SVM-based Real-Time Hardware Trojan Detection for Many-Core Platform

Amey Kulkarni[1], Youngok Pino[2], and Tinoosh Mohsenin[1]

[1]Department of Computer Science & Electrical Engineering , University of Maryland, Baltimore County
[2]Information Sciences Institute , University of Southern California

***Abstract*—Hardware Trojans inserted during design or fabrication time by untrustworthy design house or foundry possesses important security concerns. These Trojans lead to un-desired change in functionality of the design and provide easy access to sensitive information. Trojans attacks or malicious activities are triggered based on very rare conditions, which can evade test-time Trojan detection but can arise during long hours of field operation. In this paper we propose a run-time Trojan detection architecture for a custom many-core based on Machine Learning technique. We exploit Support Vector Machine (SVM) supervised machine learning algorithms. The Data-set is generated based on many-core router behavior under normal and Trojan triggered settings. The paper targets different communication attacks triggered by Hardware Trojans, namely core address spoofing, traffic diversion, route looping attack. Support Vector Machine (SVM) algorithm has detection accuracy in the range of 94% to 97%. We implemented a framework for many-core architecture with SVM kernel while triggering Trojans based on two different conditions. To demonstrate the performance of proposed security framework, we implement a bio-medical seizure detection application as a case study. The algorithm is mapped on 64 processing cores and it takes 2.1μS to execute whereas with the proposed security framework it requires 4.8μS execution time. The Distributed Attack Detection Framework is implemented with each attack detection module having 2% area overhead.**

***Index Terms*—Hardware Security, Trojan Detection, Many-Core Design, Machine Learning, Support Vector Machine**

I. Introduction

Increased focus on R&D and reduction in time-to-market window in most of the semiconductor companies, a new trend has started to rely on Third Party Intellectual Properties (3PIP) and outsourcing fabrication process [1]. This raises serious security concern about Hardware Trojan inclusions in recent years. The Trojans in 3PIPs lead to malfunctioning and can create a backdoor to leak essential information to the attackers [2]. These type of hardware Trojans create new set of challenges for the user and impose an urgent need for trust validation in designs from third party vendors.

The Trojan detection can be performed at design-time, test-time and run-time. Run-time approaches have significant advantages over its counterparts: 1. Detecting Trojans for entire lifetime of the IC, 2. Trojans escaped during design and test time are detected at run-time, 3. It allows us to deploy a Trojan-inserted-IC while avoiding Trojan-infected-logic. Though run-time Trojan detection has several advantages, implementing run-time detection has large overheads. Trojans inserted at design or fabrication phase by untrustworthy third party vendors can certainly escape design and test-time Trojan detection methods. Therefore, low overhead run-time hardware Trojan detection is very important.

In this paper, we propose a low overhead run-time hardware Trojan detection framework using Machine Learning (ML) algorithms. ML algorithms are advantageous in several domains and have recently gained attention in hardware community for various applications [3]. Although practicing ML techniques on hardware is advantageous, it faces various challenges: 1. Relevant features extraction and its hardware implementation feasibility 2. Required number of observations for training data (memory requirement) 3. Hardware complexity of the ML algorithm 4. Effective placement of ML kernel etc.

The main contribution to the research include:

- Trojan Detection analysis on four commonly used Supervised ML algorithms.
- Feature data set generation for many-core (with 16 and 64 processing cores) architecture, feature selection based on correlation analysis and hardware implementation feasibility.
- Hardware implementation analysis of Support Vector Machine algorithm and its integration with custom many-core router architecture that can target both FPGA and ASIC.
- Fully placed and routed implementation of the proposed security framework on Xilinx Virtex-7 FPGA.
- Adopting the proposed security framework to demonstrate secure processing of a bio-medical seizure detection application on many-core platform.

The rest of the paper is organized as follows: Section II presents Trojan Insertion technique considered for the proposed work. Feature extraction and its optimization is discussed in Section III. Section IV discusses Trojan detection accuracy analysis for Supervised ML algorithms and explains SVM Supervised ML algorithm. Section V describes our proposed security framework for real-time Trojan detection using SVM Supervised ML algorithm and implements seizure detection algorithm as case study. Finally, Section VI discusses results and analysis.

II. Trojan Insertion Methodology

In this paper, we assume that IP cores, processing cores and memories are secured similar to Fiorin et.al. [4]. Therefore, Trojan can trigger an attack only through communication

TABLE I
Example Observations from "Golden Data Set" for Real-time Trojan Detection

Features /Obs	Source Core	Dest. Core	Route 0	Route 1	Route 2	Route 3	Route 4	Distance	Class	Attack Type
1	11	2	C	R0_2	R1_0	R0_0	B	4	Secured	-
2	11	2	C	R0_2	R1_0	R0_1	B	4	Trojan	Traffic Diversion
3	11	2	C	R0_2	0	0	C	2	Trojan	Route Looping
4	11	2	C	R0_2	R1_0	R0_3	D	4	Trojan	Core Address Spoofing
5	11	2	C	R0_2	0	0	A	2	Trojan	Core Address Spoofing

network on many-core platform. In this section, we discuss Trojan Insertion methodology considered for proposed work.

The Trojan is implemented at Design-phase and activated internally. There are two different ways Trojans can be activated internally: Always-on and Condition-based. As the name suggests, always-on Trojans are always active and can include malicious activity at any-time whereas condition-based Trojans are activated under specific conditions. In this paper, condition-based Trojan activation is implemented on internal logic state after particular number of core-to-core transfers or clock cycles.

Attacks on many-core router can affect network packet transfer rate, network/processing core availability and interruption in core communication [5]. The router (communication network) can be attacked externally through memory architecture interface, specialized core interface or internally by corrupting routing table to include different attacks such as Traffic diversions, Routing loops, Core spoofing attacks. All three attacks are also called as Denial-of-Service (DoS) attack, where in a specific core under attack is made unavailable. Table I shows example observation from "Golden Data Set" for the considered attacks.

- Traffic Diversion Attack : It is a very common attack in many-core router architecture. Under this attack, the router selects a random core to transfer the data. This attack affects the deadline for the other cores, which are dependent on the attacked transfer packet.
- Routing loop attack: Under this attack, the packets are routed back to the source core. The source core is made unavailable to other communicating cores, thereby causing latency in other core transfers.
- Core Spoofing Attack: This attack transfers all packets to randomly chosen (address) destination. The attack saturates the core and makes it unavailable for other cores.

Change in a single bit by an attacker at the router hop (any level) can modify the destination core address. To secure router from above mentioned attacks, we use different features. The feature values are formulated based on hardware functionality to form an observation. The supervised model is trained based on these observations to detect an attack in real-time.

III. Feature Extraction and Optimization

Collecting relevant data based on hardware behavior analysis is the first and most important step in this research. In a good ML data-set, each feature must contribute to the class i.e better correlation between feature and the class, but not among the features. Relevant attribute selection will aid both, increasing accuracy of Trojan detection and hardware implementation. Removing irrelevant features will reduce data-set thereby decreasing hardware complexity and memory usage. Therefore, we select relevant features based on feature correlation analysis. We consider following features for Trojan detection:

- Source Core : Source Core Number
- Destination Core : Destination Core Number
- Packet Transfer Path : Packet transfer between the two cores has a unique path which alters in case of Trojan. In this paper, we experiment on a 8×8 NoC i.e many-core architecture with 64 processing cores. It has three levels of router hop and therefore highest number of hops to be traveled by packet can be 5 for inter-cluster communication.
- Distance : At each router hop, distance *vector* is incremented by 1. For example, when core 11 is transferring packet to core 62, distance *vector* is incremented at $R0\_2$ (Distance=1), $R1\_0$ (Distance=2), $R2\_0$ (Distance=3), $R1\_3$ (Distance=4), $R0\_3$ (Distance=5). Since there will be six vertices (5 router hops and 1 processing core) and five edges, highest distance is 6.

Table I gives an example of expected test records received from router in case of Trojan / attack (Class "0") and without Trojan (Class "1") when packet is transferred from core 11 to core 2. Observation 3 shows routing loop attack whereas observation 4 and 5 show core spoofing attack.

IV. Machine Learning Algorithms

We examine commonly used Supervised ML algorithms such as Support Vector Machine (SVM), Decision Tree (DT), Linear Regression (LR), and K-Nearest Neighbors (KNN) for the Trojan detection analysis for many-core architecture. We also evaluated Un-supervised algorithms for Trojan detection,which has shown detection accuracy in the range of 50% to 80% [6]. Matlab statistics toolbox and Linear Library

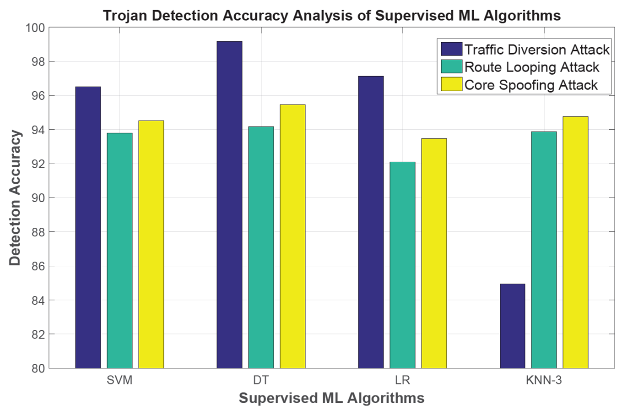

Fig. 1. Detection Accuracy Analysis of Supervised ML Algorithms, SVM-Support Vector Machine, DT-Decision Tree, LR-Linear Regression, KNN-K-Nearest Neighbor

TABLE II
TIME COMPLEXITY ANALYSIS OF SUPERVISED ML ALGORITHMS, WHERE n SIZE OF TRAINING DATA, m SIZE OF TEST DATA AND p FEATURES

Algorithm	Complexity of Learning Model	Complexity of Prediction Model
K-NN	-	$O(knp)$
Decision Tree	$O(pn^2 \log(n))$	$O(m)$
Linear Regression	$O(p^2n)$	$O(pm)$
Support Vector Machine	$O(pn^3)$	$O(pm)$

(LibLinear 1.94) is used for implementation and accuracy analysis of ML algorithms. The training data set consists of 716 different observations with 8 attributes and a binary class whereas, test data set consists of 290 observations.

A. Accuracy Analysis on Supervised Learning Algorithms

Figure 1 shows accuracy analysis on supervised algorithms. Accuracy is calculated as number of correctly labelled observations in total labelled observations. The analysis is performed for each attack separately. Decision tree has better detection accuracy for traffic diversion attacks among all Supervised learning algorithms. Linear regression algorithm detects 97% of traffic diversion attacks however, it performs worst for route looping and core address spoofing attacks among all supervised algorithms. KNN algorithm with ($K = 3$) neighbors, has lowest detection accuracy for traffic diversion attacks. Decision tree algorithm performs better than Linear regression irrespective of types of attack. SVM performs better than K-NN and Linear Regression algorithms. It can be noticed that SVM has detection accuracy in range of 94% to 97%.

B. Hardware Complexity Analysis of Supervised Learning Algorithms

In this section, we perform complexity analysis of Supervised learning algorithms. Classification of a test observation, in Supervised ML algorithm involves two steps. First kernel is to form a supervised model using training data set, whereas second kernel involves predicting a test observation based on trained supervised model.

Table II shows Learning and Prediction complexity of the Supervised Algorithms. In most of the supervised algorithms complexity of learning model is greater than the complexity of prediction model. Thus to reduce hardware complexity, we train the model offline by using “Golden Data set”. The “Golden Data set” is created by emulating real traffic loads on many-core hardware while randomly injecting attacks on the routers and obtaining *feature_sample* for each communication, based on the three types of attacks as discussed in Section II. The Golden Data Set consists of 60% - Feature data without Trojan (Class-1) and 40% - Feature data with Trojan (Class-0). Golden Data (training) set is built to detect the three attacks (as mentioned in Section II) efficiently.

TABLE III
HARDWARE COMPLEXITY ANALYSIS OF SUPERVISED ML ALGORITHMS WHEN TRAINED OFF-LINE, WHERE n SIZE OF TRAINING DATA, m SIZE OF TEST DATA AND p FEATURES

Algorithm	Multiplications	Additions	Memory Requirement
K-NN	$p \times n \times m$	$(p-1) \times n \times m$	$n \times p$
Decision Tree	$p \times n$	$n \times 2^{p+1} - 1$	p
Linear Regression	$p^2 \times n + p^3/3$ $-p^2/2 + p/6$	$p^2(n-1)$ $+(p^3/3 - p/3)$	$n \times p$
Support Vector Machine	$p \times m$	$m \times (p-1)$	p

Table III shows the hardware complexity of the supervised algorithms when training is performed off-line. For K-NN algorithm, though training cannot be performed offline, computation requirements are almost equal to Decision tree algorithm. LR and SVM have same computational and memory requirements as both algorithms give weight vectors after training. The obtained weight vector is used to predict the test class. Training ML algorithm offline reduces two overheads i.e resource utilization and execution time. Based on the complexity and detection accuracy analysis, SVM is the best option among these algorithms for further hardware implementations.

C. Support Vector Machine Algorithm

Support Vector Machine (SVM) is an efficient Supervised ML algorithm which provides good generalization performance for both classification and regression exercises. SVM algorithm adapts training data consisting of features and its desired class, to model and construct the weighted function for test data prediction. It consists of two phases: “learning phase” where SVM identifies closest data points to decision boundary known as Support Vectors (SVs), which forms best separation among the classes. These SVs are used to predict the class of test record in the “prediction phase”.

Figure 2 shows the binary classification problem. The aim of SVM is construction of decision surface (*W.X -b=0*) to find maximum separation between the classes. Thus, future test record can be predicted using equation 1.

$$f(x) = sign(\sum(W_i \times X_i) + b) \quad (1)$$

where, W is weight vector formed by using SVs, X is test record and b is bias.

D. Hardware Implementation of Support Vector Machine

Implementing hardware architecture for ML algorithms faces several challenges such as pre-processing of fea-

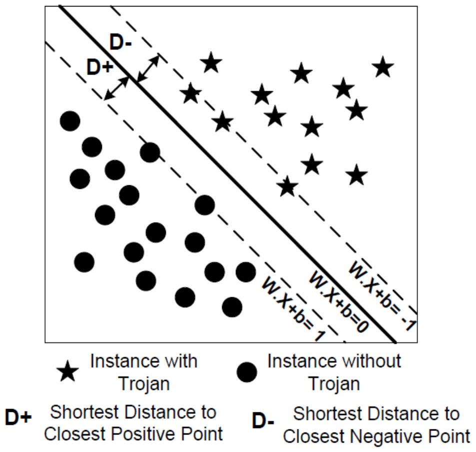

Fig. 2. Support Vector Machine Separating Hyperplane

tures, computational model implementation, managing memory transfers etc. The SVM is trained offline i.e. the Support Vectors are found offline and test records are predicted based on Support Vectors on-line. In this study, SVM is trained by using “Golden Data Set” on Matlab toolbox and SVM Linear Library (liblinear-1.94). Since the SVM is trained offline using “Golden Data Set”, it already knows the patterns of communication with and without Trojan. The SVM uses “Golden Data Set” to formulate function (in terms of weight vector and bias), the test feature is then mapped onto function to detect a Trojan.

We consider Source Core, Destination Core, Packet Transfer Path (router hop), and Distance features for Trojan detection. For 16-core many-core architecture 716 training records each with 8 features are used, whereas for 64-core many-core architecture 7260 training records each with 10 features are used to build “Golden (Training) Data Set”. The weight vector is, 8×1 for 16-core architecture whereas in case of 64-core router it is 10×1 and each vector is 6-bit. The 6-bit Bias is obtained from training data set. The SVM architecture consists of three main blocks: Test Feature Extraction, SVM Computation and Post Processing, as shown in Figure 3. These three blocks are interdependent, and designed in pipeline to reduce execution cycles. The input to SVM is *feature_sample*, which is formed by router architecture. The 16-core router *feature_sample* is 24-bit, whereas 64-core *feature_sample* is 30-bit. In Test *feature_sample* Extraction Module, the *feature_sample* is separated to form each *feature vector*. This *feature vector* is then padded with zero to match with the *weight vector*. The SVM Computation block consists of Dot Product and Bias Vector Addition blocks. The predicted class is calculated by using equation 1. Finally, Post Processing block converts calculated predicted class to the binary class i.e. feature data with Trojan or without Trojan.

V. Case Study: Security Framework for Seizure Detection Algorithm on Many-Core Platform

In order to demonstrate the efficiency of the proposed security framework, we implement a bio-medical seizure detection algorithm. Approximately 1% of world’s population

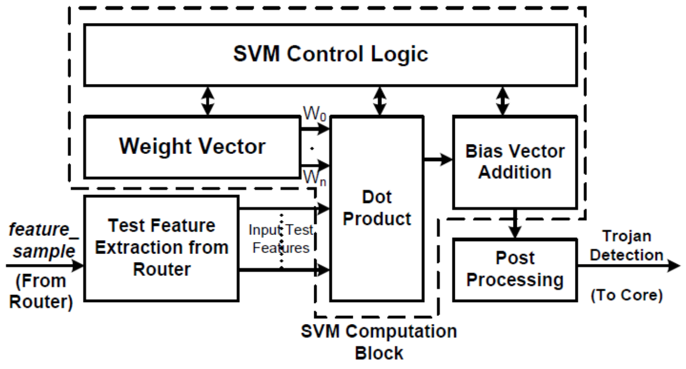

Fig. 3. Architecture for Support Vector Machine Kernel

suffers from Epilepsy. Epileptic patients experience seizure causing involuntary motion, loss of cognition etc. The diagnosis of epilepsy is performed based on seizure detection which requires placement of electrode (sensing channel) on the patient’s scalp. Therefore, seizure detection should have high accuracy and low hardware overhead. Previously proposed seizure detection algorithm mapping on many-core platform targeted low-power and reduced latency detection [7]. In this paper, we mapped Seizure detection algorithm with 16-channels on low power many-core platform. Figure 4 shows mapping of Seizure Detection algorithm on many-core platform and it consists of three main kernels: 1. Detection Phase 2. Analysis Phase 3. Band Analysis Phase. The detection phase is performed in time-domain by determining EEG high frequency components and comparing their magnitude with predetermined threshold values that are calibrated for each patient. In analysis phase, signals are converted in frequency domain by FFT-blocks, where we implement 128-pt FFT decomposed in 8-pt and 16-pt FFT blocks. Data Energy separation in four frequency bands: Theta (4-7 Hz), Alpha (8-12 Hz), Beta (13-29 Hz) and Gamma (30-50 Hz) is performed in Band Analysis phase. The seizure detection algorithm requires 61 cores, where communication instructions (60%) dominates execution time and takes 2.1μS to execute.

A. Many-Core Test Setup

Figure 5 shows the 64-Core test setup implemented on Xilinx Virtex-7 FPGA. The test setup consists of three important modules : 1. 64-Core Many-core Architecture, 2. Attack Detection Module and 3. Trojan Insertion Module. The seizure detection algorithm is mapped on 64-Core many-core architecture. It takes 458 inter-cluster and 1088 intra-cluster communications, where each cluster consists of four processing cores to execute the application. For each core to core communication, *Router Packet* is generated which has data to be transferred and also address of the destination core. *feature_sample* is updated at each communication hop and it contains features discussed in Section III. At the source core, *feature_sample* is updated with two features i.e. source core, destination core, and other feature vectors are initialized as *zeros*. At each router hop, *feature_sample* updates other features i.e path and distance. Finally at destination router i.e

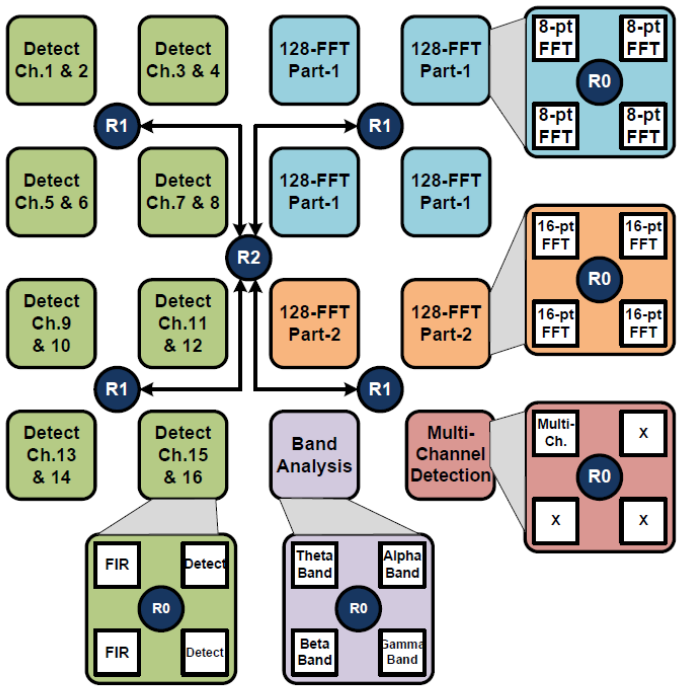

Fig. 4. **Seizure Detection Algorithm Mapping on Many-Core Platform, where R0- Router Level-0, R1-Router Level-1, and R2-Router Level-2**

before the destination core, *feature_sample* is transferred to the Attack Detection Module which is designed based on SVM as discussed in Section IV-D.

We test our real-time post deployment architecture for Trojan detection by emulating the three attacks in hardware. We consider that all routers are malicious and the Trojan can be triggered on routers using Trojan Insertion Module. The Trojan insertion module is condition based and triggers the attack after either 100 clock cycles or 8 *Router Packet* transfers. Upon trigger, destination core field in *Router Packet* is altered to a random number. In case of an attack, it corrupts router data in turn corrupting the *feature_sample*. The Attack Detection Module detects the attack using SVM kernel. In case of attack detection, the Attack Detection Module outputs $Class-0(low)$ and the attacked packet is dropped. However, in case of no-attack, it sends *"Destination Core Enable"* signal to the destination core to accept the packet. To reduce the communication latency we implemented Distributed Attack Detection Module for each four clusters of cores (i.e. for each 16-core) and one for inter-cluster communications. The proposed framework is scalable, Distributed Attack Detection Framework can be implemented in case of large NoC by placing SVM Kernel at every router level.

VI. Implementation Results

Many-Core architecture consisting of 64 processing cores is fully placed and routed in Xilinx Virtex-7 FPGA. We implement Distributed Attack Detection Framework by placing SVM Kernel at two different router levels. Each SVM kernel will detect intra-cluster attack separately and hence reduction in latency of operations. Also, as shown in Table IV, the area requirement for SVM kernel is very less as compared to router architecture and hence concern for area or power bottleneck doesn't arise. Table IV shows area analysis and security overhead. The security kernel overhead is due to Attack Detection Module and peripheral combinational logic. Security kernel adds 3 cycles to each data transfer between 16-cores (intra-cluster), whereas 4 cycles for inter-cluster data transfer. The latency overhead of security kernel is calculated by using a counter, which runs at many-core frequency. The counter is enabled when data packet reaches to the destination (desired or replaced) core. The feature_sample contains information which is updated at each router hop (For example as shown in Table I). The feature_sample is transferred through separate bus from each hop to another hop and then to attack detection module. The area overhead analysis includes the additional bus for feature_sample. Therefore, feature_sample will introduce area overhead and does not affect bandwidth of the router. The seizure detection algorithm requires 4.8 μS to execute with the proposed security framework. SVM kernel achieves 93% average Trojan detection accuracy for seizure detection. The seizure detection application the data transfers are performed at 88 Kbps, thus security overhead will not affect detection performance. Table V compares hardware overhead of the proposed platform with previously published papers.

TABLE IV
Area Analysis on Xilinx Virtex-7 FPGA

Logic Utilization	Many-Core Only	Many-Core with Security	Security Overhead	Available
Slice Count	55,072	55,220	148 (0.26%)	75,900
Register Count	49,472	49,830	358 (0.72%)	607,200
LUT Count	142,008	142,281	273 (0.2%)	303,600
Distributed Memory Count	11,244	11,244	-	130,800

TABLE V
Comparison of Security Architecture with Previous Works, Where S= # Slices, R= # Registers, L= # LUTs. Both Virtex-6 and Virtex-7 slices contain four LUTs and eight flip-flops

	Platform Kernel	Area Overhead	Latency Overhead (Cycles per Communication)
Cotret et.al. [8], [9]	*Virtex-6*	S×138 +R×123+L×293	8
Diguet et.al. [10]	*Virtex-6*	S×10750	-
This Work Many-Core	*Virtex-7*	S×29 +R×98+L×57	3

VII. Conclusion

In this paper we present low-overhead security framework using Machine Learning techniques. We assume that processing cores and memories are secured and Trojans are included only through router. The Trojan attack corrupts the router packet by changing the destination address in terms of traffic diversion, route looping or core spoofing attack. We built "Golden Data Set" based on hardware feature analysis and Trojan insertion effects to train ML model. In this work, we experimented four commonly used Supervised Ma-

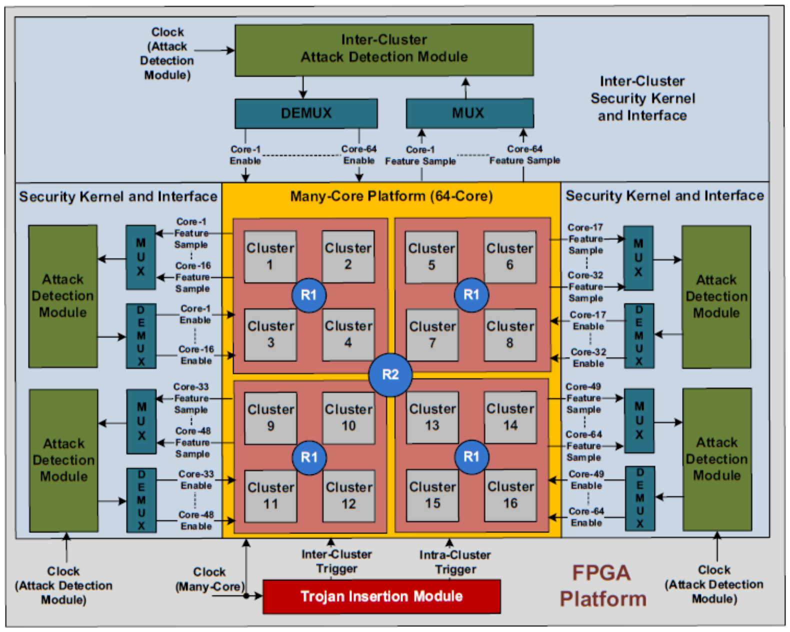

Fig. 5. Test Setup for Many-Core Platform with 64 Processing Cores on Xilinx Virtex-7 FPGA, Intra-Cluster trigger is used to attack all Level-1(R1) routers and Inter-Cluster trigger is used to attack Level-2(R2) routers

chine Learning techniques. We chose Support Vector Machine (SVM) to be implemented on hardware based on Accuracy and Hardware Complexity analysis. We implemented Support Vector Machine on hardware to detect Trojans at run-time. To demonstrate the performance of the proposed security framework, seizure detection algorithm is implemented on many-core platform with 64 processing cores. To reduce latency of Trojan detection, we implement distributed attack detection module, where each attack detection module has 2% area overhead. The results shows that the proposed security framework achieves average of 93% detection accuracy for seizure detection application and executes it in 4.8μS.

References

[1] S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M. Hsiao, J. Plusquellic, and M. Tehranipoor, "Protection against hardware trojan attacks: Towards a comprehensive solution," *Design Test, IEEE*, vol. 30, no. 3, pp. 6–17, June 2013.

[2] C. Liu, J. Rajendran, C. Yang, and R. Karri, "Shielding heterogeneous mpsocs from untrustworthy 3pips through security- driven task scheduling," *Emerging Topics in Computing, IEEE Transactions on*, vol. 2, no. 4, pp. 461–472, Dec 2014.

[3] S. Cadambi, I. Durdanovic, V. Jakkula, M. Sankaradass, E. Cosatto, S. Chakradhar, and H. Graf, "A massively parallel fpga-based coprocessor for support vector machines," in *Field Programmable Custom Computing Machines, 2009. FCCM '09. 17th IEEE Symposium on*, April 2009, pp. 115–122.

[4] L. Fiorin, S. Lukovic, and G. Palermo, "Implementation of a reconfigurable data protection module for noc-based mpsocs," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, April 2008, pp. 1–8.

[5] F. Gebali, H. Elmiligi, and M. El-Kharashi, *Networks-on-Chips: Theory and Practice*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2009.

[6] A. Kulkarni, Y. Pino, M. French, and T. Mohsenin, "Real-time anomaly detection framework for many-core router through machine learning techniques," *ACM Journal on Emerging Technologies in Computing*.

[7] J. Bisasky, D. Chandler, and T. Mohsenin, "A many-core platform implemented for multi-channel seizure detection," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, May 2012, pp. 564–567.

[8] P. Cotret, J. Crenne, G. Gogniat, and J.-P. Diguet, "Bus-based mpsoc security through communication protection: A latency-efficient alternative," in *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*, April 2012, pp. 200–207.

[9] P. Cotret, F. Devic, G. Gogniat, B. Badrignans, and L. Torres, "Security enhancements for fpga-based mpsocs: A boot-to-runtime protection flow for an embedded linux-based system," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2012 7th International Workshop on*, July 2012, pp. 1–8.

[10] J.-P. Diguet, S. Evain, R. Vaslin, G. Gogniat, and E. Juin, "Noc-centric security of reconfigurable soc," in *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, May 2007, pp. 223–232.

[11] A. M. Kulkarni, H. Homayoun, and T. Mohsenin, "A parallel and reconfigurable architecture for efficient omp compressive sensing reconstruction," in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, ser. GLSVLSI '14. New York, NY, USA: ACM, 2014, pp. 299–304.

[12] M. Khavari Tavana, A. Kulkarni, A. Rahimi, T. Mohsenin, and H. Homayoun, "Energy-efficient mapping of biomedical applications on domain-specific accelerator under process variation," in *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*, ser. ISLPED '14. New York, NY, USA: ACM, 2014, pp. 275–278.

[13] A. Kulkarni and T. Mohsenin, "Accelerating compressive sensing reconstruction omp algorithm with cpu, gpu, fpga and domain specific many-core," in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, May 2015, pp. 970–973.

[14] A. Kulkarni, A. Jafari, C. Sagedy, and T. Mohsenin, "Sketching-based high-performance biomedical big data processing accelerator," in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, May 2016.

[15] A. Page, C. Sagedy, E. Smith, N. Attaran, T. Oates, and T. Mohsenin, "A flexible multichannel eeg feature extractor and classifier for seizure detection," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 62, no. 2, pp. 109–113, Feb 2015.

[16] A. Page, A. Kulkarni, and T. Mohsenin, "Utilizing deep neural nets for an embedded ecg-based biometric authentication system," in *IEEE Biomedical Circuits and Systems (Biocas) Conference*, Oct 2015.

[17] A. Jafari and T. Mohsenin, "A low power seizure detection processor based on direct use of compressively-sensed data and employing a deterministic random matrix," in *IEEE Biomedical Circuits and Systems (Biocas) Conference*, Oct 2015.