Physical Computing for Hopfield Networks on a Reconfigurable Analog IC

Pranav O. Mathews and Jennifer O. Hasler
Department of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, Georgia 30332
Email: jennifer.hasler@ece.gatech.edu

***Abstract*—This paper discusses physical computing for solving optimization problems using a Hopfield network built on a Field Programmable Analog Array (FPAA). A core Hopfield circuit is presented that uses a programmable Vector-Matrix-Multiply (VMM) and Transconductance Amplifier (TA). The circuit dynamics of the VMM and effects of mismatch are discussed. The analog Hopfield network is evaluated by inputting a graph to the network and solving the NP-hard max-cut problem. Experimental results show convergence time in the order of microseconds towards a optimal solution on a four and ten node graph.**

***Index Terms*—FPAA, Hopfield, Floating-Gate, Max-Cut**

I. Energy Surface Minimization

Energy surface minimization is a technique which can be used to solve difficult problems that conventional computing algorithms struggle with [1]. Problem metrics are assigned an energy, and a solution is reached by discovering network parameters that minimize said energy. One example of this is a standard neural network classifier, which forms an energy surface referred to as a ”cost function” and minimizes it through gradient descent and backpropogation.

Methods of representing a problem on an energy surface can vary. Some early models connect to Ising's model of magnetism [2] which then inspired Hopfield's networks [3]–[6], and other Ising network implementations [8], [9]. The parameters of these networks form an energy surface that is automatically minimized through network dynamics. When applied to difficult problems such as NP-hard tasks both Hopfield and Ising networks provide good solutions most to all of the time, and can even solve the task optimally [5], [6].

Analog computing allows for efficient processing of complex processes by recreating dynamical equations in a physical setting. Traditionally Hopfield networks have been expressed in this space, even present in class laboratory experiments. By building cost minimization structures in an analog setting, one can realize efficient solutions to difficult problems.

This work presents an analog computing framework for Hopfield networks. The circuit is built and tested on a reconfigurable analog substrate, the Field Programmable Analog Array (FPAA) [11]–[15]. The core circuit and its relation to Hopfield network dynamics are discussed in Section II. Circuit and calibration details are discussed in Section III, and experimental results of an analog Hopfield network solving the NP-hard max-cut problem are presented in Section IV. Finally, Section V contains concluding thoughts and discussions on future research directions.

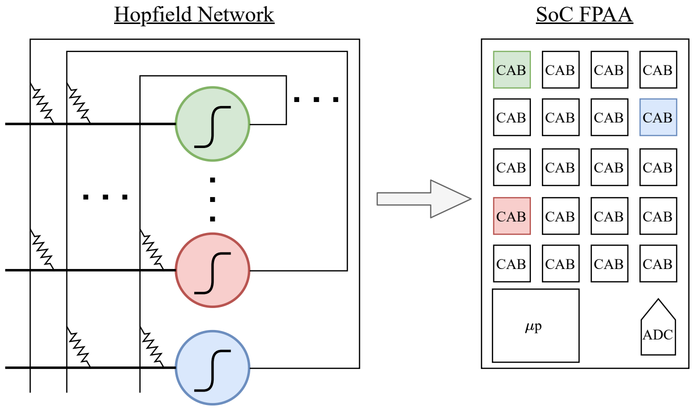

Fig. 1. This work aims to construct a Hopfield network on the SoC FPAA [14] to solve difficult problems through energy surface minimization. Hopfield neurons are mapped to programmable analog blocks and routed together to form a larger network.

II. Analog Hopfield Network

A Hopfield network consists of multiple neurons wired together in an all-to-all configuration, or in other words a fully connected graph. Each of these neurons has some output state S_i which gets multiplied by a weight W_{ij} to form an output between neurons O_{ij}. S_i is set by comparing the weighted sum of its inputs to a thresholding function:

$$S_i = \begin{cases} 1, & \sum_j O_{ji} \geq T \\ 0, & \sum_j O_{ji} < T \end{cases} \tag{1}$$

where T is some threshold value.

One can implement this structure on existing SoC FPAA devices [11]–[15] as shown in Figure 2; this work uses the FPAA described in [14]. The FPAA enables a mixed signal and analog computing platform due to the flexibility of its routing fabric and analog components. Analog computing elements are located in Computational Analog Blocks (CABs), giving access to devices such as Transconductance Amplifiers (TAs), current mirrors, NFET, and PFET devices among others. Routing locally within a CAB and globally between CABs is implemented with programmable floating gate (FG) PFET switches that can also be used for analog computation.

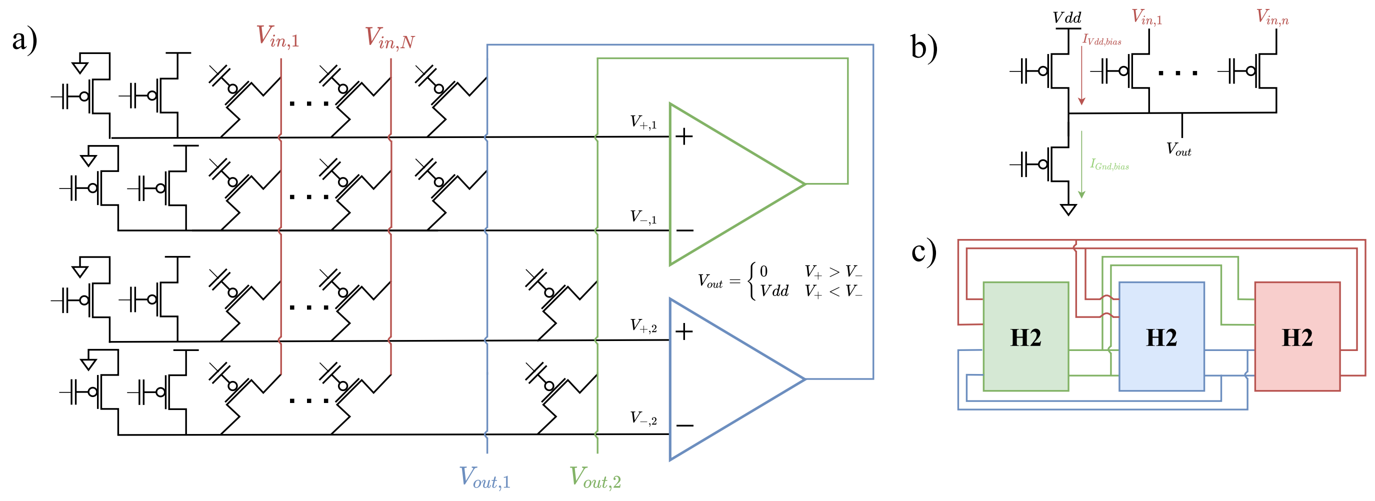

Fig. 2. a) Circuit schematic of a two node Hopfield network block. A VMM made of reconfigurable floating gate (FG) PFET devices with biases makes up the inputs. An TA compares positive and negative weighted inputs, amplifying the difference by a programmed gain. This gain is made high so that a small difference causes a large output swing between Gnd and Vdd. b) Schematic of the input VMM portion of the Hopfield circuit. Input voltages come in through the source of a PFET and get weighted by a charge programmed at the gate. This weighted current then gets converted to a voltage through the PFET connected to Gnd. c) Block diagram of a six node Hopfield network made of three two node blocks wired together. Each two node circuit as shown in (a) fits into one block of the FPAA; this block diagram represents three of these units wired together to form a larger six node network.

The threshold function of a Hopfield neuron can be implemented by an TA programmed to have a high gain. Negative weighted inputs get sent to the negative row, and positive inputs to the positive row. A difference in voltage between the two rows leads to an almost binary output similar to that of a Hopfield network threshold

$$V_{out} = \begin{cases} 0, & V_+ > V_- \\ Vdd, & V_+ < V_- \end{cases} \tag{2}$$

A Vector Matrix Multiply (VMM) created with floating gate (FG) PFETs is used to emulate the weighted inputs of a Hopfield network in the analog circuit as shown in Figure 2a and 2b. Each PFET takes a voltage input at its source and converts it to a current weighted by the gate voltage, which is then added together with all other currents according to KCL. The PFET gate voltage can also ”turn off” the transistor, allowing flexibility in choosing between positive and negative weights.

III. VMM Operation and Calibration

FG devices allow for charge to be tunneled onto the transistor gate, and are used to enable programmable gate voltages. The current through a FG PFET in subthreshold is given by

$$I = I_{th} e^{\kappa((V_{dd}-V_{fg}-V_{T0})-(V_{dd}-V_s)+\sigma(V_{dd}-V_d))/U_T} \tag{3}$$

where V_{fg} accounts for capacitive coupling of the control gate input and charge programmed onto the floating gate.

$$V_{fg} = \frac{C_1}{C_T} V_g + \frac{C_{ov}}{C_T} V_d + V_Q \tag{4}$$

Assuming a steady V_g, $C_T >> C_{ov}$, and $\sigma \approx 0$, (3) can be rewritten to clearly show how a FG PFET transistor weights a current according to the programmed floating gate charge and source voltage input

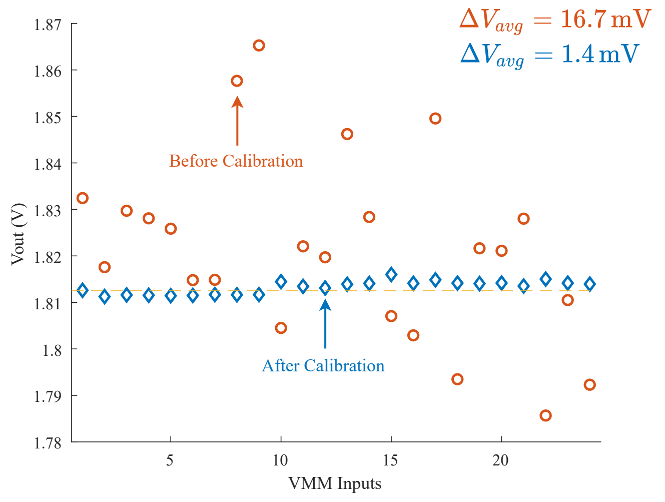

Fig. 3. Before and after calibration of input FG PFETs in a VMM. Mismatch variation in V_{out} is reduced by an average of 15.3 mV after a few calibration steps, equalizing the strength of weighted inputs to a Hopfield neuron.

$$I = e^{-\kappa V_Q/U_T} \times e^{V_s/U_T} \times I_0 = W \times X_{in} \times I_0 \tag{5}$$

where

$$I_0 = I_{th} e^{\kappa((V_{dd}-\frac{C_1}{C_T}V_g - V_{T0})-V_{dd}/U_T} \tag{6}$$

When these devices are connected together, KCL adds up all the weighted currents to create a VMM.

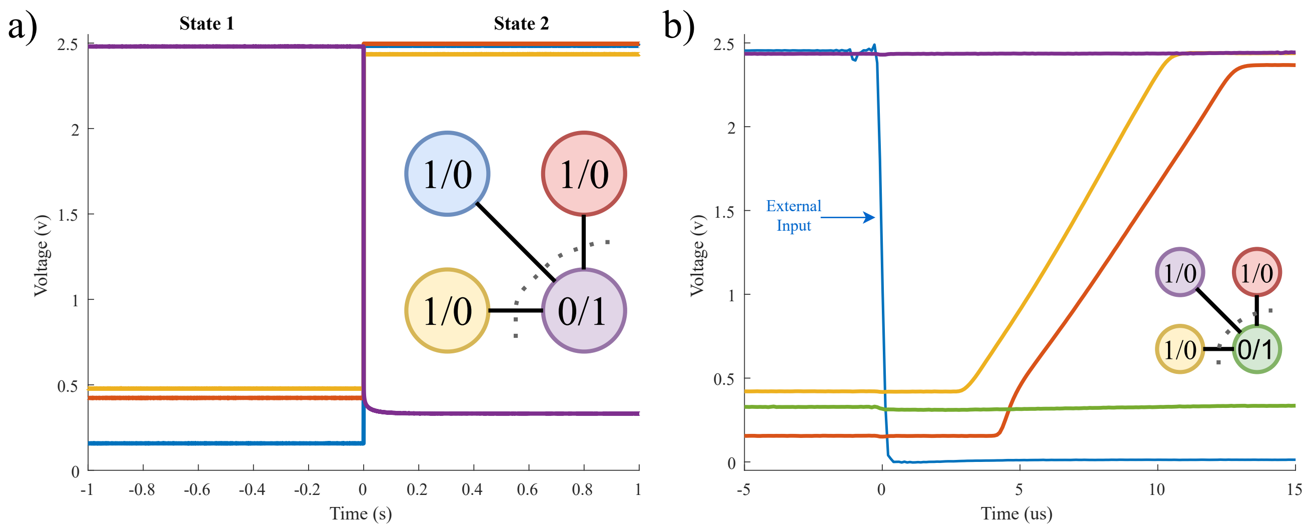

Fig. 4. Experimental data of a four node Hopfield network implemented on the FPAA solving a max-cut problem. a) Two stable max-cut solutions of the input graph. By perturbing external inputs at $t = 0$, the network switched between the two correct steady state solutions. These solutions are exact inverses of each other but represent the same partition of the input graph. b) Time domain plot of a four node Hopfield network converging from a forced unstable state to a stable state in the order of microseconds. Response time is limited by the slew of the TAs used to compare positive and negative weighted sums.

A. *Current to Voltage Translation*

If we want a voltage to voltage system the weighted current output of the VMM needs to go through a current to voltage transformer. A simple implementation is to use another PFET connected to ground, creating a source follower like structure where the ”input” is steady and the VMM modifies the bias current as shown in Figure 2b.

If we assume that inputs to the VMM are binary (Gnd or Vdd) and that every transistor is identical we can then write V_{out} according to KCL:

$$I_{vbias} + e^{Vdd/U_T} I_0 \sum_n W_n = I_{gbias} + I_0 \sum_m W_m \qquad (7)$$

Assuming that $I_{Gnd,bias} >> I_m$ and $I_{Vdd,bias} << I_n$.

$$V_{out} = V_{dd} + U_T ln\left(\frac{\sum_n (W_n)}{W_g}\right) \qquad (8)$$

for N inputs at Vdd.

B. *Mismatch and Calibration*

Previously we assumed that each transistor was identical. In general this is not true, process variations and slight programming offsets result in an effective shift of V_{th} for each transistor. This variation can be calibrated out of a VMM by adjusting the programmed V_Q to compensate for a shifted V_{th}.

A source follower structure can be used to find a ΔV_Q that is needed to reach a reference V_{out} [16]. Calculating ΔV_Q for all transistors allows an effective V_{th} match across different devices. Using (8) we can find that

$$\Delta V_{out} = V_{out,ref} - V_{out} = -\kappa \Delta V_Q \qquad (9)$$

The Hopfield network VMM uses three kinds of FG PFETs: Inputs, Vdd bias, and Gnd bias. Mismatch in any one of these affects V_{out}. Instead of calibrating all three types individually to a standard reference we can calculate ΔV_Q for only the input FETs using its unique Vdd and Gnd biases as a reference, essentially adjusting for all three sources of mismatch at once as shown in Figure 3.

Using (9) and stepping one input of the VMM to Vdd at a time allows us to find a ΔV_Q for each input transistor, adjusting for mismatch in the row. In this experiment a Digilent Analog Discovery was used both for stepping and measuring voltages, however any power supply and voltage measurement tool would suffice. Additionally the FPAA contains onboard DACs and ADCs, so this could be done entirely on chip as shown in [16].

We have assumed that κ is the same across all transistors, but this isn't generally correct and combined with slight programming variations gives some error from the ideal ΔV_Q. To adjust for this the calibration procedure can be repeated so that V_{out} converges tightly to variations of a few millivolts across inputs, minimizing the mismatch effect on calibration.

IV. Max-Cut on an Analog Hopfield Network

The max-cut problem is one of the NP-hard class of tasks that can be solved through energy surface optimization. If there is some graph with edges, E, and vertices, V, that is then divided into two sets the maximum cut is the partition that contains the largest amount of edges between the sets. A bipartite graph is an example of the most extreme possible solution, where every edge goes between the two sets. These edges can be weighted or unweighted, and a solution is evaluated by taking the tTAl cost of all edges between the two sets.

A max-cut problem can be mapped to a Hopfield network by setting weights depending on if an edge between two vertices in the problem graph exists,

$$W_{ij} = \begin{cases} -W, & E_{ij} \subset E \\ +W, & E_{ij} \not\subset E \end{cases} \tag{10}$$

where W is some weight constant [17], [18]. In the case of this analog Hopfield network W is a voltage programmed onto the gate of the FG transistors in the VMM between neurons i and j. Vertices are considered in the same set if they have the same state after the network has settled. Setting the weights in this way encourages vertices sharing an edge to be in different sets, which is what we want to find the maximum cut.

A max-cut problem was built on an FPAA and programmed onto a four and ten node Hopfield network; solutions are shown in Figure 4 and Figure 5. W was set to ten nanoamps and calibrated onto VMMs as discussed in Section III. One CAB in the FPAA contains two regular TAs as shown in Figure 2, so the four node network took up two CABs worth of area while the ten node network took up five. Each CAB has thirteen input lines so the theoretical maximum size with the presented design is a fifteen node network on the FPAA, however the remaining lines were left free for external inputs in this paper.

There are two equivalently correct optimal solutions for any max-cut problem programmed to a Hopfield network: a solution state and its inverse. Figure 4a shows the network switching between the two correct states of a graph programmed onto a four node network through an external input perturbation. The optimal state settled with a small convergence delay after programming.

To study this convergence time an incorrect initial condition was set using external inputs and then released as shown in Fig 4b and Fig 5 for both the four and ten node networks. The correct solution is reached after around thirteen microseconds in the four node network and thirty microseconds for the ten node. This delay is due to the TA gain not being infinite: it takes some time for the difference in V_+ and V_- to register at the output. Additionally, to prevent networks settling with every node at zero, the Vdd bias transistor on the positive row was set slightly higher than the negative row. This causes nodes to initially move towards a high output if all inputs are zero, which is seen happening to nodes three, seven, nine, and ten in Fig 5 before network dynamics push them back down.

V. Summary and Discussion

In this paper we have presented an analog computing structure for solving optimization problems based on the Hopfield Network. The circuit was built on a programmable FPAA, and experimental results demonstrated a correct solution for the max-cut problem on both a four and ten node network. Both networks reached an optimal solution in microseconds.

Future work will build upon the base Hopfield circuit to adjust for TA slew, build larger VMMs, and improve the general design. Larger networks will be tiled together and programmed to test how well this design scales, and detailed energy measurements will be taken to show efficiency gains from this analog computing implementation.

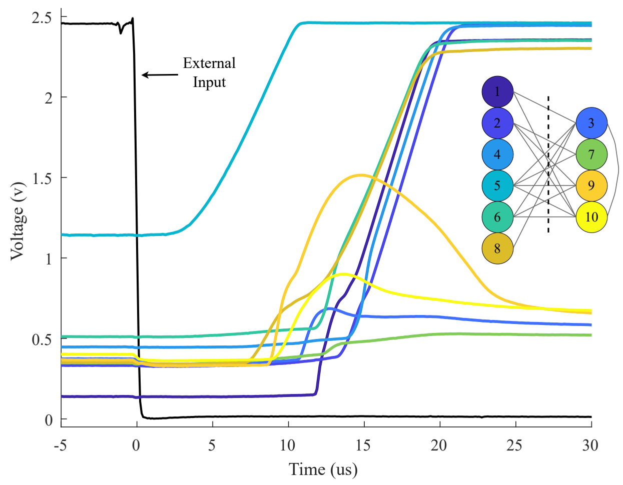

Fig. 5. Time domain plot of a ten node Hopfield network converging from a forced unstable state to a stable state in the order of microseconds. Response time is limited by the slew of the TAs used to compare positive and negative weighted sums. Additional dynamics appear on the nodes that converge to a low output due to an increased bias on the positive row of each node. The asymmetry is set to ensure the network does not settle at a state where every output is Gnd, and initially affects all neurons before network dynamics force the output back low, increasing convergence time.

Ising networks are derived from the same physical phenomena that inspired Hopfield networks, and have very similar behavior in that they also minimize the same energy surface as a Hopfield network and can be used to equivalently solve complex problems. However it is debated which implementation, Hopfield or Ising, can be more efficient. Future work will investigate this question by implementing an Ising network alongside the Hopfield network proposed in this work, allowing for an equal comparison in the same programmable analog substrate.

References

[1] LeCun, Yann, Chopra, Sumit, Hadsell, Raia, Ranzato, M, and Huang, F. A tutorial on energy-based learning. Predicting structured data, 1:0, 2006.
[2] S.K. Vadlamania, T. P. Xiaob, and E. Yablonovitch, "Physics successfully implements Lagrange multiplier optimization," PNAS, pp. 1-12
[3] R. Rojas, "The Hopfield Model," Chapter 13, Neural Networks, Springer-Verlag, Berlin, 1996.
[4] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," Proceedings of NationalAcademy of Science, vol. 79, 1982, pp. 2554.
[5] J. Hopfield, "Neurons with graded responses have collective computational properties like those of two-state neurons," Proceedings of NationalAcademy of Science, vol. 81, 1984, pp. 3088-3092.
[6] J.J. Hopfield, D.W. Tank, Neural computation of decisions in optimization problems, Biological Cybernetics, vol. 52, 1985, pp. 141-152.
[7] J. Hopfield and D. Tank, "Computing with neural circuits: a model," Science, vol. 233, no. 4764, 1986. pp. 625-633.
[8] "Oscillator-based Ising Machine," T. Wang and J. Roychowdhury
[9] OIM: Oscillator-based Ising machines for solving combinatorial optimisation problems," T. Wang and J. Roychowdhury
[10] J. Chou, S. Bramhavar, S. Ghosh, W. Herzog, "Analog coupled oscillator based weighted Ising machine," Scientific Reports, Nature Research, vol. 9, October 2019.

[11] B. Rumberg and D. W. Graham, “A low-power field-programmable analog array for wireless sensing,” in Proc. ISQED, Mar. 2015, pp. 542–546

[12] G. E. R. Cowan, R. C. Melville, and Y. P. Tsividis, “A VLSI analog computer/digital computer accelerator,” IEEE J. Solid-State Circuits, vol. 41, no. 1, pp. 42–53, Jan. 2006.

[13] “A continuous-time field programmable analog array (FPAA) consisting of digitally reconfigurable GM-cells,” in Proc. ISCAS, May 2004, pp. I.1092–I.1095.

[14] S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, R, Wunderlich, S. Nease, and S. Ramakrishnan, “A programmable and configurable mixed-mode FPAA SoC,” IEEE Transactions on VLSI, vol. 24, no. 6, 2016, pp. 2253-2261.

[15] J. Hasler, “Large-scale field programmable analog arrays,” IEEE Proceedings, vol. 108. no. 8. August 2020. pp. 1283-1302.

[16] S. Kim, S. Shah and J. Hasler, ”Calibration of floating-gate SoC FPAA system,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 9, pp. 2649-2657, Sept. 2017, doi: 10.1109/TVLSI.2017.2710020.

[17] Rong Long Wang, Zheng Tang, Qi Ping Cao, A parallel algorithm for maximum cut problem using gradient ascent learning of Hopfield neural networks, IEEEJ Transactions on Electronics, Information and Systems, 2002, Volume 122, Issue 11, Pages 1986-1994

[18] L. yun Wu, X. sun Zhang, and J. liang Zhang, “Application of discrete hopfield-type neural network for max-cut problems,” 2001