NoC-level Threat Monitoring in Domain-Specific Heterogeneous SoCs with SoCurity

Naorin Hossain
IBM Research
naorin.hossain@ibm.com

Alper Buyuktosunoglu, John-David Wellman, Pradip Bose
IBM Research
{alperb, wellman, pbose}@us.ibm.com

Margaret Martonosi
Princeton University
mrm@princeton.edu

***Abstract*—Heterogeneous systems-on-a-chip (SoCs) are increasingly used to meet low-power, high-performance computational requirements but are vulnerable to on-chip resource availability attacks. In this work, we explore design opportunities with fw, a widely applicable SoC design enhancement that provides protection against resource availability attacks through anomalous activity detection. We demonstrate the effectiveness of using SoCurity for anomaly detection with a case study on real SoC implementations for a connected autonomous vehicle system and find up to 96.4% detection accuracy. We then discuss how this work can be used to protect against other threats and to enhance system reliability through detection of hardware faults. We propose that a solution like SoCurity can help ease and enable focused threat and fault mitigation built into our systems.**

I. Introduction and Motivation

Systems-on-a-chip (SoCs) consolidate various processing components on a single die to offer high-performance processing ability, while maintaining low area and power overheads. They are increasingly being adopted to power digital systems, including personal computers, servers, and embedded devices, such as smartphones, autonomous vehicles, medical devices, data centers, cyber-physical systems, and the internet-of-things [3, 36]. Heterogeneous SoCs are designed to the target device's computational needs by bringing together specialized processing units tailored to accelerate the device's tasks, all on a single die. SoC components can include general purpose processing cores, hardware accelerators, memory interfaces, and I/O interfaces. Some of the SoC components may also be black-box third-party intellectual property (IP) units. State-of-the-art SoCs use network-on-chip (NoC) interconnects for efficient communications between these SoC components [1, 9].

Although heterogeneous SoCs make notable improvements in power and performance, they also introduce more severe security and reliability challenges than prior system designs. The diverse set of components in heterogeneous systems invites an increased possibility of vulnerabilities or hardware faults to occur in a variety of ways throughout the system. Such problems could potentially lead to fundamental system failures that must not occur in many safety-critical edge devices. Furthermore, integration of components from third-party vendors presents the added risk of unknown vulnerabilities introduced by internal component designs. It is thus a substantially greater challenge for systems designers to ensure security and reliability throughout a heterogeneous SoC.

Protecting embedded devices from security and reliability issues comes with added challenges that must be addressed by realistic solutions. Like the low-power needs met by the domain-specific SoCs that operate these devices, preventative measures must feasibly run under low-power and area constraints without requiring high performance processing that can exceed the device's capabilities. Additionally, embedded devices are often designed for long field lives, such as autonomous vehicles which may be used for over a decade. This requires defenses to be adaptable over time against evolving threats. Finally, a general solution is needed that is applicable to the custom and fast-changing SoC designs that are used across various devices.

Although prior works used hardware performance counters for malware and hardware fault detection in traditional CPU architectures [5, 12, 16, 20–22, 29, 30], they are insufficient for SoCs as they may not be able to monitor the different processing units, memory, and I/O interfaces on SoCs that may be at risk. Further, black-box SoC components may not expose desired hardware performance counters to be used in such approaches. With rapidly changing SoC designs, it is critical for these needs to be met.

In this work, we explore design opportunities with SoCurity [17], a widely applicable SoC design enhancement that provides protection against resource availability attacks through anomalous activity detection. We first discuss SoCurity and our experiments that show it can be effective for availability attacks like denial-of-service (DoS) on domain-specific architectures. We then discuss how SoCurity can be used to protect against other threats and to enhance system reliability through detection of hardware faults. We propose that a solution like SoCurity can help ease and enable focused threat and fault mitigation built into our systems.

II. The SoCurity Approach for SoC Monitoring

Our presented ideas center around methodology developed in the SoCurity work, which developed a design method for enhancing security in heterogeneous SoCs at the NoC interconnect level. SoCurity presents the first use of *NoC-based hardware counters* to monitor ongoing SoC activity. NoC-level monitoring ensures there is no reliance on the internal design of individual SoC components. NoC-based counters can thus provide a foundation for hardware-level

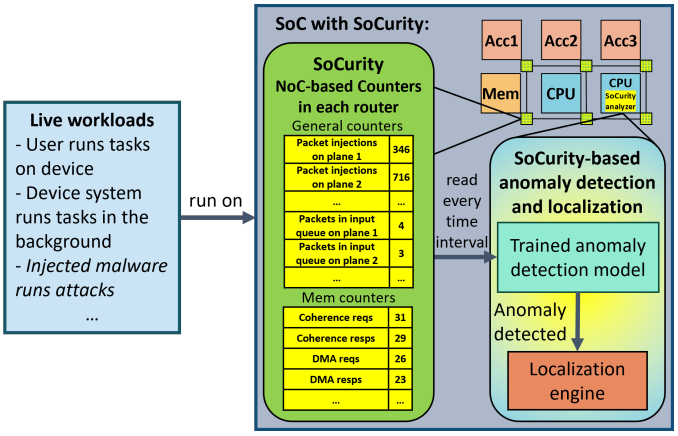


Figure 1: SoCurity introduces counters to NoC routers for each tile in an SoC to monitor ongoing interactions between tiles (example counters shown). These counters enable a constantly active anomaly detection and mitigation system while live workloads run on the SoC.

security and reliability measures to be incorporated into a broad scope of SoC designs.

Built on top of SoCurity's NoC-based SoC monitoring, we developed a fast, lightweight anomalous activity detection system, called the SoCurity analyzer, that can capture hardware availability attacks and identify where anomalous activity is occurring in the SoC. Here, lightweight indicates low algorithmic complexity with compact processing and memory requirements resulting in swift execution, even on small-scale embedded systems. To account for the growing attack space as SoC designs rapidly change and become more heterogeneous, no prior knowledge or characterization of attacks is required with our approach, enabling detection of existing attacks and novel future availability attacks. To this end, we used *semi-supervised machine learning (ML) models* that train only on benign SoC behavior captured by the NoC counter data. After training, the model is deployed on the SoC to identify and flag any runtime-collected counter data that deviates from training data as *anomalous* activity, such as excessive NoC traffic from a DoS attack on an SoC component. Furthermore, the anomalous NoC counter data provides useful insight on *where* in the SoC the anomalous behavior is occurring.

In this section, we briefly discuss the proposed NoC counters, ML models, and end-to-end SoCurity detection system, as summarized in Figure 1.

A. NoC-Based SoC Hardware Counters

In a heterogeneous SoC, the presence of black-box third-party IP and various on-chip processing units requires that any holistic approach to observing hardware activity throughout the SoC either a) requires a specific interface be implemented by each component to monitor and expose its internal activity, or b) brings monitoring outside these components to the NoC-level where usage of and communications between components can be tracked. SoCurity takes the latter approach, as the NoC interconnect granularity provides a reliable medium for monitoring activity throughout an SoC and does not impose additional requirements on SoC component designs. This approach is ideal for open-source hardware designs, such as those supported by the ESP project [26], which provides a plug-and-play platform integrating accelerators into SoCs within a 2D mesh NoC. SoCurity is a natural fit for open-source SoC design platforms like ESP.

NoC-based counters can capture a holistic view of ongoing SoC activity by monitoring: 1) NoC packets injected by each tile, 2) NoC congestion at each tile, 3) memory tile requests and responses, and 4) accelerator usage. Counters can be implemented as protected registers at the router for each tile (examples in Figure 1). NoC packet counters can increment at each source and destination router for each packet. NoC congestion counters can track the size of packet input queues at each router. For memory interface tiles, some counters can track coherence and direct memory access (DMA) requests by incrementing (on arrival) for packets from cores and accelerators, respectively. Routers for accelerators can count usage cycles by recording cycles between a packet arrival from a core and a departure of a response packet to that core. These proposed counters are generic and applicable for tiles in any SoC design.

B. Semi-Supervised Anomaly Detection Models for SoCurity

The goal for the SoCurity detection system is to use a lightweight algorithm to quickly flag uncharacteristic hardware behavior without searching behaviors specific to known attacks. When developing SoCurity, we considered four semi-supervised anomaly detection models. They are each based on common supervised models, but designed for one-class classification, training only on a set of benign data. These models are well-studied anomaly detection algorithms and their simplicity, speed, accuracy, and general applicability best fit our goals. Collected training data is used to define bounded regions of possible benign activity. Samples that fall outside of these regions get labeled *anomalous*. With no anomalous training data, these models are capable of capturing both existing and novel future attacks, a particularly attractive quality that inspired their use in our system. In our experiments (§III), we used available implementations of the models from the Python scikit-learn library [32], except for one-class nearest neighbors (OCNN) which is not in the library. We wrote our own OCNN implementation based on the k-Nearest Neighbors (kNN) module in the scikit-learn library.

1) One-Class Nearest Neighbors: Nearest neighbors algorithms like kNN use distance between samples for prediction [14]. Training data is simply stored as feature vectors, and distances from test data to each training data point are calculated, with a majority vote among the k nearest neighbor labels used to classify a test sample. In the "one-class" anomaly detection variant, OCNN, a distance boundary determines whether a sample is close enough to training data to label it benign. Of the OCNN variants described in [19], we used 11NN.

2) One-Class Support Vector Machines: Support vector machines (SVM) distinguish classes by projecting data to a higher dimension where samples can be separated by a hyperplane [10]. The nearest training points to the hyperplane are the support vectors, and the optimal hyperplane maximizes distance to them. In one-class SVM (OCSVM), samples are lifted to a higher dimension but the optimal hyperplane maximizes distance between support vectors in the benign training set and the origin [34]. The model learns a frontier where training samples are densely populated and anomalies fall outside of the region when raised to the higher dimensional space. We used OCSVM with the non-linear radial basis function (RBF) kernel.

3) Isolation Forest: Isolation forest (iForest) is a one-class, anomaly detection variant of the supervised random forest model [24]. Random forest is an ensemble model composed of several decision trees that break up a classification problem [6]. iForest is similarly composed of isolation trees (iTrees) that split on random features in the training set until all data resides at a leaf node. The basis of anomaly detection with iForest is the notion that anomalies are distinct enough from regular data that they can be quickly and easily sifted out. Anomalies thus have a shorter path down an iTree than a regular sample. A sample receives a score for the length of its traversal path for each iTree and averaged scores are used to flag anomalous samples.

4) Local Outlier Factor: Local outlier factor (LOF) is a distance-based algorithm similar to kNN but also accounts for local density of samples [7]. The key idea is that regular samples are densely packed in a space and anomalies would thus be farther away from these regions, in sparser areas of the space. LOF scores indicate a sample is benign when it falls within a dense cluster of its neighbors.

C. End-to-end Anomaly Detection System

Model Training: Effective anomaly detection with a semi-supervised approach relies on strong training data that accurately portrays regular system behavior. Collecting representative data sets is feasible since SoCs are designed for specific uses. The detection model is statically trained with counter data collected during representative workloads in a secure, offline setting. Once deployed, the trained model can be updated periodically with data that is collected during the device's use and found to be benign. These future updates ensure the model remains robust to regular usage.

Secure Implementation: When deployed on an SoC, the base configuration for the proposed anomalous activity detection system is to execute it on an isolated core at the highest privilege level, independent of the software stack, protecting it against attackers (Fig. 1). Alternatively, the detection system can be implemented and executed as a standalone hardware unit, enhancing its security from intruders. For either approach, the NoC counters should only be accessible to the detection system through a dedicated NoC plane that is unavailable to other SoC components. Memory for the detection model and counter data should also be isolated in a trusted, secure space that only the detection system can access and do so without interrupting ongoing program executions.

Full defense process for malware-injected attack detection:

1) NoC counter data is collected while running representative workloads on the SoC in a secure, offline setting.
2) The anomaly detection model is trained with the collected data.
3) The trained model is deployed on the SoC for runtime detection.
4) NoC counter data is periodically read and input to the model.
5) The detection model predicts if the counter data is anomalous.
6) If the data is anomalous, it is provided to the localization engine.
7) The localization engine pinpoints the anomalous activity in the SoC. Location information is then used to find the associated anomalous process for mitigation. Alternatively, it may be determined that the data was mispredicted.
8) Counter data labeled benign by the detection or localization engine is used to update the detection model over time.

III. SoCurity for Connected Autonomous Vehicles

To demonstrate SoCurity monitoring and detection, we performed a case study on real SoC implementations of connected autonomous vehicle (CAV) systems. CAVs are being designed to improve reliability of navigational decisions through "swarm" communications with nearby vehicles, smart infrastructure, and the cloud for sharing knowledge on surroundings [38]. CAV SoCs are complex and subjected to high input variability based on real-world conditions.

A. Threat Model

SoCurity was designed to provide a hardware-level security solution for maintaining the performance gains made possible by heterogeneity in SoCs [17]. In particular, it was designed to protect the SoC against attacks that disturb the regular operation and accessibility of components on the SoC. Possible avenues to introduce such an attack on an SoC include hardware trojans incorporated at design time into third-party IP tiles and malware injected at runtime through connected devices or the cloud. Though the techniques presented in the SoCurity work are able to detect anomalous hardware activity resulting from attacks loaded on the SoC through any of these means, our study focuses on malware injections. We assume the malicious attacker is aware of accelerators on the SoC and is able to send tasks to them. In this study, we particularly focus on detection of DoS attacks that flood and congest the NoC with tasks for victim SoC components. We select DoS attacks as they are the most common hardware availability attack and are a concerning vulnerability for SoCs with NoCs [9, 23].

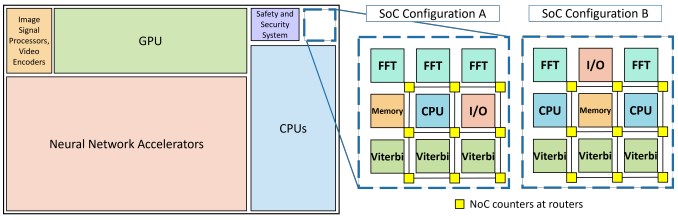


Figure 2: Overview of the implemented SoC subsystem layouts for each configuration. The SoCs are shown within a larger CAV compute model (based on Tesla Full Self-Driving chip [1]). They consist of RISC-V Ariane cores, memory tiles, I/O tiles, FFT accelerators, and Viterbi decoder accelerators. SoC A has a single RISC-V CPU, while SoC B has two.

B. Experimental Setup

We implemented two SoC prototype configurations for the swarm communication component of a full CAV SoC for this case study (Fig. 2). Both SoCs were emulated on a Xilinx Virtex UltraScale+ VCU118 FPGA running at 78MHz using the ESP platform [26]. Each SoC has a 3-by-3 tiled layout. SoC A has one 64-bit RISC-V Ariane core [15, 40], a memory tile, an I/O tile, three FFT accelerators, and three Viterbi decoder accelerators. In contrast, SoC B includes one additional RISC-V core that replaces one of the FFT accelerators. The RISC-V core has a 16KB L1 I-cache and a 32KB L1 D-cache. In SoC B, the CPU tiles also have a 32KB private L2 cache. The memory tile has a 256KB LLC. Tiles are connected by a 2D mesh NoC with six planes for specific data packets [26]. Fig. 2 shows a high-level sketch of the implemented SoCs and how they might fit into a larger CAV SoC.

For each SoC, we implemented hundreds of NoC-based counters using the lightweight hardware monitoring system from the Cohmeleon work for ESP [42]. The counters spanned the following:

- **General:** NoC packets, backpressure cycles by tile/plane
- **Memory:** off-chip memory accesses, coherence requests/responses, DMA requests/responses, LLC hits/misses
- **Accelerator:** total/memory/TLB cycles, invocations[1]

1) CAV Workload: For our experiments, we ran a representative workload on the SoC implementation that addresses characteristics of a real-world CAV application, with a focus on swarm communication. A generated trace of upcoming obstacle positions drives the workload. The workload loops on the following tasks during each *time step*:

1) Calculate distance to surrounding objects using radar readings (via the FFT accelerators).
2) Messages are received from other vehicles, infrastructure, or the cloud (i.e., swarm communications). We assume a Viterbi encoding scheme for messages and use Viterbi decoding accelerators to process them [39].
3) Combine learned knowledge to plan and execute actions. This step is driven by information from steps 1 and 2.

[1]The accelerator invocation counter was added to SoC B only. We found that it strongly correlates with the other accelerator counters.

Attack	Attack Time (s)	Total Time (s)	Impact
No attack	–	46.56	–
FFT	11.55	55.59	5.75
Viterbi	94.13	146.46	4.89
FFT + Viterbi	104.13	155.74	5.00
Memory	40.89	61.71	2.00

Table I: Average DoS attack durations, average total durations of CAV workload during attacks, and impact of each attack (higher values indicate higher impact). Absolute times reflect 300 time steps of the workload running on a 78MHz FPGA for SoC A. Runtimes for SoC B follow similarly.

2) DoS Attack Targets: We ran four DoS attack variants on the SoCs by flooding the NoC with packets for victim tiles. The attacks slowed down CAV workload computations by keeping targeted SoC components busy, negating performance gains achieved by the heterogeneous design. The attacks are:

1) **FFT:** 1000 FFT tasks are inserted into the network, causing traffic on all FFT accelerators and slowing down distance calculations.
2) **Viterbi:** 1000 randomly sized Viterbi decoding tasks are sent for all Viterbi units, slowing down incoming message decoding.
3) **FFT + Viterbi:** 1000 tasks are sent to both FFT and Viterbi tiles.
4) **Memory:** 500,000 memory requests are made for random addresses in a space that is twice the size of the LLC. Frequent cache misses slow down legitimate memory requests.

3) Evaluation Methods: To develop an anomaly detection system using data from the SoCurity counters, we implemented the four semi-supervised ML models described in §II-B. They were trained offline using NoC counter data collected while the regular CAV workload ran on the SoCs. They were then tested with counter data collected while the DoS attacks ran alongside the CAV workload on the SoCs.

C. Results

Each DoS attack impacted the CAV workload running on the SoCs. For SoC A, Table I gives each attack duration, the time to run the CAV workload for 300 time steps during each attack, and an impact metric for each attack's overall effect on the workload. SoC B runtimes follow similarly. As each attack varies in length, overall workload slowdown does not provide a good comparison for the impact each attack had on the CAV workload. Instead, we include an impact metric for fair comparison. Impact is quantified as the ratio of the workload's slowdown ($\frac{t_{total}}{t_{no\_attack}}$) to the proportion of the run during which the attack is executed ($\frac{t_{attack}}{t_{total}}$). In this section, we present results from our anomaly detection experiments using these attacks on SoCs A and B.

1) Comparing Models for DoS Attack Detection: To evaluate anomaly detection performance of each semi-supervised model on SoC A, we counted true positive (TP), false positive

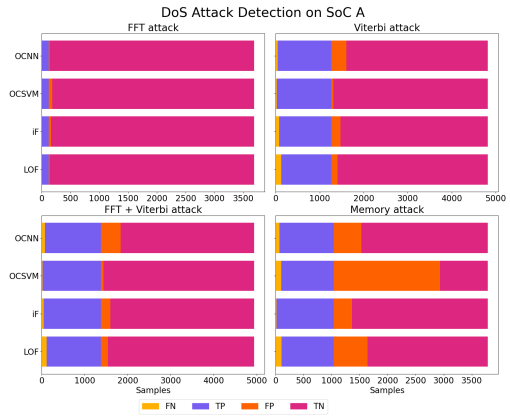


Figure 3: FN, TP, FP, TN for each model and attack. Low FN and FP are better. All models performed well on FFT attacks. OCSVM performed best for Viterbi and FFT + Viterbi attacks. All models had high FPs for memory attacks.

(FP), true negative (TN), and false negative (FN) predictions for the test data sets, where the positive class is anomalies and predictions were compared against automated ground truth labels (automated using clustering models like k-means [25]). When tuning model parameters, we prioritized low FNs (<10% on SoC A) to minimize missed anomalies and low FPs to avoid system resource costs.

Fig. 3 presents FNs, TPs, FPs, and TNs for each model and DoS attack on SoC A. We found attacks with higher *impact* on the CAV workload (Table I) were more easily detected. All models detected FFT attacks best; there were no FNs. OCSVM had the lowest FNs and FPs for Viterbi and FFT + Viterbi attacks. All models had high FPs for memory attacks which had the least impact and thus less accurate clustered ground truth labels than other attacks. In online tests (§III-C3), far fewer FPs occurred for memory attacks. Overall, all models were highly effective, but OCSVM best distinguished variable message traffic decoding from Viterbi-based attacks in SoC A.

Semi-Supervised vs. Supervised Models: To underscore the detection models' enhanced ability to detect unknown attacks compared to supervised models, we compared the semi-supervised models with their supervised variants, using similar parameters for training. We examined KNN, SVM, and random forests models, training them with anomalous data from FFT attacks, and compared their abilities to detect Viterbi attacks against the semi-supervised models.

Fig. 4 presents receiver operating characteristic (ROC) curves for SoC A comparing Viterbi attack detection with the supervised models against our detection system's semi-supervised models that were only trained with CAV workload data. False positive rate (FPR: $\frac{FP}{FP+TN}$, lower is better) is on the x-axis and recall ($\frac{TP}{TP+FN}$, higher is better) is on the y-axis. Areas under each model's curve (AUCs) are given in the legend. AUCs closest to one have the best model performances. We found KNN to be the least effective model with a 0.500 AUC (equivalent to random guessing), while its

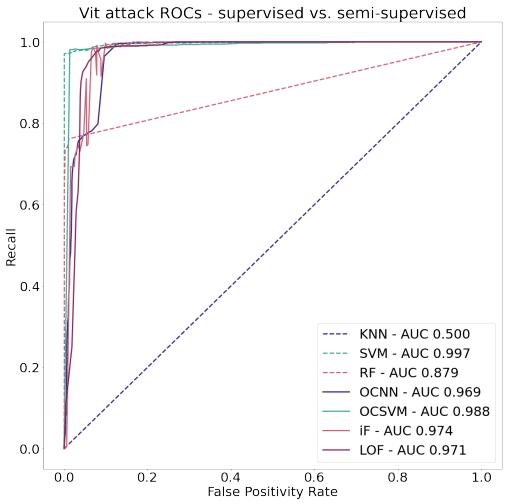
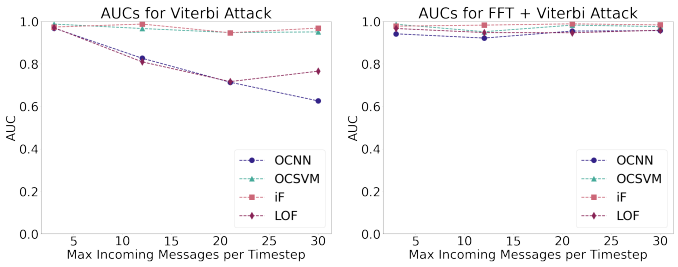


Figure 4: ROC curves for Viterbi attack detection with SoC A using supervised (dashed) and semi-supervised (solid) models, with AUCs (areas under curves, higher is better) in the legend. Most semi-supervised detection models outperformed their supervised counterparts that were trained on FFT attack data.



(a) Viterbi Attack (b) FFT + Viterbi Attack

Figure 5: AUC impact for each model when incoming message traffic variability of CAV workload increases (higher is better). OCSVM and iForest maintain high AUCs for both attacks.

semi-supervised counterparts OCNN and LOF showed robust performance with AUCs of 0.969 and 0.971, respectively. Random forests had a 0.879 AUC, but it was surpassed by its semi-supervised counterpart iForest which had a 0.974 AUC. SVM slightly outperformed its semi-supervised counterpart OCSVM, but both had the two largest AUCs at 0.997 and 0.988, respectively. These results show that semi-supervised ML models used in our anomaly detection system are superior at detecting "unknown" attacks compared to supervised models that are trained with data from similar attacks.

Storage Requirements: While iForest and LOF models had comparable detection accuracy to OCSVM, iForest required significantly more storage at about 807.3KB and LOF required 316.0KB. The OCNN model required 187.0KB. We found the OCSVM model required the smallest amount of storage at just 2.9KB, making it lightweight and ideal for fast detection.

2) SoCurity Robustness: We stress tested the SoCurity detection system to see how Viterbi-based attack detection is impacted under conditions with much higher incoming

Attack	Detection Cycles	Accuracy	FPR
FFT	0.96	0.96	0.04
Viterbi	2.82	0.87	0.05
FFT + Viterbi	1.06	0.87	0.06
Memory	2.22	0.60	0.05

Table II: Average detection cycles to flag the attack (lower is better), accuracy (higher is better), and FPR ($\frac{FP}{TN+FP}$, lower is better) for each attack on SoC A emulated on the FPGA. Accuracy correlates with impact (Table I).

message traffic for the CAV workload, simulating extreme urban settings with large CAV presence. Message traffic was randomized with up to 3, 12, 21, and 30 incoming messages of random size per time step on SoC A. Beyond three messages, Viterbi accelerators faced increased task loads. Fig. 5 shows AUCs for detecting Viterbi and FFT + Viterbi attacks under these conditions for each semi-supervised model on SoC A. For Viterbi attacks (Fig. 5a), OCSVM and iForest maintained high AUCs with increasing variability, while OCNN and LOF had decreased AUC with higher variability. As distance-based models, OCNN and LOF struggled to distinguish between high Viterbi activity due to incoming message traffic and the Viterbi attack. In contrast, OCSVM and iForest incorporate feature relations for a stronger model. For FFT + Viterbi attacks (Fig. 5b), all models maintained high accuracy. OCNN and LOF distinguished anomalous activity better with the presence of increased FFT activity from the attack, tracking with their high detection performance for the FFT-only attack (Fig. 3). Overall, using OCSVM or iForest for detection, SoCurity is robust to highly variable swarm communication conditions processed by the CAV workload.

3) Fast and Lightweight SoCurity in Action on the CAV SoC: We selected OCSVM for online experiments as it was the most lightweight model, with two orders of magnitude less storage than others. It also performed well across all offline tests on SoC A and was robust against higher variability introduced to the base workload. We statically trained the model and deployed it on the FPGA with m2cgen [41]. When run on the 78MHz FPGA, the OCSVM model had a 320µs prediction time (32µs on an ASIC [18]). This *prediction* time better represents *detection* time for an SoC with a dedicated SoCurity analyzer unit. In this study, the SoCurity analyzer shared the single core in SoC A with the CAV workload and the attacks, limiting detection cycles to every 100ms.

Table II presents average detection cycles required to flag attacks after they are launched, accuracy ($\frac{TP+TN}{TP+FP+TN+FN}$), and FPRs ($\frac{FP}{TN+FP}$) for each DoS attack on the FPGA. TP and FN were counted as prediction cycles during each attack where counters were predicted anomalous and benign, respectively. TN and FP were counted as the cycles before and after each attack where counters were predicted benign and anomalous, respectively. The OCSVM model had low FPRs for each attack and high accuracy for accelerator-based DoS attacks. Due to the memory attack's lower impact (Table I), some counter cycles were not as recognizably anomalous so the model had lower detection accuracy. Nonetheless, the attack

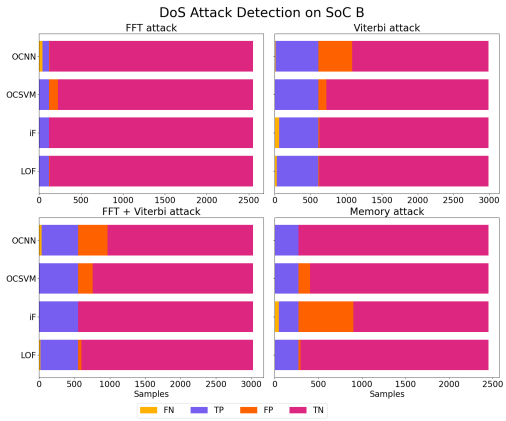


Figure 6: FN, TP, FP, TN for each model and attack. Low FN and FP are better. All models performed well on FFT attacks. LOF performed well for all attacks.

was detected within a few detection cycles, so this anomalous behavior would be quickly investigated by the localization engine. These results demonstrate SoCurity's strong abilities for effective anomaly detection in heterogeneous SoCs.

4) DoS Attack Localization on the Multicore SoC B: The multicore design of SoC B was used to test localization capability with SoCurity's NoC counters. In this paper, we do not go into the details around our localization methodology, but we briefly summarize our detection and localization results from experiments on SoC B in this section.

NoC counter data that is flagged by the anomaly detection engine is sent to the localization engine so first, we needed to evaluate anomaly detection performance of the semi-supervised models on DoS attacks running on SoC B. Fig. 6 shows the FNs, TPs, FPs, and TNs resulting from running detection with each model on each attack on counter data collected from SoC B. These tests revealed LOF to perform the best overall for detecting each attack with low FNs and FPs. We thus used the LOF model in localization tests for SoC B. Leveraging explainable AI techniques [2, 27], we found that DoS attacks were able to be located with up to 99% accuracy.

Overall our results indicate that SoCurity could successfully be used to build a secure foundation into hardware designs for domain-specific SoCs such as those powering CAVs.

IV. Advancing Security and Reliability for SoCs

While our study showed that SoCurity performed well for DoS attack detection within an SoC, the SoCurity approach has the potential to provide protection against other threats as well. In the next part of this work, we discuss how the SoCurity methodology may benefit security and reliability in SoCs beyond DoS attack detection.

SoCurity may be useful in detection of several other threats. For example, SoCurity counters can detect *black hole router* attacks launched by hardware trojans, where one node in the NoC drops all received packets, creating a "black hole" in the network [11]. Such attacks would result in clear impacts on hardware activity monitored by SoCurity counters. Likewise,

side channel attacks like Prime+Probe [28] that repeatedly access shared resources to learn secrets can be detected as prior work showed that they impact hardware counters [12]. SoCurity counters can also be expanded to support power token monitoring [35] and detect if malicious actors interject and manipulate token flow, which could cause imbalanced power and overheating in the SoC.

Additionally, SoCurity can be used to improve hardware reliability. Resources may fail to operate as expected, causing slowdowns that are reflected in SoCurity counters. One example is silent data corruption (SDC), in which a CPU produces incorrect computational results [13]. Typically, SDC is considered to be a major problem at large scales, such as in data centers where there is a much higher potential for errors to occur and propagate [31]. SoCs that power edge devices may not be as susceptible to such SDC errors in isolation. However, many edge devices are backed by some larger cloud system where SDC may present a more significant threat. When edge devices rely on the cloud for important or computationally expensive information, they are also exposed to this SDC risk. Particularly as we see larger and larger AI models (e.g., generative AI models) provide services to edge devices, whether through the cloud or within the device itself, having built-in protection against otherwise untraceable hardware faults such as SDC will be invaluable. Though SDC errors are not identifiable by existing error handling systems, it may be possible to use SoCurity counters to detect NoC communication pattern changes caused by SDC within an SoC or resulting from SDC-impacted data received from the cloud. Prior works have shown some success with using CPU hardware performance counters on single and multicore systems [5, 16, 21, 22]. Extending SDC analysis to SoC layouts using NoC counters can close the gap made by the presence of non-CPU cores within an SoC that may be impacted by SDC. This area in particular may benefit from anomaly localization methodology that we have found to work well with SoCurity in our case study (§III-C4). In future studies, we plan to experiment with such SDC detection and localization possibilities using SoCurity.

V. Related Work

Over the last decade, several prior works explored CPU hardware performance counters for anomaly and attack detection using ML models [4, 12, 20, 29, 30]. As discussed in [9], several works have also explored security measures for NoC-based SoCs in which **CPU** performance counters are not a sufficient solution. [30] evaluated malware detection with CPU performance counters from cores on an SoC and found high FPRs averaging 11.3%. They also explored an approach similar to [37] in which coherence NoC traffic information is extracted from packets to train and detect DoS attacks. SoCurity takes a simpler approach by integrating counters in the NoC that do not require extracting additional information from packets. Further, the NoC counters provide a broad set of data that is not limited to coherence packets and is thus better suited for heterogeneous SoCs in which accelerators may make DMA requests instead. On the other hand, [8] took an unrealistic approach for DoS detection in an SoC by relying on predictability of NoC traffic latencies and [33] explored detection of bandwidth denial attacks by a malicious NoC.

VI. Conclusion

In this work, we presented SoCurity as the basis for developing security and reliability solutions in domain-specific heterogeneous SoCs through hardware activity monitoring at the NoC-level with NoC-based hardware counters that track interactions between SoC components. We discussed experimental studies performed with SoCurity and explored several other use cases for expanding on its capabilities. As our systems get more complex with higher demands for security and reliability needs to be met, it is critical that we take these aspects into consideration at earlier design stages such that hardware-level protection can be provided through strong, yet lightweight means such as SoCurity.

References

[1] P. Bannon, G. Venkataramanan, D. D. Sarma, and E. Talpes, "Computer and redundancy solution for the full self-driving computer," in *2019 IEEE Hot Chips 31 Symposium (HCS), Cupertino, CA, USA, August 18-20, 2019*. IEEE, 2019, pp. 1–22. [Online]. Available: https://doi.org/10.1109/HOTCHIPS.2019.8875645

[2] A. Barbado, Ó. Corcho, and R. Benjamins, "Rule extraction in unsupervised anomaly detection for model explainability: Application to OneClass SVM," *Expert Syst. Appl.*, vol. 189, p. 116100, 2022. [Online]. Available: https://doi.org/10.1016/j.eswa.2021.116100

[3] S. Borkar and A. A. Chien, "The future of microprocessors," *Commun. ACM*, vol. 54, no. 5, pp. 67–77, 2011. [Online]. Available: https://doi.org/10.1145/1941487.1941507

[4] M. Bourdon, P. Gimenez, E. Alata, M. Kaâniche, V. Migliore, V. Nicomette, and Y. Laarouchi, "Hardware-performance-counters-based anomaly detection in massively deployed smart industrial devices," in *19th IEEE International Symposium on Network Computing and Applications, NCA 2020, Cambridge, MA, USA, November 24-27, 2020*. IEEE, 2020, pp. 1–8. [Online]. Available: https://doi.org/10.1109/NCA51143.2020.9306726

[5] F. A. Bower, D. J. Sorin, and S. Ozev, "A mechanism for online diagnosis of hard faults in microprocessors," in *38th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-38 2005), 12-16 November 2005, Barcelona, Spain*. IEEE Computer Society, 2005, pp. 197–208. [Online]. Available: https://doi.org/10.1109/MICRO.2005.8

[6] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[7] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, W. Chen, J. F. Naughton, and P. A. Bernstein, Eds. ACM, 2000, pp. 93–104. [Online]. Available: https://doi.org/10.1145/342009.335388

[8] S. Charles, Y. Lyu, and P. Mishra, "Real-time detection and localization of distributed dos attacks in noc-based socs," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 39, no. 12, pp. 4510–4523, 2020. [Online]. Available: https://doi.org/10.1109/TCAD.2020.2972524

[9] S. Charles and P. Mishra, "A survey of network-on-chip security attacks and countermeasures," *ACM Comput. Surv.*, vol. 54, no. 5, pp. 101:1–101:36, 2022. [Online]. Available: https://doi.org/10.1145/3450964

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: https://doi.org/10.1007/BF00994018

[11] L. Daoud and N. Rafla, "Analysis of black hole router attack in network-on-chip," in *62nd IEEE International Midwest Symposium on Circuits and Systems, MWSCAS 2019, Dallas, TX, USA, August 4-7, 2019*, H. Lee and R. L. Geiger, Eds. IEEE, 2019, pp. 69–72. [Online]. Available: https://doi.org/10.1109/MWSCAS.2019.8884979

[12] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. J. Stolfo, "On the feasibility of online malware detection with performance counters," in *The 40th Annual International Symposium on Computer Architecture, ISCA'13, Tel-Aviv, Israel, June 23-27, 2013*, A. Mendelson, Ed. ACM, 2013, pp. 559–570. [Online]. Available: https://doi.org/10.1145/2485922.2485970

[13] H. D. Dixit, S. Pendharkar, M. Beadon, C. Mason, T. Chakravarthy, B. Muthiah, and S. Sankar, "Silent data corruptions at scale," *CoRR*, vol. abs/2102.11245, 2021. [Online]. Available: https://arxiv.org/abs/2102.11245

[14] E. Fix and J. L. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," *International Statistical Review/Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989.

[15] D. Giri, K.-L. Chiu, G. Eichler, P. Mantovani, and N. Chandramoorthy, "Ariane + NVDLA: Seamless third-party IP integration with ESP," in *CARRV*, 2020.

[16] S. K. S. Hari, M. Li, P. Ramachandran, B. Choi, and S. V. Adve, "mswat: low-cost hardware fault detection and diagnosis for multicore systems," in *42st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42 2009), December 12-16, 2009, New York, New York, USA*, D. H. Albonesi, M. Martonosi, D. I. August, and J. F. Martínez, Eds. ACM, 2009, pp. 122–132. [Online]. Available: https://doi.org/10.1145/1669112.1669129

[17] N. Hossain, A. Buyuktosunoglu, J. Wellman, P. Bose, and M. Martonosi, "Socurity: A design approach for enhancing SoC security," *IEEE Comput. Archit. Lett.*, vol. 22, no. 2, pp. 105–108, 2023. [Online]. Available: https://doi.org/10.1109/LCA.2023.3301448

[18] T. Jia, P. Mantovani, M. C. dos Santos, D. Giri, J. Zuckerman, E. J. Loscalzo, M. Cochet, K. Swaminathan, G. Tombesi, J. J. Zhang, N. Chandramoorthy, J. Wellman, K. Tien, L. P. Carloni, K. L. Shepard, D. Brooks, G. Wei, and P. Bose, "A 12nm agile-designed SoC for swarm-based perception with heterogeneous IP blocks, a reconfigurable memory hierarchy, and an 800MHz multi-plane NoC," in *48th IEEE European Solid State Circuits Conference, ESSCIRC 2022, Milan, Italy, September 19-22, 2022*. IEEE, 2022, pp. 269–272. [Online]. Available: https://doi.org/10.1109/ESSCIRC55480.2022.9911456

[19] S. S. Khan and A. Ahmad, "Relationship between variants of one-class nearest neighbors and creating their accurate ensembles," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1796–1809, 2018. [Online]. Available: https://doi.org/10.1109/TKDE.2018.2806975

[20] A. P. Kuruvila, S. Karmakar, and K. Basu, "Time series-based malware detection using hardware performance counters," in *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2021, Tysons Corner, VA, USA, December 12-15, 2021*. IEEE, 2021, pp. 102–112. [Online]. Available: https://doi.org/10.1109/HOST49136.2021.9702291

[21] M. Li, P. Ramachandran, S. K. Sahoo, S. V. Adve, V. S. Adve, and Y. Zhou, "Trace-based microarchitecture-level diagnosis of permanent hardware faults," in *The 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2008, June 24-27, 2008, Anchorage, Alaska, USA, Proceedings*. IEEE Computer Society, 2008, pp. 22–31. [Online]. Available: https://doi.org/10.1109/DSN.2008.4630067

[22] M. Li, P. Ramachandran, S. K. Sahoo, S. V. Adve, V. S. Adve, and Y. Zhou, "Understanding the propagation of hard errors to software and implications for resilient system design," in *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2008, Seattle, WA, USA, March 1-5, 2008*, S. J. Eggers and J. R. Larus, Eds. ACM, 2008, pp. 265–276. [Online]. Available: https://doi.org/10.1145/1346281.1346315

[23] Y. Li and Q. Liu, "A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments," *Elsevier Energy Reports*, vol. 7, pp. 8176–8186, 2021.

[24] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*. IEEE Computer Society, 2008, pp. 413–422. [Online]. Available: https://doi.org/10.1109/ICDM.2008.17

[25] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–136, 1982. [Online]. Available: https://doi.org/10.1109/TIT.1982.1056489

[26] P. Mantovani, D. Giri, G. D. Guglielmo, L. Piccolboni, J. Zuckerman, E. G. Cota, M. Petracca, C. Pilato, and L. P. Carloni, "Agile soc development with open ESP : Invited paper," in *IEEE/ACM International Conference On Computer Aided Design, ICCAD 2020, San Diego, CA, USA, November 2-5, 2020*. IEEE, 2020, pp. 96:1–96:9. [Online]. Available: https://doi.org/10.1145/3400302.3415753

[27] H. Núñez, C. Angulo, and A. Català, "Rule extraction from support vector machines," in *10th Eurorean Symposium on Artificial Neural Networks, ESANN 2002, Bruges, Belgium, April 24-26, 2002, Proceedings*, M. Verleysen, Ed., 2002, pp. 107–112. [Online]. Available: https://www.esann.org/sites/default/files/proceedings/legacy/es2002-51.pdf

[28] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: The case of AES," in *Topics in Cryptology - CT-RSA*, 2006.

[29] M. Ozsoy, C. Donovick, I. Gorelik, N. B. Abu-Ghazaleh, and D. V. Ponomarev, "Malware-aware processors: A framework for efficient online malware detection," in *21st IEEE International Symposium on High Performance Computer Architecture, HPCA 2015, Burlingame, CA, USA, February 7-11, 2015*. IEEE Computer Society, 2015, pp. 651–661. [Online]. Available: https://doi.org/10.1109/HPCA.2015.7056070

[30] Z. Pan, J. Sheldon, C. Sudusinghe, S. Charles, and P. Mishra, "Hardware-assisted malware detection using machine learning," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021*. IEEE, 2021, pp. 1775–1780. [Online]. Available: https://doi.org/10.23919/DATE51398.2021.9474050

[31] G. Papadimitriou and D. Gizopoulos, "Silent data corruptions: Microarchitectural perspectives," *IEEE Trans. Computers*, vol. 72, no. 11, pp. 3072–3085, 2023. [Online]. Available: https://doi.org/10.1109/TC.2023.3285094

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://dl.acm.org/doi/10.5555/1953048.2078195

[33] R. J. S., D. M. Ancajas, K. Chakraborty, and S. Roy, "Runtime detection of a bandwidth denial attack from a rogue network-on-chip," in *Proceedings of the 9th International Symposium on Networks-on-Chip, NOCS 2015, Vancouver, BC, Canada, September 28-30, 2015*, A. Ivanov, D. Marculescu, P. P. Pande, J. Flich, and K. Pattabiraman, Eds. ACM, 2015, pp. 8:1–8:8. [Online]. Available: https://doi.org/10.1145/2786572.2786580

[34] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, S. A. Solla, T. K. Leen, and K. Müller, Eds. The MIT Press, 1999, pp. 582–588. [Online]. Available: http://papers.nips.cc/paper/1723-support-vector-method-for-novelty-detection

[35] P. Shah, R. G. Shenoy, V. Srinivasan, P. Bose, and A. Buyuktosunoglu, "Tokensmart: Distributed, scalable power management in the many-core era," *ACM Trans. Archit. Code Optim.*, vol. 20, no. 1, pp. 4:1–4:26, 2023. [Online]. Available: https://doi.org/10.1145/3559762

[36] S. Simpson, "Systems on a chip comes to the data center," *McKinsey Digital*, 2022, https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/tech-forward/systems-on-a-chip-comes-to-the-data-center.

[37] C. Sudusinghe, S. Charles, and P. Mishra, "Denial-of-service attack detection using machine learning in network-on-chip architectures," in *NOCS '21: International Symposium on Networks-on-Chip, Virtual Event, October 14-15, 2021*, T. Krishna, J. Kim, S. Abadal, and J. S. Miguel, Eds. ACM, 2021, pp. 35–40. [Online]. Available: https://doi.org/10.1145/3479876.3481589

[38] A. Vega, A. Buyuktosunoglu, and P. Bose, "Towards "smarter" vehicles through cloud-backed swarm cognition," in *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018*. IEEE, 2018, pp. 1079–1086. [Online]. Available: https://doi.org/10.1109/IVS.2018.8500627

[39] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, 1967. [Online]. Available: https://doi.org/10.1109/TIT.1967.1054010

[40] F. Zaruba and L. Benini, "The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-ghz 64-bit RISC-V core in 22-nm FDSOI technology," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 27, no. 11, pp. 2629–2640, 2019. [Online]. Available: https://doi.org/10.1109/TVLSI.2019.2926114

[41] I. Zeigerman, V. Yershov, and N. Titov, "m2cgen: Model 2 code generator," 2019, https://github.com/BayesWitnesses/m2cgen.

[42] J. Zuckerman, D. Giri, J. Kwon, P. Mantovani, and L. P. Carloni, "Cohmeleon: Learning-based orchestration of accelerator coherence in heterogeneous socs," in *MICRO '21: 54th Annual IEEE/ACM International Symposium on Microarchitecture, Virtual Event, Greece, October 18-22, 2021*. ACM, 2021, pp. 350–365. [Online]. Available: https://doi.org/10.1145/3466752.3480065