AKER: Milestone Report

Chi Chow, Brandon Erickson, Hosein Yavarzadeh

May 19, 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 are developing an adaptation of AKER onto network-on-chip (NoC) applications to form a high-bandwidth access control wrapper (ACW). Our adaptation is to be implemented onto ESP, an open-source rapid SoC development platform created by the University of Columbia. We have successfully set up the ESP project, and will integrate a simplified AKER access control module into the project in the coming weeks.

1 Introduction

Modern System on a Chip (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 – 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. This Project is going to implement the AKER access control system on the ESP platform which is an open source platform for heterogeneous system-on-chip design that combines a tile-based architecture and a flexible system level design methodology.

2 Implementing Access Control for NoC

2.1 The ESP Development Platform and Architecture

The target platform for this project is ESP, 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 NoC for packet-based data transfer. 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.

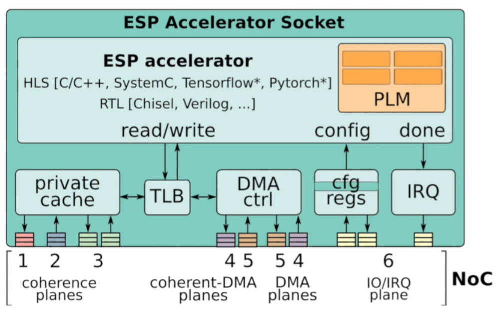
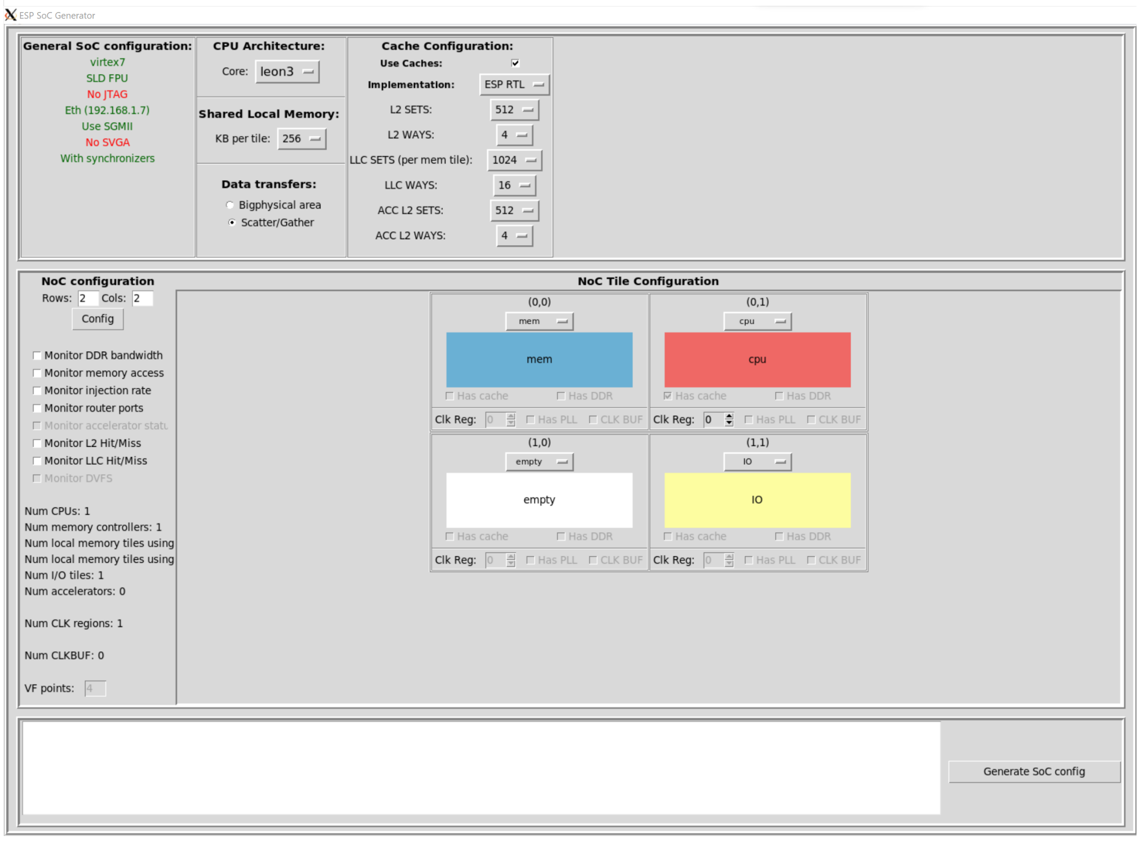


Figure 1: ESP Accelerator Socket



General SoC configuration:
virtex7
SLD FPU
No JTAG
Eth (192.168.1.7)
Use SGMII
No SVGA
With synchronizers

CPU Architecture:
Core: leon3

Shared Local Memory:
KB per tile: 256

Data transfers:
Bigphysical area [✓ Scatter/Gather]

Cache Configuration:
Use Caches: ✓
Implementation: ESP RTL
L2 SETS: 512
L2 WAYS: 4
LLC SETS (per mem tile): 1024
LLC WAYS: 16
ACC L2 SETS: 512
ACC L2 WAYS: 4

NoC configuration
Rows: 2 Cols: 2

- [ ] Monitor DDR bandwidth
- [ ] Monitor memory access
- [ ] Monitor injection rate
- [ ] Monitor router ports
- [ ] Monitor accelerator statu
- [ ] Monitor L2 Hit/Miss
- [ ] Monitor LLC Hit/Miss
- [ ] Monitor DVFS

Num CPUs: 1
Num memory controllers: 1
Num local memory tiles using
Num local memory tiles using
Num I/O tiles: 1
Num accelerators: 0

Num CLK regions: 1

Num CLKBUF: 0

VF points: 4

NoC Tile Configuration

(0,0)	mem	Has cache ☐ Has DDR ☐ Clk Reg: 0 Has PLL ☐ CLK BUF ☐
(0,1)	cpu	Has cache ✓ Has DDR ☐ Clk Reg: 0 Has PLL ☐ CLK BUF ☐
(1,0)	empty	Has cache ☐ Has DDR ☐ Clk Reg: 0 Has PLL ☐ CLK BUF ☐
(1,1)	IO	Has cache ☐ Has DDR ☐ Clk Reg: 0 Has PLL ☐ CLK BUF ☐

Figure 2: ESP's GUI to configure SoC tiles. We plan to add an option to select using AKER access control during SoC configuration.

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.

2.2 ESP Architecture Implementation

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

- Generate SoC configuration
- Create caches
- Run RTL simulation

We will implement AKER's functionality within the RTL code base of the ESP repository. The 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.

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

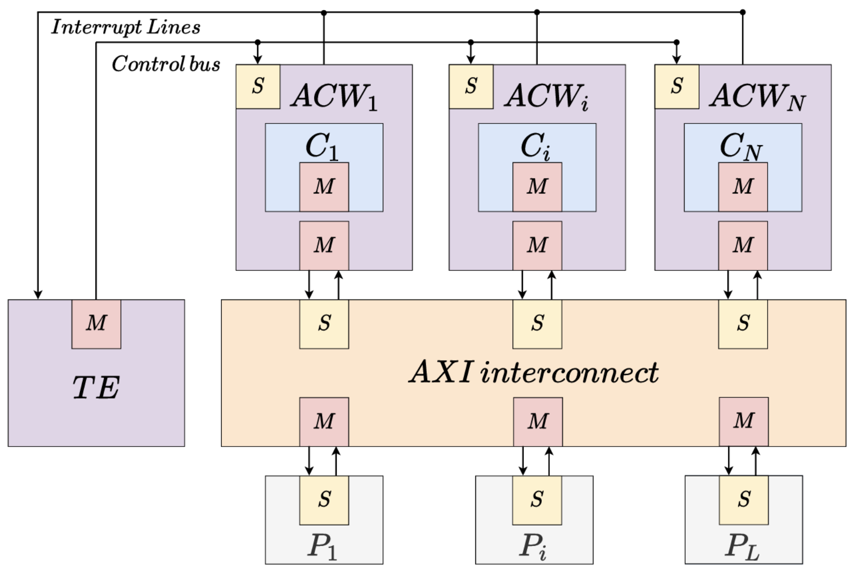


Figure 3: Extended AXI multi-controller, multi-peripheral architecture incorporating an AKER-based access control system. The Trusted Entity TEmanages the ACWmodules. Only legal requests are transmitted to the AXI interconnect, i.e., the peripherals receive only legal AXI transactions.

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.

2.3.2 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).

2.3.3 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.

2.3.4 Simplified Version of AKER

Detailed structure of the ACW is shown in Figure **??**. 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.

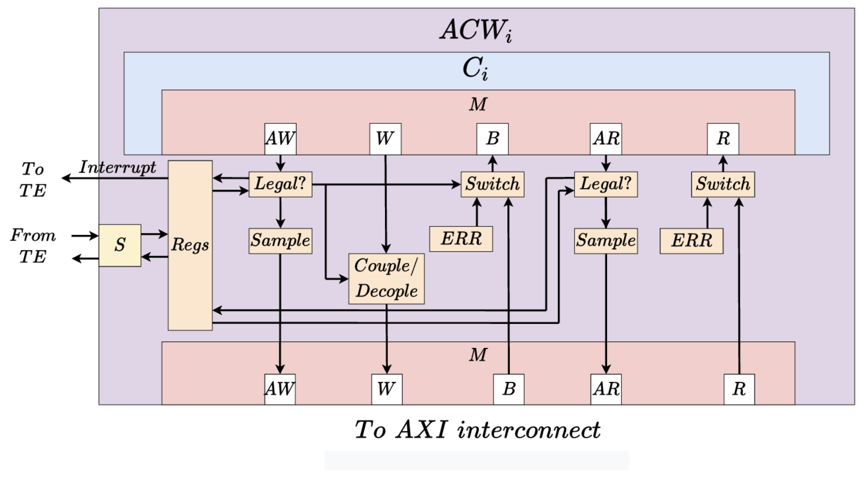


Figure 4: ACW_i architecture: C_i is the controller module using an AXI M interface. Regs are the configuration registers holding $LACP_i$. The AXI S interface is connected to the $HRoT$.

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

We will implement the simplified version of the AKER on ESP platform in the next phase!

3 Project Progress

Our progress on this project is in line with milestones provided in the Project Specification. Over the past two weeks, we have researched the ESP architecture and formulated a plan for AKER's integration into the socket layer. We have also completed setup of the ESP project and have performed simple experiments through simulation. Within the next two weeks, we will create a simplified AKER access control wrapper for integration into the ESP project.

3.1 Project Milestones and Updates

An updated list of project milestones and their completion status are provided below.

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 Sections 2.1 and 2.2 for accelerator specification and evaluation.
2. ESP Project Modifications (Brandon)
   - 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: In progress. ESP's socket layer provides a modular interface for third-party accelerators, so it is expected that few modifications to ESP are necessary for AKER's integration.
3. AKER Project Simplification (Chi, 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: In progress. See Section 2.3 for details on the AKER framework and its components.
4. Implementation of AKER on ESP (Brandon, 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: Not Started. Modifications to ESP and AKER simplification must be completed prior to integration.
5. Final Evaluation and Debugging (Brandon, 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: Not Started.

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: Not Started.

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: Not Started.