A Hardware Root-of-Trust Design for Low-Power SoC Edge Devices

Alan Ehret, Eliakin Del Rosario, Karen Gettings †, Michel A. Kinsy
Adaptive and Secure Computing Systems (ASCS) Laboratory, ECE Department, Texas A&M University
{ehretaj, eliakin.drosario, mkinsy)@tamu.edu
† MIT Lincoln Laboratory
karen.gettings@ll.mit.edu

Abstract— **In this work, we introduce a hardware root-of-trust architecture for low-power edge devices. An accelerator-based SoC design that includes the hardware root-of-trust architecture is developed. An example application for the device is presented. We examine attacks based on physical access given the significant threat they pose to unattended edge systems. The hardware root-of-trust provides security features to ensure the integrity of the SoC execution environment when deployed in uncontrolled, unattended locations. E-fused boot memory ensures the boot code and other security critical software is not compromised after deployment. Digitally signed programmable instruction memory prevents execution of code from untrusted sources. A programmable finite state machine is used to enforce access policies to device resources even if the application software on the device is compromised. Access policies isolate the execution states of application and security-critical software. The hardware root-of-trust architecture saves energy with a lower hardware overhead than a separate secure enclave while eliminating software attack surfaces for access control policies.**

Keywords—Hardware Security; System-on-Chip; Low-Power

I. Introduction

Increasingly efficient System-on-Chips (SoC) have enabled numerous new applications for low-power embedded systems. Much of the efficiency and performance improvements in modern SoCs are attributable to the inclusion of heterogeneous hardware accelerators [1]. Example accelerators include Digital Signal Processing (DSP) cores, neural network inference engines, and Fast-Fourier Transform (FFT) cores. Energy efficiency and performance gains have allowed SoC-based devices to perform more of their computations locally. Devices that compute on data locally are commonly referred to as "edge" devices, because they operate at the edge of a network, between data sources (sensors) and a centralized infrastructure [2]. Without the need to frequently offload data to a centralized infrastructure, edge devices can be deployed in environments lacking a stable network connection or power grid access.

Common examples of edge devices include solar-powered smart signs that can change their display after deployment and connected trashcans that alert a city's sanitation department when they are full. Other examples include unattended ground sensor nodes, unmanned aerial vehicle (UAV) swarms, and mobile payment processing systems. These examples are all solar- or battery-powered, but the amount of power they consume can vary by several orders of magnitude. Edge devices, such as smartphone-based mobile payment systems, may consume between 100s of mW to several Watts. Meanwhile, unattended ground sensors may consume power on the order of nW to μW. Achieving nW power consumption requires the use of sub-threshold CMOS ASICs, as in [3]. Despite a wide range of power consumption, the above examples all share long duration deployments and local computation requirements. In the case of a smart sign or ground sensor, ideally, device deployments would last months or years without maintenance or physical access on the part of the device owner. UAVs or ground sensor nodes require local computation because of latency or security requirements.

Connected edge devices, such as a wireless sensor in a security system or a ground sensor node, frequently handle security-critical functions, making them prime targets for attacks. The advances in power efficiency that made edge applications and devices possible have also led to the deployment of these devices in outdoor and public spaces. Furthermore, the nature of edge applications means that edge devices are frequently left unattended by those responsible for them. Unattended deployment means that an attacker can gain uninterrupted physical access to a device. Physical access presents a security challenge when devices must ensure the integrity of their program results to achieve their objectives. Further advances in power efficiency will only exacerbate the vulnerability of these devices as they are deployed in greater numbers, for longer periods of time and in more remote locations. Anti-tamper protections can mitigate physical access-based attacks. Previous works have used battery-backed volatile memory to prevent tampering with sensitive memory such as FPGA bit streams [4]. However, the power budgets for edge devices generally preclude powering volatile memories for the entire lifetime of the device. The security challenges faced by unattended edge devices has motivated previous works, including surveys of hardware-based security defenses [5][6].

In this work, we design a security-focused low-power SoC architecture for use in edge devices. The architecture mitigates deployment-time threats faced by edge devices while meeting power and energy constraints with a hardware root-of-trust.
The SoC architecture is named RECORD, short for Reconfigurable Edge Computing for Optimum Resource Distribution. The RECORD architecture continues the trend of accelerator-focused SoC designs with the inclusion of a single RISC-V core and several reconfigurable hardware accelerators.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.

The RECORD SoC is designed to serve as a multi-purpose wireless audio sensor device. The RECORD SoC supports a microphone as a peripheral device and includes hardware accelerators for processing microphone sensor data. Accelerators are designed to operate in a hierarchy, with each one consuming more resources than the last. When data must be processed, RECORD activates the lowest power accelerator first and evaluates the output to determine if further processing is warranted. When additional processing is needed, more resource intensive accelerators are activated.

The RECORD SoC ensures the integrity of the execution environment when deployed in uncontrolled locations with the Root-of-Trust (RoT) Unit. The RoT Unit is made up of several individual hardware modules. A non-volatile boot memory module includes an E-fused [7] write port to prevent changes to the device boot loader or other security-critical code after device deployment. Digitally signed programs can be loaded into non-volatile programmable instruction memory before or after deployment. Verification of the digital signature ensures the application program is from a trusted source. Occasionally, the RoT unit requires access to a resource shared with application code, such as the CPU or data memory. To prevent information leakage and unauthorized resource usage, a programmable finite state machine (FSM) enforces access control policies. Access control policies prevent simultaneous access to a resource by the RoT Unit and application code. The RoT Unit's FSM and access controls allow it to temporarily take control of the SoC, pause application execution, and execute one of its own security-critical functions implemented in software. The RoT Unit helps prevent information leakage by reseting the state of a shared resource when a security-critical operation has completed, allowing the application code to be safely granted access again. Also included are Built-In Self Test and E-fused programmable interrupt controller modules. To meet energy and security constraints, the RoT Unit architecture considers both a tight area and power budget, as well as the physical access to the device attackers have.

The RECORD SoC and the included RoT Unit demonstrate how threats faced by edge devices during their deployment can be mitigated using a hardware root-of-trust deeply integrated with the rest of the SoC. The design philosophy applied to RECORD and the RoT Unit in this work are not specific to microphone sensor-based devices. The techniques presented here can be adopted in future edge designs regardless of the application, sensors, or accelerators.

II. RECORD Accelerator Hierarchy

The RECORD SoC is designed as a multi-purpose device for edge systems with audio-based applications. Potential applications include speech detection or recognition, perimeter monitoring, or noise identification and classification. Four hardware accelerators are included in the SoC: a noise/energy detector, a Finite Impulse Response (FIR) filter, a Fast Fourier Transform (FFT) module, and a Support Vector Machine (SVM). The SoC uses a RISC-V core to perform general purpose operations. The selected accelerators have been chosen to minimize the energy consumed processing sensor data. Accelerators can be reconfigured to further optimize their energy consumption for different use cases, such as different input sizes. In this case, a microphone serves as the sensor but other sensors could be used.

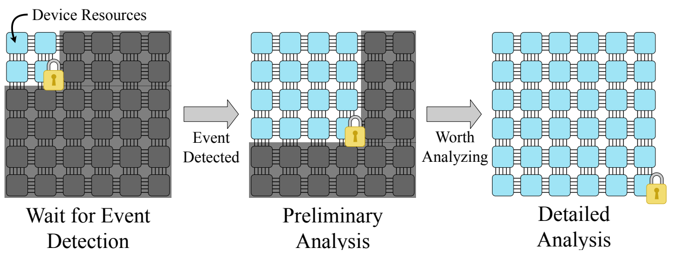

Fig. 1. Hierarchical resource usage

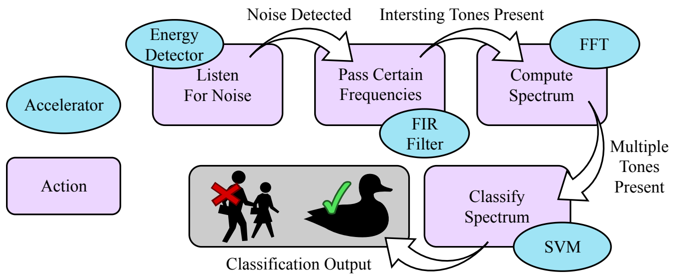

Fig. 2. Classification accelerator hierarchy.

The four hardware accelerators form a hierarchy of computation. The lowest level of the hierarchy consumes minimal power while waiting to detect an event in the sensor data. When an event is detected, the next level of the hierarchy begins preliminary processing of the data. The application program on the RISC-V core checks the accelerator output to determine if data processing should continue. Only events selected by the program will continue processing with higher power accelerators. This hierarchy of processing allows events irrelevant to the edge system application to be quickly dropped, saving energy and compute resources for relevant events. Fig. 1 shows how more device resources are allocated to a computation as processing progresses through the hierarchy. Using a RISC-V core to analyze accelerator output and determine which events are relevant gives different RECORD SoCs the flexibility to support different edge applications.

III. Example Application

To guide the design of the RECORD SoC, we selected an example application to run on the completed edge device. In the application, the RECORD SoC-based edge device listens for noise and identifies it as either human or wildlife. A hierarchy of computation can greatly improve SoC efficiency. Consider that edge systems deployed in uncontrolled environments are likely to encounter numerous events (noises in this case) that are not relevant to the application. Wind, and rain are examples of irrelevant noises likely to be detected by an edge device running the classification application example. A hierarchy of computation quickly eliminates irrelevant noises from the human/wildlife detection algorithm, saving energy and compute resources.

In the first step in the hierarchy, the RECORD SoC uses the energy detector accelerator to listen for noise. When the energy detector is active, other accelerators and the RISC-V core are put in a sleep state. When a noise triggers the energy detector, the detector wakes up the RISC-V core to configure the FIR filter and eliminate unwanted frequencies from the recorded audio

TABLE I. ATTACKS AND COUNTERMEASURES FOR EDGE DEVICES

Attack	Scan-Chain state modification	Off-Chip RAM modification	Timing Side Channel	Power Side Channel	EM Side Channel	Power Fault Injection	Clock Fault Injection
Countermeasures	Disable scan-chain, built-in self-test	On-chip RAM	Constant runtime, no caches	Obfuscate power consumption	EM noise circuits	On-chip capacitors	On-chip PLL
Mitigated in RECORD	Yes	Yes	Yes	No	No	No	Yes

signal. The RISC-V core checks if enough energy is left in the signal to warrant FFT processing. If a significant signal is still present, the FFT module computes the transform of the signal. When the transform is complete, the RISC-V core performs simple sanity checks on the result, ensuring the spectrum has a reasonable distribution, i.e., more than a single tone is present. Finally, the SVM accelerator is enabled outputs a classification. The complete classification hierarchy is shown in Fig. 2.

IV. Threat Model and Attack Overview

Sections II and III described how the RECORD SoC achieves the energy efficiency and performance necessary for edge devices and provided an application example as a reference. This section examines the threats faced by edge devices. The security mechanisms used to mitigate some of those threats are introduced and described in detail in the following Section V.

A. *Threat Model*

The nature of edge device deployment creates a challenge when trying to create a trusted execution environment. Deployment in an uncontrolled environment means an attacker can gain physical access to a device. With physical access, edge devices can be attacked with both software and hardware modifications. The limited power budget of edge devices means that mitigating all of the threats they face is not feasible. Instead, the RECORD SoC architecture mitigates a subset of software and PCB level hardware modification-based attacks that aim to achieve arbitrary code execution on the device.

Attackers are assumed to be capable of (1) interfacing with any device ports accessible by an end user, (2) disassembling and inspecting the device's internal design, including any PCBs in the device, (3) analyzing signals on PCB traces or IC pins, (4) accessing debug ports, such as PCB test points or the scan-chain, and (5) making PCB-level modifications, such as removing ICs, shorting traces, and opening traces. Attackers are not assumed to be capable of decapsulating the RECORD SoC or probing internal chip wires.

With the previously listed capabilities, a variety of attacks are possible. Attacks considered during the design of RECORD include: scan-chain attacks, off-chip memory probing or modification, timing/power/electro-magnetic (EM) side-channel information leakage, power fault injections, and clock fault injections. These attacks all occur after the device has been deployed. Table I summarizes considered attacks and their potential countermeasures. The RECORD SoC is assumed to be uncompromised until it is deployed in an uncontrolled environment.

B. *Attack Descriptions*

In scan-chain attacks, an attacker uses the debugging scan-chain to observe or alter the internal state of an IC. Attackers can use unprotected scan-chains to dump firmware or upload malicious code. Off-chip memory is vulnerable to snooping or modification by an attacker. PCB traces from an SoC to a memory IC can easily be probed by an attacker. Complex systems can suffer from timing-based microarchitectural side channel attacks, such as the Spectre [8] family of attacks. These attacks leverage access-based cache-timing side-channels to leak information. Power side-channel attacks including, Simple Power Analysis and Differential Power Analysis, can leak information about which instructions are executing or the data they execute on. This information could allow an attacker to learn secret keys or other information about a program's execution. Similarly, EM side-channels can reveal secret information about a program by comparing EM noise from different inputs. Power or clock fault attacks reveal encryption keys by comparing the ciphertexts produced under fault-free and faulty conditions. Detailed description of these hardware-based attacks is beyond the scope of this work but can be found in [9].

V. RECORD SoC Architecture

This section describes the architecture of the RECORD SoC and how it mitigates hardware-based attacks discussed in the threat model. Section V.A describes the logical organization of the RECORD SoC. Section V.B presents a detailed view of the SoC architecture. Section V.C describes the design and functionality of each RoT Unit feature.

A. *RECORD Logical System View*

As an accelerator-focused SoC, RECORD includes a single RV32I RISC-V Core and four reconfigurable hardware accelerators. Fig. 3 presents a logical view of the RECORD SoC. The RISC-V core has isolated instruction and data buses. Accelerators share a dedicated bus and communicate with the RISC-V core through a second port on the data memory. All SoC memory is on-chip to mitigate memory probing/tampering attacks. The RoT Unit is not logically connected to the rest of the system. Logical isolation means the RECORD compute modules (shown in blue) cannot observe the RoT Unit's internal state.

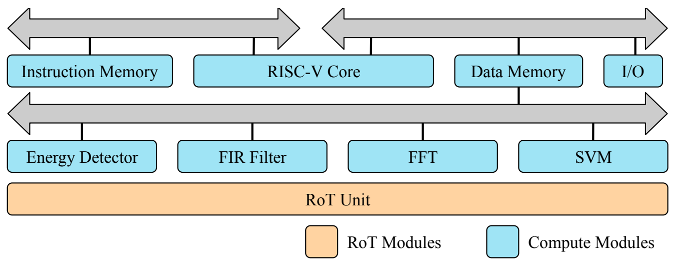

Fig. 3. A logical view of the RECORD SoC

B. *RECORD Detailed Architecture*

Fig. 4 presents a detailed block diagram of the architecture. In addition to the RISC-V core, accelerators, memory and I/O shown in the logical system view (Fig. 3), specific RoT modules are shown in orange. Access control modules and the

Programmable Finite State Machine (FSM) isolate RoT Unit and application code state when the RoT Unit must use shared resources. Access controls can also be applied to create multiple isolated memory partitions within RoT Unit or application code. Section VI describes how the FSM sets and updates access control policies. Also included in the RoT unit are the Programmable Interrupt Controller (PIC) shown in yellow and digitally signed instruction memories. The device includes a hard coded, memory-mapped public key used to verify the instruction memory signatures. We assume the device designer and end user have previously arranged a secure process to sign authentic instruction memory images. Device I/O is sanitized with an RoT control module to prevent I/O related side channels.

RECORD's area budget forces many performance enhancing micro-architecture features (including caches) to be omitted, mitigating many common timing side-channels. Additionally, the signed instruction memory and access controls prevent attackers from executing the code needed to exploit popular access-based cache timing side-channels.

C. *RoT Unit*

1) Built-In Self Test: The Built-In Self Test (BIST) hardware module checks the RoT circuitry for faults. Energy and area limitations prevent the BIST module from performing testing extensive enough to ensure the RoT hardware is implemented as designed. Instead, the BIST module and the rest of the RoT Unit are considered the trusted computing base of the RECORD system. Mitigating design-time attacks such as hardware trojans is beyond the scope of this work.

With the included BIST module, the RoT hardware does not need to be connected to an external scan-chain. Eliminating scan-chain connections on security critical hardware mitigates scan-chain based attacks that attempt to change or observe the internal hardware state.

2) Programmable Finite State Machine: Security constraints are enforced in hardware with a programmable FSM. Prior to deployment, the FSM configuration can be programmed to implement any N-input, L-output, K-state Moore FSM. N, L and K are parameters chosen at SoC design time. FSM inputs are fed into Look-Up-Tables (LUT) to compute the next state. The system state is fed into more LUTs to compute outputs. Fig. 5 presents the FSM architecture. At deployment time, an E-fuse is used to prevent further writes to the configuration memory, locking the FSM configuration.

The FSM inputs include "done" status signals from each of the accelerators and a two bit Control Status Register (CSR) from the RISC-V core. The two bit CSR represents system events such as the completion of the next accelerator configuration. For each accelerator, the FSM outputs a two bit signal to the associated access control module to select one of four permission configurations. Three additional FSM outputs are fed into a decoder connected to eight interrupt lines. Interrupts can indicate that an accelerator computation is complete or that the RoT Unit must take control of the system to process a security related event. Decoding output bits into interrupt lines or access control states reduces the number of output bits, saving a significant amount of area that would otherwise be needed for output LUTs.

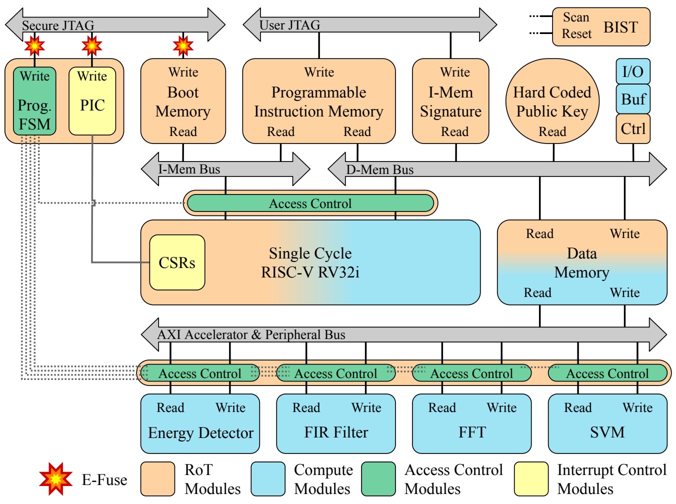

Fig. 4. Detailed RECORD Architecture

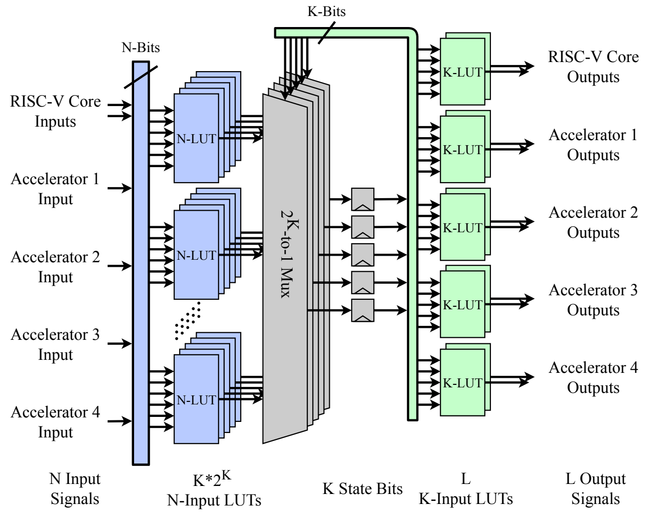

Fig. 5. Programmable FSM architecture. Interrupt outputs are not shown.

The programmable FSM mitigates attacks that leverage unauthorized or unexpected use of SoC compute resources. The access control policies selected by the FSM ensure software vulnerabilities or rogue accelerators cannot overwrite sensitive sections of memory or improperly use I/O resources. As the application execution progresses through the hierarchy of accelerators, the FSM state changes to set the appropriate access control policy.

3) Boot Memory: A dedicated non-volatile memory module, called the boot memory, is used to store code that the RECORD SoC executes first on power up. The boot memory stores the instructions used for all RoT software features. One RoT Unit software feature is the verification of the digital signature of the programmable instruction memory against the hard-coded public key. Additional application or deployment specific RoT software features can be added before the RECORD SoC deployment. The RECORD SoC currently uses an elliptic curve digital signature algorithm (ECDSA, [10]) for the programmable instruction memory verification. The boot memory and hard-coded public key of future designs could easily be updated for new key sizes or algorithms. In the example application, programmable instruction memory verification is the only software-based RoT Unit functionality. Similar to the programmable FSM, the boot memory includes an E-fused write port. At deployment time, the boot memory is made read-only by blowing the fuse, providing a trusted, immutable section of memory for security and boot code. The programmable interrupt controller and programmable FSM are flexible enough that (prior to deployment) additional RoT software-based functionality can be added to the boot memory without hardware changes.

4) RISC-V Core: The single-cycle RISC-V core is used to execute both application code and RoT Unit code. The other RoT Unit hardware modules enforce isolation between the two states of execution. The RISC-V core is based on The BRISC-V Platform, an open-source RISC-V design space exploration platform [11]. Several custom Control Status Registers (CSR) have been added to the core to support interrupts, as described in V.C.6. These CSRs are modeled after the interrupt CSRs defined in the privileged RISC-V specification. Power and area constraints make the full privileged specification impractical.

5) Programmable Instruction Memory: The programmable instruction memory provides non-volatile memory for application code. All software in the programable instruction memory must be digitally signed by the private key associated with the hard coded public key on the RECORD SoC. The programmable instruction memory can be altered after deployment. The programmable FSM and access control modules ensure the programmable memory passes the signature verification check (performed by the RoT Unit software in the boot memory) before it can be executed.

6) Programmable Interrupt Controller: The Programmable Interrupt Controller (PIC) enables the RISC-V core to react to asynchronous events such as accelerator done signals or FSM triggered security events. Core interrupts are delayed by one cycle to allow the FSM to detect the interrupt condition and update the access controls if necessary. The PIC configuration memory stores an interrupt handler PC value for each interrupt line. To support interrupts, the RISC-V core includes two custom CSRs that are not included in the RV32I base specification. The first CSR stores the next instruction address when an interrupt is triggered. The second CSR stores a status code written by the PIC based on the interrupt condition. The PIC configuration memory (the handler PC and status code associated with each interrupt) includes an E-fuse to prevent writes after deployment. When an interrupt is triggered, the interrupt handler will read the status register to determine what action to take. When interrupt handling is complete, the interrupt handler returns to the stored PC address. A dedicated RoT interrupt handler function is stored in boot memory. Other handlers are stored in programmable instruction memory. Access controls prevent non-RoT related software from executing the RoT interrupt handling code. Nested interrupts are not supported.

Interrupts are the only mechanism to transition control of the RECORD SoC to the RoT Unit. When a RoT related interrupt occurs, access controls prevent accelerators from accessing data memory. Using interrupts to transition control of the RISC-V core to the RoT Unit allows RoT software to enter a known state of execution, independent of the application program state.

7) Data Bus Access Control: The access control modules restrict the range of allowable read or write addresses for each bus master in the RECORD SoC. Each access control module supports four access policies. Each policy consists of a minimum and maximum address with read and write permission settings. A two-bit signal from the programmable FSM selects the current policy. Access control policies are written with the programmable FSM configuration and become read-only after the programmable FSM's fuse is blown.

A minimum of two access policies must be configured to isolate RoT memory from application memory (one for RoT mode, another for application mode). The FSM ensures the appropriate access control policy is selected during RoT or application code execution. Additional policies can be used to create isolated regions within application or RoT memory. For example, policies could be used to prevent accelerator reads or writes when an accelerator is supposed to be inactive. Alternatively, multiple isolated RoT memory regions could be created for different interrupt routines.

VI. RECORD PROGRAMING AND DEPLOYMENT

This section describes how all of the RoT Unit modules work together to prevent arbitrary code execution attacks on the RECORD SoC. The development, deployment and operation of the RECORD SoC are described for the application example discussed in Section III.

In the development phase, application software is written and any necessary changes to the RoT configuration are made. In this phase, the RECORD SoC is assumed to operate in a trusted environment where it will not be attacked. System developers test and verify the application and RoT configuration. When satisfied that a device is ready for deployment, an end user (the one who will deploy the device) loads the final RoT configuration and blows the E-fuses to prevent further changes. Then, they can load a stable version of the application program into the programmable instruction memory. Finally, the end user deploys the device in its final, uncontrolled location.

Once deployed, the RECORD SoC is powered on. The Built-In Self-Test (BIST) holds the hardware reset high while it tests the RoT circuits for faults. If the test is passed, the reset signal is lowered and the start-up process continues. After a hardware reset, the RoT Unit's Finite State Machine (FSM) is configured to grant control of the system to the RoT Unit and the RISC-V core PC is set to an address in boot memory. At this point in the start-up process, all accelerators are disabled and no application code has executed. The RoT Unit executes the signature verification function in boot memory to ensure the programmable instruction memory contains a trusted program. Once the signature verification is complete, the RoT Unit software writes to the FSM input CSR, indicating the RoT Unit is about to cede control of the RECORD SoC to the application program. One cycle after writing to the CSR, the RoT software jumps to the application entry point to begin application execution. The FSM updates access control accordingly.

In the application example given in Section III, the energy detector (ED) accelerator is the first to be configured. The application program sets the desired threshold to trigger an interrupt and writes to the FSM CSR to indicate the application has entered a new phase. In this phase, the energy detector is given access to the data memory while the rest of the accelerators and the RISC-V core enter a sleep mode without data memory access. When the ED threshold is passed, the accelerator triggers an interrupt. The FSM detects the interrupt and updates the access controls to grant the RISC-V core data memory access. The RISC-V core begins logging sensor data for the remaining accelerators to process. Energy limitations prevent the ED accelerator from constantly logging sensor data. When logging is complete, the application configures the FIR Filter accelerator to begin processing the logged sensor data. Another FSM CSR write indicates that the program phase has changed. The FSM updates access controls. The accelerator interrupt, FSM access control update, and data checking process continue for each of the accelerators until the application program receives a final classification from the SVM accelerator or rules out the logged data as an event of interest.

After a predetermined amount of time, the RoT Unit will trigger an interrupt to take control of the RECORD SoC and re-verify the programmable instruction memory. The FSM will disable each of the accelerators and enable RoT memory access for the RISC-V core. The RoT Unit interrupt handler begins executing. First, the RoT interrupt handler must save the application register file state to a predetermined point in the RoT memory. With the application state saved, a new known, safe state is loaded into the register file. After gaining control of the SoC and placing the RISC-V core in a known state, the RoT Unit will re-verify the signature of the programmable instruction memory to ensure no tamping has taken place during program execution. Finally, the RoT Unit interrupt handler restores the application register file state, writes to the FSM CSR to indicate the end of the RoT interrupt and jumps to the address stored in the "next address" CSR. The FSM updates the access controls, and application execution resumes.

VII. Results and Analysis

The RoT Unit hardware modules, including the microcontroller, were synthesized for the Xilinx Artix-7 XC7A200T-2FBG676C. Table II presents the synthesis results. Results are included for each submodule of the RECORD SoC top module. The domain specific accelerators are excluded. FPGA resources are reported as Slices (four 6-input LUTS and eight flip flops) and Block RAM (BRAM) Tiles. Dedicated RoT Unit modules are listed in bold. Collectively, these modules are the hardware overhead of the RoT Unit in the RECORD SoC. The RoT Unit carries a 13.7% slice overhead and a 34.4% BRAM Tile overhead.

Each programmable instruction memory verification takes approximately 180M cycles with the libecc ECDSA implementation [12]. During verification, the application execution will not progress. The frequency of verifications can be adjusted to meet application specific performance or security requirements. The remaining RoT Unit features (access control and programmable FSM) only impart an area overhead. They do not impact the performance of accelerators or application code.

TABLE II. RISC-V Core FPGA Synthesis Results

Module	Slices	BRAM Tiles	Module	Slices	BRAM Tiles
RECORD SoC (Top)	1471	192	**Instr. Bus Access Control**	9	0
Boot Memory	32	64	Microcontroller	1151	0
Data Memory	6	64	**PIC**	8	0
Data Bus Access Control	9	0	**Programmable FSM**	99	0
Instruction Memory Signature	13	2	Programmable Instruction Memory	29	62
UART	192	0	**Public Key**	31	0

VIII. Conclusion and Future Work

This work has presented the RoT Unit hardware root-of-trust architecture. The RECORD accelerator-based SoC, a design tightly integrated with the RoT Unit, was developed. An example application was described to illustrate the intended use case for the RECORD SoC. Future work will focus on estimating battery lifetimes for given deployment durations and battery sizes. Analysis will focus on the energy used by different RoT Unit features to mitigate specific attacks. Such analysis will facilitate design decisions for future edge system designs.

References

[1] M. D. Hill and V. J. Reddi, "Accelerator-level parallelism," CoRR, vol. abs/1907.02064, 2019. [Online]. Available: http://arxiv.org/abs/1907.02064

[2] W. Shi and S. Dustdar, "The Promise of Edge Computing," in Computer, vol. 49, no. 5, pp. 78-81, May 2016.

[3] S. Jeong et al., "Always-On 12-nW Acoustic Sensing and Object Recognition Microsystem for Unattended Ground Sensor Nodes," in IEEE Journal of Solid-State Circuits, vol. 53, no. 1, pp. 261-274, Jan. 2018.

[4] S. M. Trimberger and J. J. Moore, "FPGA Security: Motivations, Features, and Applications," in Proceedings of the IEEE, vol. 102, no. 8, pp. 1248-1265, Aug. 2014.

[5] A. Ehret, K, Gettings, B. Jordan, and M. Kinsy, "A Survey on Hardware Security Techniques Targeting Low-Power SoC Designs," IEEE High Performance extreme Computing Conference, Waltham, MA, 2019.

[6] M. Isakov, V. Gadepally, K. M. Gettings and M. A. Kinsy, "Survey of Attacks and Defenses on Edge-Deployed Neural Networks," 2019 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 2019, pp. 1-8.

[7] C. Kothandaraman, S. K. Iyer, and S. S. Iyer, "Electrically programmable fuse (efuse) using electromigration in silicides," IEEE Electron Device Letters, vol. 23, no. 9, pp. 523–525, Sep. 2002

[8] P. Kocher et al., "Spectre Attacks: Exploiting Speculative Execution," 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2019, pp. 1-19.

[9] S. Bhunia and M. Tehranipoor, "Hardware Security: A Hands-on Learning Approach". Morgan Kaufmann, 2018.

[10] Johnson, D., Menezes, A. & Vanstone, S., "The Elliptic Curve Digital Signature Algorithm (ECDSA)," IJIS 1, 36–63, 2001.

[11] S. Bandara, A. Ehret, D. Kava, and M. Kinsy. "BRISC-V: An Open-Source Architecture Design Space Exploration Toolbox," In Proceedings of the 2019 International Symposium on Field-Programmable Gate Arrays (FPGA '19). ACM, New York, NY, USA

[12] Libecc Library. https://github.com/ANSSI-FR/libecc.