Hardware Implementation of Artificial Neural Networks

Mats Forssell

***Abstract*—Artificial Neural Networks (ANNs) have long been used to solve complex machine learning problems (“deep learning”). The inherent distributed component of ANNs in both memory and computation means that implementing them directly in hardware can allow large gains when scaling the newtork sizes, since the von Neuman bottleneck (computations limited by memory access time) does not have to distributed architectures. Therefore both analog and digital circuits have been used to implement these circuits without the use of a general purpose CPU architecture. Promising results have been obtained for various circuit architectures, with commercial applications close by. Challenges remain in the scaling of these networks, since the number of synapses grows quadratically, making wiring difficult. Novel devices, in particular different memristor-like devices, can be used to more efficiently design and fabricate ANNs, although the reliability of the fabrication remains to be demonstrated.**

***Index Terms*—Aritifical Neural Network, Neuromorphic Computing, Memristor**

I. Introduction

ARTIFICIAL Neural Networks (ANNs) are a powerful paradigm in machine learning inspired by the computation and communication capabilities of the brain. As such they have been the basis for many powerful algorithms with applications in pattern recognition, memory, mapping, etc… Recently there has been a large push toward a hardware implementation of these networks in order to overcome the calculation complexity of software implementations: the power budget of the human brain is around 15W, and its computation capabilities range in the 10^{17} FLOPS[1] [1], far better than the best supercomputers. This has lead to an entire new class of circutis, dubbed *“neuromorphic circuits”*, which closer emulate the behavior of neurons: high network connectivity, simple base processing element, and distributed memory and computation. It is important to keep in mind that the purpose of these circuits is not to accurately reproduce the processing performed in the brain; rather, they are inspired by the brain and emulate some charcteristics of brain processing, but do so using very different elements.

A. Types of Artificial Neural Networks

From a structural perspective, ANNs can be divided into two main categories, *feed-forward networks*, in which the computation is performed in a layer-by-layer fashion from the input to the output of the network; and *recurrent networks* which have an interconnected network structure including cycles. The main application of the former class of networks is supervised classification performed by a perceptron algorithm.

[1]In a manner of speaking. Of course direct comparison with CPU processig power is not really meaningful.

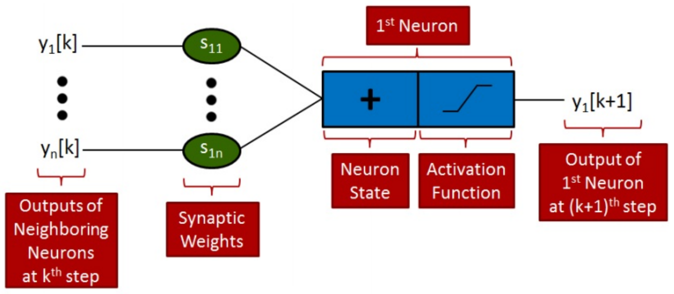


Fig. 1. Symbolic representation of a neuron-synapse model. Inputs from neighboring neurons are summed using the synaptic weights, and a nonlinear activation function then determines the output of the neuron [4].

The latter class of networks is more diverse and applications include self-organizing maps, associative memory (Hopfield netowork), Boltzmann networks, etc… [2] Finally, artificial neurons can also be used to implement digital-like logic using spikes, and therefore reproduce a universal Turing machine [3].

B. General Structure of ANNs

Although the final application may differ, the two components of ANNs remain the same: in analogy with biological systems, they are referred to as *neurons* and *synapses*, and correspond to the vertices and the edges of the graph respectively. The neuron-synapse model is shown in Fig. 1. The challenges in hardware implementation of the ANN are:

1) *Synapse*
   - Network wiring: Number of synapses scales qudratically with the number of neurons
   - Synaptic weight: Weights have to be defined with high precision in order to ensure proper convergence of the algorithms
   - Learning: synaptic weights have to be updateable
2) *Neuron*
   - Neuron state: Summation of weighed input must be performed
   - Activation function: Highly nonlinear function calculation

As ANNs scale in size the number of synapses grows quadratically for fully connected networks, which quickly becomes impractical to wire. Today’s integrated circuit fabrication techniques are essentially stacked 2D structures, further limiting the possibility of full interconnection. Therefore most implementations limit the wiring to some neighborhood of each neuron.

C. Learning Rules

The general operation of most ANNs involves a learning stage and a recall stage. During the learning stage the weights of the network are adjusted to fit the application at hand. In the case of the perceptron, this involves using the backpropagation algorithm on a classified training set; in the case of associative memory this involves setting the weights to ensure the desired memories act as local attractors. During the recall stage, new inputs are introduced and the network is left to equilibrate (single-pass feed-forward for the perceptron and evolution to equilibrium for associative memory for example). While the recall stage is always performed on the physical network itself, the learning stage can be performed in advance. There are three training strategies:

1) *Off-chip learning* involves performing the learning stage on a simulated network in software. This allows the training claculations to be performed faster and more accurately than could be done using the hardware network. However, manufacturing variation of the hardware is not taken into account.
2) *Chip-in-the-loop learning* involves both the hardware network and an external software computation. It relies on software to perform the learning algorithm but uses the hardware network to perform the computations. In the case of the backpropagation algorithm for example, the forward pass is performed by the network while the weight updates are performed in software. The precision of the calculations is therefore not limited by the hardware capabilities, while the actual behavior of the network is still taken into account.
3) *On-chip learning* Uses only the hardware chip to perform the learning. While slower and less precise in the weight calculations than the other two methods, this technique does not involve external manipulation in the learning stage. This makes it more realistic for embedded hardware, and networks designed with this approach in mind will be able to update their learning over time. However the design is inherently more complicated and less flexible, since the algorithm performing the learning has to be implemented in hardware.

II. CMOS Implementation

A. Digital Circuitry

ANNs can be implemented using CMOS digital circuits [5]–[8], which have the advantage of being easy to design and build. They rely on existing logic elements and can take full advantage of decades of advances in digital circuits. Synaptic weights can be implemented using digital memory cells, or even latches. The number of bits used to store the synaptic weights is critical to the accuracy of the algorithm, especially in the learning phase [2]. Although this can be mitigated in some cases by using off-chip learning (meaning the algorithm used to set the weights occurs on an external software having comparatively high precision, and the final result is then quantized to the chip precision), having an efficient implementation of the synaptic weights is important. The neuron state summation can be easily implemented as well using common multipliers and adder stages. However for large numbers of incoming neurons, the total number of these elements can grow to be non-negligible.

The activation function is typically more complicated to implement given that it has to be highly nonlinear by definition. While a single threshold function can easily be implemented, its capabilities are rather limited. On the other hand, a more complex activation function, such as a sigmoid function, requires look-up tables; these slow down the computations considerably and require significant power and area if a good precision is desired.

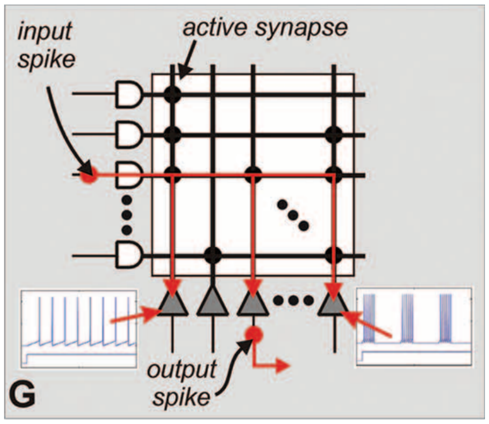
Although adapting CMOS digital logic for ANNs leads to fairly simple designs, the result is inherently not power and area optimized. However, another advantage of a CMOS implementation is that it can easily be integrated alongside the standard circuitry, which can even be fabricated using the same CMOS process. Recent applications have been demonstrated, such as the Synapse chip by IBM [8] (Fig. 2) or the Zeroth processor by Qualcomm [9].

Figure 2 shows the structure of the IBM Synapse chip. Computations are performed on a local core, but spikes can be targetted to any neuron on the chip. In order to lower power consumption, event-driven communication is used, meaning that only spikes are communicated; this results in active power proportional to spiking activity. However a global synchronization clock has to be maintained throughout the chip; the global clock rate is 1 kHz (about equivalent to the "clock rate" of the brain) Each core requires 1.2 million transistors, equivalent to less than 20 transistors per synapse.

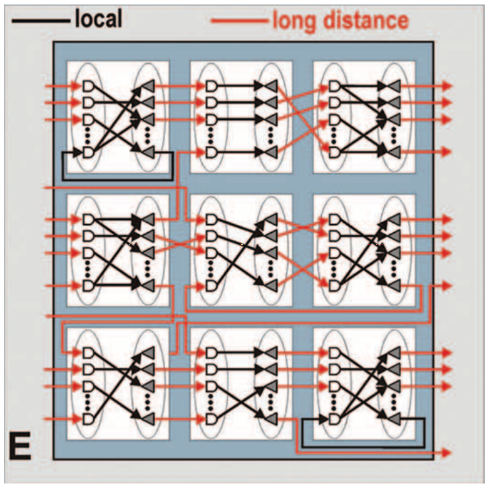
B. Analog Circuitry

In general, integrated circuit design is much more complicated in the case of analog circuits compared to digital circuits, especially when scaling to large number of gates. This is no different in the case of ANN circuits: while analog design can lead to more power and area efficient circuits [10], [11] (which is not typically the case for traditional ICs, where the footprint of the analog portion is dominating), their design cannot be automated to the same extent as that of digital circuits. Because it is inherently more difficult to scale the design of analog circuits, general-purpose circuits are more difficult to achieve; however for a given application, analog circuits can be significantly more efficient than their digital counterparts.

The synaptic weight can be implemented by storing a digital weight, as is done for the digital synapse. Performing the computation, will then require ADC/DAC elements. Not only can this introduce an undesirable delay in computation, the power and area scaling will also be dependent on the precision given to the synaptic weights. The alternative is to store the weights using analog elements, such as resistors or capacitors, which can be done directly if the weights are to be fixed by design. Other methods exist to store digitally programmable



(a)



(b)

Fig. 2. Functional circuit showing (a) one neuromorphic core and (b) a multiple-core connection of IBM's Synpase chip. The core has 256 fully connected neurons, and the chip consists of an array of 64x64 fully connected cores [8].

weights using reconfigurable analog elements [10] more efficiently than storing the weights in digital memory. This will still require ADC/DACs for the learning step (i.e. updating the weights).

Summation of the neuron inputs is usually not an issue in an analog circuit, since the neuron input and outputs will be represented as either currents or voltages. Currents are summed by joining the branches in parallel, while voltages are summed by joining them in series; in both cases, no extra element is required to perform the computation. An alternative is the class of *oscillatory neural networks*, which use phase to represent the neuron state [12]. In this case, a phase-locked-loop (PLL) is required to sum the neuron inputs.

The activation function is usually implemented using an amplifier, which presents strong nonlinearity in saturation regime. Essentially arbitrary activation functions can be built using relatively simple circuits.

The manufacturing variations inherent to anaolg circuits limits the achievable accuracy of analog ANNs. This means that in order to achieve a certain level of precision, circuits must be made large enough to limit the effect of manufacturing tolerance. However neural networks implementing associative memory are known for their fault tolerance, meaning they can allow some variation in the basic circuit elements.

III. Post-CMOS Implementation

Implementing ANNs using CMOS processes in both digital and analog case suffers from significant drawbacks, especially when one attempts to scale the networks to large numbers of neurons and synapses. As was already explained, the number of synapses required soon makes the circuit wiring impossible, especially for complete graphs (of which the Hopfield network is an example). There are however several promising novel devices which might enable easier implementation of the neural network synapse.

Chief among these is the memristor, long-anticipated fourth circuit element [13], [14] which is currently approaching commercial feasibility. A simplified view of the memristor consists of seeing it as a two-port programmable resistance. It is clear then that if we use resistance to encode synaptic weights, the memristor will enable an efficient programmable analog synapse without requiring heavy additional digital circuitry [15].

Over the past few years, several types of memristive-like devices have appeared, such as oxide thin-film, spin transfer torque, phase change materials, or optically gated field effect transistors. Better yet, the memristive effect in many of these devices occurs at the nanoscale, which is promising with regards to area scaling. The challenge of these devices is the reliability in their fabrication, and integrationg with standard CMOS manufacturing process

RRAM

Resistive Random Acess Memory (RRAM) is the name given to the non-volatile memory application of the memristor. It is the most product-ready of all the novel ANN devices, with commercial applications to be launched within the year. RRAMS are implemented using thin-film oxide structures which change their resistance after the application of a temporary voltage; the switching energy is only a few pJ. Moreover, spike-timing dependent plasticity, a well knwon feature of the brain (Hebb's law), can be implemented using RRAMs [16], creating an inherent memory/synapse link. Weight updating happens automatically in the network, with only one transistor and one memristor required at every synapse; this is one of the most compact possible implementations for ANNs.

IV. Logic circuits

The neuron-synapse pair is the basic element of Artificial Neural Networks. However in some applications, the neuron can be implemented in a stand-alone fashion and used as the basic building block of logic circuits. In this case it is also called *neuristor*, since it replaces the transistor used in traditional CMOS circuits. The logic circuits can be implemented using spiking devices [3], or oscillatory devices [17] (Fig. 3).

Universal Turing machines can be constructed based on the gates in Fig. 3. Therefore neuromorphic hardware can

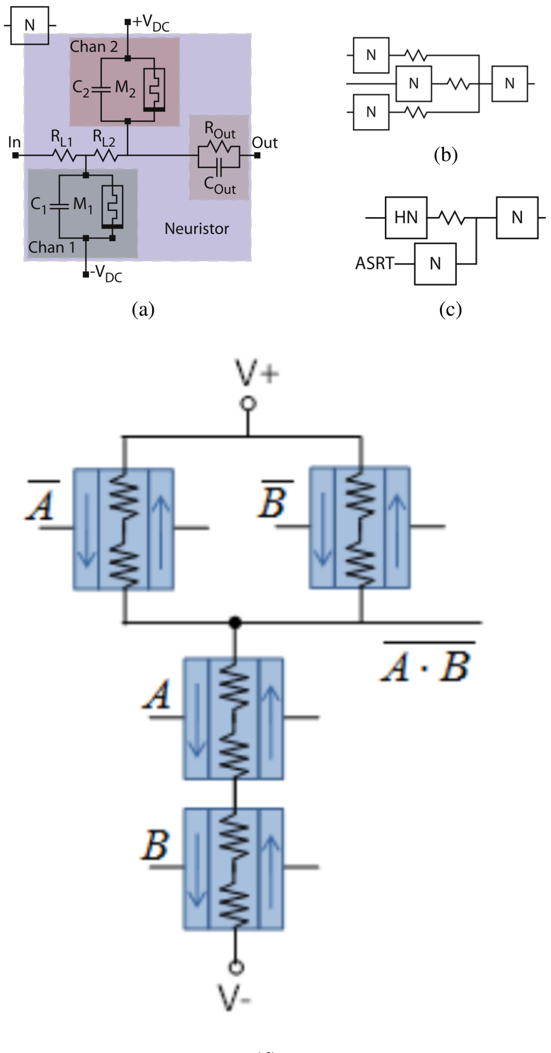


Fig. 3. Neuristor implementation and basic logic gates. (a) Spiking neuristor design based on memristor circuit, (b) AND logic gate, (c) NOT logic gate. The assertion is required to ensure timing of spikes. (d) Oscillatory NAND gate based on magnetic logic gate.

be used to perform general computations, without requiring an additional CMOS logic layer. The ANNs based on spikes or oscillations could be directly integrated with these neuromorphic logic circuits, without requiring a costly transcription of instructions from standard CMOS logic (TTL) into the neuromorphic ANN.

V. Conclusion

Hardware implementation of ANNs has been succesfully achieved using either analog or digital neuron-synapse circuits. Future devices can make the ANN design and fabrication more efficient. The full power of hardware ANNs has not been seen yet, but with the coming release of commercial chips implementing arbitrary neural networks, more efficient algorithms will no doubt be realized in those domains where neural networks are known to dramatically improve performance (pattern recognition, associative memory, etc…).

References

[1] A. H. Marblestone, B. M. Zamft, Y. G. Maguire, M. G. Shapiro, T. R. Cybulski, J. I. Glaser, D. Amodei, P. B. Stranges, R. Kalhor, D. A. Dalrymple *et al.*, “Physical principles for scalable neural recording,” *Frontiers in Computational Neuroscience*, vol. 7, 2013.

[2] P. Moerland and E. Fiesler, “Neural network adaptations to hardware implementations,” *Handbook of Neural Computation*, vol. 1, p. 2, 1997.

[3] M. D. Pickett and R. S. Williams, “Phase transitions enable computational universality in neuristor-based cellular automata,” *Nanotechnology*, vol. 24, no. 38, p. 384002, 2013.

[4] V. Calayir, T. Jackson, A. Tazzoli, G. Piazza, and L. Pileggi, “Neurocomputing and associative memories based on ovenized aluminum nitride resonators,” in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–8.

[5] R. C. Frye, E. A. Rietman, and C. C. Wong, “Back-propagation learning and nonidealities in analog neural network hardware,” *Neural Networks, IEEE Transactions on*, vol. 2, no. 1, pp. 110–117, 1991.

[6] S. Jung and S. S. Kim, “Hardware implementation of a real-time neural network controller with a dsp and an fpga for nonlinear systems,” *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 1, pp. 265–271, 2007.

[7] H. Hikawa, “{FPGA} implementation of self organizing map with digital phase locked loops,” *Neural Networks*, vol. 18, no. 56, pp. 514 – 522, 2005, {IJCNN} 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608005001103

[8] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, 2014. [Online]. Available: http://www.sciencemag.org/content/345/6197/668.abstract

[9] V. Aparin and J. A. Levin, “Methods and systems for cmos implementation of neuron synapse,” Apr. 8 2014, uS Patent 8,694,452.

[10] P. W. Hollis and J. J. Paulos, “Artificial neural networks using mos analog multipliers,” *Solid-State Circuits, IEEE Journal of*, vol. 25, no. 3, pp. 849–855, 1990.

[11] G. Indiveri and T. K. Horiuchi, “Frontiers in neuromorphic engineering,” *Frontiers in Neuroscience*, vol. 5, 2011.

[12] F. C. Hoppensteadt and E. M. Izhikevich, “Pattern recognition via synchronization in phase-locked loop neural networks,” *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 734–738, 2000.

[13] L. O. Chua, “Memristor-the missing circuit element,” *Circuit Theory, IEEE Transactions on*, vol. 18, no. 5, pp. 507–519, 1971.

[14] 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, 2008.

[15] Y. V. Pershin and M. Di Ventra, “Experimental demonstration of associative memory with memristive neural networks,” *Neural Networks*, vol. 23, no. 7, pp. 881–886, 2010.

[16] S. Ambrogio, S. Balatti, F. Nardi, S. Facchinetti, and D. Ielmini, “Spike-timing dependent plasticity in a transistor-selected resistive switching memory,” *Nanotechnology*, vol. 24, no. 38, p. 384012, 2013.

[17] V. Calayir and L. Pileggi, “Fully-digital oscillatory associative memories enabled by non-volatile logic,” in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–6.