Applied Computer Science, vol. 17, no. 4, pp. 20–33
doi: 10.23743/acs-2021-26

Submitted: 2021-10-21 | Revised: 2021-11-29 | Accepted: 2021-12-10

Keywords: hardware trojan, chips, logic test, machine learning, microcontroller

Kadeejah ABDULSALAM [0000-0001-7856-4269]*, *John ADEBISI* [0000-0002-1105-7219]*, *Victor DUROJAIYE**

IMPLEMENTATION OF A HARDWARE TROJAN CHIP DETECTOR MODEL USING ARDUINO MICROCONTROLLER

Abstract

These days, hardware devices and its associated activities are greatly impacted by threats amidst of various technologies. Hardware trojans are malicious modifications made to the circuitry of an integrated circuit, Exploiting such alterations and accessing the level of damage to devices is considered in this work. These trojans, when present in sensitive hardware system deployment, tends to have potential damage and infection to the system. This research builds a hardware trojan detector using machine learning techniques. The work uses a combination of logic testing and power side-channel analysis (SCA) coupled with machine learning for power traces. The model was trained, validated and tested using the acquired data, for 5 epochs. Preliminary logic tests were conducted on target hardware device as well as power SCA. The designed machine learning model was implemented using Arduino microcontroller and result showed that the hardware trojan detector identifies trojan chips with a reliable accuracy. The power consumption readings of the hardware characteristically start at 1035-1040mW and the power time-series data were simulated using DC power measurements mixed with additive white Gaussian noise (AWGN) with different standard deviations. The model achieves accuracy, precision and accurate recall values. Setting the threshold probability for the trojan class less than 0.5 however increases the recall, which is the most important metric for overall accuracy acheivement of over 95 percent after several epochs of training.

1. BACKGROUND

Hardware Trojan (HT) is a malicious modification of the circuitry of an integrated circuit (IC) (Salmani, Tehranipoo & Plusquellic, 2011). Most hardware trojans are specially designed to change the functionality, reduce the reliability and/or leak valuable information from the host circuit. An HT's payload is the whole activity the trojan performs when triggered. Malicious trojans generally attempt to bypass or disable a system's safety arcade; and private data can be leaked by radio emission. HTs could also disable, derange or ruin the whole chip

* University of Lagos, Electrical Electronics and Computer Engineering Department, Nigeria, kabdulsalam@unilag.edu.ng, adebisi_tunji@yahoo.com, durojaiyevic@gmail.com

or its parts (Salmani, Tehranipoo & Plusquellic, 2011). Albeit trojans are designed for various purposes, they can be functional or parametric, small or large, tight or loose, logic or sensor-based, trigger based or always on among others. Different techniques can be used for trojan detection, each with its strengths and weaknesses. Some of the weaknesses could be attached to the test nature (Grus, 2015); such as – destructive tests, which involve removing the chip from the package and scanning the layers to get the net lists, possibly using an electron microscope (Grus, 2015). The floor plans gotten from the process can then be compared with the floor plans of the actual IC.

This involves reverse-engineering (He et al., 2014). However, this process renders the chip useless, and the presence (or absence) of trojans in a chip does not guarantee the presence (or absence) in another chip (Grus, 2015). In this case, non-destructive tests are alternatives which do not involve the reverse-engineering of the chip. Other tests include; Logic test where the input ports of the chip are stimulated with test vectors, and the output is compared to what is expected. A deviation from the expected output could indicate the existence of a trojan (Wang & Luo, 2011). Power SCA; observes power traces from the device under test (DUT) and compares the results to what is obtained from a "golden chip" (a trusted chip without any trojans). The existence of a large trojan would imply an obvious difference in power consumption. However, for smaller trojans, more sophisticated approaches involving machine learning (ML) would need to be used to identify the subtle differences between chips which contain trojans and those which do not (Ni et al., 2014). Delay SCA: uses a time delay in the path of DUT, which is more than the time delay observed in a golden chip. It could indicate the presence of a trojan in a path while runtime test monitors a running system, actively checking for indications that a trojan may exist, but contains an interrupt mechanism to stop the system once a trojan is detected, protecting the system (Cui et al., 2016). A summary of the different types of hardware trojans and the detection methods is presented in Table 1.

Tab. 1. Summary of hardware trojan detection methods

Trojans	Logic test	Power SCA	Delay SCA	Run time
Parametric	×	✓	✓	×
Big	?	✓	?	✓
Small	✓	×	✓	?
Tight	✓	✓	✓	?
Loose	✓	?	✓	?
Always-on	×	×	✓	×
Leak info	×	✓	×	✓

where: ✓ – good, × – bad, **?** – depends.

2. LITERATURE REVIEW

In literature, hardware trojan is a deliberate and unwanted alteration of an integrated circuit (Jahan, Sajal & Nygard, 2019; Salmani, Tehranipoo & Plusquellic, 2011). They present emerging security concerns and can have devastating effects when used in sensitive

environments. Hence, research of this nature is needed to detect them. Physical inspection could be destructive and sometimes involves reverse-engineering which makes the chip unusable, the results of this test do not bring a lot of confidence; the presence (or absence) of a trojan is not guaranteed and such test is not standard enough to be used in a dynamic environment (Paul, Suman & Sultan, 2013).

Research has been done on the detection of hardware trojans using machine learning involving Principal Component Analysis (PCA), Latent Dirichlet Analysis (LDA) or Support Vector Machine (SVM). In (He et al., 2014), a combination of logic testing and side-channel testing was used as a novel hardware trojan detection method. An 18-bit Intellectual Property (IP) core was used as a golden circuit, while a 2-bit counter was used as a trojan circuit. Testing was done using the Xilinx ISE (Integrated Synthesis Environment), LabVIEW software and a high-precision oscilloscope. The power traces from the devices were monitored and PCA was used to process them, extracting three projections on three corresponding largest eigenvectors. In the resulting 3D graph, there was a clear difference between the devices as the power traces for each class clustered together. The system was shown to have a trojan detection sensitivity of 0.1%.

The use of machine learning for power SCA was explored in (Shende & Ambawade, 2016). Here, the experimental setup involves a golden chip; an AES (Advanced Encryption Standard) core, and another with a trojan inserted. The hardware is synthesized with using Xilinx ISE and implemented on a Xilinx Spartan-6 Field Programmable Gate Array (FPGA). Power measurements were obtained using power analysis exploratory data analysis (EDA) tools, dimensionality reduction is done using PCA and then classified using LDA. The setup is able to differentiate between the golden and trojan devices to a very high degree of accuracy, theoretically 100%.

A research on "detection technique for hardware trojans using machine learning in frequency domain", which is an application of SCA on the power consumption waveform data limited to the frequency domain using discrete Fourier transform (DFT) of the waveforms is presented in (Iwase, Nozaki, Yoshikawa & Kumaki, 2015). An AES core and several other variants containing trojans were built on an FPGA using Verilog. The Fourier transform of the power traces is then used to train an SVM machine learning model with some level of accuracy recorded. The work of (Bai, 2018; Cui et al., 2016) focused more on Cluster Analysis of Mahalanobis Distance in their detection approach; a concept associated with LDA. An AES encryption circuit was designed on an FPGA, and on another design, a trojan is added. Power SCA was then carried out. Results from the experiments show that in performing cluster analysis on the data points, the Mahalanobis distance gives far more accurate results than the Euclidean distance.

The Influence on Sensitivity of Hardware Trojans Detection by Test Vector" was investigated in (Ni et al., 2014), although the research was a proposal of power relative variation (PRV) parameters to analyze the relation between test vectors and detection sensitivity. S-box circuit noise model was introduced to an AES core and the power traces of various sizes of trojans were simulated using HSPICE. Different sets of test vectors are applied as well. It was found that the power of the entire circuit is reduced by 41% and the PRV value increased 12.71% and 3.34% at most, corresponding to combinational and sequential trojan circuits. A power characteristic template for hardware trojan detection which is a novel hardware detection method using PCA and applying the Mahalanobis distance was developed in (Zhang et al., 2016). The output could adapt to a variety of different

functions on the same chip. The RS232 serial port hardware trojan implanted in DES and AES algorithms was used for verification. Hardware trojan detection rate was high, with low computational cost due to dimensionality reduction. In the work of (Wang & Luo, 2011), a power analysis based experimental approach was used coupled with two FPGAs; one used for implementing the 64-bit DES cipher (called "Test FPGA") and the other used to generate test vectors (called "FPGA pattern generator"). Two trojans were designed for the DES cipher and the power traces were analyzed through singular value decomposition (SVD). This detection method works, even for trojans about two orders of magnitude smaller than the main circuit.

A General Framework for Hardware Trojan Detection in Digital Circuits by Statistical Learning Algorithms" – presented in (Chen et al., 2016), uses a Bayesian inference-based calibration technique to check for the existence of a trojan and map to the sparse solution of the linear system A batch of underdetermined linear systems are solved together by the well-studied simultaneous orthogonal matching pursuit algorithm to get their common sparse solution. This framework gives high trojan detection rates with low measurement cost. It has been observed power SCA is the leading method of identifying hardware trojans in literature. This research leverage on the work, along with the concept of logic testing. Machine learning was used to create a smart and accurate trojan detection system. It was observed that the use of deep learning techniques would provide better results when working with time-series data (power traces of chips), compared to more traditional machine learning approaches such as PCA, LDA and SVM. Hence, deep learning would be used for power SCA, more specifically a mix of convolutional neural networks (CNN) and recurrent neural networks (RNN).

3. DESIGN METHODOLOGY

This section contains the procedures and methods used in building the proposed trojan detector. It further, highlights the software and hardware tools used for model implementation. Theoretical frameworks for the different components and subsystems involved in the design are also considered in this section.

3.1. Field programmable gate array (FPGA)

This is a programmable hardware used to implement any logic circuit; combinational or sequential. It serves as an alternative to designing an application-specific integrated circuit (ASIC). FPGAs contain programmable logic blocks, programmable interconnects and input/output (I/O) blocks as shown in Figure 1.

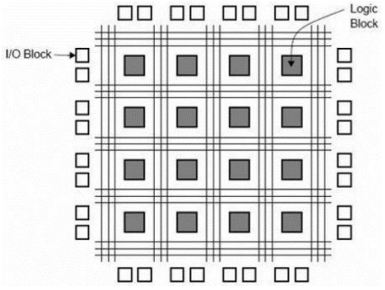


Fig. 1. Block diagram of an FPGA [source: (Tutorial Point, 2020)]

Several hardware description languages (HDLs) can be used for programming FPGAs. The most popular ones are Verilog and VHDL (Very High Speed IC Hardware Description Language). For this research, VHDL was used due to its deterministic and better error-checking capabilities. Architectures can be modelled using a structural description (which shows the interconnections of components), a behavioural description (a high-level description of how the circuit should behave) or a combination of both. In this paper a Cyclone II EP2C5 Mini Development Board was used for the hardware design. It has 4608 logic elements (16 logic elements per block), 26kb memory blocks, 13 embedded multipliers, 2 phase-locked loops (PLLs) and 89 usable I/O ports. Its compact size, low cost and high efficiency makes it a good choice for the design.

3.2. The chip model design

The base chip model designed is a synchronous 10-bit binary counter, which counts the number of occurrences of a trigger signal (clock), incrementing the count whenever a clock pulse occurs. The n-bit counter used contains n flip-flops which can hold values from 0 to $2^n - 1$, where n is the number of bits used for the counter. Hence, the 10-bit counter designed counts from 0 (0000000000) to 1023 (1111111111). Typically, once counters reach maximum count, go back to 0 and start all over. However the counter module of this research would be triggered by a positive-going clock transition, from 0 to 1 while, an asynchronous reset/clear input would be present, resetting the count to 0 whenever triggered. The elaborated design for the golden chip is shown in Figure 2. The VHDL source can be seen in Figure 3.

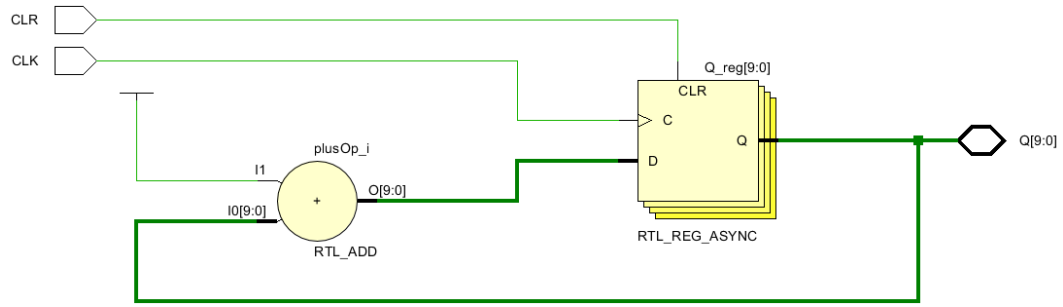


Fig. 2. Golden chip design

Given that the clock period is 10ms, the time it takes to complete a data gathering round is approximately 10 ms; 1024 = 10240 ms = 10.24 seconds. The trojan chip shares a lot of similarities with the golden chip, as expected for a device with a trojan. However, for the trojan device, count 512 (1000000000) is replaced with 1023 (1111111111) and it remains in this state until reset. This acts as a malicious modification to the base circuit. The trojan chip design is shown in Figure 4. The VHDL source can be seen in Figure 5.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity golden_chip is
    Port ( CLK : in STD_LOGIC := '0';
           CLR : in STD_LOGIC := '0';
           Q : inout STD_LOGIC_VECTOR (9 downto 0) := (others => '0'));
end golden_chip;

architecture Behavioral of golden_chip is
begin
    process(CLK, CLR)
    begin
        if CLR = '1' then
            Q <= (others => '0');
        elsif rising_edge(CLK) then
            Q <= Q + 1;
        end if;
    end process;

end Behavioral;
```

Fig. 3. Golden chip VHDL code

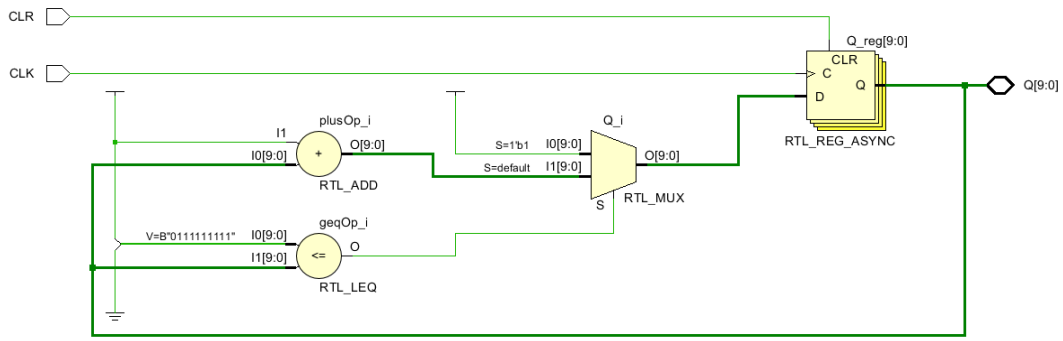


Fig. 4. Trojan chip design

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity trojan_chip is
    Port ( CLK : in STD_LOGIC := '0';
           CLR : in STD_LOGIC := '0';
           Q : inout STD_LOGIC_VECTOR (9 downto 0) := (others => '0'));
end trojan_chip;

architecture Behavioral of trojan_chip is
begin
    process(CLK, CLR)
    begin
        if CLR = '1' then
            Q <= (others => '0');
        elsif rising_edge(CLK) then
            if Q >= "0111111111" then
                Q <= (others => '1');
            else
                Q <= Q + 1;
            end if;
        end if;
    end process;

end Behavioral;
```

Fig. 5. Trojan chip VHDL code

3.3. Data acquisition with Arduino microcontroller

In this research, data used for the trojan detection was acquired using an Arduino microcontroller due to its flexibility and durability, albeit in previous works, LabVIEW was used for data acquisition, where data was read off the chip and sent to the computer for processing. Arduinos typically have a USB port where they can be interfaced with a computer for programming, and also for serial communication. The Arduino Uno used for this research addressed power measurements issues for SCA purposes, in addition, a INA219 current sensor was used for both voltage and current data acquisition and measurements, thus, the power computed by multiplying the instantaneous voltage and current readings. Figures 6 and 7 show the printData function and the microcontroller setup.

A python script was used for the extraction of the data at the other end of the serial COM port; pushed into the COM port by the Arduino sketch (the data acquisition system). The number of rounds is first specified as an input to the script, determining the amount of data to be gathered from the chip. The type of chip is also specified as an input, be it a golden or a trojan chip, so as to store the dataset in the appropriate location. Once the parameters have been determined, the script first resets the counter by communicating with the Arduino writing a custom 'RESET' command to the COM port. Afterwards, data is then collected a number of times, as specified by the input parameters, saving the data in .csv form.

```
void printData(float t) {
  int Q9 = digitalRead(2);
  int Q8 = digitalRead(3);
  int Q7 = digitalRead(4);
  int Q6 = digitalRead(5);
  int Q5 = digitalRead(6);
  int Q4 = digitalRead(7);
  int Q3 = digitalRead(8);
  int Q2 = digitalRead(9);
  int Q1 = digitalRead(10);
  int Q0 = digitalRead(11);

  float shuntvoltage = ina219.getShuntVoltage_mV();
  float busvoltage = ina219.getBusVoltage_V();
  float current_mA = ina219.getCurrent_mA();
  current_mA = abs(current_mA);
  float loadvoltage = busvoltage + (shuntvoltage / 1000);
  float power = current_mA * loadvoltage;

  Serial.println((String) t + "," +
                 (String) Q9 +
                 (String) Q8 +
                 (String) Q7 +
                 (String) Q6 +
                 (String) Q5 +
                 (String) Q4 +
                 (String) Q3 +
                 (String) Q2 +
                 (String) Q1 +
                 (String) Q0 + "," +
                 (String) power);
}
```

Fig. 6. An example of printData function

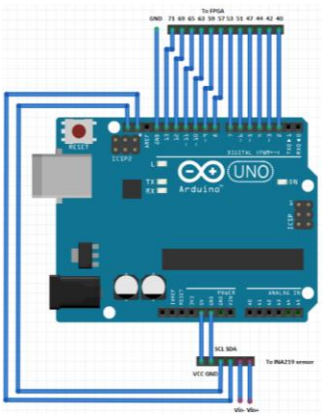


Fig. 7. Arduino data acquisition system

3.4. The Machine Learning model

A binary classification machine learning model which processes time-series data (power traces) and classifies each time-series data was built for the logic testing. It provides insight into whether or not the chip of interest contains a trojan. This is an hardware trojan detection technique which involves running test vectors through the input, checking for discrepancies at the output of the chip. A discrepancy thus indicate the presence of a trojan. If a chip fails the logic test; there is no need to proceed any further; it is tagged as containing a trojan. However, more sophisticated trojans, especially trigger-based and parametric trojans, may pass the logic testing stage. Nonetheless, a more sophisticated detection method would be required. This test result was used in this case to generate training data for the machine learning model. For both the golden and trojan chips, logic test was carried out severally and the power traces obtained were written into files, for later use in training the model. One hundred iterations were done to obtain large data in training the model for accuracy. However, there is a tradeoff between the accuracy of the model and how long it takes the model to train, as more training data increases the training time of the model.

The power SCA hardware trojan detection methodology was employed through monitoring of the power consumption (power trace) of the chip and analyzing the time-series to find patterns and detect whether or not a trojan is present in the device of interest. The power trace is monitored during logic testing, in anticipation of the chip passing the test. Once this happens, power SCA would be used to get more insight into the trojan status of the chip. This involves a sophisticated machine learning model which has been pre-trained. The power trace for any chip is bound to be different every time due to stochastic variations. However, there are underlying patterns in the power consumption for the chips, which is difficult for the human eye to observe, but easy for a sensitive machine learning model to discern. An elegant solution used in this research was a collection of values evenly spaced in time. These values, structured as one-dimensional arrays of length n, can be seen as representing a point in n-dimensional space. The time-series was then processed by the machine learning algorithm in this form; as a single data point. This is not far-fetched from the fact that deep learning approaches was used. The trained model is a neural network. See Figures 8 and 9 for the architecture of the neural network and the code which creates this model.

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1024, 32)          64
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 1024, 32)          204832
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 512, 32)           0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512, 32)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               53200
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 101
=================================================================
Total params: 258,197
Trainable params: 258,197
Non-trainable params: 0
```

Fig. 8. Neural network architecture

```
# create the model
model = Sequential()
model.add(Dense(32, input_shape=(N,1)))
model.add(Conv1D(filters=32, kernel_size=200, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.1))
model.add(LSTM(100))
model.add(Dropout(0.1))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

Fig. 9. Neural network code

4. MODEL IMPLEMENTATION

The software tools used for this research work are: **Altera Quartus II** (for programming the Cyclone II EP2C5 FPGA), **Arduino IDE** (where the sketches were written, compiled and loaded to the Arduino) and **IDLE** (the official IDE for Python). Incremental software development model was used due to its flexibility and the requirements nature of the entire research which are clear and specific.

4.1. Trojan detecting procedure

A comprehensive summary of the trojan detection procedure is illustrated in Figure 10, as a flow chart.

The implementation was setup on GitHub – Version Control System (VCS) which allows developers to track changes in code. During development, the different states of the Arduino, VHDL and Python source codes at various points were organised in snapshots called commits. This proved useful in a number of cases where changes which broke a feature had to be removed by rolling back to a previous commit or getting a few lines of code which existed in an older version of the source file of interest. GitHub served as a remote/online version of the model implementation.

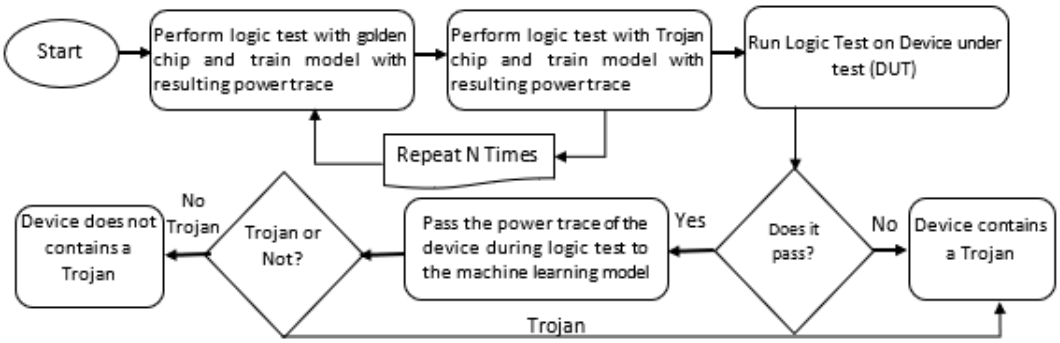


Fig. 10. Trojan detection procedure

4.2. Training the model

The model is a neural network which contains convolutional layers from CNNs and feedback layers as seen in RNNs, or more specifically, long short-term memory (LSTM) networks. A Python library Keras was used for building the network using Scikit-learn as a scale for the data and compute the model metrics (accuracy, precision and recall). At this stage all the data stored in text files are extracted with the aid of load_data and load_series python functions. See Figure 11 for code snippet.

```
def load_series(folder, file):
    t = []
    series = []
    with open(join(folder, file)) as f:
        data = f.read().split('\n')
        for line in data:
            linearr = line.split(',')
            t.append(float(linearr[0]))
            series.append(float(linearr[2]))
    return t, series
```

Fig. 11. An example of 'load_series' function

Afterwards, the data was normalized and the model trained on the training dataset. Following the training, and device testing using power SCA (i.e. after passing the logic test), the time-series is then converted to a data point and passed as input to the trained neural network. The neural network outputs a probability; the probability that the chip contains a trojan. Based on a predefined threshold, the probability is converted to a binary decision afterwards; whether or not the chip contains a trojan. Testing was then done using the testing dataset to determine the accuracy, precision and recall of the model.

4.3. Discussion of result

The data acquisition system was able to collect counter state data and power consumption data for both the golden and trojan chips. The data was structured in CSV using the format "time (in seconds), counter state, power consumption (in mW)". Figures 12 and 13 show some data samples for the golden and trojan chips respectively. Observe that, as expected, the trojan chip deliberately gives the wrong state for counts after 511.

```
5.00,0111110100,1026.32
5.01,0111110101,1026.33
5.02,0111110110,1023.51
5.03,0111110111,1025.32
5.04,0111111000,1024.50
5.05,0111111001,1024.83
5.06,0111111010,1026.00
5.07,0111111011,1026.82
5.08,0111111100,1025.32
5.09,0111111101,1026.32
5.10,0111111110,1025.33
5.11,0111111111,1023.51
5.12,1000000000,1024.84
5.13,1000000001,1025.65
5.14,1000000010,1024.50
5.15,1000000011,1026.82
5.16,1000000100,1025.50
5.17,1000000101,1026.50
5.18,1000000110,1025.82
5.19,1000000111,1026.32
5.20,1000001000,1024.33
```

Fig. 12. Some data samples for the golden chip

```
5.00,0111110100,1020.54
5.01,0111110101,1021.04
5.02,0111110110,1020.71
5.03,0111110111,1020.21
5.04,0111111000,1021.04
5.05,0111111001,1021.04
5.06,0111111010,1021.54
5.07,0111111011,1021.54
5.08,0111111100,1020.54
5.09,0111111101,1021.04
5.10,0111111110,1020.05
5.11,0111111111,1021.54
5.12,1111111111,1021.54
5.13,1111111111,1020.04
5.14,1111111111,1023.02
5.15,1111111111,1020.04
5.16,1111111111,1021.04
5.17,1111111111,1020.04
5.18,1111111111,1021.54
5.19,1111111111,1022.03
5.20,1111111111,1020.54
```

Fig. 13. Some data samples for the trojan chip

The model was trained, validated and tested using the acquired data, for 5 epochs. As expected, the accuracy increased and the model loss decreased. The validation and testing set also performed well on the data. More training epochs are possible but there would be an increased risk of overfitting the model. Figures 14 and 15 show graphs for the model performance over training epochs.

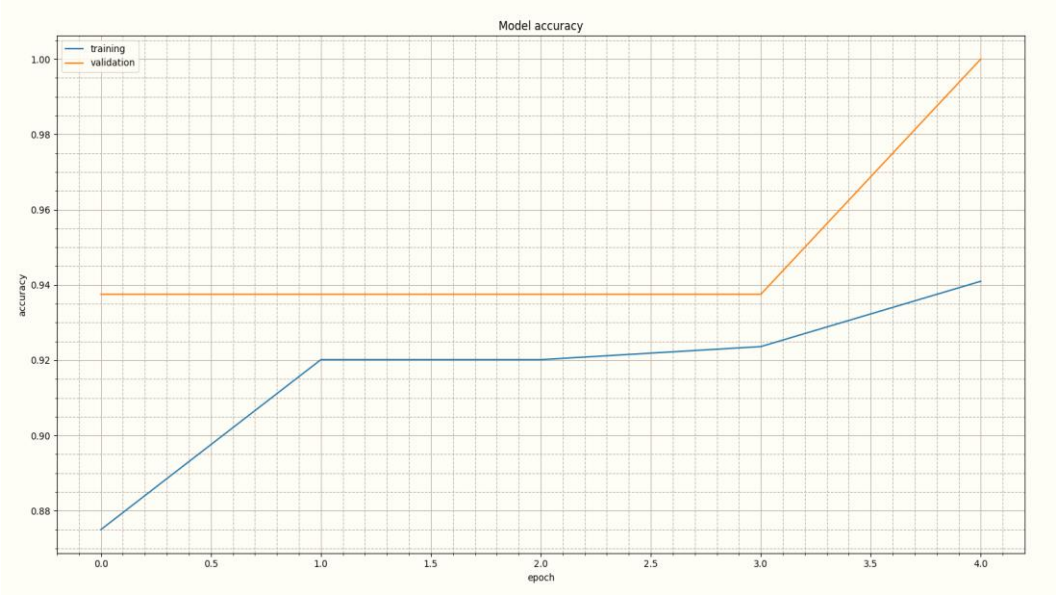
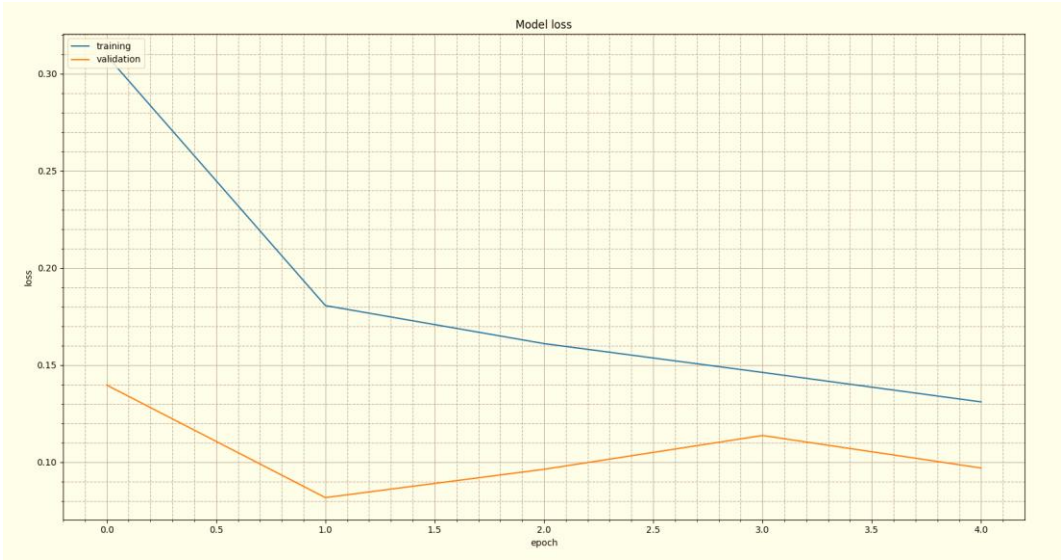


Fig. 14. Model accuracy



Fig, 15. Model loss

Following the successful training of the model, the system as a whole was tested using golden and trojan chips. The results are shown in Figures 16 and 17 respectively. Observe that both logic and power SCA tests give the same result.

```
Connected to: COM12
Test running...
Logic test: Passed

Trojan probability: 0.091
Power SCA test: Passed
>>>
```

Fig. 16. Testing a golden chip

```
Connected to: COM12
Test running...
Logic test: Failed

Trojan probability: 0.898
Power SCA test: Failed
>>>
```

Fig. 17. Testing a trojan chip

As depicted in the results, the trojan detection system works to a high degree of accuracy. The power consumption readings of the hardware characteristically start at 1035–1040mW. The power Time-series data were simulated by using DC power measurements mixed with additive white Gaussian noise (AWGN) with different standard deviations. The model was then trained and tested based on the simulated data. The recall improved by adjusting the threshold variable, although at the expense of too many false positives in the model's prediction. At the early stage, the model achieves accuracy, precision and recall values

of 80 to 90 percent. Setting the threshold probability for the trojan class less than 0.5 however increases the recall, which is the most important metric for the system later achieved overall accuracy, precision and recall values of over 95 percent after several epochs of training.

However, it should be noted that the detection system built was trained with specific implementation of a trojan; that alters the count of the counter in a very unique way. To kin future, the model could be subjected to varieties of trojan implementations for the base chip. The more trojan implementations the model learns from, the more accurately it can identify patterns in golden and trojan chips, and correctly classify them. Also, although the neural network works reasonably well, its architecture was mostly arbitrary. A possible follow-up research work is finding the optimal network architecture for training the model. Search techniques such as grid search, or optimisation techniques with a touch of genetic algorithms can also be applied to find the optimal hyperparameters for training the model.

5. CONCLUSION

Two chips – one golden and the other trojan were designed – both having hardware and software interface. A machine learning model was then built in Python and trained with data gathered from the hardware. This project illustrates a concept which solves the problem stated earlier in this paper. Using sophisticated and cutting-edge machine learning techniques, a system which can detect modification to integrated circuit designs has been built. This work provided a different and better approach to such malicious modifications to hardware considering the emerging security concerns. While novel methods are being created to combat them, the more recent trojans are intelligently written and are capable of evading detection by most methods. This work takes a more sophisticated approach in detecting these trojans, using machine learning.

REFERENCES

Bai, X. (2018). Text classification based on LSTM and attention. *2018 Thirteenth International Conference on Digital Information Management (ICDIM)* (pp. 29–32). IEEE. https://doi.org/10.1109/ICDIM.2018.8847061

Chen, X., Wang, L., Wang, Y., Liu, Y., & Yang, H. (2016). A general framework for hardware trojan detection in digital circuits by statistical learning algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (vol. 36, no. 10, pp. 1633–1646). IEEE. https://doi.org/10.1109/TCAD.2016.2638442

Cui, Q., Sun, K., Wang, S., Zhang, L., & Li, D. (2016). Hardware trojan detection based on cluster analysis of mahalanobis distance. *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)* (pp. 234–238). IEEE. https://doi.org/10.1109/IHMSC.2016.65

Grus, J. (2015). *Data Science from Scratch. 1005 Gravenstein Highway North*. O'Reilly Media.

He, C., Hou, B., Wang, L., En, Y., & Xie, S. (2014). A novel hardware Trojan detection method based on side-channel analysis and PCA algorithm. *2014 10th International Conference on Reliability, Maintainability and Safety (ICRMS)* (pp. 1043–1046). IEEE. https://doi.org/10.1109/ICRMS.2014.7107362

Iwase, T., Nozaki, Y., Yoshikawa, M., & Kumaki, T. (2015). Detection technique for hardware Trojans using machine learning in frequency domain. *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)* (pp. 185–186). IEEE. https://doi.org/10.1109/GCCE.2015.7398569

Jahan, I., Sajal, S. Z., & Nygard, K. E. (2019). Prediction model using recurrent neural networks. *2019 IEEE International Conference on Electro Information Technology (EIT)* (pp. 1–6) IEEE. https://doi.org/10.1109/EIT.2019.8834336

Ni, L., Li, S., Chen, J., Wei, P., & Zhao, Z. (2014). The influence on sensitivity of hardware trojans detection by test vector. *2014 Communications Security Conference (CSC 2014)* (pp. 1–6). IEEE. https://doi.org/10.1049/cp.2014.0756

Paul, L. C., Suman, A. A., & Sultan, N. (2013). Methodological analysis of principal component analysis (PCA) method. *International Journal of Computational Engineering & Management*, *16*(2), 32–38.

Tutorial Point. (2020). Retrieved October 8, 2021 from https://www.tutorialspoint.com

Salmani, H., Tehranipoor, M., & Plusquellic, J. (2011). A novel technique for improving hardware trojan detection and reducing trojan activation time. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *20*(1), 112–125. https://doi.org/10.1109/TVLSI.2010.2093547

Shende, R., & Ambawade, D. D. (2016). A side channel based power analysis technique for hardware trojan detection using statistical learning approach. *2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN)* (pp. 1–4). IEEE. https://doi.org/10.1109/WOCN.2016.7759894

Wang, L.-W., & Luo, H.-W. (2011). A power analysis based approach to detect Trojan circuits. *2011 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering* (pp. 380–384). IEEE. https://doi.org/10.1109/ICQR2MSE.2011.5976635

Zhang, L., Sun, K., Cui, Q., Wang, S., Li, X., & Di, J. (2016). Multi adaptive hardware Trojan detection method based on power characteristics template. *2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)* (pp. 414–418). IEEE. https://doi.org/10.1109/CCIS.2016.7790294