. Zhi, and W. Tao, "Downgrade Attack on TrustZone," *Computing Reseach Repository (CoRR)*, Jul. 2017.

[20] N. Jacob, J. Heyszl, A. Zankl, C. Rolfes, and G. Sigl, "How to Break Secure Boot on FPGA SoCs Through Malicious Hardware," in *International Conference on Cryptographic Hardware and Embedded Systems (CHES 2017)*, Taipei, Taiwan, Sep. 2017, pp. 425 – 442.

[21] L. Moritz, G. Daniel, S. Raphael, and M. Stefan, "ARMageddon: Last-Level Cache Attacks on Mobile Devices," *Computing Reseach Repository (CoRR)*, vol. abs/1511.04897, Jun. 2016.

[22] P. Cotret, G. Gogniat, and M. J. S. Flrez, "Protection of heterogeneous architectures on FPGAs: An approach based on hardware firewalls," *Microprocessors and Microsystems*, vol. 42, pp. 127 – 141, May 2016.

[23] E. M. Benhani, C. Marchand, A. Aubert, and L. Bossuet, "On the security evaluation of the ARM TrustZone extension in a heterogeneous SoC," in *Proc. 30th IEEE International System-on-Chip Conference (SOCC)*, Munich, Germany, Sep. 2017, pp. 108 – 113.

[24] J. H. Saltzer, "Protection and the control of information sharing in multics," *ACM SIGOPS Operating Systems Review*, vol. 17, no. 7, pp. 388 – 402, Jul. 1973.

[25] R. S. Sandhu and P. Samarati, "Access control: principle and practice," *IEEE Commun. Mag.*, vol. 32, no. 9, pp. 40 – 48, Sep. 1994.

[26] P. Amthor, "A uniform modeling pattern for operating systems access control policies with an application to SELinux," in *Proc. 12th IEEE International Joint Conference on e-Business and Telecommunications (ICETE)*, Colmar, France, Jul. 2015, pp. 88 – 99.

[27] N. Jacob *et al.*, "Securing FPGA SoC configurations independent of their manufacturers," in *Proc. 30th IEEE International System-on-Chip Conference (SOCC)*, Munich, Germany, Sep. 2017, pp. 114 – 119.