Revealing the Secrets of Spiking Neural Networks

The Case of Izhikevich Neuron

Caetano Garaffa, Luíza ; Aljuffri, Abdullah; Reinbrecht, Cezar; Hamdioui, Said; Taouil, Mottaqiallah; Sepulveda, Johanna

DOI
10.1109/DSD53832.2021.00083

Publication date
2021

Document Version
Accepted author manuscript

Published in
2021 24th Euromicro Conference on Digital System Design (DSD)

Revealing the Secrets of Spiking Neural Networks: The Case of Izhikevich Neuron

Luíza C. Garaffa, Abdullah Aljuffri, Cezar Reinbrecht, Said Hamdioui, Mottaqiallah Taouil
Delft University of Technology – EEMCS Faculty
Delft, The Netherlands
m.taouil@tudelft.nl

Johanna Sepúlveda
Airbus Defense and Space
Munich, Germany
johanna.sepulveda@airbus.com

***Abstract*—Spiking Neural Networks (SNNs) are a strong candidate to be used in future machine learning applications. SNNs can obtain the same accuracy of complex deep learning networks, while only using a fraction of its power. As a result, an increase in popularity of SNNs is expected in the near future for cyber physical systems, especially in the Internet of Things (IoT) segment. However, SNNs work very different than conventional neural network architectures. Consequently, applying SNNs in the field might introduce new unexpected security vulnerabilities. This paper explores and identifies potential sources of information leakage for the Izhikevich neuron, which is a popular neuron model used in digital implementations of SNNs. Simulations and experiments on FPGA implementation of the spiking neurons show that timing and power can be used to infer important information of the internal functionality of the network. Additionally, the paper demonstrates that is feasible to perform a reverse engineering attack using both power and timing leakage.**

I. Introduction

Designed to emulate the brain's dynamics with a higher degree of biological plausibility, Spiking Neural Networks (SNNs) are a strong candidate for future machine learning [1]. SNNs can obtain the same accuracy as classic artificial neural networks (ANNs) for a wide variety of applications, while only using a fraction of its power [2]–[4]. As a result, the popularity of SNNs is expected to increase in the near future, especially for low-power applications like IoT. However, SNNs will only become a reality for IoT devices when they run in specialized platforms known as neuromorphic hardware, which are optimized for performance and power. As a result, such neuromorphic platforms might introduce new and unexpected security vulnerabilities to the IoT segment as the underlying hardware works completely different than conventional Von Neumann architectures. Therefore, it is extremely important to investigate the vulnerabilities of SNNs running in neuromorphic hardware before they are widely deployed.

The amount of research that investigated security aspects of SNNs has largely increased in the recent years. A significant part of the published articles focus on evaluating the robustness of SNNs when exposed to adversarial input noise [5]. However, degrading the SNN's integrity and performance are not the only possible forms of attack. The leakage of sensitive information, such as the neuron's weights, facilitates reverse engineering attacks of targeted neural networks, whose proper training usually requires a considerable amount of effort, time, and financial resources. These kind of attacks have already been proven to be effective against traditional ANN [6] and could also be a severe threat to SNNs. In that context, theoretical threat models are proposed in [7] involving side channel, fault injection and IC attacks against neuromorphic hardware. With regard to countermeasures against these kind of attacks, in [8] the obsolescence effect of memristors is employed to deceive unauthorized users with physical access to the hardware, and in [9] deep learning algorithms are used for intrusion detection. However, the study field is still incipient. It is important to reinforce that SNNs run in neuromorphic hardware when applied to cyber-physical systems. Given that possible attacks could have severe economic consequences, it is clear that further vulnerability analysis is required in order to deploy SNNs securely.

This paper explores and studies potential leakage sources of a widely employed neuron model, i.e., the Izhikevich neuron [10]. Simulations and experiments based on FPGA implementation shows the existence of timing and power leakage. In summary, our contributions are: i) A methodology to investigate vulnerabilities of a spiking neuron model; ii) The characterization of timing and power behavior for the Izhikevich neuron; and iii) A discussion of how timing and power leakages could be used in reverse engineering attacks to recover the neuron's weights of SNN applications running in neuromorphic hardware.

II. Spiking Neural Networks

For decades, the scientific community has been putting efforts into developing biologically-inspired devices, aiming to reproduce the human brain's impressive processing and energy efficiency characteristics. With a volume between 1.2 and 1.5 liters, the brain is able to process up to 100 trillions of synapses with an average power consumption of only 20W [11]. In that context, the neuromorphic computing field studies brain-inspired implementations that emulate the biological neural structure. With the imminent end of Moore's law and the necessity of replacing Von Neumann-based architectures, neuromorphic computing is being highlighted as a promising alternative due to its massive parallel processing, real-time performance, scalability, and low power consumption [1].

Neuromorphic devices are especially appropriate to execute machine learning applications. Spiking Neural Networks (SNNs), referred to as the third generation of artificial neural networks, fit ideally in this scenario. SNNs are event-driven networks in which the neurons' communication occurs through discrete events, i.e., spikes. Information is encoded through

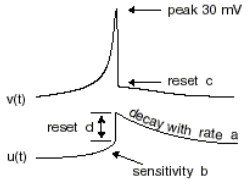

Fig. 1. Izhikevich Spike

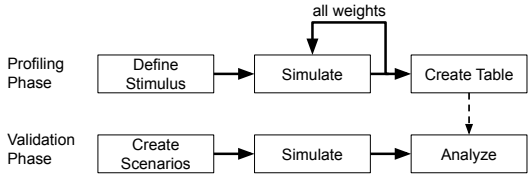

Fig. 2. Timing Vulnerability Methodology

spiking rates and spiking moments. The basic processing units are called neurons, which are modeled by equations that aim to reproduce the behavior of biological cells. In the last decade, several articles have been exploring the applicability and performance of such networks. Despite training challenges due to the non-differentiable nature of spikes, significant progress has been made. Some noteworthy achievements include the use of SNNs in classic supervised learning applications like object recognition and biological signal classification [2], and the implementation of reinforcement and unsupervised learning applications using the Spike Timing Dependent Plasticity training method [4].

Due to memory and communication bottlenecks, the high processing speed and energy efficiency of SNNs are not fully explored in traditional computing architectures. Therefore, neuromorphic hardware specifically designed for SNNs are being increasingly studied. Essentially, these architectures consist of neurosynaptic cores that operate in parallel. The cores represent the neural networks' nodes and are usually composed of processor, data bus, and memory, analogous to the cell body, axon, and synapse of the biological neuron. The number of published papers on hardware for neural networks has grown almost exponentially in the last decade [1]. Despite current technical challenges, efforts from academia and industry indicate that SNN hardware will be employed in an increasing number of applications, thereby playing a fundamental role in the technological development of the coming years.

Izhikevich Neuron

It is of critical importance to define a suitable neuron model as a basic building block of an SNN. Several models of spiking neurons try to reproduce the behavior of a biological cell. A popular neuron model that provides an acceptable compromise between computational efficiency and biologically realistic behavior is the one proposed by Izhikevich [10]. The model is widespread in the neuromorphic literature because it presents a biological plausibility similar to the Hodgkin-Huxley neuron [12] while being computation-wise as efficient as an Integrate-and-Fire model [13]. It is also capable of simulating large-scale spiking neurons in real-time [1].

The Izhikevich neuron model is represented by Equations 1, 2 and 3. In the equations, $v(t)$ denotes the membrane potential over time, $u(t)$ the recovery factor, I the synaptic input, C the capacitance and R the leaky resistance. The other parameters such as k, a, b, and c are predefined constants controlling the spike shapes. The relations between the variables and constants are provided in Figure 1.

$$C\frac{dv(t)}{dt} = k(v(t) - V_{rest})(v(t) - V_{th}) - u(t) + I \quad (1)$$

$$\frac{du(t)}{dt} = a[b(v(t) - V_{rest}) - u(t)] \quad (2)$$

$$\text{if } v(t) \geq V_{peak} : v(t) = c, u(t) = u(t) + d \quad (3)$$

III. Methodology

A. Sources of Leakage

Before devising an attack against a specific system, it is necessary to investigate and characterize its vulnerabilities. The target device in this paper is a neuromorphic hardware platform running SNN applications with the Izhikevich neuron model. Leakage based on timing is an interesting candidate as the neurons are modeled with equations that have different timing responses depending on the applied input. As a direct result, the power consumption is also a potential candidate. The system's operation logic indicates that the consumed power varies depending on the spiking activity, which is not constant throughout the network. In addition to the above arguments, time and power are some of the main exploitable features in side channel attacks. Based on the former premises, a methodology is formalized and employed in order to verify whether time and power are leakage sources.

B. Timing Vulnerability Analysis

The methodology to exploit timing leakage is divided into two phases: profiling and validation. The goal of the profiling phase is to collect measurements from a single neuron and build a model that captures the neuron's input rate, weight, and output timing characteristics. Thereafter, the validation phase evaluates if the model can infer correctly sensitive information about a neuron, such as its weight. Figure 2 shows the performed steps during the profiling and validation phases.

Profiling - First, a set of input stimuli with constant spiking rates are defined. Next, these inputs are applied to a single neuron in a controlled simulation environment. For the simulation, it is possible to use a neuron modeled at a high abstraction level like C or Python, or at a hardware abstraction level like Verilog or VHDL. Any abstraction level suffices, as long as the simulation appropriately derives the timing behavior at the output. Next, each defined stimulus is applied to neurons for all the possible weight values. For each case, the output spiking rate is recorded and stored. The profiling phase results in a table that matches input spiking rates and weight values to the expected output spiking rate.

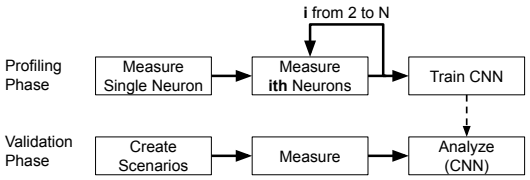

Fig. 3. Power Vulnerability Methodology

Validation - The first step in this phase is to create many different scenarios; each scenario contains a fixed input rate and an unknown weight which are both randomly defined. Thereafter, each scenario is simulated and the output spiking rate is further analyzed. The analysis uses the applied input and the output timing information to infer the weight of the neuron. To accomplish this task, the table created in the profiling phase is used. An important remark here is that some cases might lead to no spikes in the output, due to a low input rate or low weight value. In these cases, the process is repeated by changing the input stimuli. Additionally, if the random input data rate does not appear in the profiling table, the closest value is assigned. To avoid misinterpretations, the entire validation process can be repeated using various iterations and majority voting. A successful validation phase would demonstrate that it is possible to correctly model the timing leakage of the neuron to reveal its weight. However, using only timing as a side channel to recover the weights of an entire SNN is not practical as the attacker requires observability at the output of all neurons. For example, the output of the hidden layers is not accessible to the attacker since the network works as a blackbox. Therefore, in order to successfully exploit timing leakage, auxiliary techniques are required to be able to observe the output of all spiking neurons.

C. Power Vulnerability Analysis

Similarly to timing analysis, the power analysis also occurs in two stages: profiling and validation. Figure 3 shows these phases and the steps they consist of. They are described below.

Profiling - In the profiling phase, the power measurements of one or more neurons running on a hardware platform are acquired. Each measurement uses fixed inputs that are randomly selected. First, the power is characterized for a single neuron. The power trace is subsequently cut into fixed-size windows to ensure that only the power samples containing spikes are saved. Next, the same power trace collection is performed with more neurons running in parallel. All parallel neurons use the same input stimuli and weights. The concept here is to verify whether it is possible to identify multiple spikes occurring at the same time from the power trace. Thereafter, the collected power traces are organized according to the number of simultaneous spikes and stored in a database. As power traces contain noise, it is impossible to visually identify the spikes and figure out how many neurons spiked simultaneously. Therefore, to improve the accuracy of the analysis, a Convolutional Neural Network (CNN) is trained to learn the power behavior of single and multiple spiking

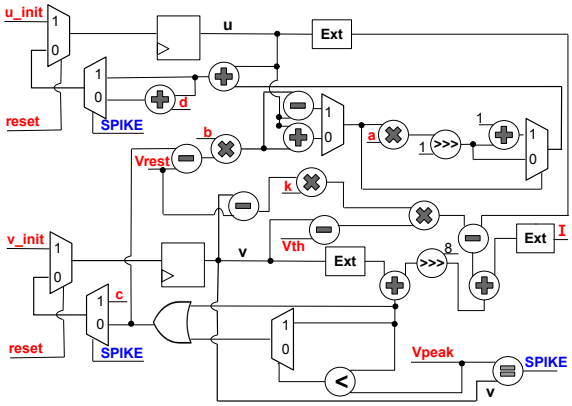

Fig. 4. Hardware Schematic of the Izhikevich Neuron Model.

neurons. The database generated in this profiling phase and power traces from neurons with different weight values are used to train this CNN.

Validation - In the validation phase, multiple neurons are executed on the target hardware each having the same fixed input spiking rate. Considering applications where the attacker has physical access to the hardware (e.g. IoT devices), measuring the device's power consumption is feasible using extra equipment. The collected power traces are applied to the trained CNN, which provides the amount of spikes that are being simultaneously fired at the given time instance. A successful power analysis reveals a spike train containing all the spikes in the system, including the ones happening in the SNN's internal layers.

IV. Experiments and Results

A. Setup

Two neuron implementations were used in the experiments. First, an Izhikevich model was written in Python language and compared to the one provided in [10]. Next, a compatible hardware Izhikevich model was designed using Verilog, according to the schematic presented in Figure 4. The hardware solution uses a fixed-point implementation and two's-complement to encode the variables in the differential equations presented in Section II. The spiking neural parameters are the same as in [14]. The hardware module was simulated using Modelsim from Mentor Systems [15] and emulated in a Xilinx Artix-7 FPGA. Important neuromorphic implementations as TrueNorth [16] and Loihi [17] are fully digital, and use neurons that constantly update their state; when the neuron state exceeds a programmable threshold, the neuron fires a spike. Our neuron implementation exhibits similar behaviour as the ones used in state-of-the-art neuromorphic architectures like TrueNorth and Loihi and hence, the analysis performed in this work could be reproduced on different implementations.

Figure 5 shows the setup and equipment used to perform the power-based experiments. The Izhikevich neuron was synthesized for the CW305 Chipwhisperer board [18] which contains a Xilinx FPGA. Next, we connected a current probe in the access ports to the power supply of the Chipwhisperer board.

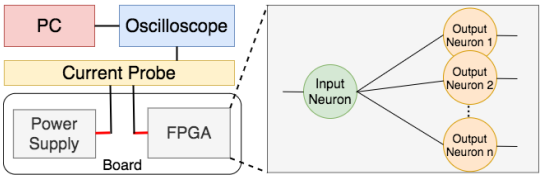

Fig. 5. Experimental Setup for the Power Measurements.

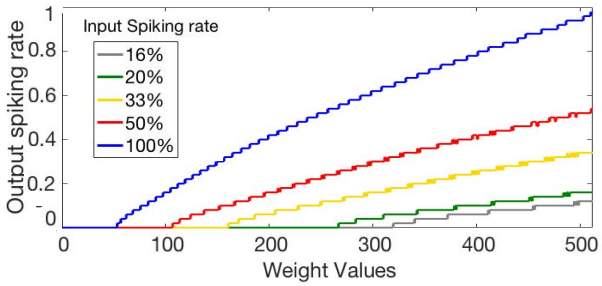

Fig. 6. Weight and Output Spiking Rate Profile

The measured current is transformed to a voltage that reflects the power behavior. The power samples were collected by the PicoScope 6000 oscilloscope using a sampling frequency of 1 Gsps. Finally, all data was sent to a desktop computer for further analysis, such as applying the traces to the CNN.

B. Timing Experiments

For the profiling phase in the timing analysis, we defined five different input spiking rates: 100% $(25MHz)$, 50% $(12,5MHz)$, 33% $(8,33MHz)$, 20% $(5MHz)$ and 10% $(2,5MHz)$. Next, we simulated the neuron under these stimuli for all possible weight values resulting in 131072 collected traces. Each simulation ran for $100\mu s$, and the output spikes were recorded. The hardware is based on a 17-bit fixed-point implementation, where 9 bits are used for the integer part and 8 for the fractional part. The output spiking rate was calculated by analyzing the time between spikes during a time window of $100\mu s$. Then, the spiking rates were stored in the form of a table, whose graphical representation is shown in Figure 6. As expected, the results show that high input spiking rates can trigger neurons with a wider weight range. Additionally, the higher input spiking rates clearly reveal the relation between input, output, and weights. Another interesting remark is that weights below 50 are useless in the analyzed model since they can not generate an output spike even with the maximum input stimulus.

For the validation phase, a data set of 500 random inputs was generated. Each input was applied to a neuron with a fixed weight during 100 μs while the output spikes were recorded. The experiments were reproduced for 3 different weight values (a small weight of 80, medium weight of 300, and a large weight of 512) using the same 500 inputs. For each experiment, where the input stimulus is provided and the output spiking rate is observed, it was possible to recover the corresponded neuron weight using the information collected during the profiling phase. The three targeted weights were inferred with a 100% of accuracy.

TABLE I
CNN FOR SPIKE DETECTION TRAINING AND VALIDATION RESULTS.

Number of neurons	Training Accuracy	Validation Accuracy
1	100%	100%
2	100%	100%
3	100%	100%
4	100%	100%
5	100%	100%
15	98.90%	95.05%
30	96.79%	74.90%
45	99.49%	85.05%
60	98.75%	79.65%
All traces	96.20%	88.77%

C. Power Experiments

Using the setup presented in Section IV-A, power traces were recorded for different neurons connected in parallel with the same weight. A fixed-rate input was set using the maximum input spiking rate (i.e., $25MHz$). Figure 7 presents some examples of the collected power traces in the profiling phase. The blue graph indicates the power, while the red indicates the real output spikes. The amplitude of the red spikes was artificially increased in order to improve visualization. The different traces show the power consumption for a different number of simultaneously spiking neurons. Only from visual analysis it is possible to conclude that the power measurements present prominent positive and negative peaks that coincide with the time that a spike is fired. In addition, it can be easily observed that the negative peaks have a higher amplitude than the positive peaks, suggesting that they may represent the most useful information when identifying the spiking activity. Furthermore, the amplitude increases proportionally with the number of firing neurons. This behavior indicates that not only the spiking times can be identified from power measurements, but also that the traces carry information about the number of active neurons in the network. Therefore, to extract such information with a high level of accuracy, our methodology employs the aforementioned CNN.

The proposed CNN has 198 input neurons, one for each sample of a power trace window, 12 hidden layers and 6 output neurons. The output neurons are used to identify from 0 up to 5 simultaneously spiking neurons. To train and validate the CNN, We created a dataset that contains traces where 1, 2, 3, 4, and 5 neurons are simultaneously activated (i.e., they receive the same input and have the same weight) and traces that contain 15, 30, 45, and 60 neurons where only up to 5 neurons can spike simultaneously. The other neurons have random weights and are added to generate noise. 70% of the dataset was used for training , and the remaining 30% for validation. For comparison purposes, the CNN was trained and validated using the traces from 1 to 60 individually and using all of them together; the results are presented in Table I. When traces that do not contain noise are used (i.e., the cases with up to 5 neurons), both a training and validation accuracy of 100% can be achieved. Even when noise is present, the accuracy rate remains high. When all traces are used, a training accuracy of $96,20\%$ and validation accuracy of $88,77\%$ can be realized. The obtained results prove that power is a critical leakage source of SNN hardware, as it can be used to understand the spiking activity of the target network.

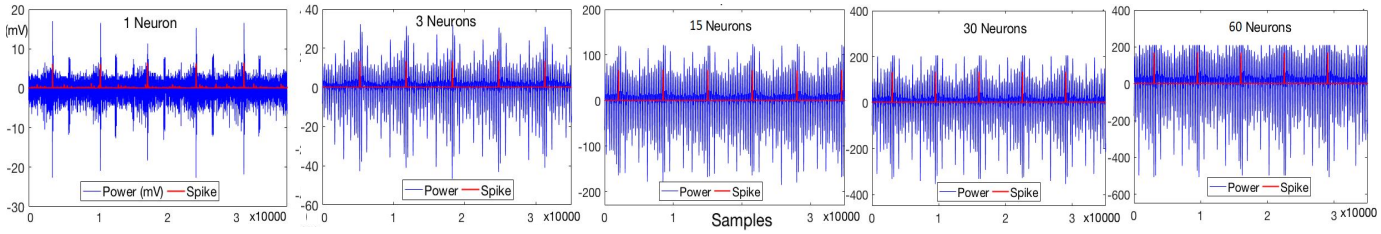

Fig. 7. Power traces collected for different number of neurons simultaneously spiking.

V. Discussion

Correctly inferring the neuron's weight by analyzing the timing of spikes is feasible. However, the attacker needs to be able to observe the neurons' input and output spikes. If the system provides some debug or test mechanism that allows users to gain such observability for each neuron, an entire SNN could be reverse-engineered. However, this degree of observability is not realistic for most devices, making time alone not enough to perform an attack, despite clearly being a leakage source. On the other hand, power leakage has shown to be very useful to identify the spikes in the system. Considering hardware employed on the field like IoT devices, it is feasible for an attacker to collect power traces. The 100% weight accuracy verified in different measurements allows the attacker to use this methodology to extract a map containing the system's spiking activity. Such a map could be used to visualize the spikes of all the network layers and enable the aforementioned observability needed for timing analysis.

Reverse Engineer of the first SNN layer: From the obtained results, a reverse engineer attack for the first hidden layer can be elaborated. Consider the SNN topology presented in Figure 5, where the target neurons are identified as *Output Neurons*. The attacker can apply a certain input and perform power analysis to retrieve the SNN spike activity. Since the input neuron does not consume substantial power, the obtained spikes correspond only to the output neurons. If the input is fixed, the spiking behavior should be periodic. Thus, it is possible to decompose the spike train into different output spiking rates through mathematical analysis. From timing analysis, all weights could be retrieved by combining the obtained rates with the input information. Note to obtain good results, the accuracy of the CNN that detects the number of spikes must be high.

Reverse Engineer of an entire SNN: Our results have shown that the combination of timing and power leakages can be used to perform reverse engineering attacks. However, more research is still required since attacking deep hidden layers is not trivial. The system's complexity increases since the attacker cannot infer the internal connections (i.e., synapses) and cannot control the input of internal layers. Hence, a more sophisticated methodology to attack an entire SNN would be required. This is part of future work.

VI. Conclusion

This paper investigated the vulnerabilities of a spiking neuron commonly implemented in neuromorphic hardware. Our results revealed that the neuron leaks information through timing and power, and we discussed how potential practical attacks on SNN can be created. At the same time that the results show that it is not trivial to exploit these leakages, they reinforce the need for security measures when applying such architectures widely in the field.

Acknowledgment

This work was labelled by the EUREKA cluster PENTA and funded by Dutch authorities under grant agreement PENTA-2018e-17004-SunRISE.

References

[1] C. D. Schuman *et al.*, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.
[2] P. Bawane *et al.*, "Object and character recognition using spiking neural network," *Materials Today:Proceedings*, vol. 5, pp. 360–366, 2018.
[3] S. R. Kheradpisheh and T. Masquelier, "Temporal backpropagation for spiking neural networks with one spike per neuron," *International Journal of Neural Systems*, vol. 30, no. 06, p. 2050027, 2020.
[4] F. Paredes-Vallés, K. Y. Scheper, and G. C. de Croon, "Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception," *IEEE txn on pattern analysis and machine int.*, vol. 42, no. 8, pp. 2051–2064, 2019.
[5] S. Sharmin *et al.*, "Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations," in *European Conference on Computer Vision*, 2020, pp. 399–414.
[6] L. Batina *et al.*, "Csi neural network: Using side-channels to recover your ann information," *arXiv preprint arXiv:1810.09076*, 2018.
[7] J. Sepulveda, C. Reinbrecht, and J.-P. Diguet, "Security aspects of neuromorphic mpsocs," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–6.
[8] C. Yang *et al.*, "Security of neuromorphic computing: thwarting learning attacks using memristor's obsolescence effect," in *Proceedings of International Conference on Computer-Aided Design*, 2016, pp. 1–6.
[9] M. Z. Alom and T. M. Taha, "Network intrusion detection for cyber security on neuromorphic computing system," in *International Joint Conference on Neural Networks*. IEEE, 2017, pp. 3830–3837.
[10] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.
[11] D. Drubach, *The brain explained*. Pearson, 2000.
[12] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500–544, 1952.
[13] N. Brunel and M. C. Van Rossum, "Lapicque's 1907 paper: from frogs to integrate-and-fire," *Biological cybernetics*, vol. 97, no. 5, pp. 337–339, 2007.
[14] A. Tavanaei and A. S. Maida, "A minimal spiking neural network to rapidly train and classify handwritten digits in binary and 10-digit tasks," *Int journal of advanced research in AI*, vol. 4, no. 7, pp. 1–8, 2015.
[15] Modeltech, "Modelsim reference manual," 2004.
[16] A. Cassidy *et al.*, "Truenorth: A high-performance, low-power neurosynaptic processor for multi-sensory perception, action, and cognition," in *Proceedings of the Government Microcircuits Applications & Critical Technology Conference, Orlando, FL, USA*, 2016, pp. 14–17.
[17] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.
[18] NewAE Technology Inc, "Chipwhisperer-Lite two part board," Available at: http://store.newae.com/chipwhisperer-lite-cw1173-two-part-version/, accessed: 2021-05-05.