AKER: Safe and Secure SoC Access Control

Chi Chow, Brandon Erickson, Hosein Yavarzadeh

June, 2022

Abstract

The growing popularity of modular system-on-chip (SoC) architectures presents an increasing need for defense against sensitive data access from malicious hardware. In security-critical applications, IP cores have different privilege levels for accessing shared resources, which must be regulated by an access control system. AKER is an existing high-performance modular access control framework for AMBA AXI4 on-chip interconnects. We have developing a simplified adaptation of AKER onto network-on-chip (NoC) applications to form a high-bandwidth access control wrapper (ACW). This adaptation is implemented onto ESP, an open-source rapid SoC development platform created by the University of Columbia. This modification enables secure protection of sensitive data against malicious hardware accelerator implementations on the ESP platform.

1 Introduction

Many modern commercial processors have opted for System-on-Chip (SoC) architectures due to their modular design and reduced power consumption compared to traditional computer system architectures. Modern SoC architectures can consist of many different resources, including processors, hardware accelerators, and IO. To handle the modularity of these SoC architectures, on-chip networks are used to communicate effectively between resources. Of course, sharing resources safely and securely is also critical. This prompts the use of access control systems that define the permissions and abilities of an SoC controller. However, it is challenging to implement secure SoC access control systems. Hardware vulnerabilities are hard to patch, often requiring a firmware rewrite, or even re-manufacture of the chip.

This problem motivates the existence of AKER [10] – a framework for the development of safe and secure on-chip access control systems targeting the requirements of modern safety and security critical applications. AKER utilizes the ACW (Access Control Wrapper) which is a high-performance and easy-to-integrate hardware module that dynamically manages access to shared resources. ACWs are used to wrap controller IP cores and to act as local access controllers.

Our project simplifies the AKER access control framework for integration onto the ESP SoC development platform. ESP [1] is an open-source research platform for streamlined modular SoC design. ESP supports multiple design flows, including RTL, high-level synthesis (HLS), and machine learning frameworks, and each design flow converges to an automated rapid SoC integration flow. Our simplified access control framework serves to improve the resilience of ESP against malicious third-party hardware accelerators.

The remainder of this document is organized as follows. Section 2 compiles information on related works, including SoC architecture, the ESP SoC development platform, and the AKER access control protocol. Section 3 details our contributions to the AKER framework, including its simplification and integration onto the ESP platform. The development process of this project, including project milestones, is also included in this section. Finally, Section 4 details the evaluation of our implementation through simulation and security analysis.

2 Related Works

2.1 System-on-Chip (SoC) Architecture

In a system-on-chip (SoC) architecture, most, if not all, components of a computer or other embedded system are integrated into a single chip. On a single chip, a basic SoC includes processing cores, memory units, I/O controllers, and hardware accelerators [11]. Some implementations of SoC also include other components such as secondary storage interfaces and GPUs.

SoC architectures have become increasingly popular among developers of embedded systems and mobile computing due to their distinct advantages over the traditional motherboard-based computer system design. For one, components in an SoC are inherently more tightly integrated than they would be in their motherboard-based counterpart. The tighter integration of components yields higher performance, reduced power consumption, and improved semiconductor area utilization.

For this reason, researchers have shown great interest in developing communication protocols that facilitate tighter integration of SoC components. Over the last decade, network-on-chip

(NoC) has emerged as the dominant communication protocol in these systems [9]. Inspired by the Internet's packet-based communication framework, NoCs facilitate scalable, low-latency, high-bandwidth communication between components in a multi-core SoC. As such, NoCs are typically structured as miniature versions of traditional networking solutions with nodes and links. Each node in an NoC consists of a processing element, a network interface, and a router. Thus processing elements in the NoC may communicate with each other without the poor bandwidth of bus communication or the poor scalability of crossbar communication.

2.2 The ESP Development Platform and Architecture

The target platform for this project is ESP [1], an open-source platform for the rapid development of SoCs and hardware accelerators. ESP's architecture implements a tile-based distributed system connected by a multi-plane network-on-chip (NoC) for packet-based data transfer [2]. This system includes RISC-V processor tiles, memory and I/O tiles, and hardware accelerator tiles. ESP provides its own hardware accelerator tiles, but also supports the integration of third-party accelerators into the architecture.

A tile may transmit and/or receive data packets by interfacing with sockets included in the NoC. The NoC exclusively interacts with these sockets, and does not interact directly with any tiles or accelerators. Through the socket layer, ESP streamlines data transfer to ensure compatibility with the network's communication protocol while maintaining modular support for third-party accelerators. A diagram of the socket layer interfacing with an ESP accelerator is given in Figure 1.

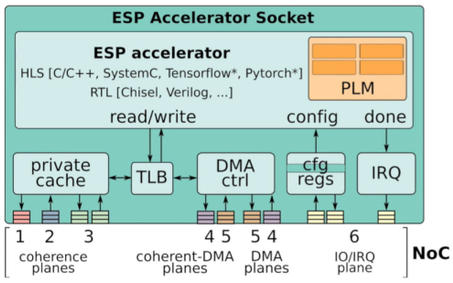

Figure 1: ESP Accelerator Socket [5]

Additionally, the NoC weighs data transfers from processor tiles and accelerator tiles equally; no priority is given to transfers to or from the processor tiles. In other words, the processor does not have the ability to preempt data transfers from external accelerators, and all data transfers are managed through the network alone.

A comprehensive access control framework, then, would interact with data transfers at the socket layer of the ESP NoC. Access control must be implemented and enforced at each plane of the NoC, including coherence planes, DMA planes, I/O planes, and interrupt request (IRQ) planes, even though third-party accelerators may not need to interface with every plane.

2.3 AKER: Safe and Secure SoC Access Control

2.3.1 Overview

In an SoC architecture, a set of controller devices communicate with a set of peripheral devices. Different processors, accelerators, and other IP cores can be assigned as a controller. As a result, they can autonomously and concurrently access shared peripheral resources on the SoC, such as a DRAM memory controller, on-chip memories, ROM, IP core control and status registers (CSRs), and GPIOs. On-chip data transfers rely on communication protocols like AMBA AXI or TileLink, which provide a flexible, asymmetric, synchronous interface designed for high-performance and low latency communications. In any SoC access control system, access to the on-chip resources must be arbitrated. To specify which resources controllers want access to, high-speed onchip communication protocols use memory mapped addressing. An access control policy specifies whether a data request is allowable at that given time. The access control system should precisely implement the access control policy while having a minimal impact on performance and area.

AKER is a design and verification framework for SoC access control systems that addresses the performance, security, and flexibility requirements of modern safety- and security-critical applications. The Access Control Wrapper (ACW) is a high-performance, programmable module that controls memory transactions from a controller.

3 Design and Implementation

3.1 ESP Architecture Implementation

Our ESP development workflow can be simplified into the following steps:

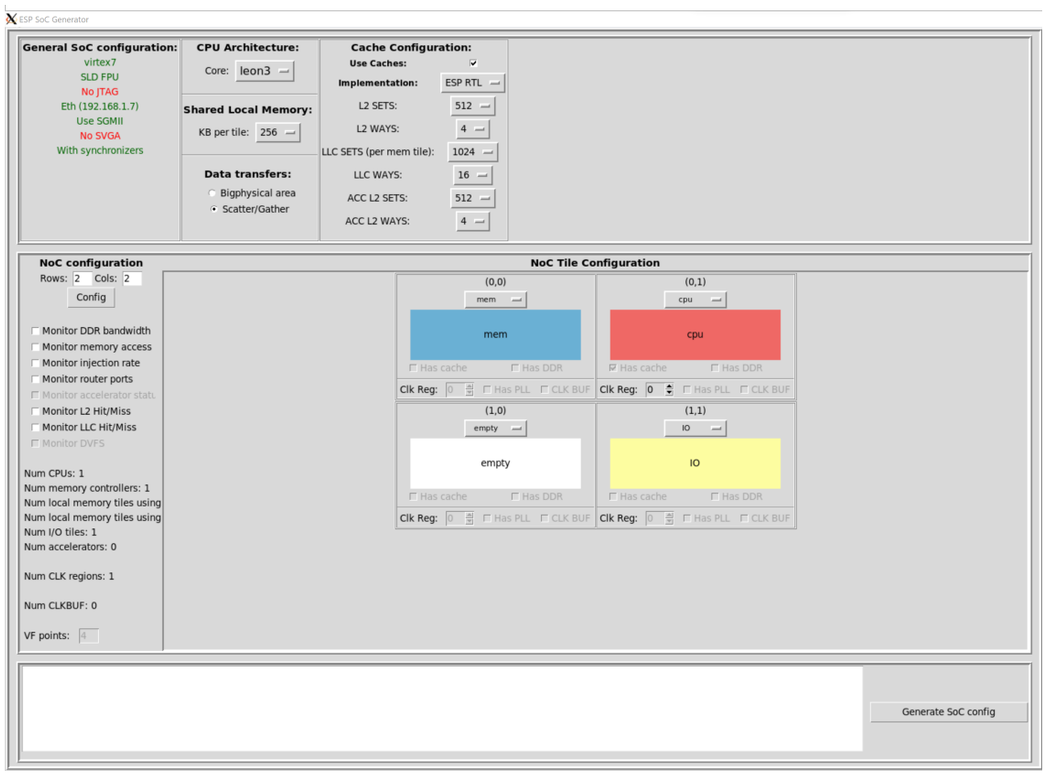

Figure 2: ESP's GUI to configure SoC tiles. Our goal is to provide an option to select using AKER access control during SoC configuration.

- Generate custom hardware accelerator configuration
- Generate SoC configuration
- Run RTL simulation

We implemented AKER's functionality within the RTL code base of the ESP repository. Our goal is to allow users to choose AKER as an option for access control during the process of generating the SoC configuration. If AKER is selected, the SoC configuration can be updated to generate a layer between different tiles of the SoC and its shared resources.

The existing ESP development workflow allows us to conveniently create Access Control Wrappers between different modules of the SoC. To test the functionality of our implementation, we can run RTL simulations in ModelSim, and create test cases for both legal and illegal accesses.

3.2 Access Control Wrapper (ACW)

The Access Control Wrapper (ACW) is a configurable access control module designed to monitor an AXI-compliant controller. The ACW exports an AXI M interface, an AXI-lite S configuration interface, and an output interrupt line. An ACW can be used on any SoC controller resource whose memory accesses require arbitration for safety or security reasons; each untrusted controller C_i is wrapped by an ACW module ACW_i.

Figure 3 provides an example of an AKER-based access control system. The M interface of ACW_i is connected to the AXI interconnect (in place of the M interface of C_i), while the S interface and the interrupt line are connected to a Trusted Entity (TE).

3.3 Simplified Version of AKER

Detailed structure of the ACW is shown in Figure 4. In this section we will describe the simplified functionality of the Access Control Wrapper. When C_i issues a read request AR through its M interface, ACW_i has the following behaviors:

- Address Check: Check AR against $LACP_i$ by comparing the address of AR against each of the allowable read regions.
- Legal Request: If AR is fully included in at least one of the allowed read regions of $LACP_i$, propagate AR to the AXI interconnect.
- Illegal Request: If AR is not fully included in any of the read regions described in $LACP_i$, AR is not propagated to the AXI interconnect. ACW_i saves internally information regarding the illegal request AR. ACW_i sends an AXI-compliant error to C_i, notifies TE, and switches into Decouple Mode.

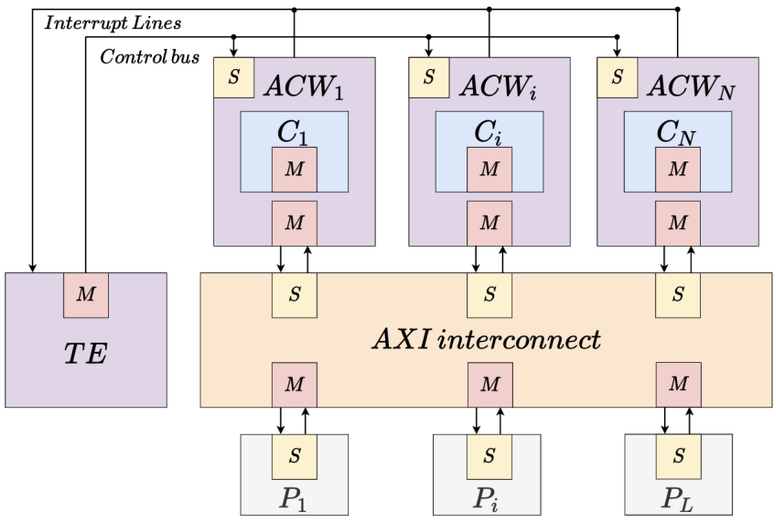

Figure 3: Extended AXI multi-controller, multi-peripheral architecture incorporating an AKER-based access control system. The Trusted Entity TEmanages the ACWmodules. Only legal requests are transmitted to the AXI interconnect, i.e., the peripherals receive only legal AXI transactions.

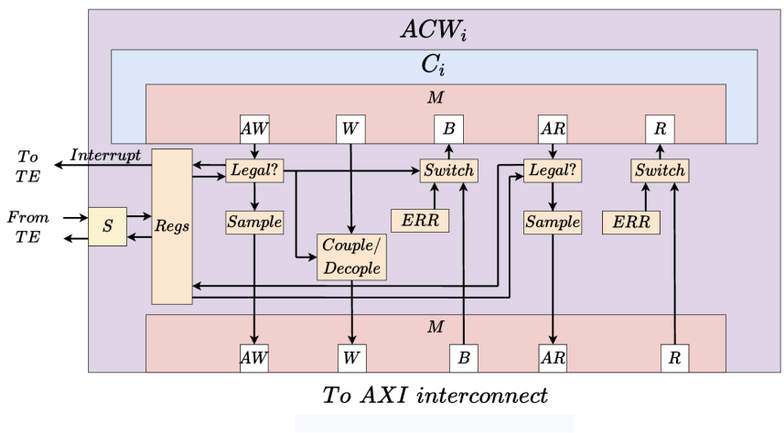

Figure 4: ACW_i architecture: C_i is the controller module using an AXI M interface. Regs are the configuration registers holding $LACP_i$. The AXI S interface is connected to the $HRoT$.

- Outstanding Transactions: Any legal outstanding transaction initiated before an illegal transaction is completed normally.

3.4 Integration of Simplified AKER onto ESP

As briefly mentioned above, the AKER development and testing process can be divided into 3 distinct steps, utilizing the ESP platform. We mainly followed ESP's RTL accelerator guide [3], adding slight modifications for our own use case.

3.4.1 Configuring a Custom Hardware Accelerator

To begin development in the RTL workflow, AKER provides a shell script to conveniently create and configure a custom hardware accelerator, which we will use for testing later. Figure 5 shows a typical hardware acceleration generation process. We can specify attributes such as the type of accelerator (in this case, RTL), and its inputs and outputs. In addition to the hardware configuration of this accelerator, a template test program is also generated. At this stage, we can also implement our test program, which will access the resources on the SoC.

3.4.2 Generate SoC configuration

Once the hardware accelerator is generated, we can select a target FPGA board to test our SoC. Any FPGA board supported by ESP can be used for testing, because AKER should work as long as the hardware accelerator uses the on-chip network to access other resources. Figure 3 shows a

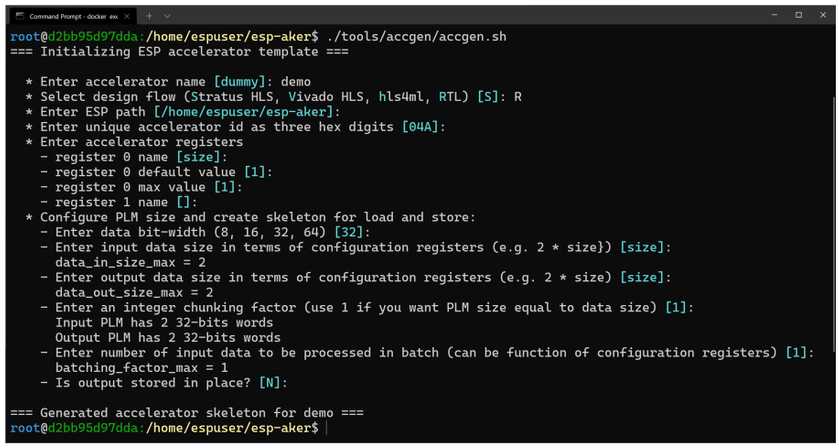

Figure 5: AKER's Hardware Accelerator Generator

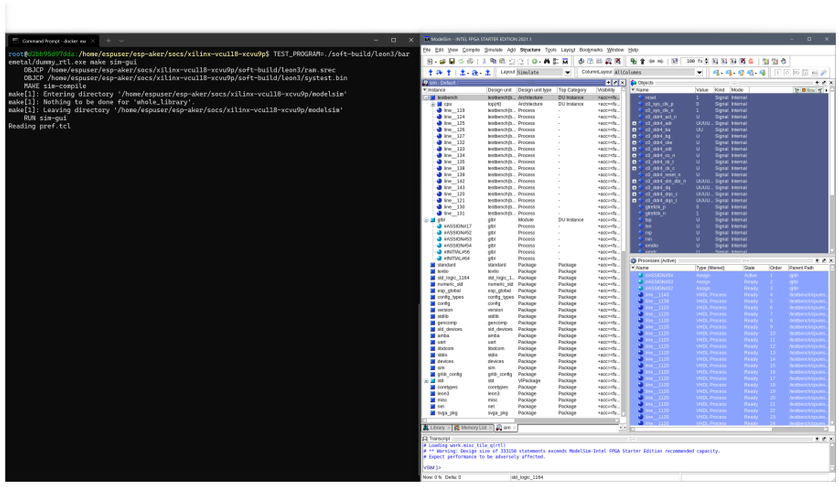

Figure 6: Running a ESP Test Program in ModelSim

simple structure of an SoC, containing one CPU, one memory, and one IO tile. We can see that the bottom left tile is empty, which we can change and set to the newly created hardware accelerator.

3.4.3 RTL Synthesis and Simulation

Once configuration and implementation is complete, we can begin RTL synthesis and simulation of our design. ESP uses Xilinx Vivado for high level synthesis [4]. It takes the targeted FPGA board's hardware configuration, and synthesizes it to be able to either run on the physical FPGA board, or be simulated in ModelSim. Figure 6 shows our SoC and test program being deployed to ModelSim.

3.5 Project Milestones

The development of the simplified AKER access control module is summarized by the milestones below. The completion status of each milestone is updated from a previous Project Progress Report.

1. ESP Project Setup (Brandon, Chi)
   - Definition: Implementation and evaluation of an accelerator using ESP. A specific accelerator (e.g. adder) will be selected from the documentation and github repository of the ESP platform.
   - Deliverables: 1-2 pages report including the setup process (requirements, installation procedure, etc.), accelerator specification, implementation results, and evaluation (functionality correctness).
   - Status: Complete. See Section 2 for accelerator specification and evaluation. This milestone was verified for completion by presentation of simulation results and the above deliverable's inclusion in the Project Progress Report.
2. ESP Project Modifications (Brandon, Chi)

- Definition: Modify the ESP platform so that specific architectures can be added on top of it.
- Deliverables: Modified Codes of the ESP (a new branch in github repository) + 1-2 pages report including the modification process (e.g. modified files) and evaluation study. The evaluation study ensures that the modified version of ESP is bug-free and the corresponding codes are compiled and installed correctly.
- Status: Complete. Because of ESP's modular socket-layer network interface, few modifications were needed to implement access control onto the ESP platform.

3. AKER Project Simplification (Hosein)
   - Definition: Simplifying the access control system such that it can be implemented in a limited time. In this milestone we will describe the detailed architectures that are required to be implemented on top of the ESP.
   - Deliverables: 3-4 pages report including the architectures (e.g. ACWs, TE, and etc.), detailed structures (interconnections) and algorithms used in simplified version of the AKER. Report will include the architectural issues to be addressed during the implementation. It will also include:
     (a) The high-level structure of the architecture (should be a standard tile-based architecture leveraging a Network-on-Chip (NoC) for data exchange)
     (b) Internals of the NoC router (i.e., any technicalities related to the implemented routing algorithm, possible routing restrictions, and possible implemented optimizations).
     (c) Details and technicalities related to the protocol used by routers for data exchanged among them.
   - Status: Complete. See Section 3.2 for details on the simplification of the AKER framework.

4. Implementation of AKER on ESP (Chi, Hosein)
   - Definition: In this milestone, we will implement the architectures, interconnections, and all required structures of the AKER on the ESP platform. All implementable structures come from the third milestone report.
   - Deliverables: Modified Codes of the ESP (a new branch in github repository branch) + 3-4 pages report including the modification process and documentation about the architectures and algorithms employed in the implementation phase.
   - Status: Complete. See Section 3.3 for details on AKER's integration onto ESP.

5. Final Evaluation and Debugging (Chi, Hosein)
   - Definition: Testing of AKER on ESP on the Zynq Ultrascale+ platform.
   - Deliverables: 3-4 pages report about the correctness, performance, and security evaluation of the implementation.
   - Status: Complete. Security evaluation and simulation results are provided in Section 4.

6. Final Project Video (Brandon, Chi, Hosein)
   - Definition: Develop a professional video describing the project.
   - Deliverables: A video (5 minutes) including team members introduction, project goals, and the final results.
   - Status: Complete. A video presentation is included in the project's Github repo: `https://github.com/cschow-ucsd/esp-aker`

7. Final Report (Brandon, Chi, Hosein)
   - Definition: Technical report summarizing the project goals and progress over the quarter
   - Deliverable: 8-10 pages report including abstract, introduction, background, implementation details, evaluation, conclusion, and references.
   - Status: Complete. This document serves as the Final technical report for this project.

4 Evaluation

4.1 IP-level Security Verification

Security verification is crucial in scenarios where the design serves a security-critical role such as implementing an access control system. AKER uses a six-step security verification process following a property-driven hardware security methodology. The security verification process describes the

threat model, identifies security assets, articulates potential weaknesses, defines security requirements, specifies security properties, and verifies the security properties. To drive our discussion, we first consider the IP-level verification of the ACW, which consists of three entities: a single controller C, a single ACW which wraps C, and a single peripheral P. We assume that the ACW's local access control policy LACP is statically configured in RTL. In later sections, we apply the same verification process to implement different SoC access control systems. Section III-C performs firmware-level security verification adding a hardware root-of-trust to configure the ACW. Section III-D describes system-level security verification that ensures that multiple ACWs adhere to a global access control policy.

1) Create Threat Model: The first step in the security verification process develops the threat model. It is crucial to articulate the relevant security concerns. Hardware threats are vast and must be assessed based upon the usage of the hardware under design. We consider an integrity scenario where C is untrusted and the ACW and P are trusted. Therefore, the threat model considers C's ability to communicate with P via the ACW in a manner which does not adhere to the statically-configured LACP. Threats related to confidentiality are similarly possible given that confidentiality is a dual to integrity [7].

2) Identify Assets: The second step identifies the assets (i.e., design signals) that will be secured via the remainder of the verification process. Each asset will eventually have at least one associated security requirement/property that will be used to verify its security. Given our design and threat model, the assets we identify are the design signals that make up the five AXI channels which connect the ACW to C and P, and the design signals that make up the configuration and anomaly registers. In later sections, we will refer to these assets as the M AXI group and config/control group, respectively, due to the similarity in their security requirements.

3) Identify Potential Weaknesses: The third step determines potential weaknesses, which are defined as any mechanism that could introduce a security vulnerability relevant to the threat model and identified assets. Identifying these weaknesses is often challenging and time-consuming since it requires designers to understand the design's specification, the design's implementation, the subtleties in the correlation between these two, and which parts of the design are most relevant to the threat model. In an effort to increase the chance of identifying security critical weaknesses, we use the threat model and design to find relevant CWEs from MITRE's extensive database following to the CWE-IFT methodology [6].

4) Define security requirements: The fourth step in the process defines plain-language security requirements for the identified weaknesses. Once a mechanism is identified as a potential weakness, designers can articulate a security requirement which addresses how that mechanism could fail as determined from the relevant CWEs and an analysis of the design. For the M AXI group of weaknesses (i.e., the AXI channels) identified in Step 2, we develop security requirements addressing the existence and the content of information flows between C, the ACW, and P. Since the ACW sits between C and P, there will always be information flows between C and the ACW and the ACW and P. However, the source of these flows dictates their allowable behaviors. Information flows in which the source is C and the destination is P (or vice versa) should only occur when the ACW is in Supervising Mode and a legal transaction is issued. In all other instances, the only information flows that should occur are those in which the source is the ACW, the destination is either C or P, and the content of the f low does not deviate from the default AXI values we have selected.

5) Specify Security Properties: The fifth step in the process specif ies a security property template for each of the security requirements. In order to verify a security requirement, it must be manually converted into a formally specified security property which uses explicit values, design signals, and operators to form an evaluable expression. Rather than specifying nearly identical security properties for each design signal that should adhere to a given security requirement, AKER provides a property generation framework which automatically generates these specific properties given a single security property template with placeholder signals and a list of target design signals. For the security requirements relevant to the M AXI group (Requirement 1), the security property templates we specify are primarily information flow tracking IFT properties. IFT properties enable us to tag information from a particular source signal and track it as it f lows through our system [8]. For example, the send aspect of the security requirement from Requirement 1 is formalized below using the following template which fails if any information originating from C during active reset flows to P.

5 Conclusion

In this project, we have successfully designed and implemented a simplified access control framework modeled after AKER for integration with open-source system-on-chip development platforms. The simplified AKER module, which includes access control wrappers and a trusted entity managing unit, enforces access control within the ESP development platform while incurring minimal additional overhead. With the addition of AKER to these platforms, researchers may improve the robustness of their prototyped computer systems against data security attacks from malicious hardware accelerators.

References

[1] Esp: the open-source soc platform. https://www.esp.cs.columbia.edu/, Last accessed Jun 8, 2022.

[2] Esp: The open-source soc platform. https://www.youtube.com/watch?v=Ybf5wNYsit0, Last accessed Jun 8, 2022.

[3] Guide – how to: design an accelerator in rtl. https://www.esp.cs.columbia.edu/docs/rtl_acc/rtl_acc-guide/, Last accessed Jun 8, 2022.

[4] Vivado. https://www.xilinx.com/support/university/vivado.html, Last accessed Jun 8, 2022.

[5] Esp accelerator specifications. 2021.

[6] Sohrab Aftabjahani, Ryan Kastner, Mark Tehranipoor, Farimah Farahmandi, Jason Oberg, Anders Nordstrom, Nicole Fern, and Alric Althoff. Special session: Cad for hardware security-automation is key to adoption of solutions. In *2021 IEEE 39th VLSI Test Symposium (VTS)*, pages 1–10. IEEE, 2021.

[7] Kenneth J Biba. Integrity considerations for secure computer systems. Technical report, MITRE CORP BEDFORD MA, 1977.

[8] Wei Hu, Armaiti Ardeshiricham, and Ryan Kastner. Hardware information flow tracking. *ACM Computing Surveys (CSUR)*, 54(4):1–39, 2021.

[9] Natalie Jerger, Tushar Krishna, and Li-Shiuan Peh. *On-Chip Networks: Second Edition*. 2017.

[10] Francesco Restuccia, Andres Meza, and Ryan Kastner. Aker: A design and verification framework for safe and secure soc access control. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2021.

[11] Agam Shah. 7 dazzling smartphone improvements with qualcomm's snapdragon 835 chip. 2017.