Real-Time Hardware-Based Malware and Micro-Architectural Attack Detection Utilizing CMOS Reservoir Computing

Sanjeev Tannirkulam Chandrasekaran, *Member, IEEE*, Abraham Peedikayil Kuruvila, Kanad Basu, and Arindam Sanyal, *Member, IEEE*

***Abstract*—In this work we demonstrate a novel CMOS reservoir computer (RC) prototype in 65nm CMOS that is capable of detecting Malware and micro-architectural attacks in real-time utilizing hardware performance counter (HPC) traces. A 65nm test chip achieves 96.8% and 96.5% accuracy when classifying Malware and micro-architectural attacks respectively, while achieving better classification performance than digital machine learning models and with lower energy consumption. The on-chip classifier consumes 38.2μW from a 1.2V supply while running at 40kHz.**

***Index Terms*—Hardware based malware detection, malware, micro-architectural attack, reservoir computing, machine learning, hardware performance counters.**

I. Introduction

IN THE era of IoT and ubiquitous computing, computing systems are increasingly under attack from sophisticated adversaries. These attackers deploy malicious software, or Malware, to extract sensitive private information or harm computing systems. Traditionally, anti-virus software (AVS) programs are used to counter the threat of Malware. However, commercial AVS programs require significant computational overhead to detect complex malware [1]. In recent years, hardware based Malware detectors (HMDs) have been shown to be a promising alternative to software AVS both in terms of computational overhead and robustness against different attacks [1], [2]. HMDs use dedicated hardware features, such as hardware performance counters (HPCs) which are special registers built into modern processors, for Malware detection. HPCs count low-level micro-architectural features such as cache misses, branch misses, etc. and will have different counts for benign and Malware. HMD can be automated by using artificial intelligence (AI) algorithms to classify HPC sequences as benign or Malware. Interestingly, the same HMD can also be used to detect micro-architectural attacks by detecting changes in HPC patterns.

The contribution of this work is to demonstrate a mixed-signal AI prototype that acts an HMD and accurately detects Malware and micro-architectural attacks from HPC traces. A reservoir computing (RC) architecture is used for the AI classifier model. An RC network uses random, non-linear projection to map the input signal from a low-dimensional plane to a high-dimensional plane, such that distinct classes in the input data become linearly separable [3], [4]. RC neural network (RCNN) uses intrinsic memory in the projection layer, which acts as reservoir, to identify patterns in a time-series signal. The raw HPC traces are used as inputs to the RCNN without feature extraction step, which reduces hardware and energy cost. All computations, except in the output layer, are performed in analog domain which eliminates energy associated with frequent memory access in digital computations. Weights in the input and reservoir layer are drawn from static random distributions, and only the output layer requires training. Thus, the same reservoir layer can be re-used to classify different categories of Malware and micro-architectural attacks by re-tuning only weights in the output layer. A prototype RCNN is fabricated in 65nm CMOS process, and demonstrated on two HPC dataset for detection of Malware and micro-architectural attacks.

Manuscript received June 21, 2021; accepted July 20, 2021. Date of publication August 4, 2021; date of current version January 31, 2022. This brief was recommended by Associate Editor S. Wang. *(Sanjeev Tannirkulam Chandrasekaran and Abraham Peedikayil Kuruvila contributed equally to this work.) (Corresponding author: Sanjeev Tannirkulam Chandrasekaran.)*

Sanjeev Tannirkulam Chandrasekaran and Arindam Sanyal are with the Department of Electrical Engineering, University at Buffalo, Buffalo, NY 14260 USA (e-mail: stannirk@buffalo.edu).

Abraham Peedikayil Kuruvila and Kanad Basu are with the Department of Electrical and Computer Engineering, UT Dallas, Richardson, TX 75080 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2021.3102526.

Digital Object Identifier 10.1109/TCSII.2021.3102526

II. Background

A. Malicious Dataset

Malware, or malicious software, can exhibit a spectrum of behavior from leaking sensitive user information to effectively shutting down a system. In this brief, we concentrate explicitly on identifying the following three types of Malware.

- *Mirai* attempts to take over an Internet-connected device in order to turn it into a controlled bot.
- *Trojans* are a type of virus that appears legitimate, but intends to take over a device in order to inflict harm.
- *Rootkits* are malicious software that allows attackers to secure privileged-level access of a device.

While Malware present a significant threat to system security, the advancement of computer architecture has engendered new windows for attackers to exploit. Computer architectures have highly improved to incorporate performance

enhancements. Some of these low-level optimization features include speculative branching and out-of-order execution. These optimization techniques enable increased concurrency, ultimately culminating in reduced run times. However, these optimizations can be exploited by several micro-architectural attacks to subvert the system, specifically to obtain leaked information or modify sensitive data. In this brief, we concentrate on distinguishing the following five micro-architectural attacks.

- *Spectre* attacks trick a processor into speculatively executing instructions that shouldn't be executed [5]. Consequently, the attacks are able to read memory addresses not belonging to the adversary.
- *Meltdown* leaks a target's physical memory by exploiting out-of-order execution. It is developed on the assumption that an instruction that produces a fault results in out-of-order instructions that were executed to be aborted [5].
- *ZombieLoad* attacks observe the contents of memory loads and stores on the CPU core. However, an attacker cannot establish which value to leak based on a target address. Instead, the ZombieLoad attack leaks whatever value currently loaded or stored in the physical CPU core [6].
- *Flush + Flush* attacks utilize the *cflush* instruction to repeatedly flush cache lines while measuring the execution time of the *cflush* instruction. This attack can be used to eavesdrop on user keystrokes.
- *Rowhammer* attacks exploit DRAM-based devices by repeatedly accessing a row of memory, thus resulting in bit flips in adjacent rows which can damage a system [7].

B. Hardware Performance Counters

Hardware Performance Counters (HPCs) are special purpose registers that keep track of low-level micro-architectural events such as branch-instructions, cache misses, etc. They are found in most processors, and the HPCs available vary depending on the processor. Based on the processor architecture, only a limited amount of HPCs, typically up to four, can be concurrently monitored [2]. HPCs have been recently used for improving system and hardware security, e.g., Malware and micro-architectural attacks detection [8], [9].

1) Hardware Performance Counter Collection: In order to collect the HPC data for both Malware and microarchitectural attacks, we utilized the Linux OS package called *linux-tools-common* which gives access to *perf*. The proper constraint parameters to the *perf stat* command, such as period of collection, enables obtaining HPCs values from an application.

In this work, we have retrieved four HPC measures per *perf stat* command every *one millisecond* from the target processor. We collected HPC samples from benign and Malware applications on an ARM Cortex A53 processor for a total of 60,000 HPC samples, referred to as Malware dataset for the rest of this brief. Similarly, we collected HPC data from benign programs and microarchitectural attacks on an Intel Core i5-4210U processor for a total of 180,000 samples, referred to as mico-architectural attack dataset for the rest of this brief.

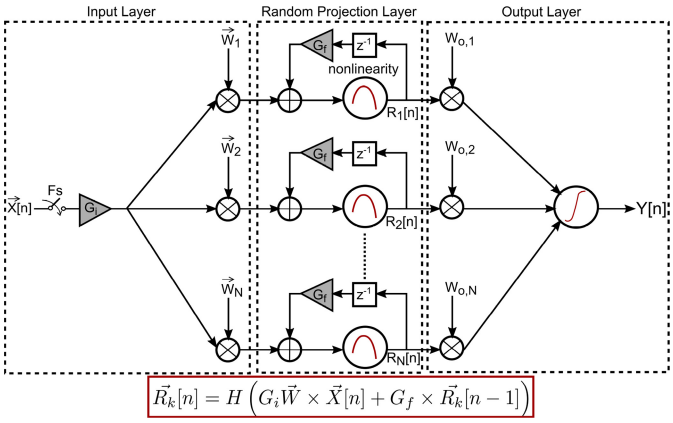


Fig. 1. Proposed reservoir computer architecture.

The Malware and microarchitectural attacks consist of the applications explained in Section II-A, while the benign program dataset consists of the Mibench, Chstone, and Phoronix benchmarks [10]–[12].

III. Reservoir Computing Based HMD

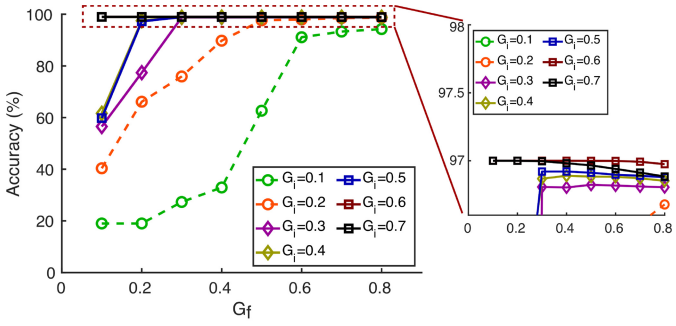
A. RCNN Architecture

RC is an excellent candidate for on-chip implementation of energy-efficient HMD due to its simple architecture and reduced training requirement. Fig. 1 depicts the proposed RC architecture with N reservoir neurons, and an output layer that performs logistic regression (LR). An input $\vec{X}$ with d features, is multiplied with an input weight matrix $\vec{W}$ of dimension $N \times d$ and passed through a non-linearity that projects the input vector to high-dimensional plane. Mathematically, state of the k-th reservoir neuron can be expressed as

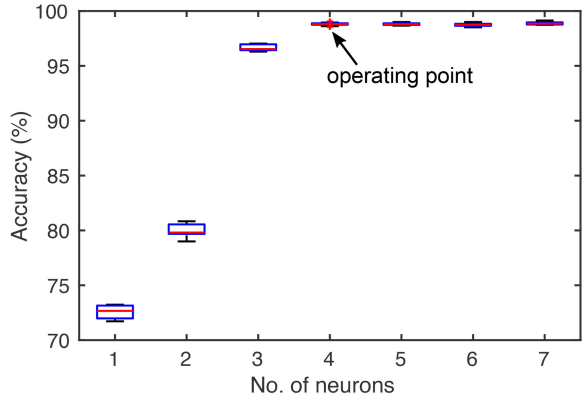
$$\vec{R}_k[n] = H\big(G_i\vec{W} \times \vec{X}[n] + G_f\vec{W}_r \times \vec{R}_k[n-1]\big) \qquad (1)$$

where $H(\cdot)$ is reservoir non-linear activation function, G_i is input scaling factor, G_f is feedback gain, and $\vec{W}_r$ is the reservoir inter-connection weight matrix. For the proposed RC, $\vec{W}_r$ is sparsely filled identity matrix, which provides memory to the reservoir, and allows the RC to exhibit properties of high dimensionality while using a small number of neurons [13]. The matrices $\vec{W}$ and $\vec{W}_r$ are derived from a static random distribution. Elements of $\vec{W}$ and $\vec{W}_r$ are restricted to $\{0/1\}$, which reduces computational cost by replacing multipliers in the input and reservoir layers with adders, and this in-turn reduces the energy consumption of the RC. The reservoir states are digitized by an analog-to-digital converter (ADC) into digital word $d[n]$, before being sent to the output layer for producing the binary prediction result, $Y[n]$. Weights in the output layer is denoted by $\vec{W}_o$ in Fig. 1.

The RCNN architecture is optimized on the Malware dataset for input scaling factor, G_i, feedback gain, G_f, and number of reservoir neurons, N. The dataset is randomly partitioned into 80% training, and 20% hold-out test set, with each set evenly balanced. Classification accuracy is calculated on the test set, and the simulation is repeated 50 times for each (G_i, G_f) pair, and for each N. Fig. 2 reports the classification accuracies on



(a)



(b)

Fig. 2. Classification accuracy as a function of a) (G_i, G_f); and b) number of reservoir neurons (N).

the test set. The highest classification accuracy is obtained for $G_i = 0.6$, $G_f < 0.6$ and $N = 4$. G_f is set to 0.4 for this design.

In order to ensure that the reservoir states are repeatable, the reservoir states should not vary significantly with changes in operating conditions, and noise in the reservoir should be low. Fig. 3(a) shows the simulated classification accuracy on the test set as the parameters G_i, G_f, N are perturbed randomly. The worst-case classification accuracy remains above 90% as long as perturbations in the RC parameters is within 11%. Fig. 3(b) shows classification accuracy on the test set as a function of noise referred to ADC input. Classification accuracy remains relatively unaffected as long as the noise remains below 1mV, rms. Thus, the RC is relatively robust against perturbations and noise.

B. RCNN Circuit Design

Fig. 4 shows the circuit architecture for the proposed RCNN which implements (1). Reservoir non-linearity, $H(\cdot)$, is realized by a common-source amplifier with resistive feed-forward path as shown in Fig. 4. Output of the non-linearity circuit is fed to a unity gain buffer which drives a 10-bit successive approximation register (SAR) ADC. The SAR ADC uses 2.4fF unit capacitor in the DAC, and employs bi-directional single sided switching (BDSS) [14] to reduce switching energy in the DAC by 86% compared to conventional SAR switching techniques. The delayed output of the SAR ADC is fed back to the input through a resistive digital-to-analog converter (R-DAC). An operational transconductance amplifier (OTA)

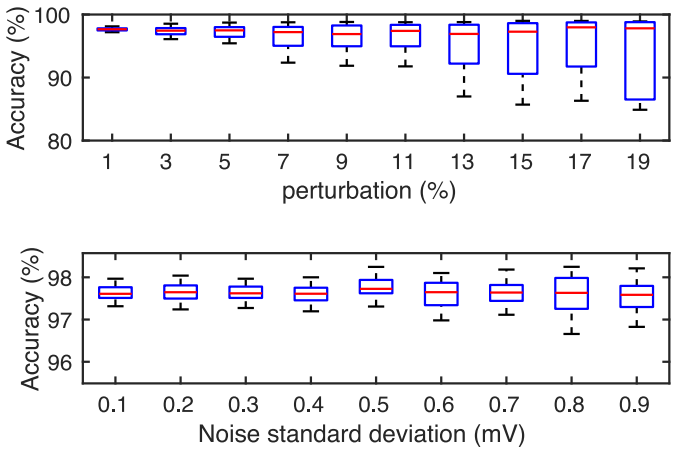


Fig. 3. Classification accuracy as function of a) perturbation in RCNN parameters, b) standard deviation of noise.

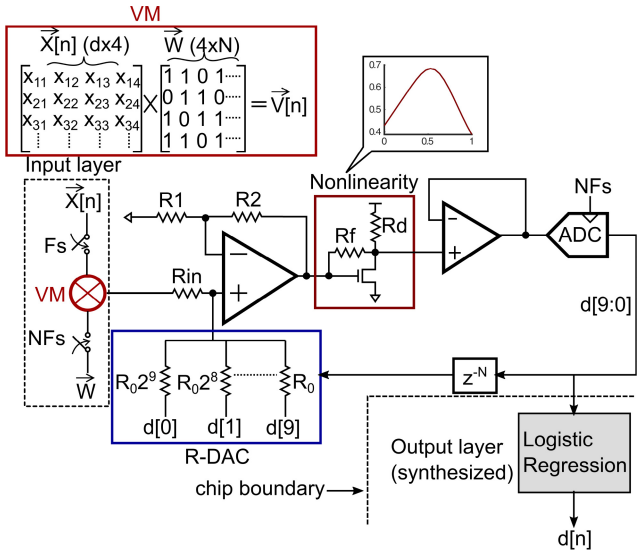


Fig. 4. Time-multiplexed RC circuit implementation.

performs summation of the weighted input with the scaled feedback signal. Indirect miller compensation is used in the OTA. The digitized reservoir states are sent to output layer for inference. In this work, the output layer is synthesized digitally and implemented off-chip. The reservoir is time-multiplexed to reduce area by $N\times$. While the input is sampled at a frequency of F_s, the rest of the circuit operates at $N \times F_s$ to realize N virtual reservoir neurons using 1 physical neuron. The vector multiplier (VM), that performs $\vec{W} \times \vec{X}[n]$ in (1), operates at $N \times F_s$. Similarly, the feedback is delayed by N-cycles of the ADC clock to realize the 1-cycle delayed feedback in (1).

Fig. 5 shows the simulated noise power spectral density (PSD) of the RC referred to the ADC input. The OTA, R-DAC, non-linearity circuit and unity-gain buffer (UGB) contribute 0.3mV$_{rms}$ noise over a 130MHz noise bandwidth, while the 10-bit ADC has an input referred noise of 0.54mV$_{rms}$ leading to total noise of 0.62mV$_{rms}$ which is not expected to degrade classification accuracy as shown in Fig. 3(b).

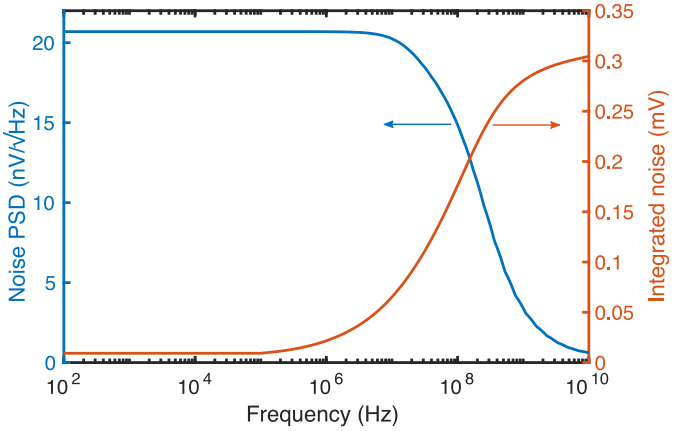
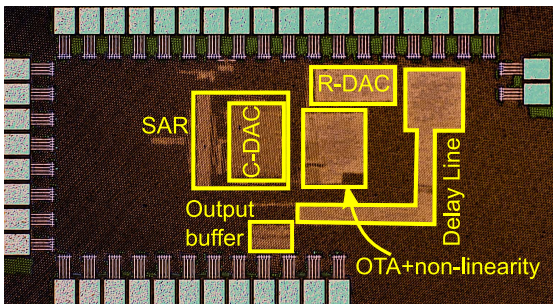
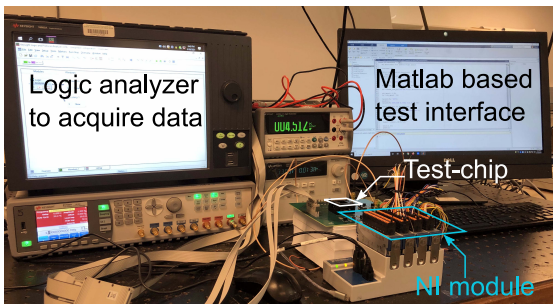


Fig. 5. Simulated noise PSD of the RC.



(a)



(b)

Fig. 6. RC chip micro-photograph.

IV. Measurement Results

The RCNN is fabricated in 65nm CMOS process and validated in the lab. Fig. 6(a) shows the chip micro-photograph, and Fig. 6(b) shows the laboratory measurement setup used to characterize the test chips. The RCNN has a core area of 0.24mm^2. The output layer is implemented digitally off-chip. The hardware performance counter data are loaded into the test chip through National Instruments Data Acquisition module, and the test-chip outputs are captured using logic analyzer. Data input to test-chip and output acquisition is synchronized through MATLAB interface running on a desktop computer.

The RCNN operates from a 1.2V supply at $F_s = 40$kHz and consumes 32μW power, while the digitally synthesized output layer is estimated to consume 6.2μW power. Thus, the complete RCNN has energy consumption of 0.95nJ/inference. The energy breakdown of the RCNN is shown in Fig. 7. The unity

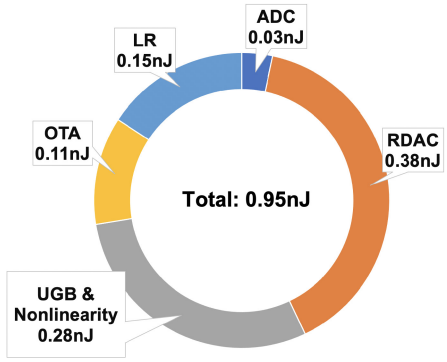


Fig. 7. Energy breakdown of RCNN.

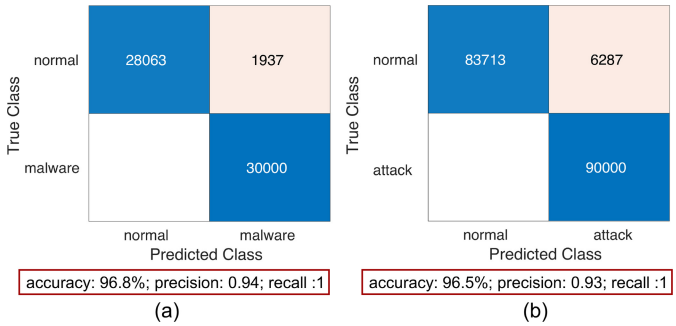


Fig. 8. Measured confusion matrices on (a) Malware; (b) micro-architectural attack dataset.

gain buffer (UGB) and non-linearity consume 29.4% of the RC energy, while the R-DAC consumes 37.6% of the overall energy.

To illustrate the efficacy of the RCNN for real-time malware detection, the RC was trained and tested with branch-misses, bus-cycles, and cache-misses information obtained from the HPCs of the two datasets described in Section II. A 5 fold cross-validation is performed on each dataset, and the results from the validation fold are collected to create individual confusion matrices for both the Malware and micro-architectural datasets as shown in Fig. 8. The RCNN achieves classification accuracy of 96.8%, precision of 0.94 and recall of 1 on Malware dataset. For micro-architectural attack dataset, the classification accuracy is 96.5% with precision of 0.93 and recall of 1. Fig. 9 shows the measured histogram of classification accuracy on the test set for 1000 repeated evaluation for both Malware and micro-architectural attack datasets. The standard deviation of accuracy is 1.19% and 1.26% respectively for the two datasets which shows robustness against noise.

Fig. 10 compares performance of 6 different software AI models on the Malware detection dataset – k-nearest neighbor (kNN), logistic regression (LR), linear support vector machine (SVM), naive bayes (NB), decision tree (DT) and random forest (RF). The RF classifier has the highest accuracy, precision and recall compared to the other classifiers. Table I compares the performance of the proposed RC based HMD with the RF classifier. The RC testchip achieves better classification performance than RF classifier, while consuming 5× lower energy/inference. If the proposed RC classifier is implemented

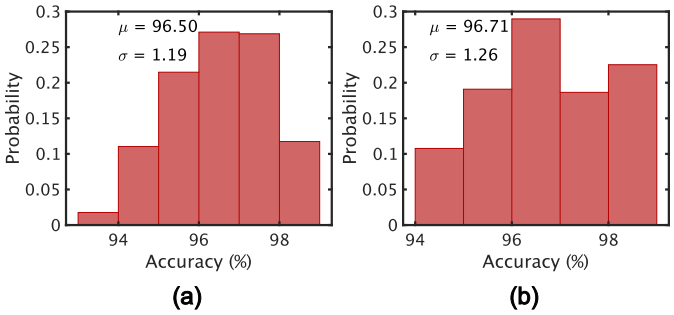


Fig. 9. Measured noise performance through repeated evaluations on (a) Malware; (b) micro-architectural attack dataset.

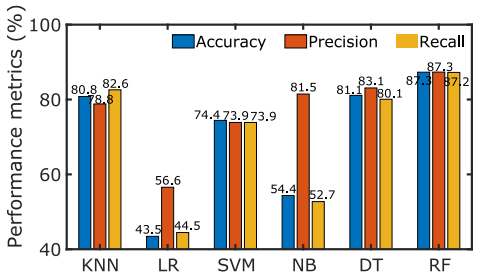


Fig. 10. Simulated performance of different software AI models on Malware detection dataset.

TABLE I
COMPARISON WITH DIGITAL AI MODELS

AI model	Accuracy	Precision	Recall	Energy
Malware detection dataset				
RC (This work)	96.8%	0.84	0.84	0.95nJ
Random Forest	87.3%	0.87	0.87	5.11nJ
Micro-architectural attack detection dataset				
RC (This work)	96.5%	0.92	0.81	0.95nJ
Random Forest	91.2%	0.90	0.92	5.11nJ

TABLE II
COMPARISON WITH PRIOR HMD WORKS

AI model	RCNN HMD	SVM [15]	DT [16]	NN [17]
Accuracy	96.8%	96.2%	87.4%	83.0%

digitally on an embedded system, such as Arm Cortex-M3 with 65nm processor, the fully digital classifier will require 238 instructions and consume 50× more energy than our prototype.

We additionally compare our detection results with existing HPC-based Malware detection techniques as shown in Table II. Prior works include SVM [15], DT [16] and neural network (NN) [17]. The NN is trained on MiBench dataset that has also been used for benign applications used in this work. The differences in classification accuracy between the AI models are attributed to the different Malware and benign programs utilized to train them. It should be noted that our RCNN HMD furnishes one of the highest accuracies at 96.8%. Therefore, our utilized HPC-based model is competent in producing performance metrics that are on par or exceed existing Malware Detection methods.

V. CONCLUSION

A reservoir-computing HMD is presented in this work that uses information in HPCs to identify Malware and micro-architectural attacks in real-time with high accuracy and low energy consumption. Energy consumption of the HMD can be further reduced by using inverter-based, single-stage amplifier and switched-capacitor DAC in the reservoir neurons. While the RCNN HMD is designed as stand-alone in this work, the RCNN can easily be integrated into most modern processors with built-in HPCs. The RCNN will collect the HPCs from the processor using bus-like communication protocol and perform real-time predictive analysis.

REFERENCES

[1] M. Ozsoy, K. N. Khasawneh, C. Donovick, I. Gorelik, N. Abu-Ghazaleh, and D. Ponomarev, "Hardware-based malware detection using low-level architectural features," *IEEE Trans. Comput.*, vol. 65, no. 11, pp. 3332–3344, Nov. 2016.

[2] K. Basu, P. Krishnamurthy, F. Khorrami, and R. Karri, "A theoretical study of hardware performance counters-based malware detection," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 512–525, 2019.

[3] S. Ortín *et al.*, "A unified framework for reservoir computing and extreme learning machines based on a single time-delayed neuron," *Sci. Rep.*, vol. 5, no. 1, 2015, Art. no. 14945.

[4] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Netw.*, vol. 20, no. 3, pp. 391–403, 2007.

[5] P. Kocher *et al.*, "Spectre attacks: Exploiting speculative execution," in *Proc. IEEE Symp. Security Privacy (SP)*, San Francisco, CA, USA, 2019, pp. 1–19.

[6] M. Schwarz *et al.*, "ZombieLoad: Cross-privilege-boundary data sampling," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, 2019, pp. 753–768.

[7] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer.js: A remote software-induced fault attack in javascript," in *Proc. Conf. Detection Intrusions Malw. Vulnerability Assess.*, 2016, pp. 300–321.

[8] J. Demme *et al.*, "On the feasibility of online malware detection with performance counters," *SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 559–570, 2013.

[9] S. Briongos, G. Irazoqui, P. Malagón, and T. Eisenbarth, "CacheShield: Detecting cache attacks through self-observation," in *Proc. Conf. Data Appl. Security Privacy*, 2018, pp. 224–235.

[10] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proc. Workshop Workload Characterization (WWC-4)*, Austin, TX, USA, 2001, pp. 3–14.

[11] Y. Hara, H. Tomiyama, S. Honda, H. Takada, and K. Ishii, "CHStone: A benchmark program suite for practical C-based high-level synthesis," in *Proc. Symp. Circuits Syst.*, Seattle, WA, USA, 2008, pp. 1192–1195.

[12] *Phoronix Test Suite Test Profiles*. Accessed: Jul. 20, 2021. [Online]. Available: https://openbenchmarking.org/tests/pts

[13] M. Le Berre, E. Ressayre, A. Tallet, H. M. Gibbs, D. L. Kaplan, and M. H. Rose, "Conjecture on the dimensions of chaotic attractors of delayed-feedback dynamical systems," *Phys. Rev. A*, vol. 35, no. 9, pp. 4020–4022, 1987.

[14] L. Chen, A. Sanyal, J. Ma, and N. Sun, "A 24-μW 11-bit 1-MS/s SAR ADC with a bidirectional single-side switching technique," in *Proc. 40th Eur. Solid-State Circuits Conf.*, Venice Lido, Italy, 2014, pp. 219–222.

[15] M. B. Bahador, M. Abadi, and A. Tajoddin, "HPCMalHunter: Behavioral malware detection using hardware performance counters and singular value decomposition," in *Proc. IEEE 4th Int. Conf. Comput. Knowl. Eng.*, Mashhad, Iran, 2014, pp. 703–708.

[16] B. Zhou, A. Gupta, R. Jahanshahi, M. Egele, and A. Joshi, "Hardware performance counters can detect malware: Myth or fact?" in *Proc. Conf. Comput. Commun. Security*, 2018, pp. 457–468.

[17] S. M. P. Dinakarrao *et al.*, "Adversarial attack on microarchitectural events based malware detectors," in *Proc. 56th ACM/IEEE Design Autom. Conf.*, Las Vegas, NV, USA, 2019, p. 164.