Hardware Trojans for Confidence Reduction and Misclassifications on Neural Networks

Mahdieh Grailoo
Tallinn University of Technology
mahdieh.grailoo@taltech.ee

Mairo Leier
Tallinn University of Technology
mairo.leier@taltech.ee

Samuel Pagliarini
Tallinn University of Technology
samuel.pagliarini@taltech.ee

2022 23rd International Symposium on Quality Electronic Design (ISQED) | 978-1-6654-9466-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISQED54688.2022.9806246

***Abstract*—With the rapid development of deep learning models, neural networks (NNs) have become a prominent solution for many complex problems. As such, NNs have been considered as hardware (HW) accelerators for best-in-class performance in these tasks. Nevertheless, once a NN is deployed in HW, it becomes susceptible to an array of attacks that have no direct counterpart in software (e.g., HW Trojan horses (HT)). The malicious logic inserted by a rogue element can alter the behavior of a NN in a stealthy way, escaping both test-based detection and human eye inspection. In this work, we propose 8 specialized HTs for NNs that are architecture-independent and cover varied adversarial goals, including misclassifications and confidence reduction. The proposed HTs require no toolchain manipulation nor access to the NN model, making their deployment feasible. Results on the MNIST set of handwritten digits show that our HTs can achieve a 100% attack success rate in all adversarial goals while incurring small overheads of about 0.2%, 2%, and 3% on average in resource usage, delay, and dynamic power, respectively. In order to quantify the HTs detectability, we have resorted to a reverse engineering technique, which reveals that the payload of some of our HTs has little impact on the netlist structure.**

***Index Terms*—Hardware Trojan Horses, Hardware Security, Neural Networks, Reverse Engineering.**

I. Introduction

Systems based on neural networks (NNs) have experienced very rapid adoption in many security-critical applications, which makes the associated security issues an urgent and severe concern [1]. Since NNs are often considered for hardware (HW) acceleration, HW attack vectors cannot be ignored [1]–[3]: a HW adversary can manipulate any neurons' weights and functionalities from any layer. Meanwhile, a software (SW) attack is almost entirely bound to manipulating input data. Pure HW attacks, especially the ones hypothesized to occur at an untrusted silicon foundry [4], have the flexibility of virtually changing any element of a circuit. SW and HW attacks in NNs share common goals of confidence reduction (CnfRdct), misclassification (MsCls), targeted misclassification (trgtd MsCls), and source/target misclassification (src/trgt MsCls). However, generally speaking, HW attacks have a much higher difficulty to mount. From a detection point of view, SW attacks that manipulate input patterns may make them visually distinguishable for the human eye, a concern that is not present for HW attacks.

Among HW attacks, the insertion of HW trojan horses (HT) is one of the most well-studied attacks [4]. A generic HT is composed of a trigger and a payload, where the former is responsible for determining the start condition, and the latter alters the behavior of the *infected* circuit. The trigger is purposefully designed to be very rare, which makes the HT stealthy by nature and hard to detect by conventional tasks within the integrated circuit (IC) design flow. In particular, traditional IC test methods fall short in detecting HTs. They are mainly geared towards identifying modeled defects; thus, they cannot reveal unmodeled malicious insertion of logic, especially when said logic is carefully hidden and do not visibly alter the IC's functionality [5], [6].

In this work, we propose **specialized HTs for NNs**. Furthermore, we investigate malicious logic that is architecture-independent and covers all adversarial goals. The herein proposed HTs does not visibly alter the NN functionality or cause it to stop operating. Instead, the prediction's distribution is manipulated towards malicious goals. The HTs need to neither access the training data nor retrain the target model, meaning that the parameters of the original model are not manipulated, making the HTs practical. In our adversarial scenario, HTs can be inserted during manufacturing or system integration by untrusted semiconductor foundries or third parties, respectively.

II. Related Works and Contribution

In [1], a "memory Trojan" is presented, which implants the malicious logic into the NN accelerator's memory controllers of off-chip memory. The HT then indirectly detects the boundaries and types of layers of the NN by observing write patterns to the external memory. To detect the boundaries, it monitors write accesses which mainly occur near the "end" of each layer. To identify the type, it monitors the write access ratio, which is usually much higher in fully-connected layers than in others. This HT uses a fractal image as a trigger to retain the data semantic with even heavy scaling during trigger image identification. Since the trigger image is different from normal ones, it can be easily recognized by visual inspection.

The authors of [2] eliminate the need for a special image to act as a trigger. Instead, the HT is triggered by a sequence of normal/valid images. As in [1], a considerable amount of execution time is required for the HT to learn the boundaries of the internal NN structure. In addition, only the goals of rgtd MsCls and CnfRdct are addressed. In [2], the classes are determined without presenting any probabilities. The authors

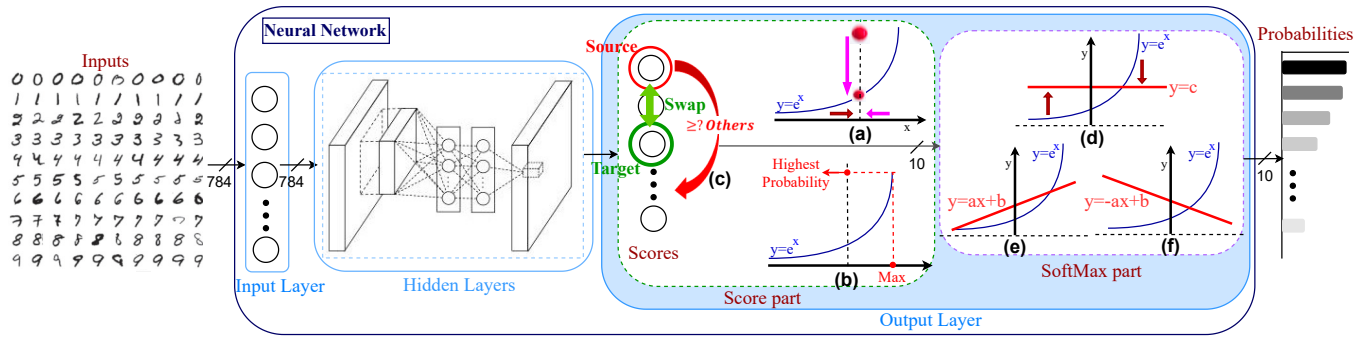

Fig. 1. HTs explained: a) moving scores toward special x-coordinate, leading to move exponential function (Exp) in blue to special y-coordinate (dashed) and MsCls, b) setting trgt class to max, causing highest probability in normalizing process and trgtd MsCls, c) after comparing src score (red), swapping sec and trgt scores (green), causing src/trgt MsCls, d) moving Exp to special point on y-axis, causing equal probabilities and MsCls, e) substituting Exp with an ascending function with smaller growth order in red causing probability reduction, f) substituting Exp with a descending function with smaller growth order (red) causing both CnfRdct and MsCls.

concentrate on the trigger part in NN accelerators such that in [2] the payload is a multiplexer (MUX) and a module on the output path to determine the targeted (trgt) class at the presence of a trigger.

In another work [3], a HT is injected into the NN internal structure, particularly on a ReLu functional block. In [7], "neural Trojans" are embedded in the weights of the neural network during the training phase. Their triggers are sampled from an illegitimate distribution determined by the NN IP vendor, which is different from the legitimate distribution. In summary, the threat models in recent studies require that the adversary accesses the NN model, toolchain, and the internal structure of the computing operations, limiting their practical adoption into NN systems while only addressing some adversarial goals.

A. Our Contributions

In this work, we overcome the aforementioned limitations of existing works by devising HTs that cover varied adversarial goals, do not visibly alter the NN functionality by preserving prediction's distribution, and do not require NN toolchain manipulation. We focus our attacks on the last layer of a NN, which is common in different NN architectures as classifiers. It typically includes a fully connected layer followed by SoftMax as the last activation function [8]. From an identification viewpoint, the last layer of a NN is relatively easy to identify by backtracking the circuit structure from the outputs. Furthermore, the SoftMax function is typically implemented by some form of precomputed values stored in memory [8], which is distinguishable from random logic. It normalizes a network's score to a probability distribution over predicted output classes. Let us look at the scores and SoftMax function individually as two possible *locations* for HT insertion.

1) Score Manipulation and Intuition for HT Design: In SoftMax, the scores go directly toward the exponent of the function, such that small score changes cause considerable modification in the NN results. As illustrated in Fig. 1 (a-c), we manipulate the scores in different ways towards different malicious goals. In Fig. 1(a), all the scores are moved across the x-axis toward a special x-coordinate and consequently the probabilities to a special y-coordinate. The NN will display equal probabilities at the outputs, which ultimately realizes the **MsCls** goal in which it is hard to determine the final class. In Fig. 1(b), the score related to the trgt class is set to max. In this state, the trgt class in the normalizing process will get the highest probability and therefore promote **trgtd MsCls**. Finally, for the **src/trgt MsCls**, the score of the source (src) class is subtracted from all other scores. If the rests are greater than or equal to zero, then the src class is detected, and the src and trgt scores are swapped as shown in Fig. 1(c).

Insight: from a HW viewpoint, approaching zero across the x-axis can be implemented with right shifting. The other score manipulations require MUX (for swapping values) and a subtractor circuit, which are relatively small elements.

2) SoftMax Manipulation and Intuition for HT Design: SoftMax function consists of three pieces: the exponential function (Exp), the summation of the Exps, and the inverted summation. In HW, each piece is calculating independently, one after the other. Therefore, these pieces can be replaced by any malicious function as illustrated in Fig. 1(d-f).

First, in order to reach **MsCls**, we try to equalize the probabilities by approaching the Exp values to a special y-coordinate as shown in Fig. 1(d). This manipulation is the SoftMax counterpart of the score manipulation shown in Fig. 1(a). Here, the most straightforward (and extreme) manipulation would be to have a constant function to squeeze the Exp across the y-axis. However, this squeeze would cause the probabilities to behave like a seesaw around the pivot midpoint, i.e., meaning that we have confidence reduction on one side and confidence promotion on the left side w.r.t. to the baseline.

To reach **CnfRdct**, a linear function with a positive slope can be utilized, as shown in Fig. 1(e). The positive slope should keep the order of the probabilities and it should present smaller growth (than Exp) to reduce the probabilities in the entire considered interval. To reach **MsCls** and **CnfRdct** goals simultaneously, a negative slope linear function is employed, as seen in Fig. 1(f).

TABLE I
PROPOSED HTS TERMINOLOGY, DESCRIPTIONS, FUNCTIONS, GOALS AND TARGETS WHERE $\ggg$ IS A SIGNED RIGHT SHIFT OPERATOR.

HT name	Description	Function	Goal	Target
HTSft10Ms	Right shifting all scores	$ht(s) = sfmx(s \ggg 1)\ (all\ scores)$	MsCls	Scores
HTSft5Ms	Right shifting half of scores	$ht(s) = sfmx(s \ggg 1)\ (5\ scores)$	MsCls	Scores
HTSft1Ms	Right shifting one score	$ht(s) = sfmx(s \ggg 1)\ (1\ score)$	MsCls.	Scores
HTMxTgMs	Setting trgt score to max	$ht(s) = \begin{cases} sfmx(max) & \text{i = trgt class} \\ sfmx(s_i) & \text{otherwise} \end{cases}$	trgtd MsCls	Scores
HTSwSTMs	Swapping trgt and src scores	$ht(s) = \begin{cases} sfmx(s_{trg}) & \text{i = src cls, \& } s_{src} \geq y_{others} \\ sfmx(s_{src}) & \text{i = trgt cls, \& } s_{src} \geq y_{others} \\ sfmx(s_i) & \text{otherwise} \end{cases}$	src/trgt MsCls	Scores
HTCstMs	Replacing SoftMax by constant function	$ht(s) = const.$	MsCls	SoftMax
HTLnCR	Replacing SoftMax by linear function	$ht(s) = as + b$	CnfRdct	SoftMax
HTIvMsCR	Replacing SoftMax by linear function	$ht(s) = -as + b$	CnfRdct & MsCls	SoftMax

Insight: from a HW viewpoint, a complex function such as SoftMax is most likely to be implemented through precomputed values stored in memory. A malicious circuit that is capable of modifying the memory contents at run time can effectively replace SoftMax with any other functions.

III. PROPOSED HARDWARE TROJANS

A. Threat Model

In our threat model, the adversary needs no knowledge of the internal details of NN architecture nor the training data. Besides, the adversary can neither retrain the target model nor change the original model's input data and parameters. However, we assume that the adversary has access to the NN circuit at some description level which varies depending on where the adversary is located. In one scenario, the entire NN can be purchased as an IP from a third party (3PIP) and already comes with malicious logic [9]. In another scenario, at design time, a rogue element can alter the RTL code to insert the malicious logic herein described [10]. These two scenarios apply to both ASIC and FPGA design. Finally, a third scenario that only applies to ASICs is the design manipulation at fabrication time by a rogue element within an untrusted foundry [6]. The difficulty in mounting an attack varies considerably between scenarios, but all scenarios have been extensively studied before. Our contributions are providing insights into the payload mechanism **without requiring any NN-specific trigger**. For a discussion on the many *generic* trigger mechanisms (e.g., time bomb, hard-to-occur combinations of inputs, counters, etc.), we redirect the reader to [1], [2], [4], [11].

B. Trojan-infested Circuits

In this section, we detail the circuitry of our proposed HTs. In NNs, let s, P and $sfmx$ denote the score, the final probability vector, and SoftMax in the inference phase in the last layer as given in (1). Suppose a HT will change the model prediction to a pre-designed $ht(s)$ function on trigger activation. A simplified HT response model can be written as follows.

$$P = ht(s) \times t + sfmx(s) \times (1 - t), t \in [0, 1] \qquad (1)$$

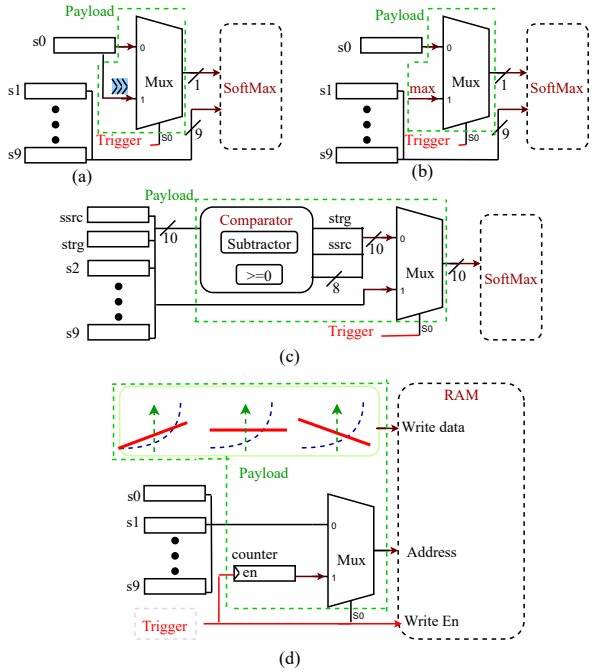

Fig. 2. HT circuits for score part in (a) (b) (c), HT for SoftMax part in (d)

where t is the trigger and behaves like a switch. Equation (1) shows that without a trigger, the model depends on $sfmx$ and the whole NN would give the correct result as expected so that users would not notice the system is compromised. Once a trigger occurs, ht dominates the model prediction. The ht implementations for the proposed HTs with their characteristics are summarized in Table I.

1) Score part HTs: Our 3 proposed score HTs are shown in Fig. 2 (a-c). For the MsCls, $ht(s)$ in Table I is approaching the scores to zero by right shifting (all 10 scores, 5 scores, or 1 single score). The payload of the HTs, termed HTSft10Ms, HTSft5Ms, HTSft1Ms, consists of a MUX utilized to select the shifted score when the trigger is activated as shown in Fig. 2(a). For the trgtd MsCls, $ht(s)$ is devised as a function that leads to a considerable rise in the trgt score, causing

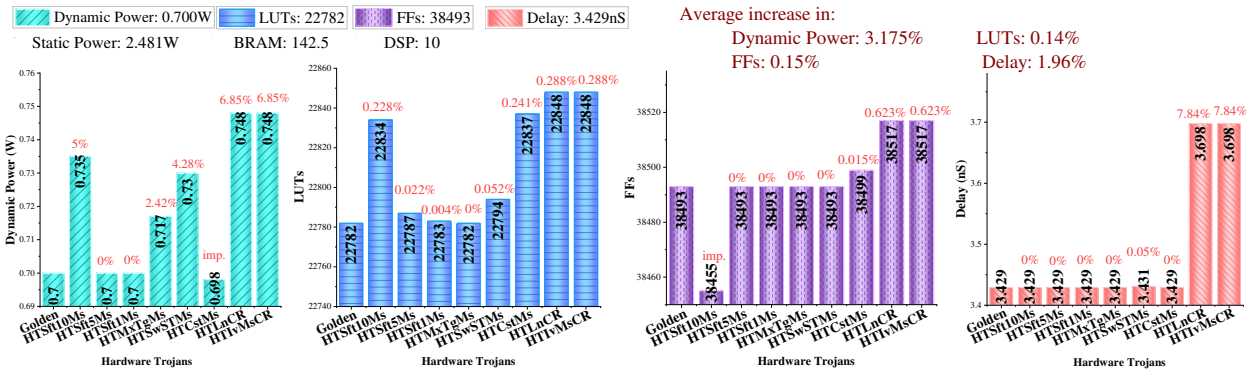

Fig. 3. Dynamic power, resource usage (LUTs, FFs, DSPs, and BRAM), and delay of proposed HTs with increase percentages (red) on the bars when SoftMax's table size is 64. The NN is a mulilayer perceptron with an input (output) layer of 784 (10) nodes and a hidden layer of 100 nodes. The zero-suppression is 75%, and reuse factor for the hidden layer is 7840, and for the output layer is 500.

probability promotion in the trgtd class. Since other scores remain fixed, while the inverted summation decreases, the other scores' probabilities will decrease. The payload of the HT, termed HTMxTgMs, consists of a MUX that selects the trgt score set as max on the trigger activation as shown in Fig. 2(b). For the src/trg MsCls, $ht(s)$ first tries to detect whether the src class is the highest. If so, then, the src and trgt values are swapped to give the trgt class the highest score. The payload of the HT, termed HTSwSTMs, consists of a subtractor and a comparator as shown in Fig. 2(c). The subtractor calculates the difference between the src class and others. The comparator is comparing the differences with zero. If all the differences are greater than or equal to zero, the swap will be done. A MUX is also utilized to select the swapped outputs on the trigger activation.

2) SoftMax part HTs: Our proposed SoftMax part HTs are shown in Fig 2 (d). For the MsCls, $ht(s)$ is now a constant function that equalizes the probabilities. For the CnfRdct, $ht(s)$ is a linear function with smaller growth order than Exp to reduce the probabilities. For both CnfRdct and MsCls, $ht(s)$ is an inverted linear function which is descending to inverse the order of the probabilities and smaller growth order to reduce the probabilities. The HTs, termed HTCstMs, HTLnCR, and HTIvMsCR, operate by modifying the memory content at runtime. The HTs' payload consists of a counter and a function generator that drives the address and write ports of the memory. The counter connects directly to the address port, and assumes one address is written per clock cycle. The counter size is the binary logarithm of the memory size. The counter requires a connection to the system's reset signal. The function generator produces the values of a given function as crafted data, which can be derived from the counter itself if the function is linear. A MUX is also utilized to select the counter when the trigger is activated. Besides the MUX select line, the trigger is also connected to the write enable pin to enable writing data. Notice that this HT requires that the trigger remains active for a number of clock cycles that correspond to the size of the memory being overwritten.

TABLE II
SUCCESS RATE IN ADVERSARIAL METRICS

HTs Name	Adversarial Goal			
	MsCls	*trgtd MsCls*	*src/trgt MsCls*	*CnfRdct*
HTSft10Ms	100%	-	-	95.81%
HTSft5Ms	98.01%	-	-	90.73%
HTSft1Ms	79.93%	-	-	87.87%
HTMxTgMs	-	97.57%	-	88.86%
HTSwSTMs	-	-	100%	21.05%
HTCstMs	100%	-	-	97.24%
HTLnCR	4.74%	-	-	100%
HTIvMsCR	100%	-	-	100%

IV. Experimental Setup and Results

Since the devised HTs are inherently architecture-independent and do not require any knowledge about the application's input data, we proceed by providing the NN study case generated using the HLS4ML package [8], and trained on the MNIST data set. It would be trivial to show results for other NNs or data sets. The model is synthesized for a Xilinx Virtex Ultra-Scale device (part number xcvu9p-flga2104-2-e). HW cost estimations are taken from Vivado 2020.1.1 synthesis with a fixed clock frequency of 100 MHz. We evaluate the success rate of the HTs to reach their goals on randomly chosen 1000 samples of the MNIST by simulation with Verilog testbenches. The results are tabulated in Table II. In the score part HTs, the HTSft10Ms for the MsCls achieves a 100% success rate. As seen, any upsetting the balance leads to a nearly satisfactory success rate in terms of the MsCls and CnfRdct goals. The HTMxTgMs for trgtd MsCls goal achieves a 97.5% success rate due to overflow issue, while this HT hits the success rate of 100% when the precision is increased. The other HTs enjoy a success rate of 100% in their adversarial goals.

Fig. 3 shows the overhead of the proposed practical HTs in comparison with the original circuit, when the SoftMax's table size is 64. The HTs overhead is shown on the top of the bars in Fig. 3 as red percentages. The average of the increases, as shown in the top right of the figure, is 3.175%,

TABLE III
DYNAMIC POWER, RESOURCE USAGE, AND DELAY OF PROPOSED HTS, WHEN SOFTMAX'S TABLE SIZE IS 128.

Circuits HW cost	Golden	HTCstMs	HTLnCR	HTIvMsCR
Power (W) **(increase)**	0.748 -	0.756 (+1.06%)	0.757 (+1.2%)	0.757 (+1.2%)
LUTs **(increase)**	22,669 -	22,676 (+0.030%)	22,681 (+0.052%)	22,681 (+0.052%)
FFs **(increase)**	38,338 -	38,345 (+0.018%)	38,363 (+0.065%)	38,363 (+0.065%)
Delay **(increase)**	3.698 -	3.709 (+0.297%)	3.698 (0%)	3.698 (0%)
BRAM **(increase)**	146.5 -	149.0 (+1.7%)	149.0 (+1.7%)	149.0 (+1.7%)

TABLE IV
SCENARIOS FOR POSSIBLE ADVERSARIES, TARGETED PLATFORMS, AND PROMINENT DETECTION SOLUTIONS

Adversary	ASIC?	FPGA?	Detection
3PIP provider	✓	✓	Verification
Rogue designer	✓	✓	Verification
Untrusted foundry	✓		Side channel, Test, Inspection

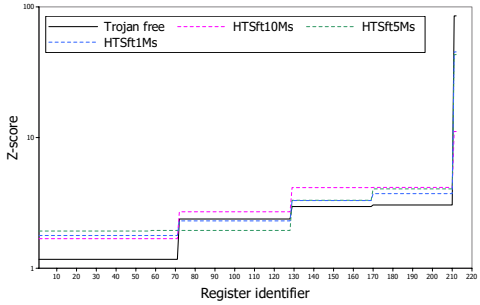

Fig. 4. Netlist analysis using RELIC for the Score part HTs.

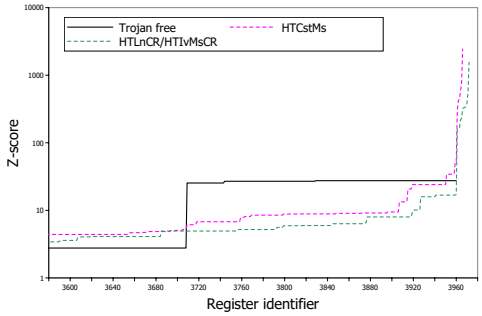

Fig. 5. Netlist analysis using RELIC for the SoftMax part HTs.

0.14%, 0.15%, and 1.96% in dynamic power, LUTs, FFs, and delay, respectively. It appears from these results that HTs with a CnfRdct goal suffer from slightly higher overheads. In another experiment, the SoftMax's table size is increased to 128. Then the overhead of SoftMax part HTs is investigated in Table III. As shown in the table, although the table size is increased, the proposed HTs remain feasible and the increase in dynamic power, LUTs, FFs, delay, and BRAM are small and less than 1.2%, 0.06%, 0.07%, 0.3%, and 1.7%, respectively. These small overheads are only possible since the proposed HTs do not modify the internal computing structure of the NN.

V. TROJAN DETECTION AND DISCUSSION

The research literature on HT detection is vast and multi-faceted. For this reason, we provide a short description of the attack scenarios previously described in our threat model (see Section III-A) and a non-exhaustive list of prominent defenses against them. This information is given in Table IV.

For defense against SoftMax part HTs, one alternative is the use of a read only memory that would prevent an adversary from changing its content. This solution would work well for ASICs. However, FPGA synthesis typically infers ROMs as initialized BRAMs for large memory sizes, so this solution is not effective in this context. For a malicious 3PIP that contains one of the HTs described in this work, detection can be carried out by verification techniques [9], [12]. The difficulty in detection comes from finding stimuli that will exercise the trigger that is carefully designed to be seldomly activated. To avoid the extensive burden of carrying out a long simulation campaign (and collecting its coverage metrics), formal verification techniques can be utilized by proving (or disproving) whether an operation with the same inputs always results in the same outputs. For a malicious modification made by a rogue designer, verification techniques can still be considered. In fact, since the code is in the clear (as opposed to a hard IP or encrypted IP coming from a 3PIP provider), detection should be relatively simpler than in the previous scenario. It is also the case that for in-house developed code, the verification effort tends to be higher than for 3PIPs (i.e., the verification effort for 3PIPs tends to focus on connectivity and configurations rather than validating design features).

In the first two scenarios, there is no conceptual difference in whether the HT targets an FPGA or ASIC platform. In the third scenario, where the adversary is an untrusted foundry, the targeted platform is only ASIC. Several detection approaches have been proposed for ASICs, including side-channel analysis [13], test vectors [14], and physical inspection [15]. However, HTs can still escape detection. In [16], HTs that incur ultra-low side-channel signatures are devised. In [17], it is shown how test vectors fare when utilized to search for HTs instead of defects. The authors conclude that even small benchmark circuits present low HT trigger coverage. For physical inspection, the issue is not HTs that escape detection – it is the scalability of the approach that is not sufficient as a layer-by-layers analysis of a modern complex chip can take weeks at a time [18].

Given all the drawbacks and limitations of HT detection approaches, we resort to **reverse engineering techniques** to quantify the detectability of our HTs. This is a considerable contribution by itself since it departs from current approaches. Importantly, our analysis is carried out only on the payload of the HTs – it already is established that the trigger mechanism

is rarely activated and generally hard to detect.

Our tool of choice for HT payload detection is RELIC2 [19], [20]. The intended use of the tool is to analyze a netlist and classify registers as either datapath or control logic. For each analyzed register, RELIC returns a numerical score termed Z-score. The higher the score, the more likely the register is to be part of the control logic. For this purpose, a series of features are considered, which include a fanin and fanout properties, reg-to-output and input-to-reg distances (measured in clock cycles), and reconvergence.

Our proposed Score part HTs are evaluated in Fig. 4. The black line represents the typical distribution of data vs. control logic for registers in a specific portion of the neural network, more precisely the location where the Score part HTs are inserted[1]. The pink line represents the distribution for the HT netlist where all 10 scores are shifted right by 1, while the green line is the netlist where 5 scores are shifted. Finally, the blue line relates to the netlist where only one score is shifted. As a general trend, all four lines appear to be in agreement, meaning that HTs that shift the scores have little impact on the structure of the netlist. The register with the highest Z-score is identical in all four netlists (the score itself for this register is 85.33, 45.23, 43.13, and 11.12 in the Trojan-free, HTSft1Ms, HTSft5Ms, and HTSft10Ms netlists, respectively).

A similar analysis is depicted in Fig. 5 for HTs that manipulate the SoftMax function. The same netlist partitioning was assumed, but here the affected circuitry contains approximately 4000 registers. For the sake of clarity, only the 400 registers with the highest Z-scores are plotted. Notice how the change in the distribution is much more pronounced in this analysis as the pink and green lines do not display the same clear step seen in the original circuit (also note that the vertical scale is logarithmic and the HTLnCR and HTIvMsCR have identical trends and are therefore represented with the same line). The reason for the discrepancy with respect to the Trojan-free circuit is due to the addition of registers that calculate the address and data to be written to the memory where the SoftMax table is stored. These registers always score higher than any other register in the original design. Without a doubt, this insight can be utilized for HT detection purposes.

VI. Conclusion and Future work

In this work, we proposed specialized and practical HTs inserted in the last layer of a NN, which is relatively easy to identify by backtracking the circuit structure from the outputs. The proposed HTs enjoy small overheads and do not always modify the structure of the circuit enough such that detection based on netlist analysis becomes possible. The proposed HTs work for representative NN architectures and cover varied adversarial goals, often with 100% precision. As future work, we will implant the HTs in different NN architectures in order to investigate overheads and success rates for adversarial goals on both ASIC and FPGA platforms.

[1]In practice, this assumption implies that netlist partitioning was executed to isolate the other parts of the circuit that are not directly affected by the proposed HTs.

Acknowledgement

This work was partially supported by the EU through the European Social Fund in the context of the project "ICT programme". It was also partially supported by the Estonian Research Council grant "MOBERC35".

References

[1] X. Hu, Y. Zhao, L. Deng, L. Liang, P. Zuo, J. Ye, Y. Lin, and Y. Xie, "Practical attacks on deep neural networks by memory trojaning," *IEEE TCAD*, vol. 40, no. 6, pp. 1230–1243, 2021.

[2] Z. Liu, J. Ye, X. Hu, H. Li, X. Li, and Y. Hu, "Sequence triggered hardware trojan in neural network accelerator," in *IEEE VLSI Test Symposium (VTS)*, 2020, pp. 1–6.

[3] J. Clements and Y. Lao, "Hardware trojan design on neural networks," in *IEEE ISCAS*, 2019, pp. 1–5.

[4] M. Xue, C. Gu, W. Liu, S. Yu, and M. O'Neill, "Ten years of hardware trojans: a survey from the attacker's perspective," *IET Computers & Digital Techniques*, vol. 14, no. 6, pp. 231–246, 2020.

[5] R. S. Chakraborty *et al.*, "A flexible online checking technique to enhance hardware trojan horse detectability by reliability analysis," *IEEE TETC*, vol. 5, no. 2, pp. 260–270, 2017.

[6] T. Perez and S. Pagliarini, "Side-channel trojan insertion – a practical foundry-side attack via eco," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021.

[7] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *IEEE ICCD*, 2017, pp. 45–48.

[8] J. Duarte *et al.*, "Fast inference of deep neural networks in fpgas for particle physics," *Journal of Instrumentation*, vol. 13, no. 07, p. P07027–P07027, Jul 2018.

[9] X. Zhang and M. Tehranipoor, "Case study: Detecting hardware trojans in third-party digital ip cores," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, 2011, pp. 67–70.

[10] J. Rajendran, A. K. Kanuparthi, M. Zahran, S. K. Addepalli, G. Ormazabal, and R. Karri, "Securing processors against insider attacks: A circuit-microarchitecture co-design approach," *IEEE Design & Test*, vol. 30, no. 2, pp. 35–44, 2013.

[11] S. Bhasin and F. Regazzoni, "A survey on hardware trojan detection techniques," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 2021–2024.

[12] X. Guo, R. G. Dutta, Y. Jin, F. Farahmandi, and P. Mishra, "Pre-silicon security verification and validation: A formal perspective," in *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.

[13] S. Narasimhan *et al.*, "Multiple-parameter side-channel analysis: A non-invasive hardware trojan detection approach," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 13–18.

[14] A. P. Fournaris, L. Pyrgas, and P. Kitsos, "An efficient multi-parameter approach for fpga hardware trojan detection," *Microprocessors and Microsystems*, vol. 71, p. 102863, 2019.

[15] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *Design Automation Conference (DAC)*, 2011, pp. 333–338.

[16] X. Wang, S. Narasimhan, A. Krishna, T. Mal-Sarkar, and S. Bhunia, "Sequential hardware trojan: Side-channel aware design and placement," in *IEEE ICCD*, 2011, pp. 297–300.

[17] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, "Towards trojan-free trusted ics: Problem analysis and detection scheme," in *Design, Automation and Test in Europe*, 2008, pp. 1362–1365.

[18] Intelligence Advanced Research Projects Activity (IARPA). (2016) Rapid Analysis of Various Emerging Nanoelectronics (RAVEN). [Online]. Available: https://www.iarpa.gov/index.php/research-programs/raven

[19] T. Meade, Y. Jin, M. Tehranipoor, and S. Zhang, "Gate-level netlist reverse engineering for hardware security: Control logic register identification," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 1334–1337.

[20] T. Meade, J. Portillo, S. Zhang, and Y. Jin, "NETA: When IP Fails, Secrets Leak," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 90–95.