Science Journal of Circuits, Systems and Signal Processing
2018; 7(2): 68-73
http://www.sciencepublishinggroup.com/j/cssp
doi: 10.11648/j.cssp.20180702.14
ISSN: 2326-9065 (Print); ISSN: 2326-9073 (Online)

Use of a Hysteresis Loop Activation Function to Enable an Analog Perceptron to Gain Memory

William Brickner, Muhammad Sana Ullah

Department of Electrical and Computer Engineering, Florida Polytechnic University, Lakeland, USA

Email address:
williambrickner0186@floridapoly.edu (W. Brickner), mullah@floridapoly.edu (M. S. Ullah)

To cite this article:
William Brickner, Muhammad Sana Ullah. Use of a Hysteresis Loop Activation Function to Enable an Analog Perceptron to Gain Memory. *Science Journal of Circuits, Systems and Signal Processing*. Vol. 7, No. 2, 2018, pp. 68-73. doi: 10.11648/j.cssp.20180702.14

Received: May 17, 2018; **Accepted**: June 14, 2018; **Published**: July 10, 2018

Abstract: With the advent of memristors, analog artificial neural networks are closer than ever. Neural computing is growing as a topic of research. In the context of analog artificial neural networks, the purpose of this research is to verify that a perceptron could gain a discrete memory from implementing a hysteresis loop in the activation function. The discrete memory is represented by the difference path of the hysteresis activation function that took from logic 1 to logic 0. To write to the memory, the input to the hysteresis loop would have to exceed threshold. To read the stored value, the input would have to be between the thresholds of the hysteresis function. In order to verify the perceptron's memory, a network with manually chosen weights is selected which acts as a shift register. The components of this network are assembled in a circuit simulation program. Functionally, the network receives two inputs: a data signal and an enable signal. The output of the network is a time-shifted version of previous input signals. A system whose output is a time-shifted version of the previous inputs is considered to have memory.

Keywords: Artificial Neural Network, Recurrent Neural Network, Memristors, Hysteresis Loop Activation Function, Analog Computing, Neural Computing, Long Short-Term Memory

1. Introduction

Artificial neural networks (ANNs) have been a known data structure in computing since the late 1940's [1]. Until the advent of stronger processing in the late 1990's, computers were not able to train at usable rates and/or solve problems more effectively than expert systems could solve [2]. Along with properly labeled data for a specific application, the current bottleneck in neural networks is the speed at which they are able to train. One contributing factor to this is that operations have distributed to a general purpose computing devices such as GPU's or CPU's. Even though these processing units are optimized in and of themselves, there is an overhead associated with being able to distribute labor to these devices and ensuring that the output of these devices align discretely with an internal clock signal. In application, specific hardware generality is lost for the gain of speed.

To measure the effectiveness of a change to the improvement of an algorithm, time complexity is the metric that used in CCT [3]. Reducing a problem's time complexity by performing operations in parallel is a technique that used in the bitonic sort [4]. Bitonic sort has algorithmic time complexity of O ($\log^2 n$) when measured with parallel time complexity [4]. This is a faster time than what considered the fastest sort because all of the operations are being executed in parallel [4]. The fastest average case sort for an unordered list in linear time complexity is considered to be quicksort with a time complexity of O (n log (n)). Reducing the algorithmic complexity of forward and back propagation by distributing the labor would allow to unprecedented sizes for scaling networks. Scaling would be able to occur in time to train the network. It would be less dependent on the size of the network and more dependent on the ability to collect labeled data due to the difference in algorithmic complexity. This is because each operation would distribute in such a way that the drudgery can do in parallel. In addition, each component executing the operation would execute substantially faster. The idea would be to take these operations are currently being performed virtually and accelerate them by distributing the labor with application specific perceptron hardware. This would revolutionize the neural network by allowing a network to be scaled larger than ever before.

The limits on an ANN are in large part of the lack of circuitry in existence, which functions as weights in the

network that can change [22]. With the change of weights, there is still a question on how precisely to set the analog weights. If the weights can be precisely set, it becomes a question of how to choose the values to set the weights too. Currently, there is a device called memristor that has explored for its potential to provide weights in an analog neural network that can change. Leon Chua [5] first theorized the memristor to exist based on symmetry of physics in the 1971 paper. It is not until 2009 that researchers at HP labs are able to document the discovery of a memristor in the nature journal [6], [10]. A memristor has the property that its resistance is dependent upon the total amount of current that has passed through the device [7], [8] and [10]. The voltage transfer characteristic of the device has the property of Lissajous hysteresis [9]. These properties has combined in an analog neural network to function as changeable weights using a memristor bridge [12]. It has also shown that analog neural networks can back propagate these weights using a random weight change algorithm [13]. A memristor could be fabricated at the nanometer scale with operating speeds of 2ns by several reported techniques [11], [14], [15], [16] and [17]. These properties allow memristors to apply in such applications as a memristor crossbar array or MRAM [18] though networks of memristors that have issues with convergence [19].

Recurrent Neural networks (RNNs) have been around since the early 1990's [24]. The idea being that there is some sort of feedback that could give a network memory to learn to solve problems that require past information such as speech recognition [25]. In 1997 with a variation on the RNN, Long Short-Term Memory (LSTM) proposed in [20]. LSTM networks became the industry standard for training networks with memory. As recently as 2016 attempts to simplify the ruleset of LSTM's into Gated Recurrent Units [21]. The general idea is that there is a memory cell, which overwritten, based on a combination of input signals. In this case, a shift register implemented in analog in the form of a Schmitt Trigger for the activation function. The Schmitt trigger shifts the register that is chosen because of the simplicity to implement in hardware.

It is in a sense that using a circuit with a hysteresis loop has a memory. This paper explores the implications that using an activation function with hysteresis in a perceptron. Therefore, in this research paper, the implementation of shift register with Multisim presented in Section 2 followed by a conclusion in Section 3.

2. Implementation of the Proposed Shift Register

The components to assemble the shift register have chosen for their simplicity to implement in Multisim. All data have collected in Multisim.

2.1. Perceptron

A perceptron is the most basic element of an ANN. A perceptron takes a weighted sum of inputs and runs then through an activation function to feed forward to another perceptron layer. A functional diagram of the perceptron that used in this research is shown in Figure 1. In Figure 1, a perceptron with a Schmitt Trigger as the Activation function is shown. The Schmitt Trigger and by extension, the perceptron has memory because of its hysteresis. The circuit components to assemble the perceptron in Figure 1 are a difference op-amp to sum the weighted inputs, resistors to weight the inputs, and a Schmitt Trigger op-amp for the activation function. In this case, the layers of the network only have a single perceptron and pass two inputs, a square data signal, and a sinusoidal enable signal. The op-amps are chosen as the basis for the difference function and the activation function because they are simple to implement having less than five (5) components each. Technique based on complementary metal oxide semiconductor (CMOS) has used to implement a summation of input signals and a Schmitt trigger [23]. However, the CMOS implementation require a more complex arrangement of parts so in this case power efficiency was forgone in the wake of circuit simplicity.

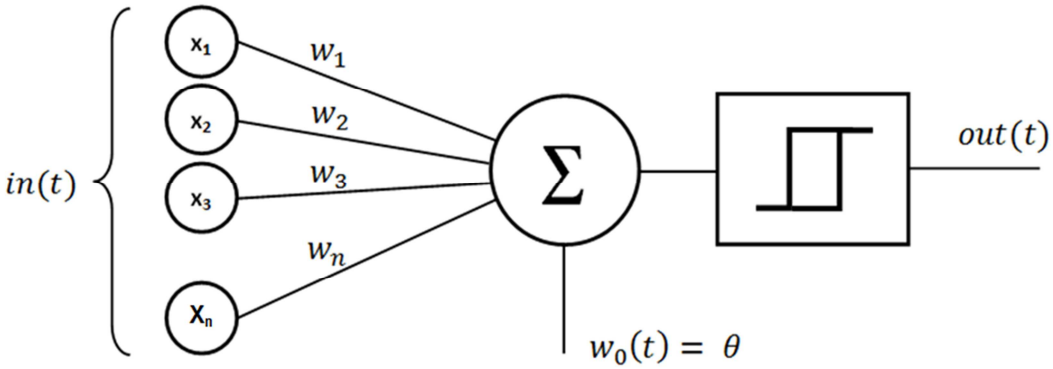

Figure 1. *Perceptron with Schmitt Trigger.*

2.2. Schmitt Trigger

A Schmitt Trigger is a device used in circuits whose characteristic transfer function has the property of hysteresis (see Figure 2). The Schmitt Trigger is used to provide a buffer to noisy input because of difference in energy to turn it on and off [23]. Hysteresis is a transfer function that has memory because it depends on the previous inputs which has a path

from one (1) to zero (0). Specifically, if the previous input has exceeded a certain voltage threshold, a value could be considered to overwrite the data in the trigger. As a result, a Schmitt trigger can function as a memory cell. A graph of the input-output characteristic of a Schmitt Trigger is shown in Figure 3. In Multisim, components are selected that produced a transfer function with the following properties: *Vo+ is 4.1 V, Vo- is -4.1V, V1 is -2.7V, V2 is 2.7V*. This graph shows the voltage input-output characteristics of a Schmitt trigger. Notice that the output of the Schmitt Trigger does not change until the input is about ±2.7 V. Functionally, the memory cell can be set when the input voltage exceeds either threshold. The memory can be read by inputting a voltage less than threshold voltage. In this case, zero (0) voltage is within both thresholds and can read the contents of the memory cell.

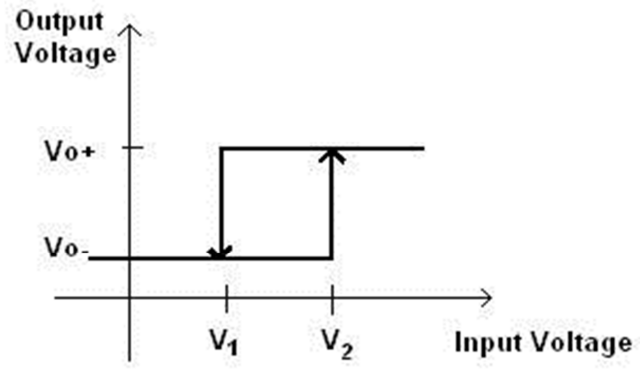

Figure 2. *Voltage Transfer characteristic of a Schmitt trigger.*

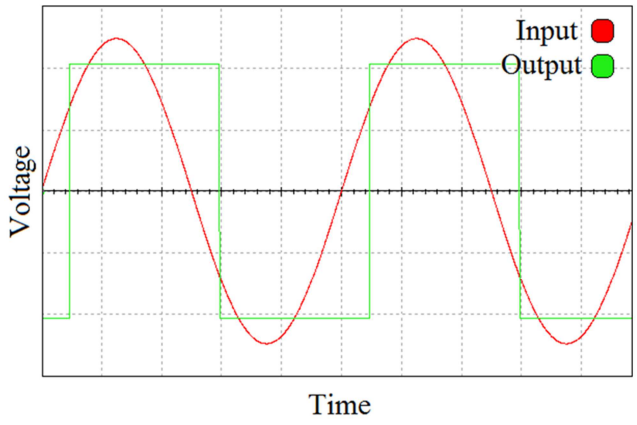

Figure 3. *Input-Output characteristics of Schmitt Trigger.*

2.3. Schmitt Trigger as a Memory Cell

In this case, a memory cell is chosen to have the properties that data can be input to the cell while there is an active an enable signal. Then once data is written to the cell it should be able to read. The enable signal is a sine wave and the data signal is a square wave.

2.3.1. Data Input

A Schmitt Trigger can have data to be written to it. By taking advantage of the property, the input data has to exceed a threshold to set the hysteresis function to follow either the high voltage path (logical 1) or the low voltage path (logical 0). When it is on either of these paths there could be considered to be a form of discrete memory. To accomplish this an input data signal is implemented in such a way that by its self it never causes the Schmitt Trigger to exceed its voltage threshold. This voltage does not exceed the Schmitt trigger threshold that is chosen to be equal to the amplitude of a second, periodic, enable signal. This ensures maintaining a control of overwriting of the data in the memory cell while the enable signal is not active. This also has a secondary effect of delaying the input until the next corresponding peak of the enable signal. An example of this behavior can be found in Figure 4. The summed input to a perceptron is displayed alongside the output of the Schmitt Trigger. In region I, the input to the Trigger includes both the sinusoidal enable signal and the square data signal so data is able to be written to the trigger and the output changes. In region II, the enable signal is not included alongside the data signal and thus no data is written to the Schmitt Trigger. In region III, the enable signal is turned on again and data can be written to the trigger again.

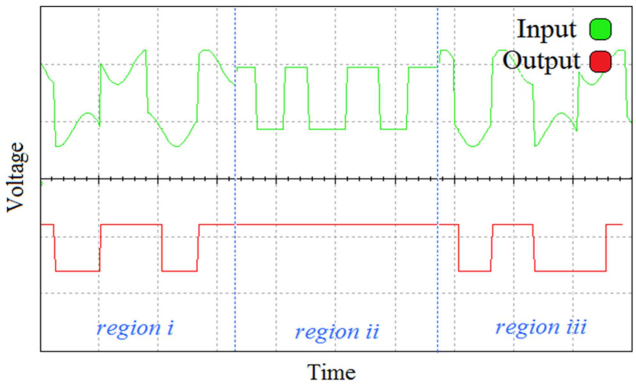

Figure 4. *Exclusive writing depending on the enable signal.*

2.3.2. Data Reading

While the enable signal does not couple with the data signal to exceed the threshold voltage to write new data to the trigger, the data present in the trigger can be read. The value that will be read corresponds to the previous value stored in the trigger.

2.4. Shift Register Network

A shift register with a single input can remember the contents of that input by first storing the value that is input into some sort of memory cell. Then, before a new value is to put in, the shift register takes the first stored value and passes it to a second memory cell. This movement of data from one cell to another is called a shift. A shift register can be useful when attempting to send large amounts of data over a single line because multiple values can pass over a single data stream by delaying each bit of information that is desired to be sent. A block diagram of the network is in Figure 5, where P is stand for perceptron. To make the perceptrons into a shift register the memory cells are connected in series. The propagation of the input can be delayed from one layer to the next by delaying the peak of the enable signal feeding into the second layer by 180 degrees. In this case, since the signal is sinusoidal, it could be choose to subtract the signal instead of adding it to delay it

by 180 degrees. This is the primary mechanism by which this shift register network is operated. Each register shifts with time delay equal to one-half the period of the enable signal. This period can be reduced by shifting the enable signal by less than 180 degrees using a different mechanism to delay it. This phenomenon can be observed in Figure 6. In Figure 7, the enable signal is turned on intermittently. It is noted that the registers do not propagate information until the enable signal is active. It also noted that the behavior continues independent of the amount of time the enable is cut off.

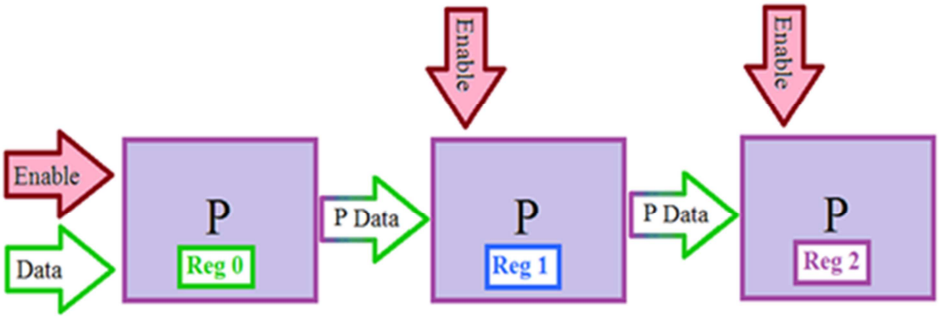

Figure 5. *Shift Register network diagram.*

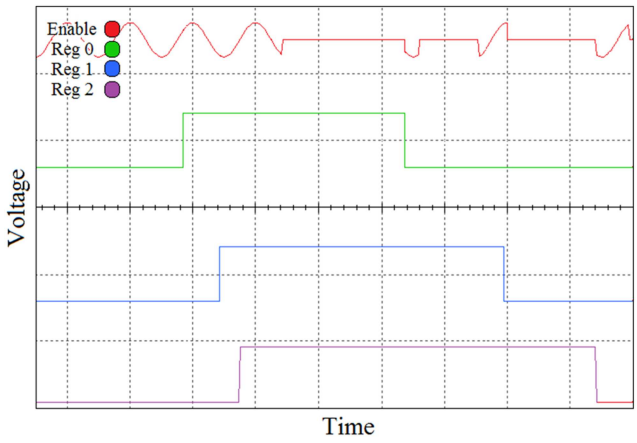

Figure 6. *Three Layer Shift Register Network's behavior by shifting the enable signal.*

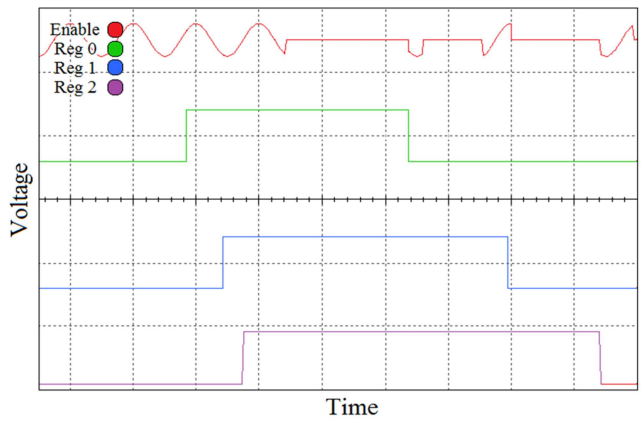

Figure 7. *Delaying the three-layer shift register network's behavior by cutting off the enable signal.*

2.5. Making the Network Turing Complete

Since the network has memory, it is possible to use cellular automata [27] and implementing rule 110 [26] to make the network can be made to be Turing complete.

To do this first, an infinite amount of inputs would be required instead of a single input. Then these inputs would feed at a 1:1 ratio into the first layer of hysteresis perceptrons. There would then not be an immediate layer of hysteresis perceptrons, instead there would be two layers that would implement rule 110. To implement rule 110 in perceptron, it would first be understood that any perceptron can be manually substituted for a logic gate [28]. Then a simplified logic function for rule 110 would be determined using Karnaugh maps. Then, using the knowledge of translating logic gates into perceptrons with the knowledge of a set of logic gates that correspond to rule 110, the first hysteresis layer would be connected to the second hysteresis layer.

This process could be redone multiple times to create a Turing complete hysteresis perceptron network. This is done with a memristor spiking network by Pickett and Williams in 2013 [29] though it should be noted that they used rule 134 instead of rule 110. Effectively a Turing complete version of the network would verify that the system had memory equally as effectively as creating a shift register, although in more steps and using far more components.

The implications of the network being Turing complete have less to do with the raw power of the system and more to do with not denying the ability of the network to do certain tasks.

2.6. Implementing a Hysteresis Function to Software

To implement the hysteresis function in software one approach might be to create a piecewise function. This piecewise function would choose an output depending on previous inputs. This would in effect create a hysteresis function. A drawback would be the increase in memory requirement of each perceptron because there would have to be a binary bit to take note of which path from 1 to 0 the activation function is taking. Although there would only need to be an increase in memory by 1 bit. A common activation function used in ANN's is the sigmoid function. One possible implementation of piecewise hysteresis function would be to linearly shift the sigmoid function to create a hysteresis loop. A version of this is graphed in Figure 8.

3. Conclusion

A system with memory is able to reproduce previous input signals to the system with a time delay. In this case, the system is a series of perceptrons whose activation function is a hysteresis loop. This series of perceptrons can be considered a shift register. An input data signal is able to propagate through this network. The speed is controlled by an enable signal at which it is propagated. Since the output of the network is a time-shifted version of the input, the hysteresis activation function perceptrons successfully created a system that has memory. While it is possible to use this technology to create a shift register, perceptrons are not a practical implementation of a shift register because they do not provide the most power efficient, smallest, or highest frequency implementation of a memory cell. The advantage of using perceptrons as memory cells is that with the aid of machine learning methods, a system can be created whose output is dependent upon an arbitrarily large rule set that may be difficult for non-machines to parse. In addition, if this hysteresis activation function network could be implemented in analog, it is conceivable that there could be an exponential increase in speed of training of the network, as opposed to traditional software implementations. This would be possible to provide several gaps in the feasibility of implementing a neural network in analog that can be filled. A large portion of these gaps could conceivably be filled by emerging memristor technologies to bring us past the microchip age into the neural chip age.

Acknowledgements

First, the author would like to acknowledge his peer, Josiah Hunsinger, for the great advice and even greater times we had discussed together regarding this paper.

The author would then like to thank the professors at Florida Polytechnic University. Mainly Dr. Richard Matyi, but also Dr. Cristopher Coughlin, Dr. David Foster, Dr. Harish Chintakunta, and Dr. Christina Drake for their advices to improve this paper.

References

[1] D. O. Hebb, The organization of Behavior: A Neuropsychological Theory. First Edition, New York, 2012.

[2] K. Anjaneyulu, “Deep Blue beats Kasparov in a rematch,” Resonance, vol. 2, no. 7, pp. 89-90, 1997.

[3] W. Dean, "Computational Complexity Theory (Stanford Encyclopedia of Philosophy)", Plato.stanford.edu, 2017.

[4] D. Nassimi and S. Sahni, "Bitonic Sort on a Mesh-Connected Parallel Computer", IEEE Transactions on Computers, vol. C-27, no. 1, pp. 2-7, 1979.

[5] L. Chua, "Memristor-The missing circuit element," IEEE Transactions on Circuit Theory, vol. 18, no. 5, pp. 507-519, 1971.

[6] D. Strukov, G. Snider, D. Stewart, and R. Williams, “The missing memristor found,” Nature, vol. 453, pp. 80-83, 2008.

[7] H. Kim, M. Sah, C. Yang, S. Cho and L. Chua, “Memristor Emulator for Memristor Circuit Applications,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 59, no. 10, pp. 2422-2431, 2012.

[8] Q. Li, A. Serb, T. Prodromakis, and H. Xu, “A Memristor SPICE Model Accounting for Synaptic Activity Dependence,” PLOS ONE, vol. 10, no. 3, pp. 1-12, 2015.

[9] M. Kumar, “Memristor - Why Do We Have to Know About It?” IETE Technical Review, vol. 26, no. 1, pp. 1-6, 2009.

[10] R. Williams, "How We Found The Missing Memristor", IEEE Spectrum, vol. 45, no. 12, pp. 28-35, 2008.

[11] C. Yakopcic, R. Hasan and T. Taha, “Hybrid crossbar architecture for a memristor based cache,” Microelectronics Journal, vol. 46, no. 11, pp. 1020-1032, 2015.

[12] M. Sah, C. Yang, H. Kim and L. Chua, “A Voltage Mode Memristor Bridge Synaptic Circuit with Memristor Emulators,” Sensors, vol. 12, no. 12, pp. 3587-3604, 2012.

[13] S. Adhikari, H. Kim, R. Budhathoki, C. Yang and L. Chua, “A Circuit-Based Learning Architecture for Multilayer Neural Networks With Memristor Bridge Synapses,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 62, no. 1, pp. 215-223, 2015.

[14] J. Capulong, B. Briggs, S. Bishop, M. Hovish, R. Matyi and N. Cady, “Effect of Crystallinity on Endurance and Switching and Behavior on HfOx-based Resistive Memory Devices,” IEEE International Integrated Reliability Workshop Final Report (IRW), South Lake Tahoe, CA, USA, pp. 22-25, 14-18 October 2012.

[15] Z. Chew and L. Li, “A discrete memristor made of ZnO nanowires synthesized on printed circuit board,” Materials Letters, vol. 91, pp. 298-300, 2013.

[16] N. Duraisamy, N. Muhammad, H. Kim, J. Jo and K. Choi, “Fabrication of TiO2 thin film memristor device using electrohydrodynamic inkjet printing,” Thin Solid Films, vol. 520, no. 15, pp. 5070-5074, 2012.

[17] N. Mou and M. Tabib-Azar, “Photoreduction of Ag+ in $Ag/Ag_2S/Au$ memristor,” Applied Surface Science, vol. 340, pp. 138-142, 2015.

[18] M. Hu, H. Li, Y. Chen, Q. Wu, G. Rose and R. Linderman, “Memristor Crossbar-Based Neuromorphic Computing System: A Case Study,” IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 10, pp. 1864-1878, 2014.

[19] W. Wang, L. Li, H. Peng, J. Xiao and Y. Yang, “Synchronization control of memristor-based recurrent neural networks with perturbations,” Neural Networks, vol. 53, pp. 8-14, 2014.

[20] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[21] G. Zhou, J. Wu, C. Zhang and Z. Zhou, “Minimal gated unit for recurrent neural networks,” International Journal of Automation and Computing, vol. 13, no. 3, pp. 226-234, 2016.

[22] J. Hopfield, “The effectiveness of analogue ‘neural network’ hardware,” Network: Computation in Neural Systems, vol. 1, no. 1, pp. 27-40, 1990.

[23] O. Law and C. Salama, "GaAs Schmitt trigger memory cell design," IEEE Journal of Solid-State Circuits, vol. 31, no. 8, pp. 1190-1192, 1996.

[24] D. Dong and J. Hopfield, "Dynamic properties of neural networks with adapting synapses," Network: Computation in Neural Systems, vol. 3, no. 3, pp. 267-283, 1992.

[25] F. Beaufays, "The neural networks behind Google Voice transcription," Google Research Blog, 11 August 2015.

[26] W. Li and M. Nordahl, "Transient behavior of cellular automaton rule 110," Physics Letters A, vol. 166, no. 5-6, pp. 335-339, 1992.

[27] F. Berto and J. Tagliabue, "Cellular Automata-The Stanford Encyclopedia of Philosophy", Plato.stanford.edu, 2017.

[28] K. Swingler, "Lecture 2: Single Layer Perceptrons," University of Stirling, Scotland, UK, 2017.

Biography

William Brickner studied and recently graduate for the degree of Bachelor of Science in Computer Engineering at Florida Polytechnic University, Lakeland, FL. He is specialized on digital logic design. Now for his Master Degree, he attended the Colorado School of Mines where he is studying in the field of Electrical Engineering with Control Systems and Signal Processing. Besides, he is working as an associate engineer of hardware and signal processing at SpaceX in Irvine, CA. His research interest includes on analog neural network, digital systems design, control systems and signal processing. His dream is to make a fully functional analog neural network.

Muhammad Sana Ullah received the B.S. degree in Electrical and Electronic Engineering from Chittagong University of Engineering and Technology (CUET), Bangladesh in 2008 and M.S. degree in Electrical and Computer Engineering from Purdue University, Hammond, IN, USA in 2013. He has finished his Ph.D. degree in Electrical and Computer Engineering from the University of Missouri-Kansas City (UMKC), Kansas City, MO, USA in 2016 and joined as an Assistant Professor of Computer Engineering in the College of Engineering at Florida Polytechnic University, Lakeland, Florida. His research interests includes a relatively new methodology and nanotechnology for the next generation of computing and other micro- and nano-electronic applications.