A simulated neuromorphic processor on a low cost consumer grade FPGA hardware for studying neuromorphic algorithms

Sprint - 01

Vivek Patel, Evan Lang, Raunak S Rajpal

Electrical and Computer Engineering

Submitted to:
Professor Osama Alshaykh

Index:

Introduction

In this report, a literature review of the paper *Exploring Neuromorphic Computing Based on Spiking Neural Networks: Algorithms to Hardware*, by Rathi et al is investigated. The paper goes through neuromorphic computing from both a hardware and software perspective, while also covering various different applications of the emergent computer architecture. This paper will serve as the foundation needed to advance our project, implementing a spiking neural network on a consumer grade FPGA. We will also discuss both the memristors and CMOS components that form the core of modern neuromorphic computing architectures. Through this report, the foundations and background knowledge needed to engineer neuromorphic chips will be explored.

Literature Review

In this paper, Nitin Rathi et al seek to chart and explore the current algorithms and hardware that have advanced during neuromorphic computing's development in recent years. They begin by describing the focus of neuromorphic computing, Spiking Neural Networks (SNNs). These networks mimic the behavior of biological neuronal systems, with the artificial neurons firing events or "spikes". Compared to typical neural networks (referred to as analog neural networks within the paper) these SNNs allow for the use of new event driven computations that utilize temporal encoding. The paper then moves into the details of the neuron models used as the foundation for modern spiking neural networks. This neuron model describes the behavior of the artificial neurons, where spikes received from other neurons build a potential within the neuron. Once this potential has exceeded a threshold, it triggers the neuron to fire a spike itself. The equation for the LIF/IF Neuron model can be seen below:

$$\tau \frac{du(t)}{dt} = -[u(t) - u_{rest}] + RI(t),$$

u is the internal state known as the membrane potential, $_{rest}$ is the resting value of the membrane potential, RI(t) is the input resistance (current), and τ is the time constant

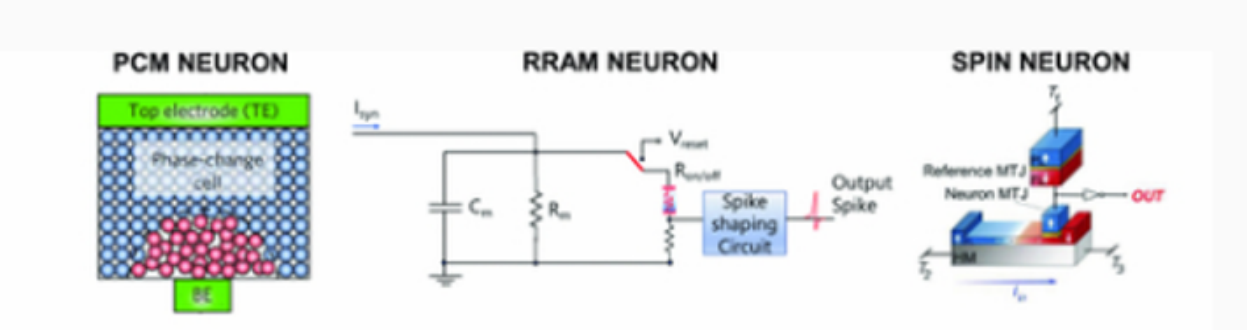
This equation describes the foundational behavior of neurons over a period of time t. The paper then describes an additional neuron model: the stochastic neuron model. This model abandons the deterministic approach of the previous one, instead adopting a probabilistic approach. Here the neurons fire stochastically, where the probability of one firing is now a

non-linear function of its inputs, rather than being based on discrete time steps. These 2 foundational models for neurons underpin all modern neuromorphic computing.

The paper then explains the encoding signals in neuromorphic computing receive. These coding methods are rate coding and temporal coding. In rate coding, the information is represented in the mean firing rate of the neuron within a time period. In temporal coding, the timing of individual spikes is crucial, as the information is encoded in the timing instances. This allows the event based behavior discussed earlier in this paper. After encoding discussion the paper moves on to discuss the network structure used in neuromorphic computing. These structures are very similar to those used in common neural networks with the 2 main structures, feedforward and recurrent networks, being the same foundation as classical neural networks.

At the end of the algorithms section of the paper, neuromorphic libraries and APIs. These include Pytorch, Tensorflow, Caffe, and more. These libraries allow for the development of neuromorphic software without worrying about how it will affect or be affected by the neuromorphic hardware.

The next topic explored are the applications of the new technology. Specifically, Image classification, gesture recognition, sentiment analysis, biomedical tooling, motion estimation, and adversarial robustness are cited as key applications. In the interest of time, I will focus discussion on what I found to be the most interesting application, adversarial robustness. Classical neural networks, built on standard Von-Neumann computing models, are generally vulnerable to adversarial attacks. An adversarial attack is when input data to the network is carefully altered to guarantee misclassification. Fortunately, defensive strategies exist such as activation pruning, input discretization, and transfer functions. SNN's inherently have these capabilities, making them good and consistent solutions to adversarial attacks. SSN's additional add in temporal encoding and non linear behavior of neurons offer an additional layer of security, compensating for modifications to the input data.



Neuron devices and circuits based on various NVM technologies, such as PCM, RRAM, Spintronics, exhibiting IF characteristics.

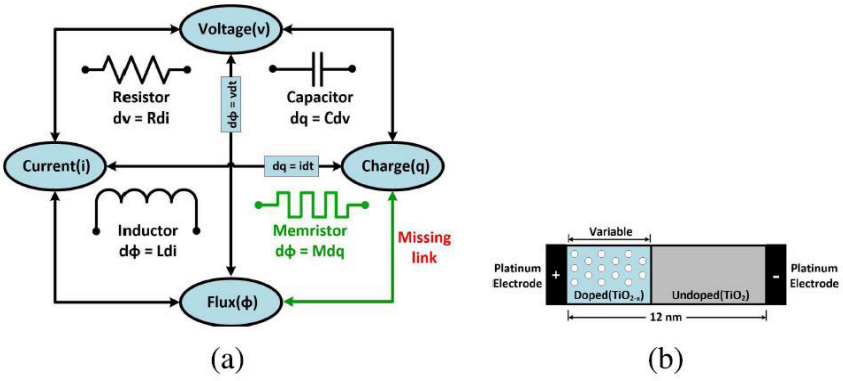
The author's then have a long discussion on neuromorphic hardware. It discusses the design requirements of making hardware for SNN's, chiefly utilizing temporal and spatial activation for SNN's. It goes into great detail on the foundational primitives used in such architecture. It discusses how CMOS based technologies can be deployed to serve as neurons,

and the special RAM configurations required to work with SNN's. In the figure taken from the paper below you can see three different artificial neurons, each with their own functionality:

The paper goes into great detail on these and many more components used for neuromorphic computers. Many of these components will need to be developed and deployed to our FPGA over the course of the project. After this there is a small discussion on the importance of hardware software codesign in neuromorphic computing. The paper then ends with a discussion on the future of research and a summary of all the content included in the paper

The Memristor

Since Leon Chua's theoretical prediction in 1971 [2], memristors have become a pivotal component in the evolution of computing technologies, especially after its groundbreaking implementation by HP Labs in 2008 [3]. Chua, in his work theorized the concept of the memristor, which completed the set of fundamental passive circuit elements alongside resistors, capacitors, and inductors, introducing a device capable of storing information in its resistance state. Unlike conventional elements, memristors can retain their resistance levels without power, making them highly valuable for non-volatile memory solutions.



(a) Memristor, the fourth circuit element (b) The design of memristor developed at HP Labs

The operational principle of a memristor is based on the modulation of resistance as a function of the history of current passing through it. This property allows memristors to "remember" their past states, essentially storing data in the form of resistance levels. Memristors consist of a thin film of a dielectric material sandwiched between two metal electrodes. The model developed by HP used titanium dioxide (TiO_2), partially doped with oxygen vacancies, placed between platinum electrodes [3]. As voltage is applied, the width of a doped region is modified within the memristor structure (TiO_{2-x}/TiO_2), altering its resistance [3].

Beyond memory storage, memristors are capable of performing logic operations directly within memory arrays, leading to the development of in-memory computing. This architecture marks a significant departure from traditional von Neumann systems, where memory and processing

units are physically separated. The inherent ability of memristors to perform both storage and computation within the same device can drastically reduce latency and bandwidth bottlenecks, making them critical for future computing systems, especially as demand for high-performance and energy-efficient technologies grows [1]. However, the most crucial drawback also results from its pioneering material design, the lack of industry standard CMOS design capabilities.

CMOS Compatibility

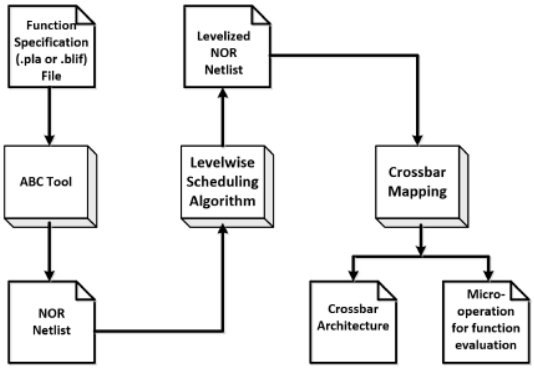
Over the decades several logic design styles compatible with CMOS design have been developed leveraging the advances in VLSI technology. The *IMPLY* methodology utilizes two memristors and one resistor to realize the material, as cited in [4]-[6]. *Memristor-Aided loGIC (MAGIC)* design style, on the other hand, is based on the principle of material implication through various logic gates, developed in [7] and [8], primarily supporting the mapping of NOR and NOT logic gates to memristor crossbar arrays [5][9]. Sophisticated control of voltage drivers along rows and columns of the crossbar array results in the implementation of microinstruction architecture, detailed by Talati et al. [9].

The behavior of the MAGIC NOR gate based on different input values can be summarized as follows [9]:

- If both inputs, in1 and in2, are 0, the voltage across the output memristor remains below its threshold voltage, so its resistance state remains unchanged, representing logic 1.
- If either of the input memristors is set to 1, the voltage across the output memristor exceeds the threshold voltage, causing the memristor to switch to logic 0.

By aligning memristor materials and processes with the standard CMOS fabrication workflow, memristors can be layered onto the same silicon substrate used for CMOS transistors, typically as part of the back-end-of-line (BEOL) fabrication process Li et al. [10].

Since any multi-output logic gate design can be converted into a NOR-NOT netlist, the MAGIC synthesis scheme works on this new NOR/NOT netlist subjecting it to level-wise scheduling. Afterwards, a crossbar mapping tool systematically allocates the arranged NOT/NOR netlist to the crossbar array, proceeding level by level. Reducing, maximum number of gates, G_{max} results in fewer rows required within the crossbar array for evaluating the gates [11][8]. Additionally, the tool generates a series of micro-operations, including the specific voltages that need to be applied to the rows and columns for evaluation [11][8]. These micro-operations are then managed and executed by a controller.



Synthesis flow from functional definition to micro-operations

Conclusion

Ultimately *Exploring Neuromorphic Computing Based on Spiking Neural Networks: Algorithms to Hardware* will serve as an exceptional resource for our FPGA implementation of a neuromorphic processor. It goes into detail on hardware and software implementations of neuromorphic computing that will serve as the bedrock of our project work. Similarly, the knowledge of neuron behavior at the silicon level given to us by studying memristors and CMOS logic used in neuromorphic computing will enable us to properly test our project's implementation and interpret the results.

References

[1] Rathi, N., Chakraborty, I., Kosta, A., Sengupta, A., Ankit, A., Panda, P., Roy, K., Nitin RathiSchool of Electrical and Computer Engineering, P. U., Indranil ChakrabortySchool of Electrical and Computer Engineering, P. U., Adarsh KostaSchool of Electrical and Computer Engineering, P. U., Abhronil SenguptaSchool of Electrical Engineering and Computer Science, P. S. U., Aayush AnkitSchool of Electrical and Computer Engineering, P. U., Priyadarshini PandaElectrical Engineering, Y. U., & Kaushik RoySchool of Electrical and Computer Engineering, P. U. (2023, March 2). *Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware*. ACM Computing Surveys. https://dl.acm.org/doi/full/10.1145/3571155#sec-1

[2] L. O. Chua, "*Memristor—The missing circuit element,*" IEEE Trans. Circuit Theory, vol. CT-18, no. 5, pp. 507–519, Sep. 1971.

[3] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "*The missing memristor found,*" Nature, vol. 453, no. 7191, pp. 80–83, May 2008.

[4] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams. *Memristive switches enable stateful logic operations via material implication*. Nature, 464(7290):873–876, 2010.

[5] S. Chakraborti, P. V. Chowdhary, K. Datta, and I. Sengupta. *BDD based synthesis of Boolean functions using memristors*. In 9th International Design and Test Symposium (IDT), pages 136–141, December 2014.

[6] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser. *Memristor-based material implication (IMPLY) logic: Design principles and methodologies.* IEEE Transactions on VLSI Systems, 22(10):2054–2066, October 2014.

[7] S. Kvatinsky et al., "*MAGIC—Memristor-aided logic,*" IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 61, no. 11, pp. 895–899, Nov. 2014

[8] R. Gharpinde, P. L. Thangkhiew, K. Datta, and I. Sengupta, "*A scalable in-memory logic synthesis approach using memristor crossbar,*" IEEE Trans. Very Large Scale Integr. (VLSI) Syst., to be published, doi: 10.1109/TVLSI.2017.2763171.

[9] N. Talati, S. Gupta, P. Mane, and S. Kvatinsky, "*Logic design within memristive memories using Memristor-Aided loGIC (MAGIC),*" IEEE Trans. Nanotechnol., vol. 15, no. 4, pp. 635–650, Jul. 2016.

[10] Li, H., Lee, M. H., et al. (2018). *CMOS-compatible fabrication of memristive devices for non-volatile memory and computing applications*. Materials Research Express, 5(12), 125026.

[11] P. L. Thangkhiew, R. Gharpinde and K. Datta, "*Efficient Mapping of Boolean Functions to Memristor Crossbar Using MAGIC NOR Gates,*" in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 8, pp. 2466-2476, Aug. 2018, doi: 10.1109/TCSI.2018.2792474.