

From Sandboxes to Zero-Trust: Securing Multiprocessor Systems on Chip Through Isolation

06-25-2026

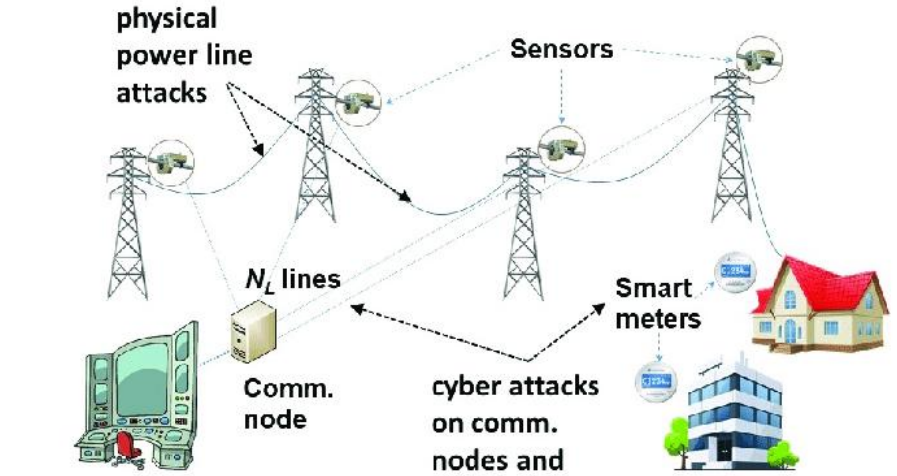
Dr. Christophe Bobda

Agenda

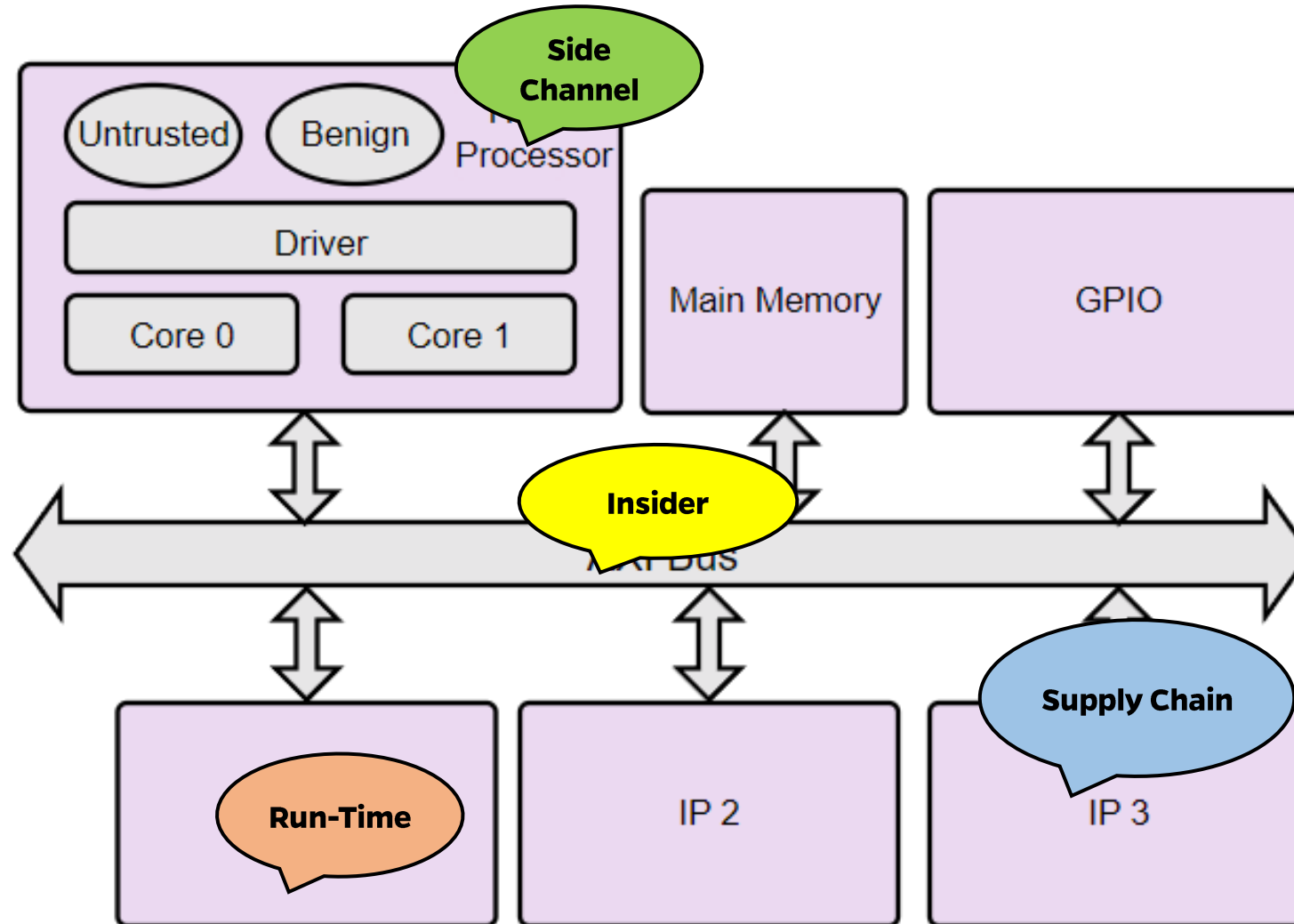
- Motivation
- SoC Security
- Isolation
- Hardware Sandbox
- FLASK-Based MAC
- Zero-Trust Design
- Physical Isolation

Introduction and Motivation

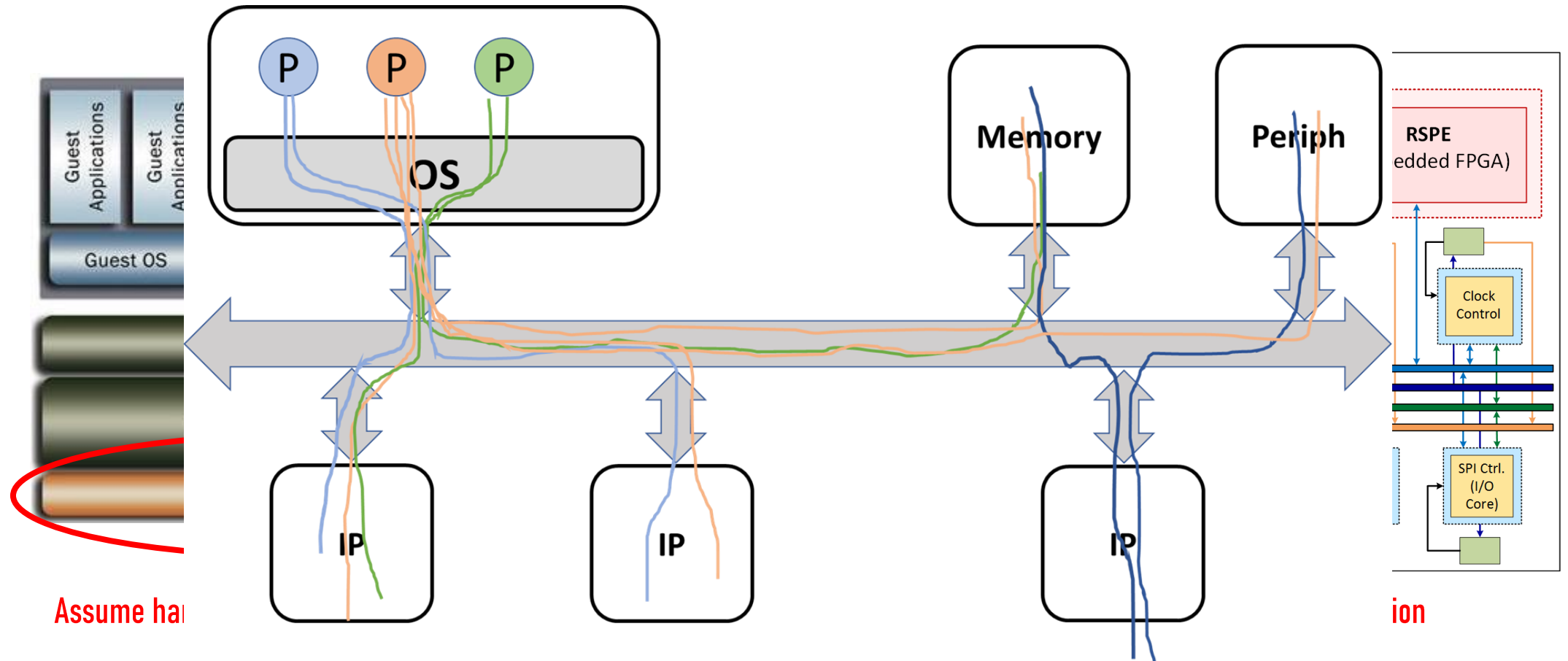
- Computing Systems at The Heart of Critical Systems and Infrastructure
- Increasingly connected



SoC-Centric Solutions



(Embedded) Computing Systems Security

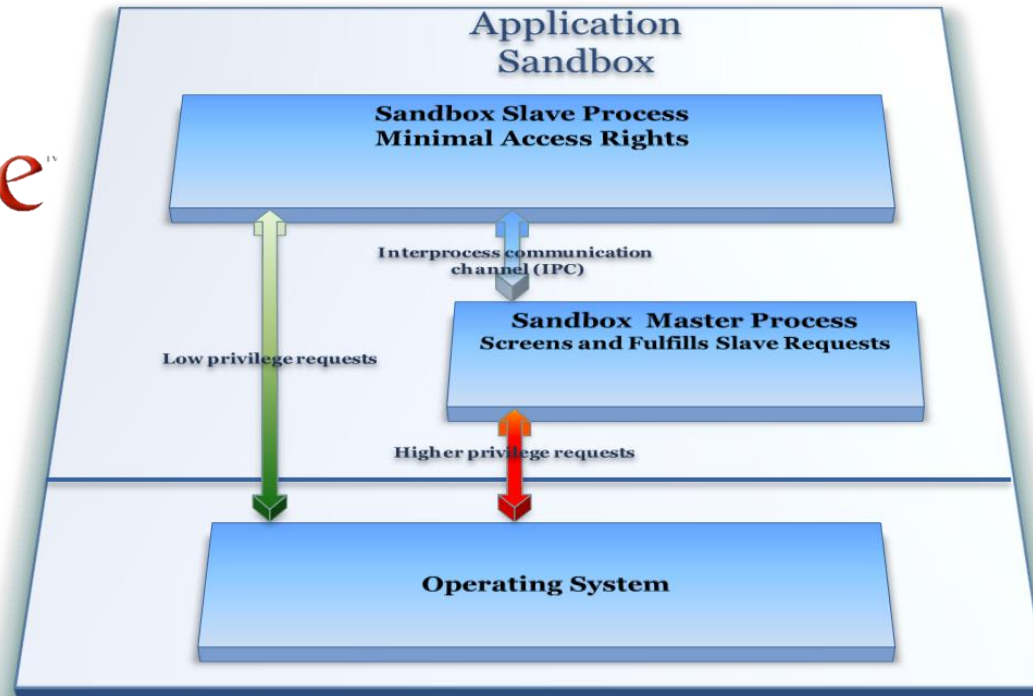


Isolation

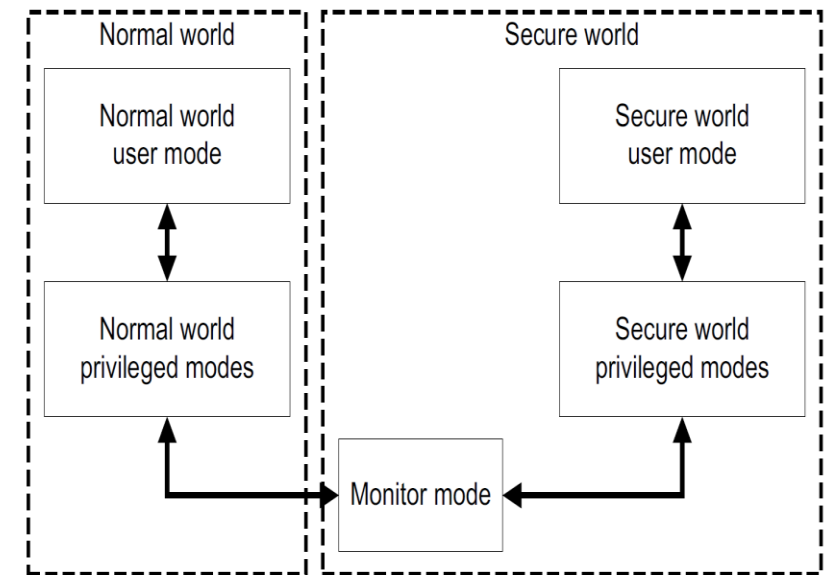
- Deception
- Separation
- Diversity
- Consistency
- Depth
- Discretion
- Collection
- Correlation
- Awareness
- Response



Separation

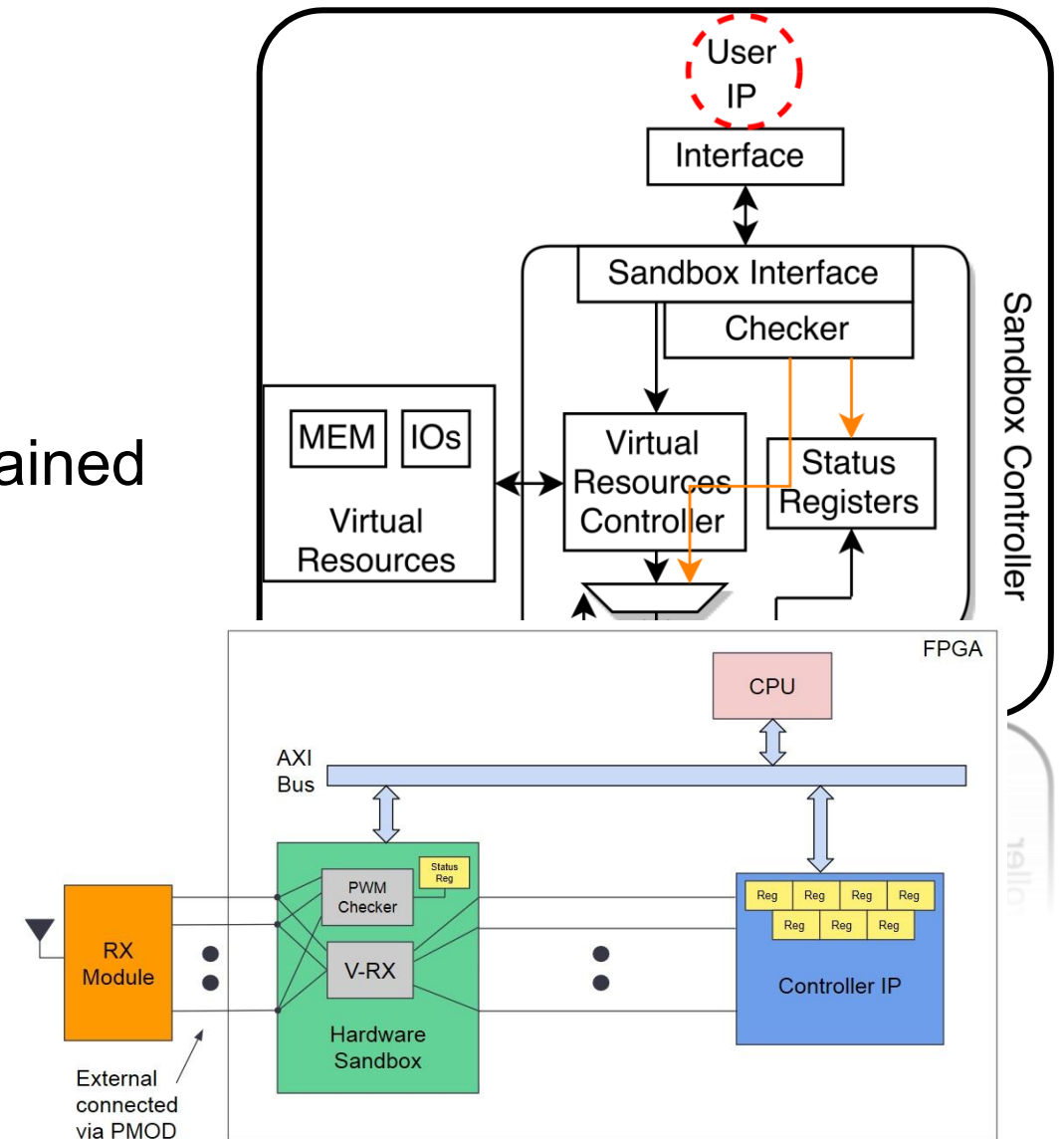
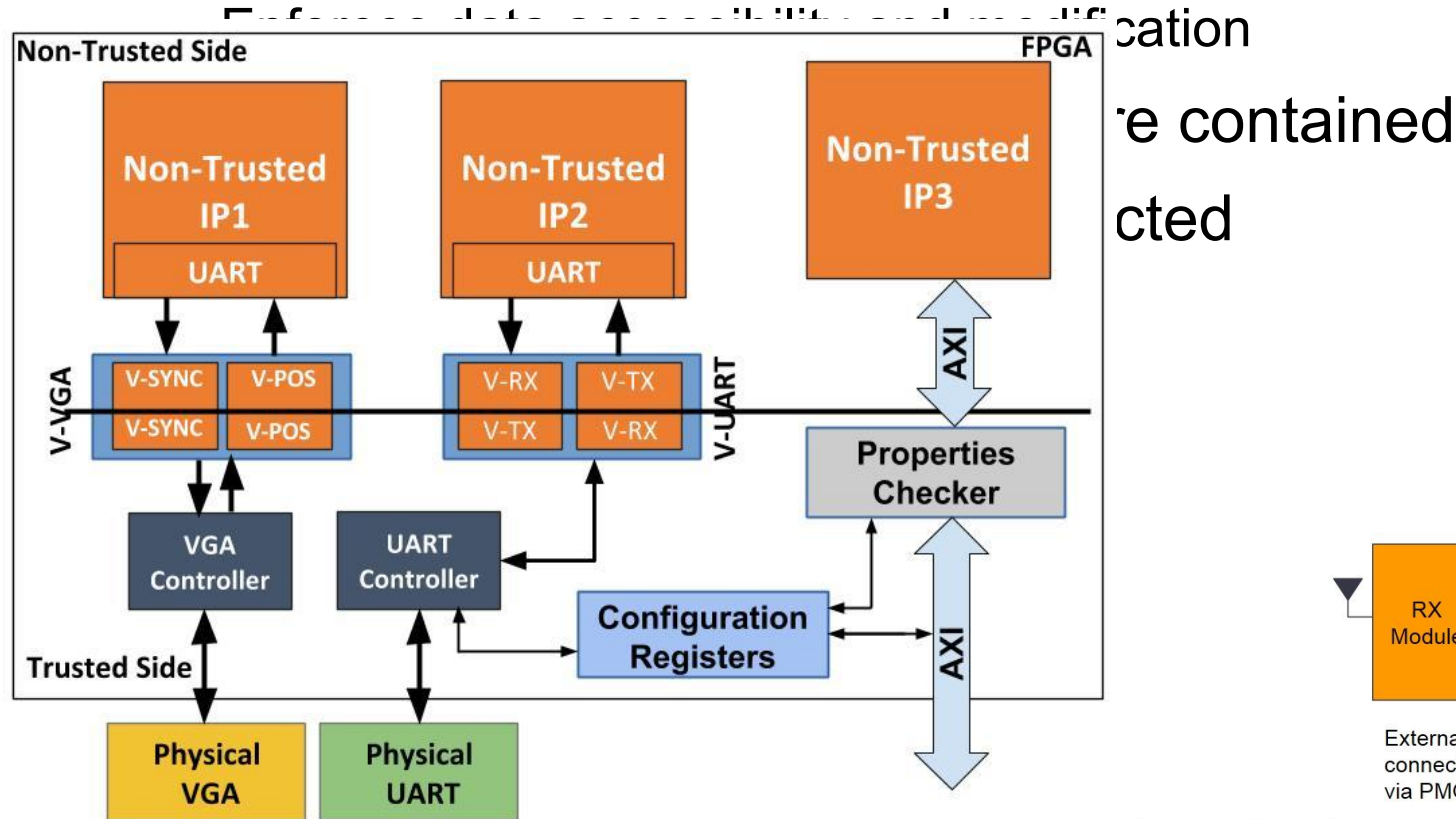


ARM®



Secured Integration of Non-Trusted Components

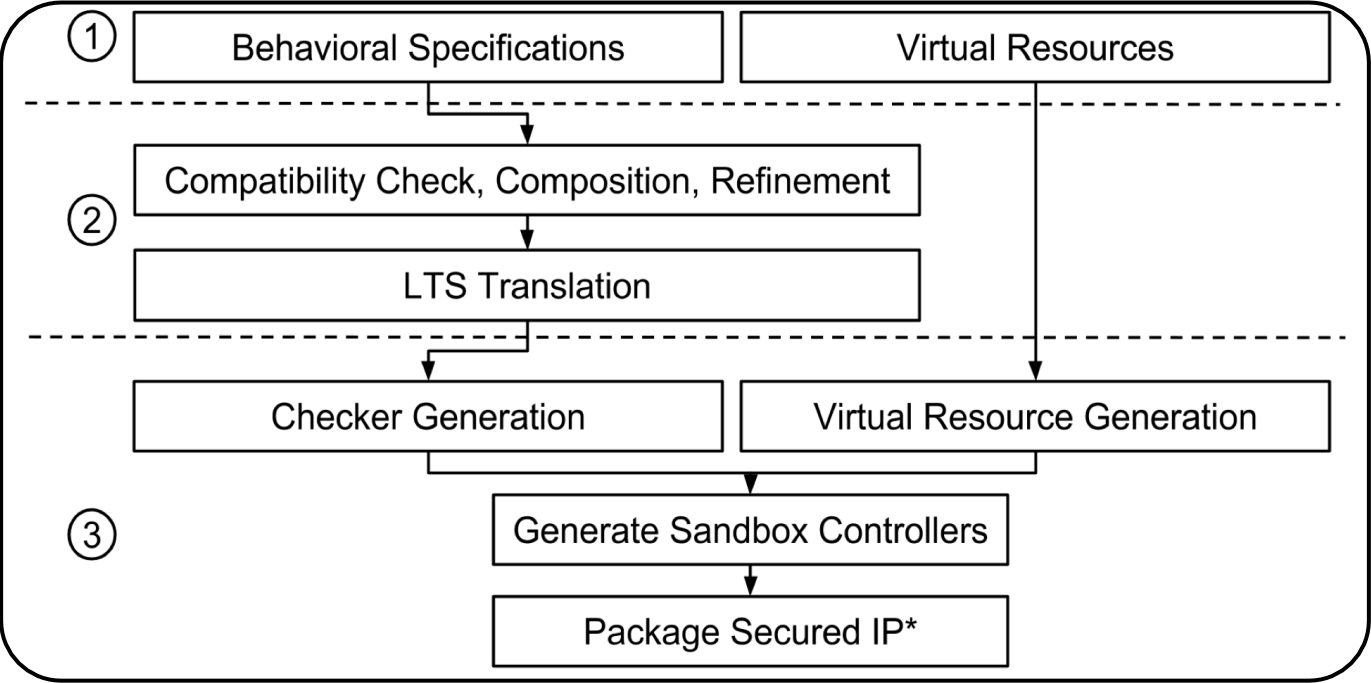
- Hardware Sandboxing
- Isolation, Integrity, Containment
 - IP limited to specified interactions



Secured Integration of Non-Trusted Components



Component Authentication Process for Sandboxed Layouts



```
<IP_name>.def{
  // Signal types: input, output
  <signal_name> : <signal_type>, // Interface ports
  <vector_name> : <signal_type>(<length>),

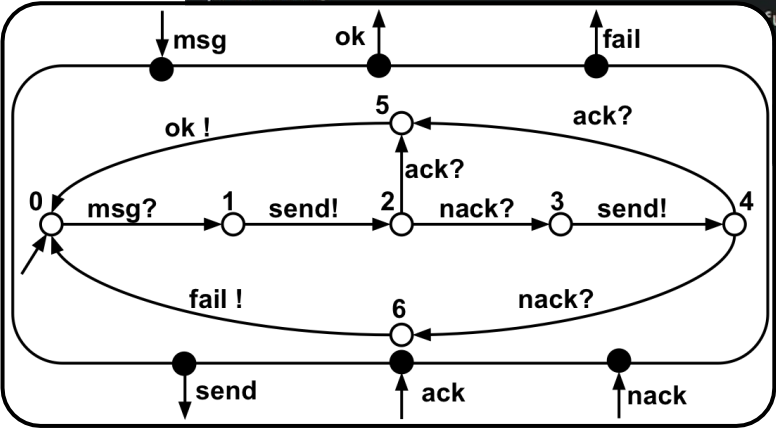
  transitions{
    s0      : <signal_name> : s<integer>, // s0 always initial state
    s<integer> : !<signal_name> : s<integer>,
  },

  resources{
    <resource_name> : { <option_name> : <option_setting> }
  }
}
```

```
logic.def{
  // Logic operators: and, or, ==, !=
  <signal_name> : <signal_name> <logic_operator> <signal_name>,

  <signal_name> : counter { // Counter example
    on: <signal_name>,
    start: <integer>,
    end: <integer>
  },

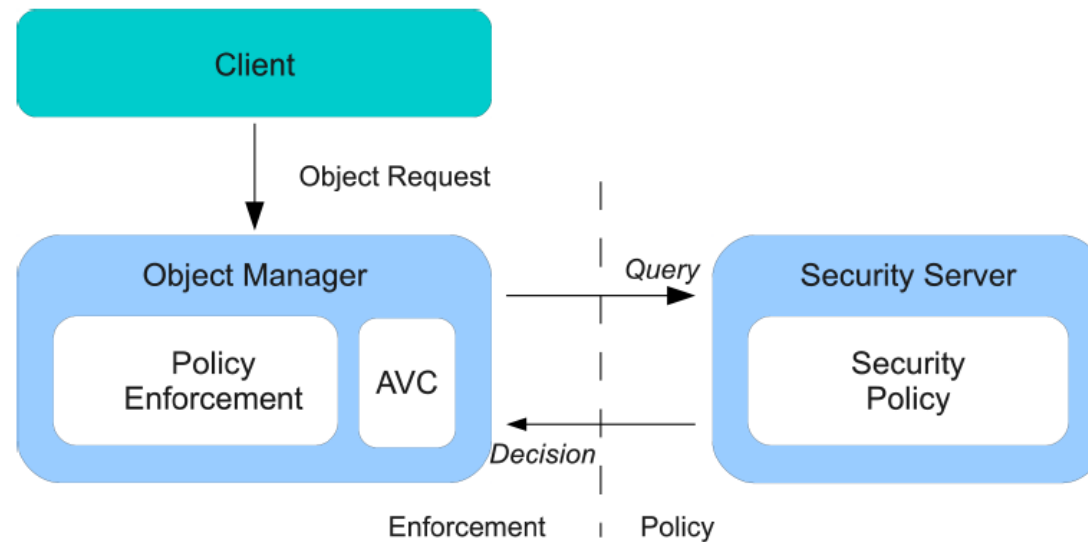
  prohibited{
```



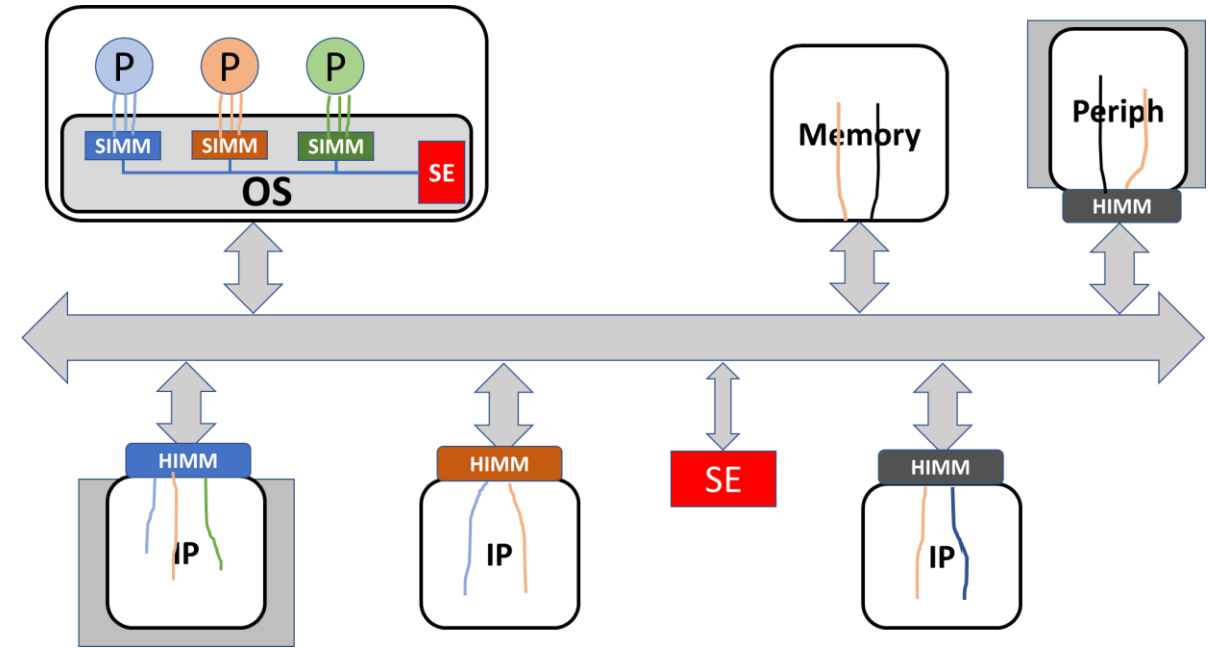
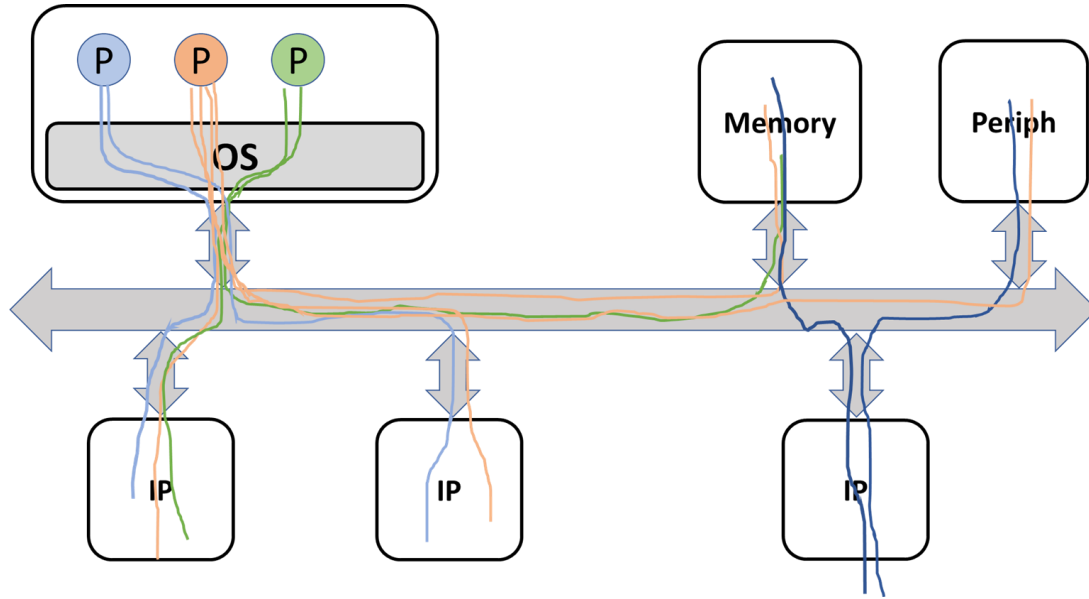
fusion
concatenation
leane star
on-consecutive repeat

FLASK-Based Mandatory Access Control

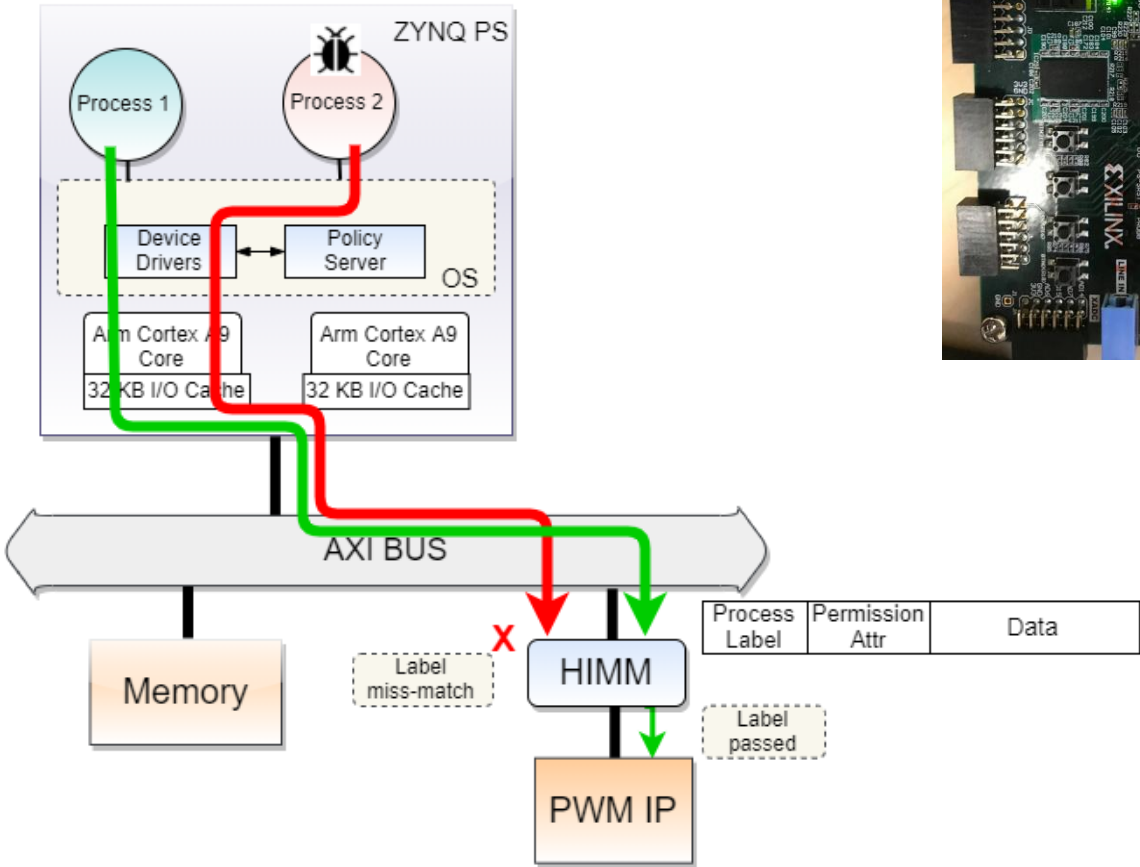
- Flux Advanced Security Kernel (FLASK) - Security Architecture
 - Open-source developed by NSA and University of Utah to provide flexible support for mandatory access control (MAC) policies in operating systems.
 - SELinux served as the template for other Flask implementations in other operating systems (Android, Xen, etc).



Target Architecture



Case Study



Linux terminal on the target SoC.

```
[root]:~  
[root]:~ ./binary 100  
current user root  
Writing to the memory...  
Operation Complete  
[root]:~ ./binary 0  
current user root  
Writing to the memory...  
Operation Complete  
[root]:~ ./binary 10  
current user root  
Writing to the memory...  
Operation Complete  
[root]:~ ./binary 2  
current user root  
Writing to the memory...  
Operation Complete  
[root]:~
```

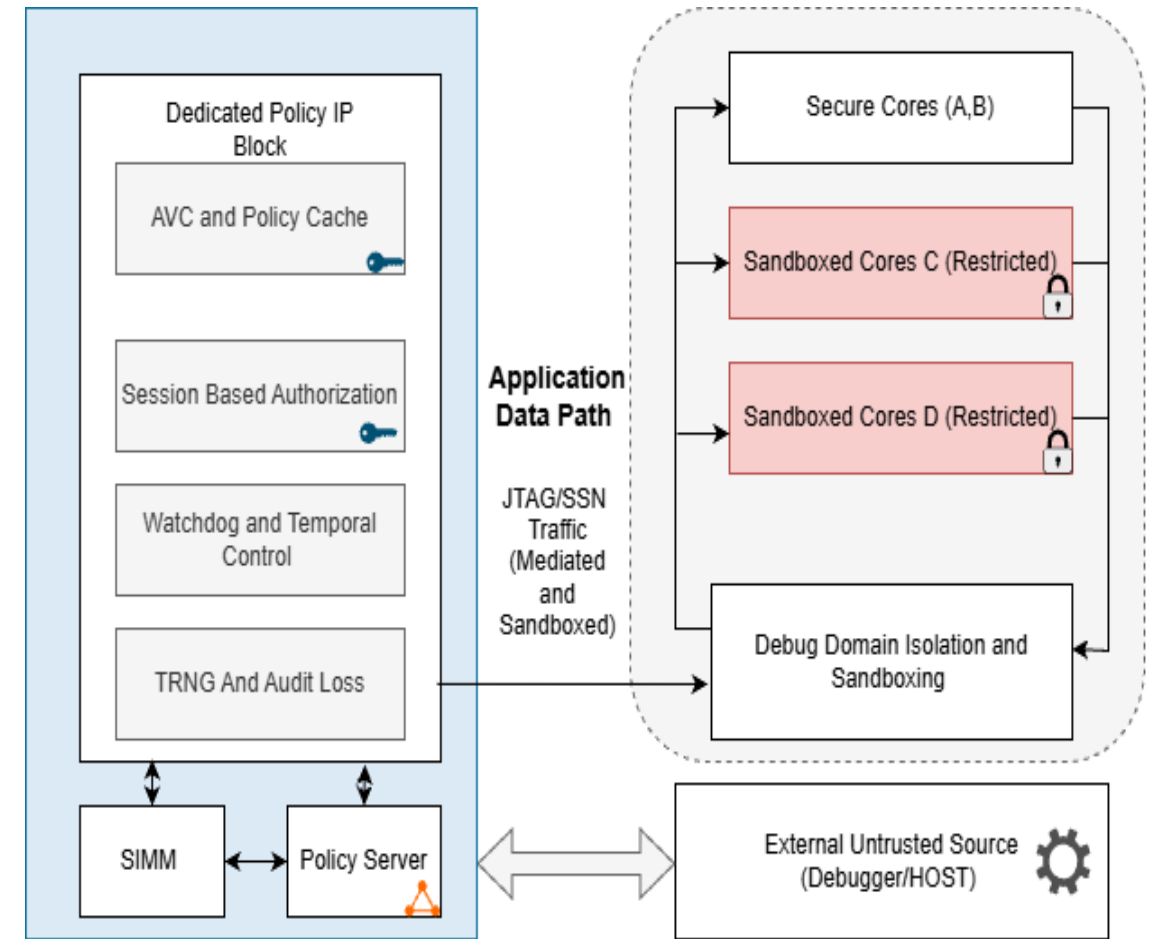


```
[malware]:~  
[malware]:~ ./binary 100  
current user malware  
Writing to the memory...  
Operation Complete  
[malware]:~ ./binary 0  
current user malware  
Writing to the memory...  
Operation Complete  
[malware]:~ ./binary 10  
current user malware  
Writing to the memory...  
Operation Complete  
[malware]:~ ./binary 2  
current user malware  
Writing to the memory...  
Operation Complete  
[malware]:~
```

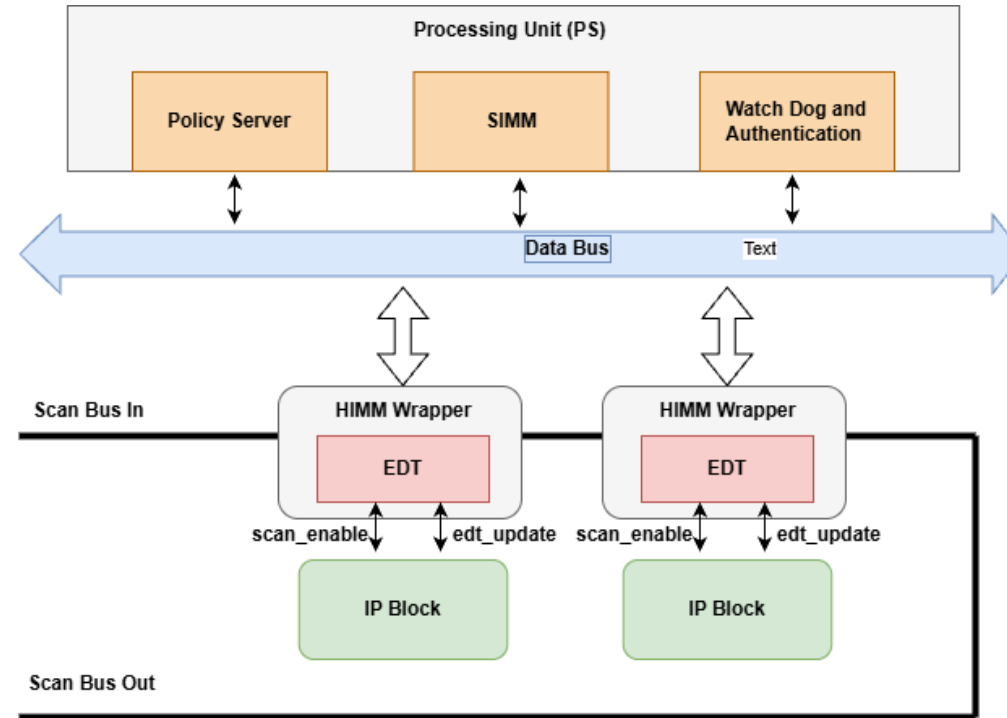


Need of Secure SSN

- **The Vulnerability:** Modern Streaming Scan Networks (SSN) lack native cryptographic authentication, exposing critical IP to data sniffing and lateral movement via JTAG TAP.
- **The Solution:** A Sandbox-Based Secure TAP (STAP) architecture that structurally decouples the security control plane from the test data plane.
- **Core Mechanisms:** Utilizes hardware-enforced instruction filtering, short-lived session authorization, and strict debug domain sandboxing



STAP Architecture



Decoupled Design

A centralized Software IP Management Module (SIMM) manages policies, while distributed Hardware (HIMM) wrappers act as local firewalls at wire-speed.

Zero Trust Access

Successfully isolates debug domains and neutralizes lateral scan-chain abuse by dropping unauthorized opcodes directly at the hardware layer.

High Performance

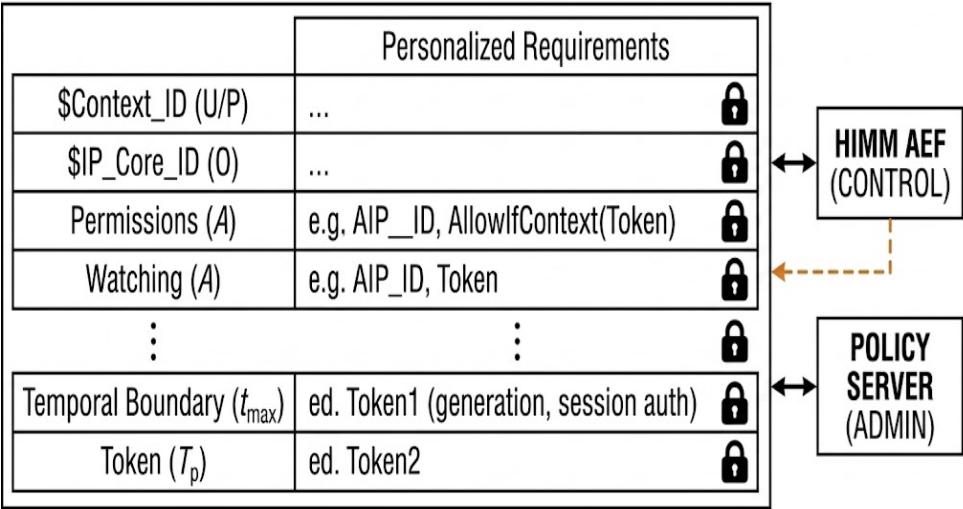
Achieves superior scalability with negligible overhead: only 0.48% area utilization per node and a minimal 0.022 μ s delay per debug transaction.

Policy Server and Context Authorization

- 1. We utilize a FLASK/SELinux-style mapping where the debugger (subject) and the SSN instrument(object) are assigned security contexts. The AEF performs an access check on every Instruction (IR) load.
- 2. Shows very little overhead in implementation
- 3. SIMM:
 - Validates the external credentials and generates a high-entropy Session Token.
 - Maps the high-level security policy to a hardware-compatible Context ID.
 - Communicates with the PolicyServer to update the HIMM's local cache upon a miss.

TABLE I: FPGA Resource Utilization of the Secure SSN System

Xilinx Zynq XC7Z010	LUTs	FFs	LUTRAMs
Available	17600	35200	6000
Used Total	2243	2290	100
- Used by four Instrumented SSHs	1899	1990	76
- Used by four HIMMs (Mediators)	344	300	24
Total Utilization	12.74%	6.5%	1.66%
- Utilized by SSHs	10.78%	5.65%	1.26%
- Utilized by HIMMs (Secure IP)	1.95%	0.85%	0.4%



Zero-Trust

- The Reality of Modern SoCs: Heterogeneous systems increasingly rely on third-party hardware attack

BeyondCorp A New Approach to Enterprise Security

RORY WARD AND BETSY BEYER

- **Dynamic Compromise:** Detect, contain, and isolate adversarial activity from the system core.
- **Strict Mission Assurance:** The system must continue to perform its primary operational mission, even while actively defending against an architecture attack.



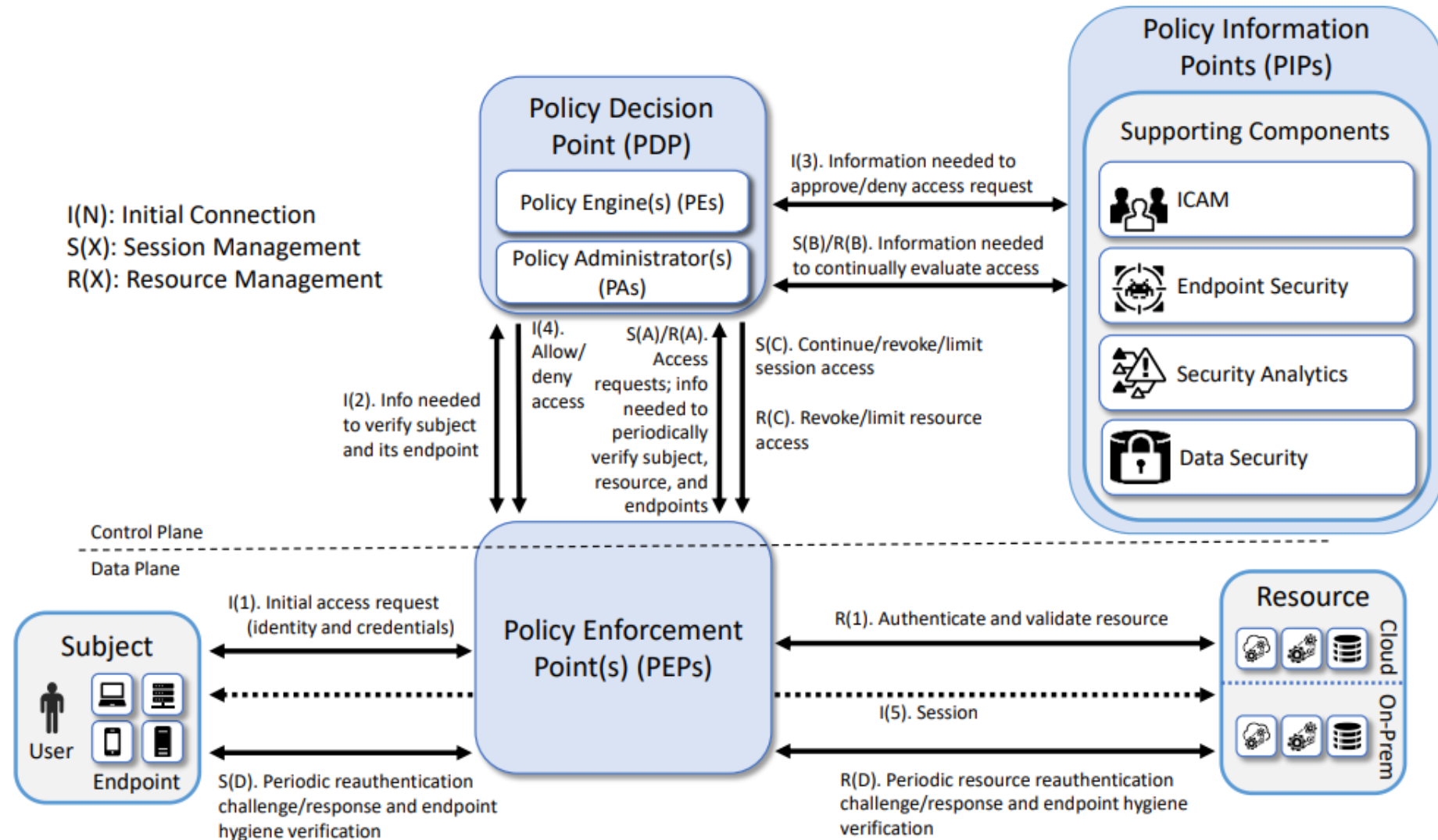
Zero Trust for AI Agents

A security framework for deploying autonomous AI agents in the enterprise.

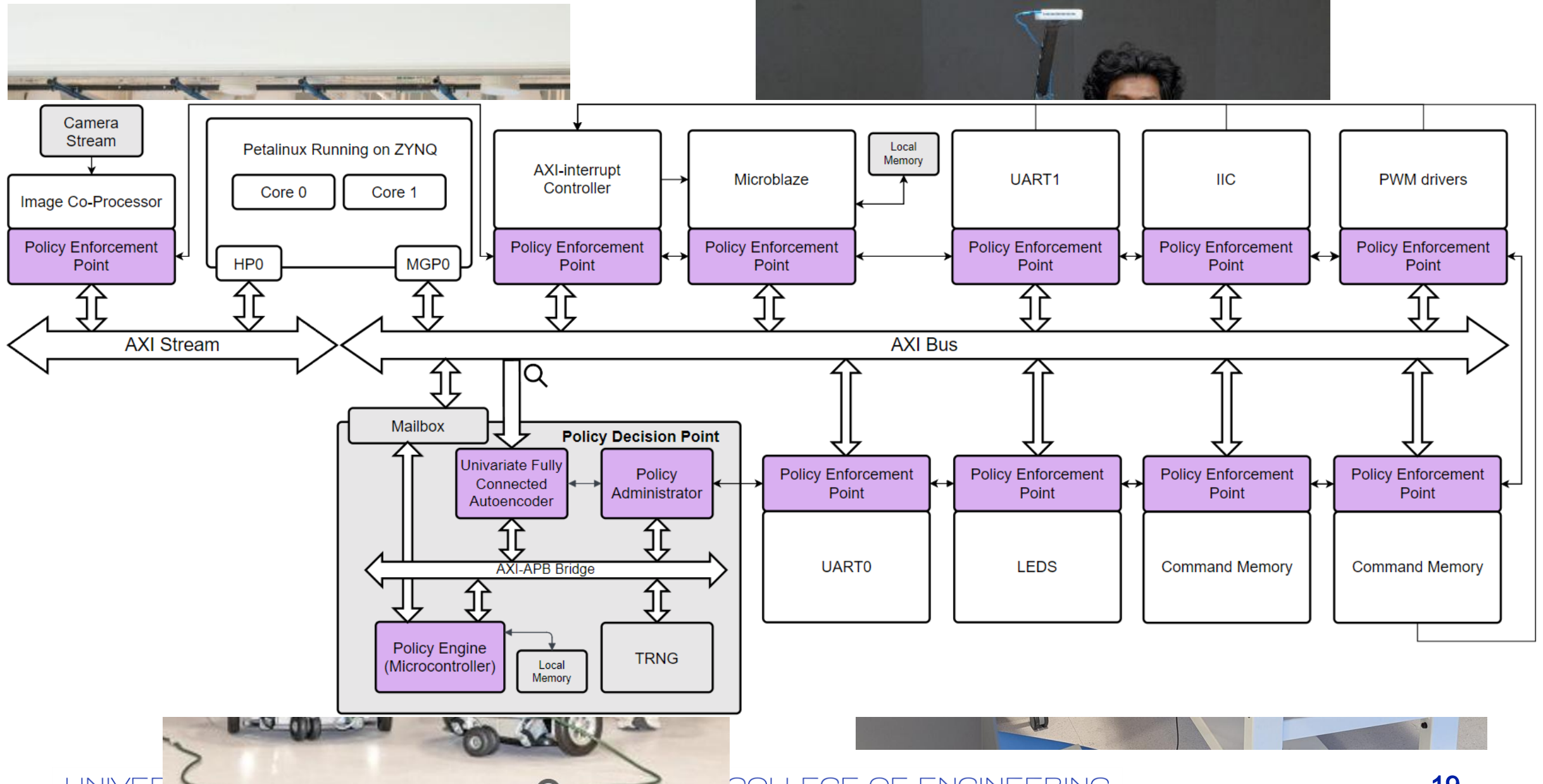
Zero-Trust

- **Assume Breach:** Treat every component, third-party IP block, or interconnect as if it is already compromised or malicious.
- **Verify Explicitly:** Always authenticate and authorize based on all available data points (identity, location, device health, anomalies) before granting access or executing a command.
- **Use Least Privileged Access:** Limit user or component access with Just-In-Time and Just-Enough-Access (JIA/JTA) models to protect data and minimize the blast radius if something goes wrong.

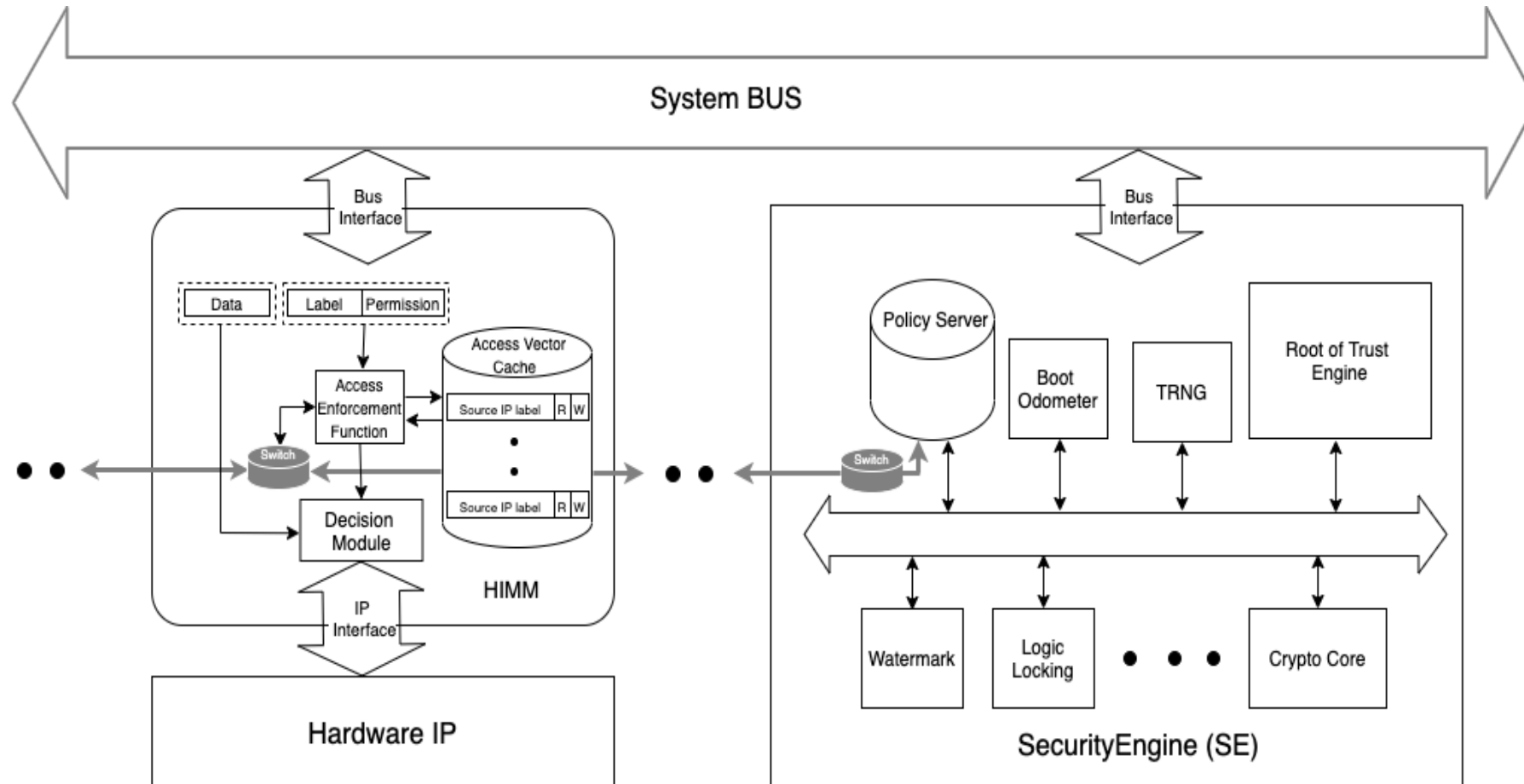
Zero-Trust in MPSoCs – NIST-Inspired



Zero-Trust in MPSoCs – NIST-Inspired

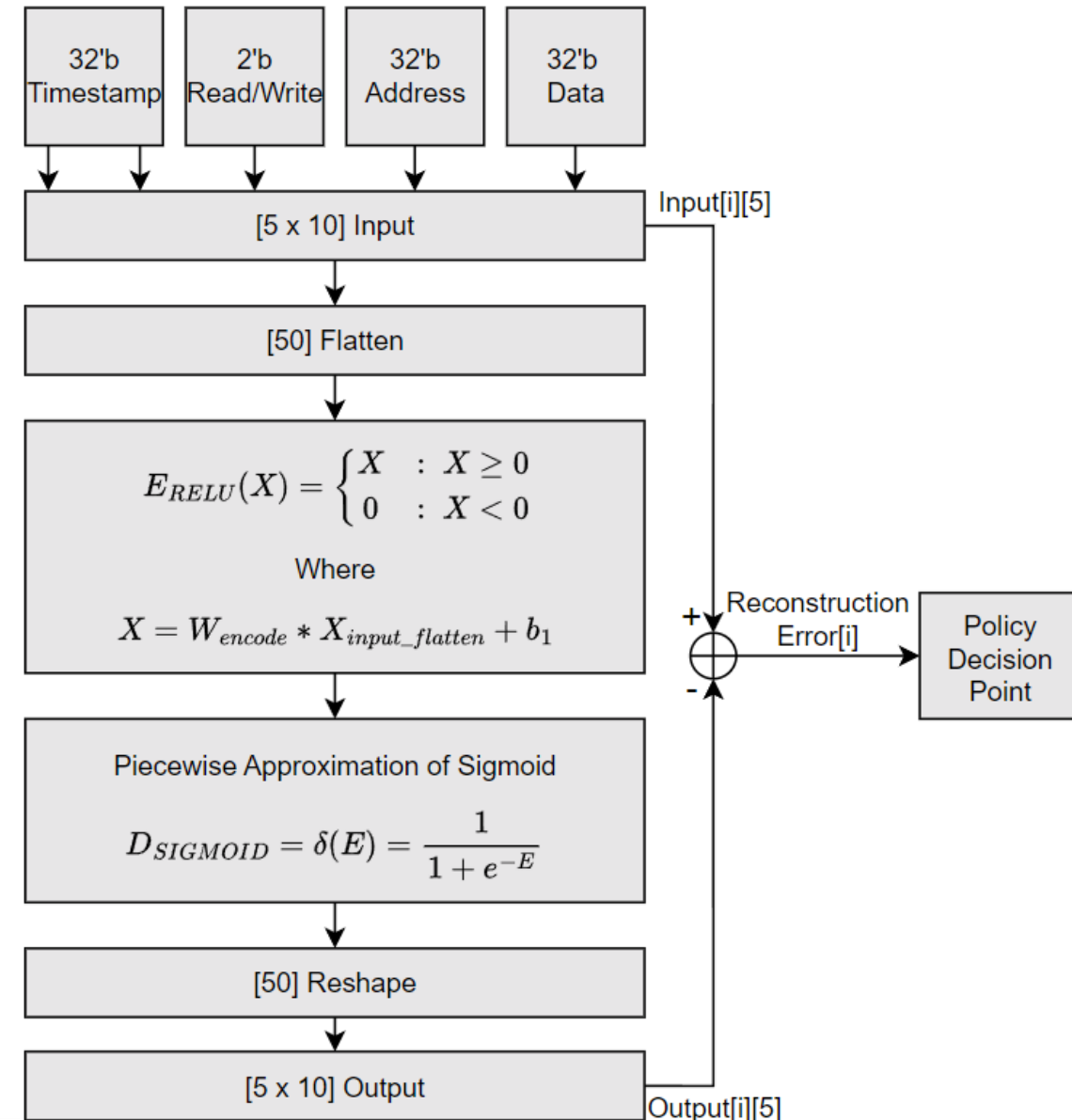


Zero-Trust in MPSoCs – Policy Decision & Enforcement



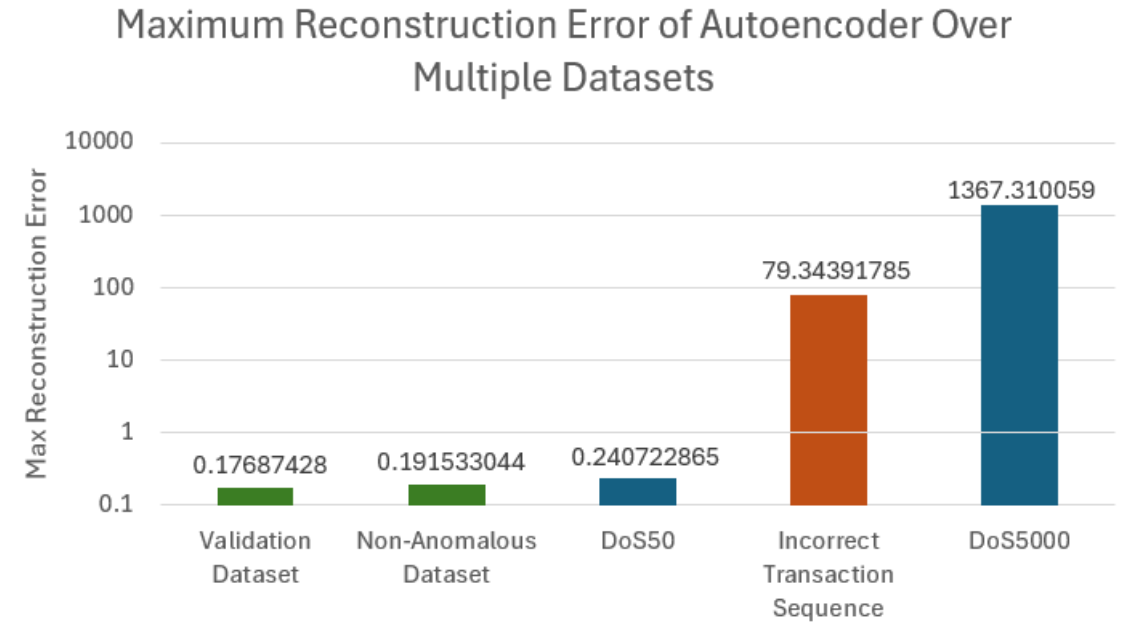
Zero-Trust in MPSoCs – Anomalies Detection

- **Autoencoder Goal:** Compress AXI-Lite transaction data down to key features, then reconstruct the original data from those features.
- Normal behavior: Input == Output
- Input Data: 32'b Timestamp Delta, 2'b Read or Write flag, 32'b Address, and 32'b Data
- Reconstruction Error = MSE(Input-Output)
- Implemented using FastML's HLS4ML [11,12]



Anomaly Detection: Results

- Collected six datasets
 - Validation
 - Non-Anomalous
 - Incorrect Transaction Sequence
 - Denial-of-Service (rand 50 pcc)
 - Denial-of-Service (rand 5000 pcc)
- Maximum reconstruction error of DoS 5000 was ~7,139x larger than non-anomalous data
- Incorrect sequence 414x larger than non-anomalous
- The threshold was established at the 95th percentile of the reconstruction error values



Resource, Power, and Latency

- Finally, resource utilization is quite large, even for a very small autoencoder implementation with only 656 trainable parameters
- Power utilization was minimal, under 984mW
- The Autoencoder can process approximately 360,896 samples/sec at 100 MHz, or ~28 cycles per sample
 - Total transaction time in a secured system is assumed between 25-29 cycles
 - +/- 3 clock cycles

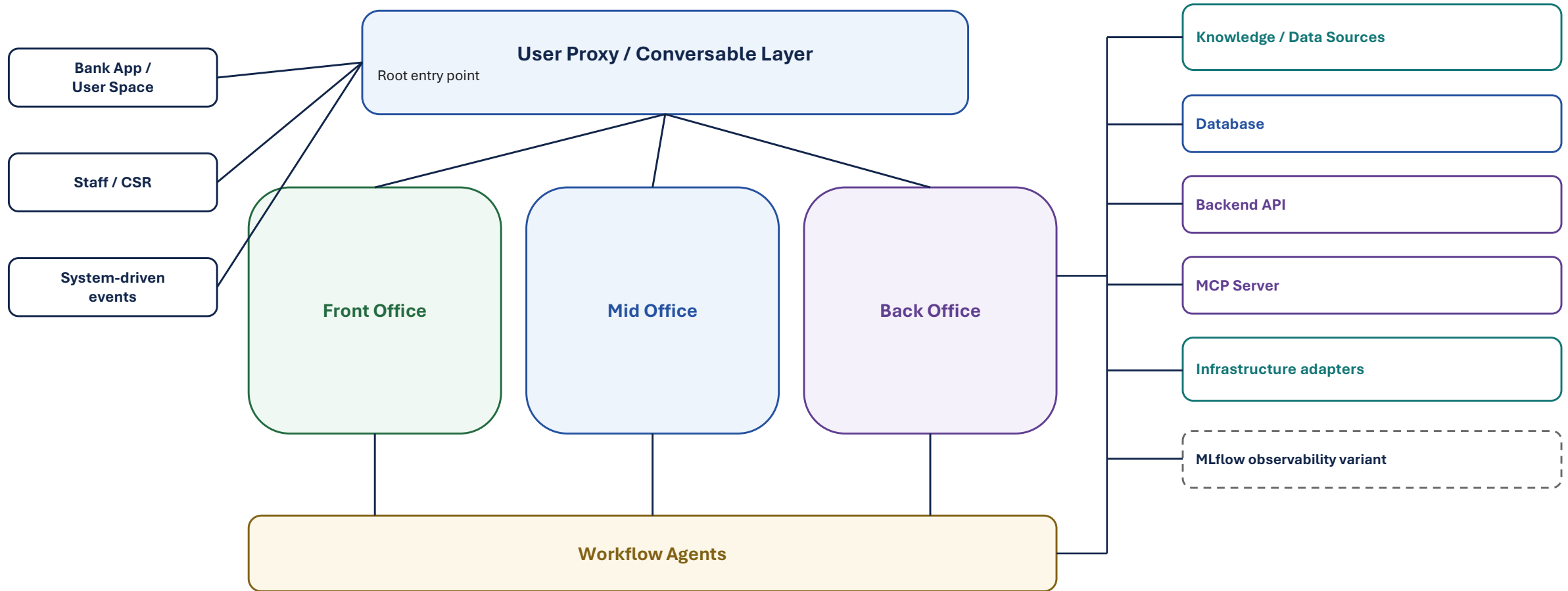
Table 4-2. FPGA Resource Utilization of Implemented Autoencoder

Resource	Utilization	Available	Utilization %
LUT	31354	53200	58.94%
FF	25261	106400	23.74%
BRAM	12.5	140	8.9%
DSPs	220	220	100%

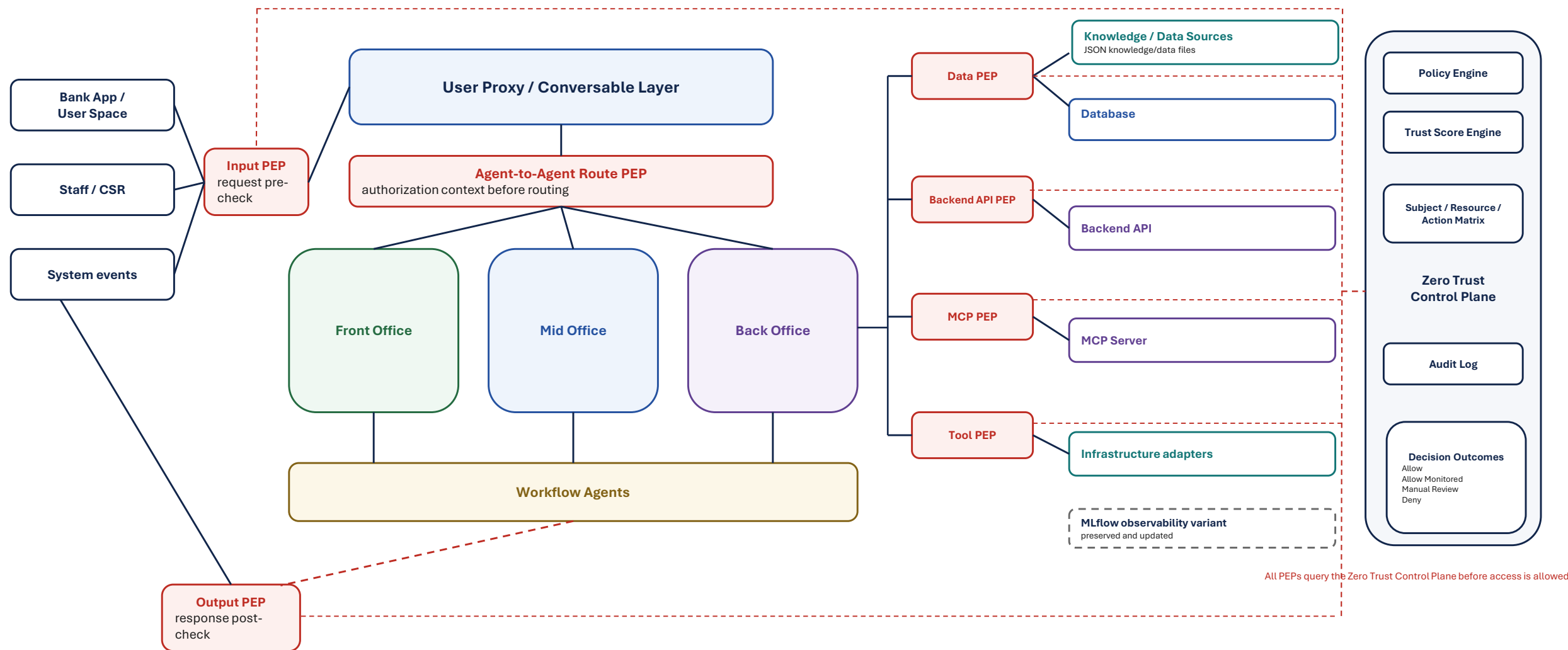
Table 4-3. Autoencoder Power Consumption in PYNQ-Z2

	Component	Power (W)	Power (%)
5*Dynamic	Clocks	0.130	7%
	Signals	0.278	39.7%
	Data	0.276	39.5%
	PL Logic	0.218	31.2%
	DSP	0.082	11%

ZTA in Agentic AI Systems



ZTA in Agentic AI Systems



What changed			
Original logic preserved	PEP at every trust boundary	Continuous trust evaluation	Least privilege + auditability

FPGA Cloud - GatorReCC



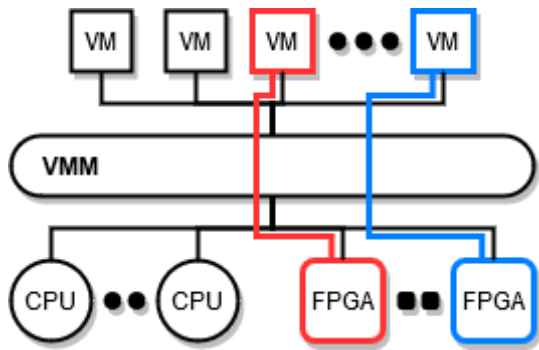
Dell EMC server with Alveo u50 board

<https://smartsystems.ece.ufl.edu/research/projects/gatorrecc/>



Multi-Tenancy FPGA Clouds

Single-tenant FPGAs

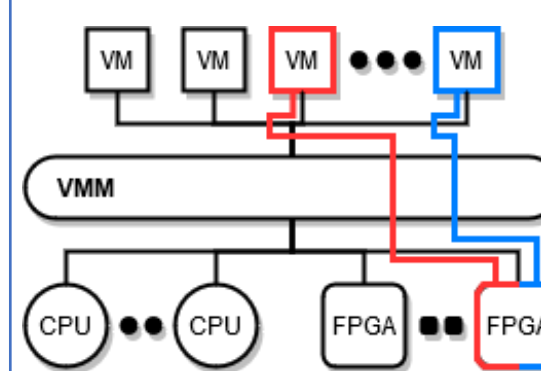


Each FPGA is entirely allocated to a VM over time.

Limitation

Resource waste and low FPGA utilization as user designs do not always use 100% of the FPGA.

Multi-tenant FPGAs



FPGA resources are time and space-shared between user workloads.

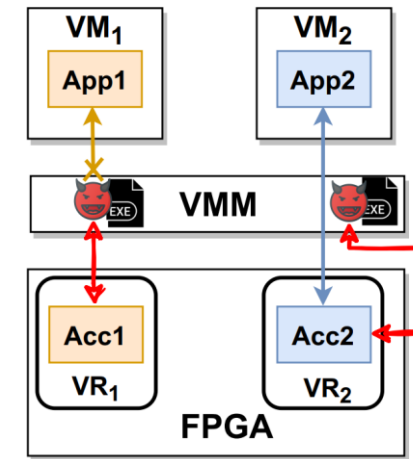
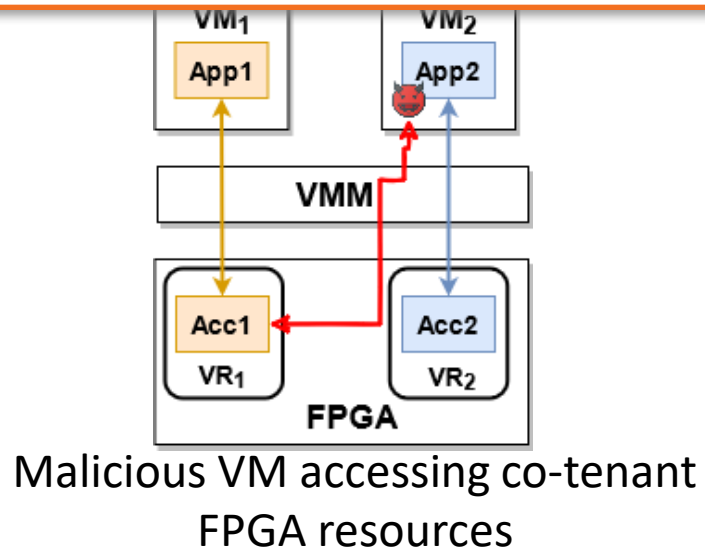
Advantage

Improved utilization of FPGA resources across cloud workloads.

Threat Model

- **Scenario1:** a VM gains access to the address space of hardware accelerators in the domain of another VM. e.g: VM2 accesses VR1

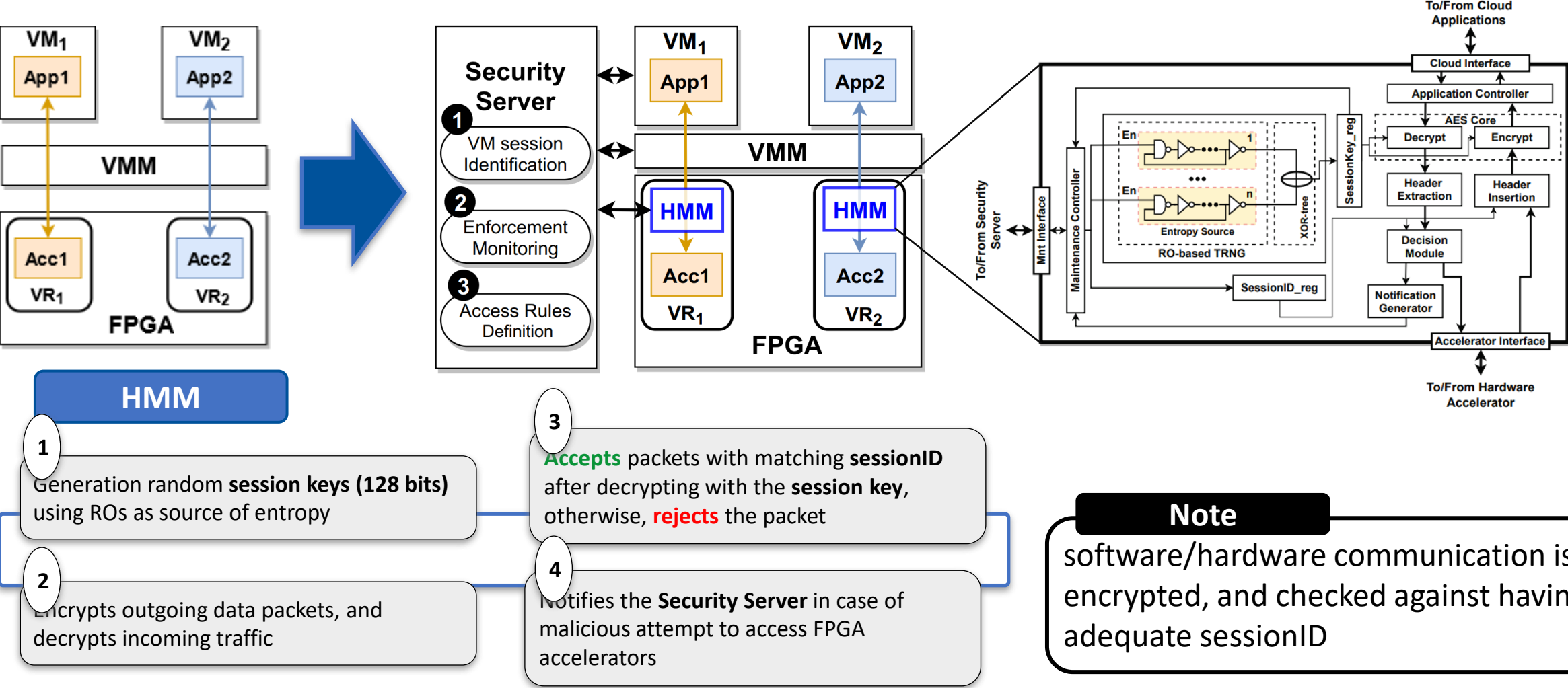
Risks: an attacker can still secret, tamper data, or compromise execution of applications



- **Scenario2:** a breached VMM is used for unauthorized access to VRs. e.g: the VMM accesses VR1 and VR2

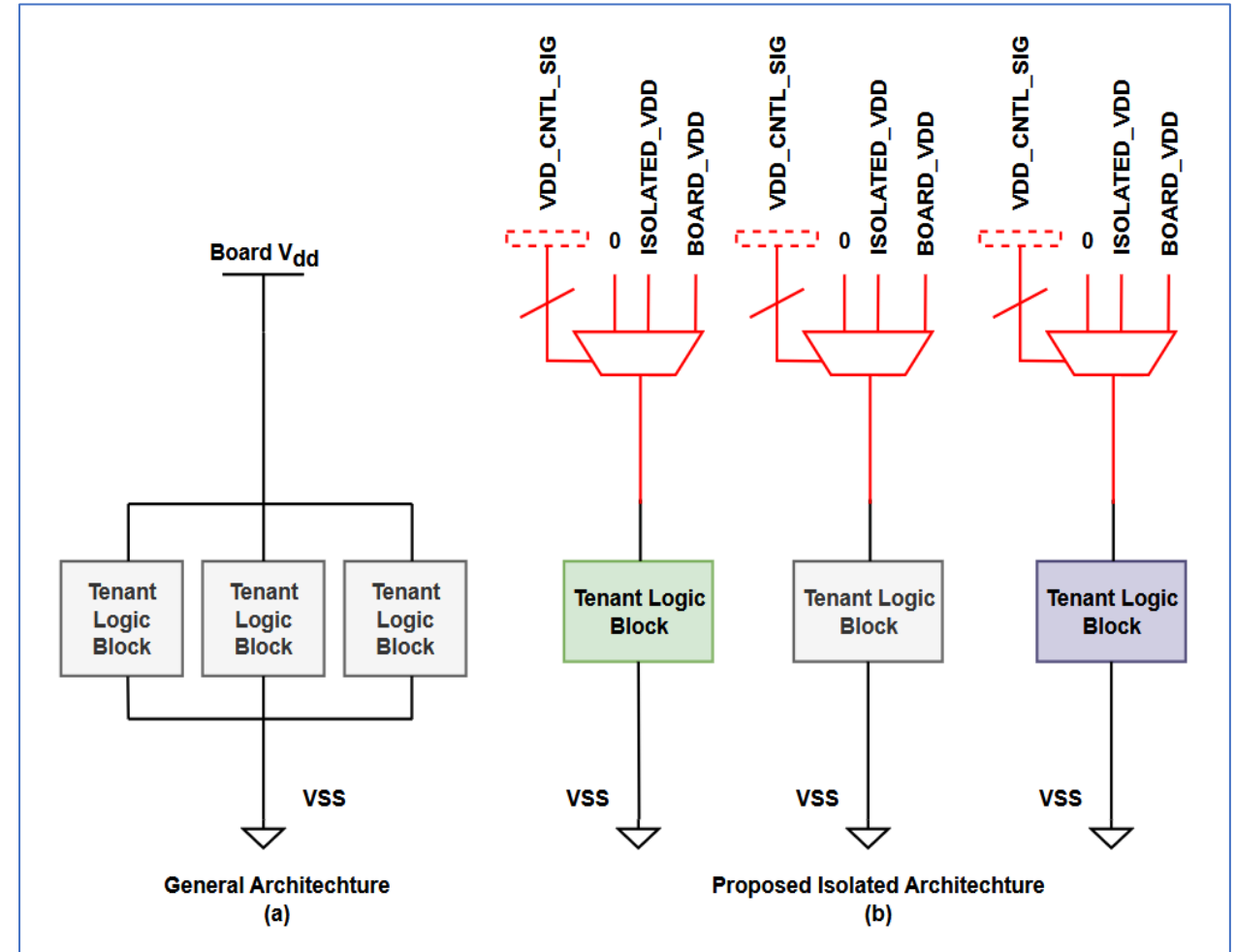
Risks: attacker can access the entire address space of the FPGA. Information leak, denial-of-service, task hiding are also rising concerns

Security Architecture



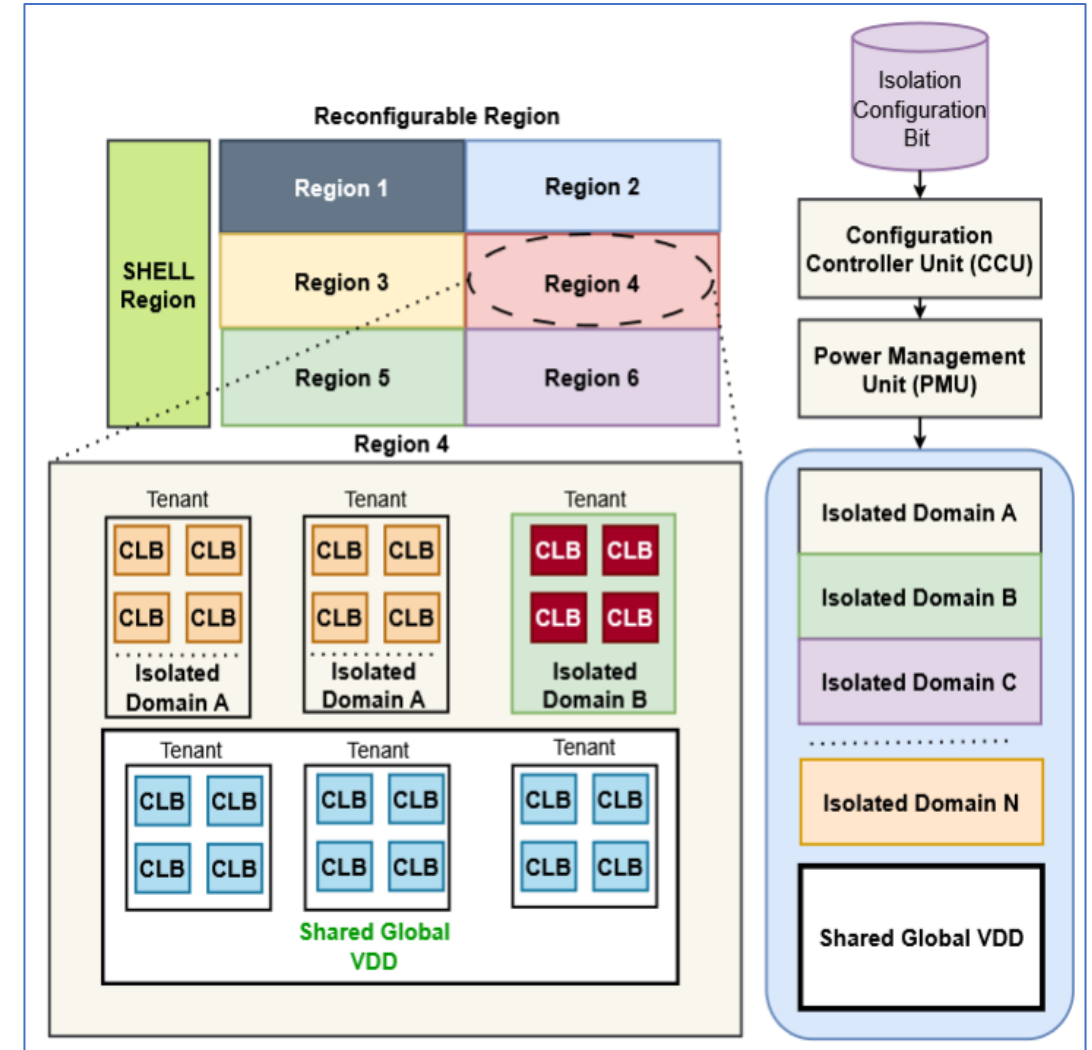
Security Architecture – Physical Isolation

- Physical capacitive isolation for each tenants.
 - Tenants will get their own voltage source !
- Tenant logic blocks are spatially isolated by a **physically separated CMOS** voltage source and ground.
- This isolation structure is controlled by a memory that can be configured, in the same way LUTs and switch matrixes are configured.

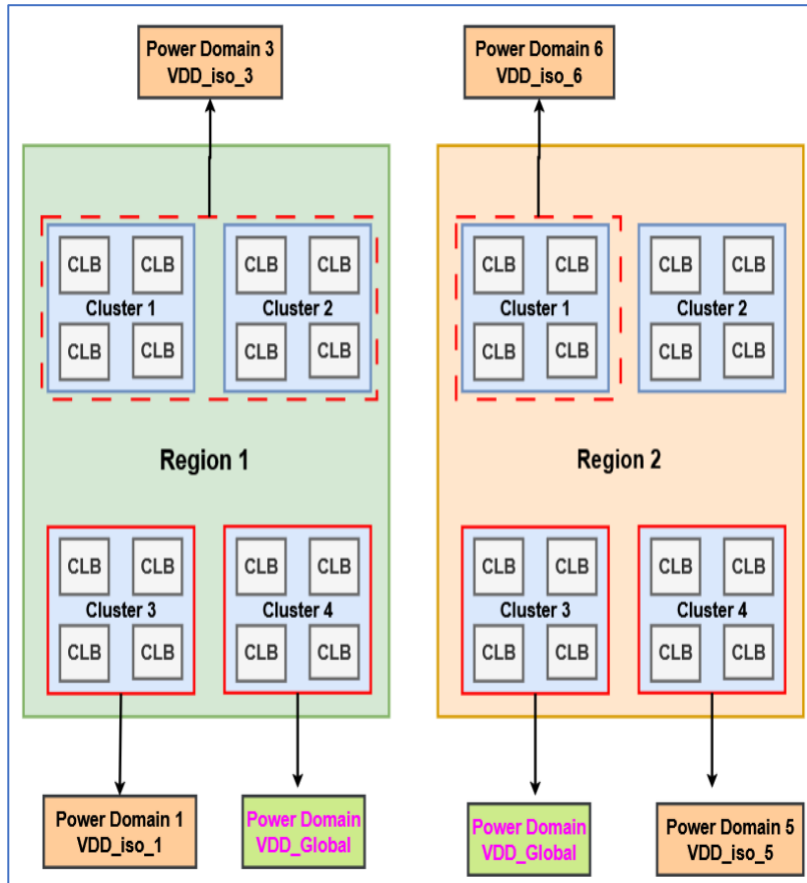


IsoFPGA Architecture

- The reconfigurable fabric is partitioned into regions, which are grouped into power-isolated domains called clusters.
- Each tenant is assigned to a specific region or cluster, ensuring their logic is physically and electrically isolated.
- A configuration controller sets isolation policies, and the power management unit enforces domain-level voltage control for secure multi-tenancy.



Attachment of Domain and Results



```
// Defining physical voltage domains
define_domain VDD_iso_1
define_domain VDD_iso_3

// Definition of clusters and their respective region
define_cluster cluster1 -region region1
define_cluster cluster2 -region region1
define_cluster cluster3 -region region1
define_cluster cluster4 -region region1

// Join clusters 1 and 2 into a merged domain
join cluster1 cluster2 domain_merged

// Physical tie of a cluster group with a physical power domain
bind domain_merged VDD_iso_3

// Physical bind of single clusters with physical power domains
bind cluster3 VDD_iso_3
bind cluster4 VDD_Global

/* Example of isolating a cluster from a power rail (VDD_iso_1) */
/* The isolate command power-gates this cluster (e.g., by disabling its PMOS switches),
effectively shutting off its voltage supply to save power or mitigate attack surfaces
when the tenant is idle. */
isolate cluster1
```

Circuit	LUTs	FF	BRAM	DSP
Attacker RO Circuit	573	650	3	4
Crypto Core SHA-256 Core	1567	1690	5	5
Configuration Controller Unit	2802	3100	5	3
Power Management Unit	3288	3314	2	0
Total Overhead(CCU + PMU)	6090	8714	7	3

Model	LUTS	FF	BRAM	DSP
Eguro et al. [11]	27,237	27,076	49	4
Joel et al. [19]	169,664	81,766	334	84
Provel engiou s et al. [22]	5,000	5,500	34	3
Zhao et al. [23]	18,471	14,857	2	0
Elrabaa et al. [24]	16,429	12566	14	4

Conclusion

- **Isolation remains** one of the most effective and foundational security mechanisms for hardware/software co-design across diverse domains.
- **Performance and area overheads** are successfully maintained within acceptable and reasonable bounds.
- **Cross-domain expertise** (e.g., VLSI design, RF engineering) is critical to addressing remaining vulnerabilities at the physical layer.

UF | Herbert Wertheim
College of Engineering
UNIVERSITY *of* FLORIDA

LEADING THE CHARGE, CHARGING AHEAD