International Journal of Engineering & Technology, 7 (x) (2018) xxx-xxx

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: xxx
Research paper, Short communication, Review, Technical paper

FPGA Realization of Deep Neural Network for Hardware Trojan Detection

Varun Reddy B[1] and Nirmala Devi M[1]

[1]*Department of Electronics and Communication Engineering, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India*
[*]*Corresponding author E-mail:varunreddy0808@gmail.com*

Abstract

With the increase in outsourcing design and fabrication, malicious third-party vendors often insert hardware Trojan (HT) in the integrated circuits(IC). It is difficult to identify these Trojans since the nature and characteristics of each Trojan differ significantly. Any method developed for HT detection is limited by its capacity on dealing with varied types of Trojans. The main purpose of this study is to show using deep learning (DL), this problem can be dealt with some extent and the effect of deep neural network (DNN) when it is realized on field programmable gate array (FPGA). In this paper, we propose a comparison of accuracy in finding faults on ISCAS'85 benchmark circuits between random forest classifier and DNN. Further for the faster processing time and less power consumption, the network is implemented on FPGA. The results show the performance of deep neural network gets better when a large number of nets are used and faster in the execution of the algorithm. Also, the speedup of the neuron is 100x times better when implemented on FPGA with 15.32% of resource utilization and provides less power consumption than GPU.

Keywords: *Deep Neural Network; Deep Learning; FPGA; Hardware Trojan; Random forest classifier*

1. Introduction

With third party vendors inserting Trojans of different kinds which may cause a behavioral change in ICs or some secret information gets leaked. It becomes a serious concern in detecting them fastly and accurately. It is challenging to propose a generalized method of HT detection, considering new types of Trojans emerging. Hence, the procedure which is automatically tuning its selection of parameters or feature and leading to a specific detection algorithm is called for. The proposed work envisages the situation and develops a Trojan independent detection method, by using a machine learning (ML) paradigm. Deep learning, a subclass of ML that has the capability of choosing the parameters on its own has been chosen for this work as it provides better accuracy when dealing with complex circuits. But the timing latency and power consumption are the major setbacks when this architecture is implemented in the general processing unit (GPU). To overcome this issue, this deep neural network (DNN) should be implemented on hardware. Re- configurable and reprogrammable computing has proved to be a better option. Among the various kinds of research targeting the speed up of complex algorithms suitable choice is field programmable gate array(FPGA) which provides fast processing compared to GPU [1] and cost-effective. In [2], emerging attack modes and hardware Trojan attacks that are violating the trust of the consumer is analyzed in detail. These Trojans inserted in different stages of hardware, different types of Trojan attacks on hardware, and the scenarios of the attacks are mentioned.[3]Presents an informatic-theoretic approach for Trojan detection. It evaluated the mathematical relationship among the signals in design and investigates how this evaluation can be done in a clustering algorithm to detect the Trojan logic. Results determine that this machine can identify the Trojan logic of Trust hub benchmark circuits with up to 100 percent coverage with low false-positive rates. [4] Showed architectures of deep learning performed better than the classic ML algorithms for applications like intrusion detection, android malware detection. The capability of DL on learning optimal feature by itself is mentioned and robustness in the adversarial environment is compared with ML algorithms.[5] Presents an overview of different types of VLSI implementations of artificial neural networks (ANN) and presented NN can provide faster speed up in training when it is implemented in hardware. In [6] an FPGA implementation of DNN is presented using VHDL and floating-point which allowed the use of physical computation of a single layer to complete the whole feed-forward step network. A DNN realization on FPGA using Stacked Sparse Autoencoders (SSAE) was one of the first to get implemented on hardware [7]. In [8] sparse autoencoder architecture with network architecture of 196 input and output neurons along with 100 hidden neurons is implemented using Verilog HDL. DNN hardware realization using a technique called SSAE was implemented in [9] which used a concept of a systolic array that allows the use of many

neurons and various layers. With the help of related works, we propose an HT detection which provides better accuracy at faster processing time. Considering the discussion, the major contributions are as follows:

- Trojan detection using random forest classifier.
- HT detection algorithm using DNN.
- Architecture of 32-10-2-5 and implementation of DNN for HT detection on FPGA.
- Validation of classification accuracy for both algorithms. Experiment results show the accuracy is slightly higher for random forest classifier. But it is restricted only on the given set of parameters.
- With 15.32% of hardware utilization, proposed network yields 100x times speedup with less power consumption when implemented on FPGA.

2. Research method

The working of a random forest classifier and the workflow for HT detection using DNN is presented here. The steps involved in classifying the HT detection by random forest classifiers are depicted in Fig.1 as pseudo-code. Fig.2 shows the training flow of the DL algorithm using sigmoid and softmax activation functions with an architecture 32-10-2-5 to classify five types of Trojans (no Trojan, type1, type2, type3, type4). For training ISCAS'85 combinational benchmark circuits have been used. Classification accuracy, weight, and weight bias sets obtained are used in implementing the feed-forward phase of HT detection. Using the weight set obtained after classification from the learned model of the DL algorithm, computation of neuron value and layer output to get the classification is executed.

ALGORITHM1: Random forest classifier

INPUT = Parameters of nets.

OUTPUT = Classified outputs.

1. Extract the trojan features for every netlist.
2. Choose the best features out of all available features.
3. Construct a decision tree for samples.
4. Get the prediction result of every tree.
5. Select the most voted prediction result as final result.
6. Using test set, compute the classification accuracy.

Figure 1: Algorithm for random forest classifier.

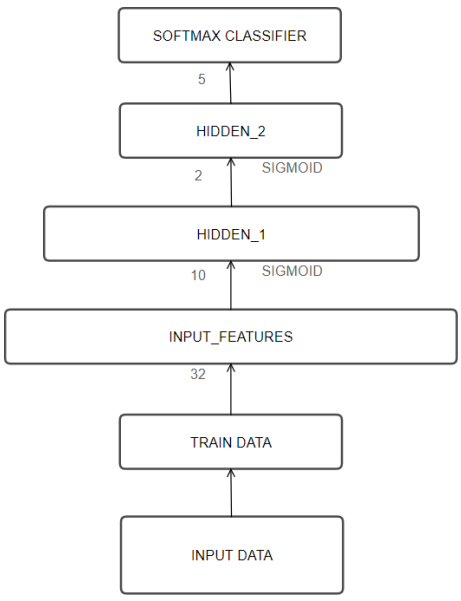

Figure 2: Training flow of DNN.

The proposed work is further focused on implementing HT detection on FPGA for faster processing time and less power consumption. The implementation is based on the deep learning algorithm and activation function equations [10]. The output (Z (n)) of the neuron can be

expressed by using weight set (w) between i-th neuron to j-th neuron of next layer, input (y) of j-th neuron of l-th layer and the weight bias (wb) of i-th layer and bias (b) as

$$Z_i^k(n) = \sum_{j=1}^{u^1} w_{ij}^k(n) * y_j^l(n) + wb_i^k(n) * b \tag{1}$$

In the above equation, the weight bias is taken as zero for the first hidden layer and for the rest of the layers, the weight bias and bias values are taken from the weight matrix obtained from the algorithm. As mentioned the sigmoid function is used as an activation function between the hidden layers. The output of that is expressed using the following equation

$$V_i^k(n) = \frac{1}{1 + e^{-z_i^k(n)}} \tag{2}$$

The V (n) will be passed as input at nth instant of the l-th layer j-th neuron

$$y_j^{l+1}(n) = V_i^k(n) \tag{3}$$

As part of the final classification of Trojans, a multi-classification function like softmax is used and it is characterized using eq (4). The output s (n) consists of the neuron value of the last layer and the number of outputs is determined by H value, here for this problem we are focusing on five classifications. The output of the function is the probability of that class

$$s_i(n) = \frac{e^{z_i^k(n)}}{\sum_{h=1}^{H} e^{z_h^k(n)}} \tag{4}$$

A network architecture which as A as the input layer, R, C as hidden layers, and H as output layer as shown in Fig.3 is implemented on FPGA. The obtained weights are fed to neurons by converting them using the IEEE754 standard.

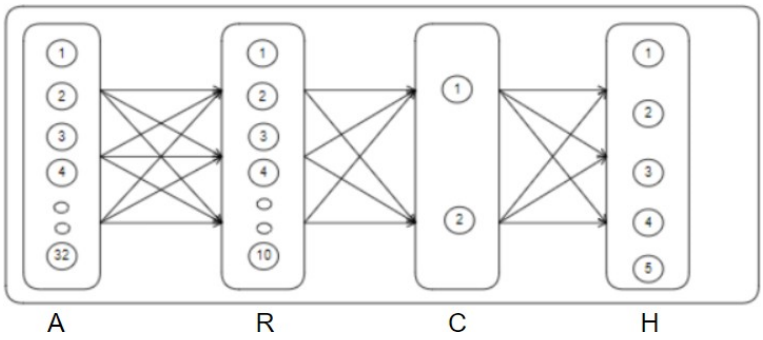

Figure 3: Network architecture.

Neuron architecture is given in Fig. 4, where the weight biases (wb) and bias (b) section is used for hidden layers other than the first. The output (z) is passed through activation function and passed on to the next layer. Fig.5 shows the pseudo steps for feed forward phase of network implementation. Individual neuron outputs are calculated with initialized weight vectors. As we are classifying the Trojans into five types, the H value is assigned five and the probabilities of circuits are calculated.

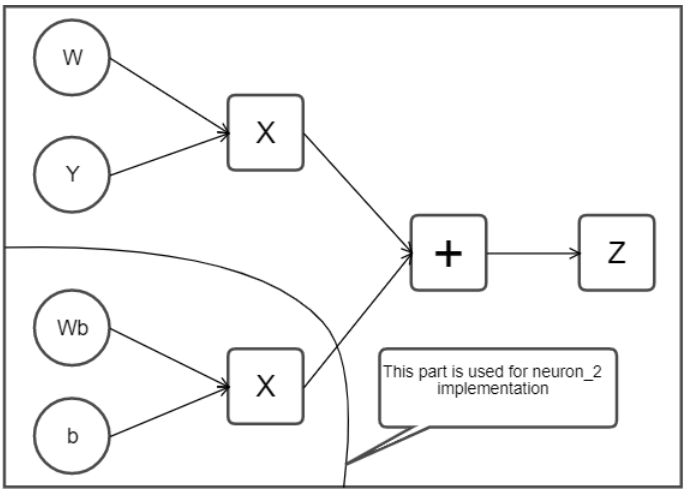

Figure 4: Neuron architecture.

```
ALGORITHM2: Feed forward phase.
INPUT = weight vectors and input vectors.
OUTPUT = HT classification (no trojan and 4 types of trojans).
\\ Neuron output classification
  1. Representing architecture as A-R-C-H, with input layer( A data inputs),
     two hidden layers (R and C inputs) and an output layer( H outputs).
  2. Choose an initial weight vector.
  3. Evaluate each individual neuron output using eq(1).
  4. Compute output of i-th neuron from kth layer at nth instant.
\\ Activation function
  5. i-th neuron from k-th layer at n-th instant is expressed using eq(2).
  6. Evaluated result is processed as input to next layer at n-th instant using eq(3).
\\ HT classification
  7. Choose an H value for the problem.
  8. Compute the probabilities using eq(4).
```

Figure 5: Pseudo code for feed forward phase.

3. Result and discussion

The accuracy of classification of the Trojans using two algorithms, viz., random forest classifier, and DL as analyzed. Further, the hardware utilization and timing of the DNN on FPGA are also presented.

3.1. Classification accuracy

Table 1 gives a summary of the accuracy of HT detection using random forest and DL algorithm classifiers also the time taken for completing the algorithm. Results show the accuracy is better for the random forest but the complexity of it is high and the time taken for an algorithm to classify is more when compared to the DL algorithm. But DL gives us better accuracy when working problems with more number of nets.

Table 1: Accuracy table.

Circuit	Random forest classifier	Time taken	DL algorithm	Time taken
c432_t0	99.12	146.023	96.80	133.587
c432_t1	99.16	147.132	96.79	134.751
c499_t0	99.01	148.016	96.91	134.677
c499_t1	98.86	148.963	96.89	136.742
c2670_t0	98.96	150.840	97.42	138.026
c2670_t1	98.75	150.753	97.53	139.458
c6288_t0	98.80	154.236	97.51	142.146
c6288_t1	98.74	157.514	97.65	143.951

3.2. FPGA realization of DNN

The hardware utilization associated with the Processing elements (PE) on FPGA is presented. For all synthesis results, the targeted FPGA used was Artix 7 BASYS 3 (xc7a35tcpg236-1) board.

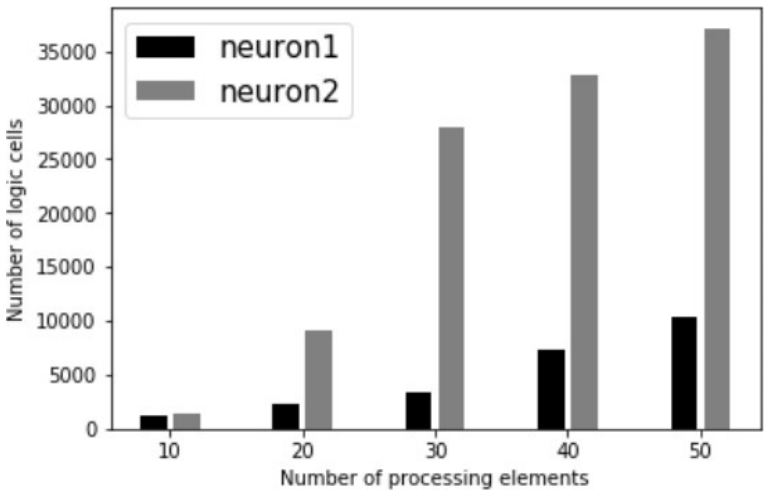

Figure 6: Hardware utilization of neurons.

Fig. 6 shows the hardware utilization i.e., the number of logic cells used in implementing neuron1 where weight biases is zero. The occupation of neuron 2 for which the weight biases and bias values are taken from the weight set obtained is also shown.
Fig.7 gives the implementation of neuron 2, which is synthesized using Vivado tool. Fig.8 shows the simulation of the layer and the sum output highlighted will be passed on to the sigmoid activation function. A 32-bit weight set and input set is used for calculation.

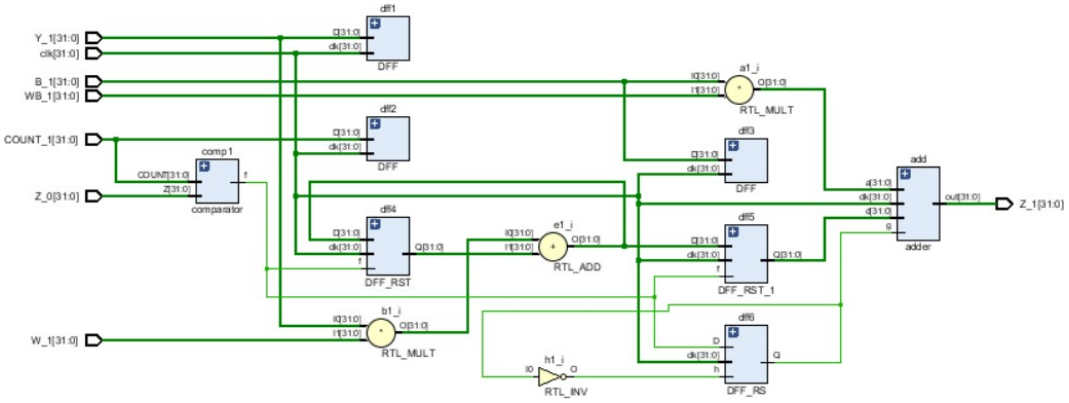

Figure 7: Neuron implementation.

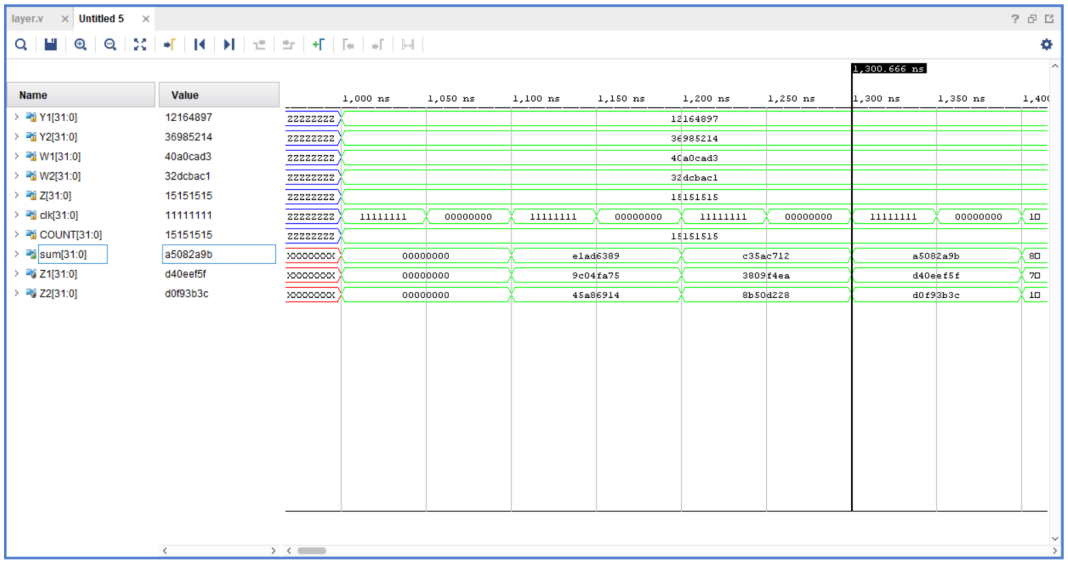

Figure 8: Simulation result of layer.

A LUT based sigmoid function is used for activation [11]. Softmax function is used as output layer as it gives best results in multi class problems. The function is built by breaking down the softmax equation given in eq.4, into four different modules adder, multiplier, division and exponential.

3.3. Timing and power analysis

From the power calculations, we could see the change from neuron1 to neuron2. Neuron2 is having more power compared to neuron1 as it has additional inputs like weight bias and bias. From the timing report we could easily infer that hardware synthesizer (FPGA) is almost 100 times faster than the software framework. We could see, layer with 2 neurons and 10 neurons have almost equal processing times. This is because the neurons compute parallel which helps in achieving faster timing.

Table 2: Timing and Power report.

Structure	Synthesizer Power (uW)	Time (us)	Hardware synthesizer Time (ns)
Neuron1	23.3253	0.64	6.3
Neuron2	37.4893	0.63	6.3
Layer1(with two neurons)	663.42	0.74	7.371
Layer2(with ten neurons)	5757.31	0.76	7.41

Fig.9 shows the hardware setup where the basys 3 board is used for evaluation purpose. Vivado synthesis tool software by Xilinx is used for generating bit stream. The overall hardware utilized is around 15.32% implementing 32-10-2-5 architecture, which gives us scope to increase the number of neurons in layer and add more hidden layers for better accuracy.

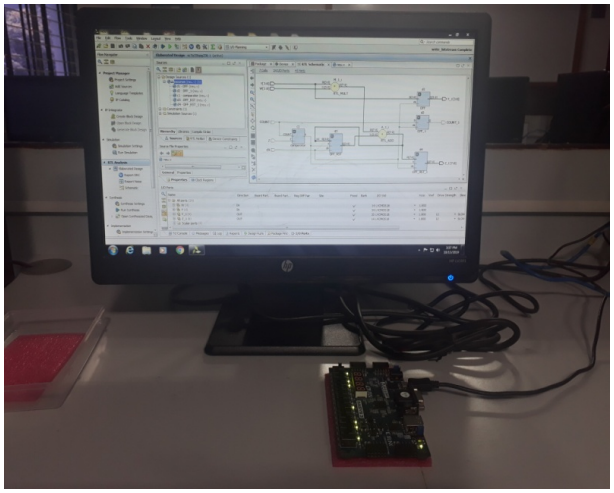

Figure 9: Hardware setup.

4. Conclusion

With high power consumption and cost, GPUs are not efficient for most of the DNN applications. In such cases, the FPGA realization of DNN is a promising platform as they provide reprogrammability and reconfigurability. FPGAs may also be used for embedded applications when the simplicity of neural computation is needed. Also, it is difficult to propose a generalized method for hardware Trojan detection. Using DL this problem can be solved to great extent. As DL selects the best features by itself. The performance of DL in classifying HTD is around 97% and in hardware, the processing time is 100x times faster and has power consumption in uW. Also, only 15.32% of hardware is utilized, which shows us the resources are available for more number of neurons. As a part of future work, this can be implemented in different architecture by varying data representation and also handling multiple Trojans and distributed Trojans.

References

[1] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie, and X. Zhou, "Dlau: A scalable deep learning accelerator unit on fpga," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 3, pp. 513–517, 2016.
[2] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: Threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.
[3] B. Cakır and S. Malik, "Hardware trojan detection for gate-level ics using signal correlation based clustering," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 471–476.
[4] R. Vinayakumar, K. Soman, P. Poornachandran, and S. Akarsh, "Application of deep learning architectures for cyber security," in *Cybersecurity and Secure Information Systems*. Springer, 2019, pp. 125–160.
[5] M. Nirmaladevi and S. Arumugam, "Vlsi implementation of artificial neural networks—asurvey," *International Journal of Modelling and Simulation*, vol. 30, no. 2, pp. 148–154, 2010.
[6] T. V. Huynh, "Deep neural network accelerator based on fpga," in *2017 4th NAFOSTED Conference on Information and Computer Science*. IEEE, 2017, pp. 254–257.
[7] J. Maria, J. Amaro, G. Falcao, and L. A. Alexandre, "Stacked autoencoders using low-power accelerated architectures for object recognition in autonomous systems," *Neural Processing Letters*, vol. 43, no. 2, pp. 445–458, 2016.
[8] Y. Jin and D. Kim, "Unsupervised feature learning by pre-route simulation of auto-encoder behavior model," *International Journal of Computer and Information Engineering*, vol. 8, no. 5, pp. 706–710, 2014.
[9] M. G. Coutinho, M. F. Torquato, and M. A. Fernandes, "Deep neural network hardware implementation based on stacked sparse autoencoder," *IEEE Access*, vol. 7, pp. 40 674–40 694, 2019.
[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
[11] M. Panicker and C. Babu, "Efficient fpga implementation of sigmoid and bipolar sigmoid activation functions for multilayer perceptrons," *IOSR Journal of Engineering*, vol. 2, no. 6, pp. 1352–1356, 2012.