A Reconfigurable VLSI Learning Array

Seth Bridges [1], Miguel Figueroa [2], David Hsu [3], and Chris Diorio [1]

(1) University of Washington, Computer Science and Engineering, Seattle, USA.
(2) Universidad de Concepción, Department of Electrical Engineering, Concepción, Chile.
(3) Hamlet, Inc., Seattle, USA.
seth@cs.washington.edu, mfigueroa@die.udec.cl,
david@hamletinc.com, diorio@cs.washington.edu

Abstract:

We present a reconfigurable array for low-power feed-forward neural networks in analog VLSI. This architecture implements a flexible computational model with coarse-grained reconfigurability, and features high computational density for a broad range of applications. Our prototype of the array, fabricated in a 0.35μm process, consumes 0.25mm^2 of area and dissipates 150μW of power on a 5V supply. In this paper, we discuss the circuits and architecture of our system, as well as experimental results.

1. Introduction

We present a reconfigurable array for low-power feed-forward neural networks in analog VLSI. Our system comprises a two-dimensional array of Programmable Learning Blocks (PLBs), each of which implements a linear synapse and a likelihood estimator. The PLBs are connected by a network that features configurable routing, global feedback, and distributed aggregation of PLB outputs. This coarse-grained architecture enables the array to implement a variety of algorithms such as silicon mixture models, single-layer perceptrons, and radial basis functions with high computational density. These algorithms are widely used in a broad range of applications such as adaptive filtering, noise cancellation, adaptive inverse control, image processing, and robotic control.

Previous approaches to reconfigurable learning in VLSI, including our own Field-Programmable Learning Array [1], feature low-level primitives and more flexible routing, and are tailored to rapid prototyping of adaptive circuits. However, they do not scale well to full-size algorithms due to the large area-overhead imposed by their fine-grained reconfigurability. Other VLSI implementations of neural networks, including single- and multi-layer perceptrons [6], radial basis function networks [8] and silicon mixture models [2] provide reconfigurable parameters, but implement a fixed algorithm. Our system also differs from more traditional Field-Programmable Analog Arrays (FPAAs) [5] in that we provide specific support for feed-forward networks with on-chip learning, rather than opamp-based filtering and signal processing.

We fabricated a prototype of the array in a 0.35μm process. The chip core measures 450μm$\times$450μm and dissipates 150μW on a 5V supply when fully configured. The rest of the paper describes the array and its underlying computational model, as well as experimental results.

2. Computational Model

Our chip implements a single-layer perceptron, a mixture model, and a radial basis function network, all feed-forward network algorithms that compute using a parallel array of local (or synaptic) functions whose outputs are aggregated in a neuron and used as inputs to subsequent layers in the network.

For linear applications such as adaptive inverse control, a *single-layer perceptron* computes the weighted sum of its inputs. The synaptic operation is the multiplication of the global input with the local weight. The neuron operation is the summation of each synaptic product.

A *mixture model* is a collection of probability density functions (PDF), typically Gaussian. In one dimension, a Gaussian distribution is defined by two scalar parameters: a mean μ and variance σ^2. An adaptive mixture-model network that learns the parameters of the distributions is useful for probabilistic data-modeling and unsupervised clustering. Assuming that each dimension is independent, we compute the log likelihood of a multi-dimensional input as the sum of the log likelihood outputs of each one-dimensional output. We can train a Gaussian mixture model in an unsupervised manner to characterize an input distribution using the Expectation-Maximization (EM) algorithm.

A *radial basis function network* is a heterogeneous two-layer network comprising a Gaussian mixture model and a linear perceptron. This type of network is commonly used in adaptive control and computer vision applications. The first layer performs a non-linear transformation of the input using a set of basis functions such as the Gaussian basis function discussed above. The second layer is a linear perceptron that weights the responses of the first layer. The output mapping of the network can be expressed as

$$y(\mathbf{x}) = \sum_{i=1}^{d} w_i \phi_i(\mathbf{x}) \qquad (1)$$

where $\phi_i(\mathbf{x})$ is one of the d basis functions and w_i is its associated weight value. The EM algorithm can be used to train the basis functions in an unsupervised manner, while the output layer can use a linear learning algorithm such as Least-Mean-Square (LMS).

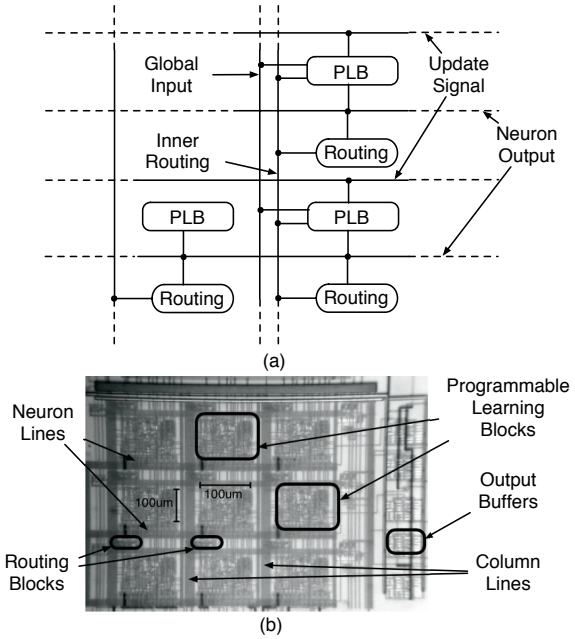

Figure 1: (a) System architecture. Our chip comprises a two-dimensional array of Programmable Learning Blocks that compute synaptic functions, and routing switches that enable multi-layer networks. (b) Micrograph. The core of our fabricated chip measures 450μm$\times 450\mu$m in a 0.35μm process. Each PLB measures about 100μm$\times 100\mu$m.

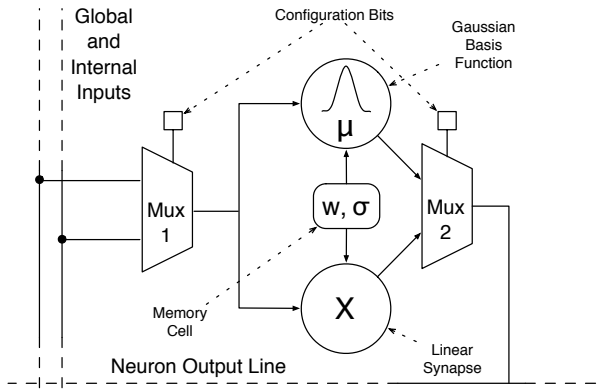

Figure 2: Programmable Learning Block. Each PLB receives a column input and a row-error signal and uses these to compute its synaptic output and local weight update. A configuration bit statically configures mux 1 to select either the global or internal input, while another bit configures mux 2 to direct the output of the selected function to the neuron. Unshown configuration bits disable unused portions of the PLB for reducing power consumption.

3. Learning Array

3.1 System Architecture

Our learning network architecture, shown in Figure 1(a), comprises a two-dimensional array of Programmable Learning Blocks (PLB) that perform a synaptic computation. The array aggregates the PLB outputs across its rows to implement a neuron function. Both the computation performed by the PLB and the interconnect network can be configured to implement the network functions described in Section 2.

The PLB is a reconfigurable block that computes a one-dimensional synaptic function of its local memory and input. Our PLB comprises an analog memory cell with linear updates, a basis function that approximates a Gaussian PDF with adaptive mean and variance, and an analog Gilbert multiplier.

The rows of PLBs form neurons and each column operates over a single input dimension. Each column has two routing lines, one for global inputs, and one for inter-layer connectivity. Each PLB has an associated reconfigurable routing block that enables its row output to be connected to the column input of other blocks in its column.

To have more flexibility in experimenting with different algorithms, this prototype implements the LMS update and winner-take-all operation for the mixture model off-chip. External circuitry generates the inputs and reads the outputs of our array to generate parameter update signals for training. The weight-update mechanism for the basis function circuits are implemented on-chip.

3.2 PLB

Figure 2 shows the block diagram of a PLB. Each block comprises a Gilbert multiplier, a basis function block, and an analog memory cell. The Gilbert multiplier computes the product between the PLB input and a synaptic weight stored in the memory cell, and the basis function block applies a Gaussian basis function to the input, with an internally-stored mean and a variance stored in the memory cell. Multiplexers controlled by static configuration bits select the input from a set of global and internal inputs running through the columns of the array, and determine the output of the PLB to the neuron output line running through the rows. Row feedback signals are used to compute memory updates and implement a variety of learning rules.

The memory cell stores its value as nonvolatile analog charge on a floating-gate pFET [2]. Pulse-driven charge updates using hot-electron injection [7] and Fowler-Nordheim tunneling [4] accurately control changes to the stored value. Floating-gate pFETs are immune to charge leakage and their pulse-update mechanisms do not suffer from the charge-injection limitations common to capacitors, thus they enable compact, high-resolution storage in VLSI neural networks.

Figure 3(a) shows the memory cell, based on our previous design [3]. Negative capacitive feedback keeps the floating-gate voltage Vmem at a constant value Vref, enabling memory updates that are linearly proportional to the number of fixed-width digital pulses on Vinc (injection) and Vdec (tunneling). A common-mode feedback circuit creates a differential version of the memory-cell output to drive the synapse and basis function circuits. Figure 3(b) shows the output of eight isolated memory cells on a chip as a function of the density of update pulses for the same reference Vref. The transfer function is linear, but device mismatch creates variations in the relative tunneling and injection strengths for each cell, leading to asymmetric updates and poor learning performance. To compensate for this effect, we set Vref independently for

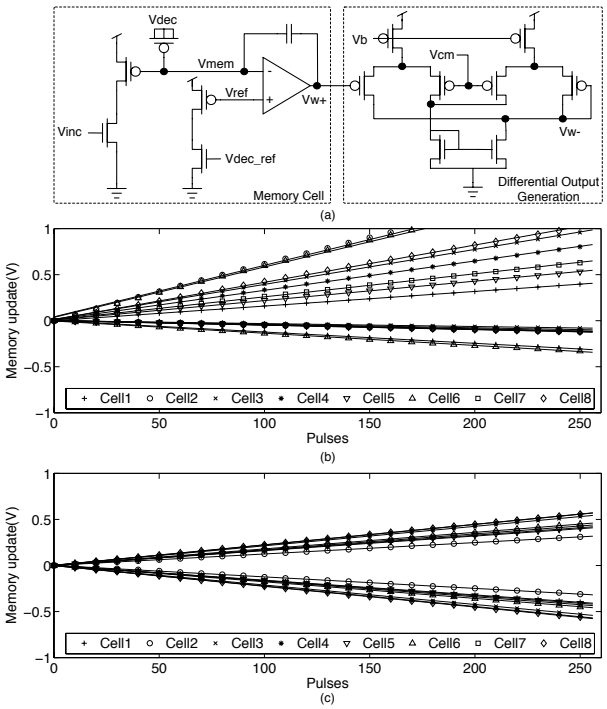

Figure 3: (a) Memory cell. This floating-gate opamp with negative feedback serves as our memory cell. Vinc and Vdec increase and decrease the value of the output Vw+. Vdec_ref calibrates the cell to have symmetric updates. A common-mode feedback circuit generates the negative output Vw- of the memory cell. (b) Pre-calibration update linearity and symmetry. (c) Post-calibration update linearity and symmetry.

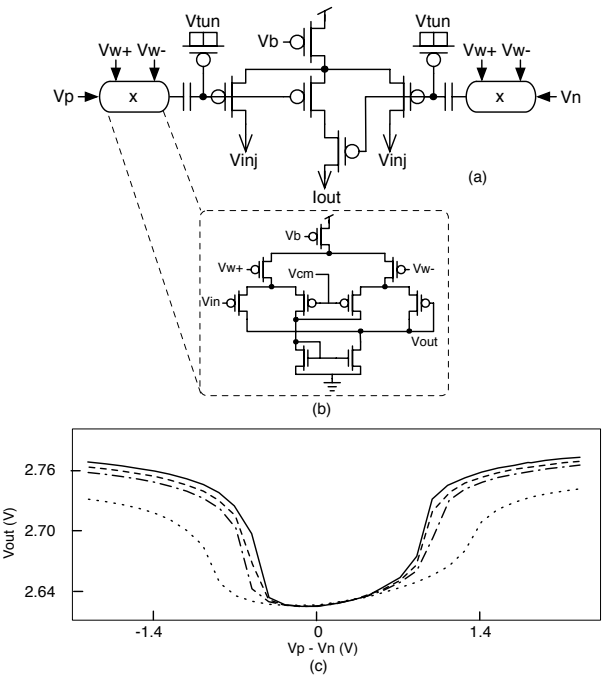

Figure 4: (a) Basis function circuit. Our variable-width basis function comprises an adaptive bump circuit and two attenuating multipliers. The differential input is Vp and Vn and we apply pulses to the Vinj and Vtun terminals to adapt the mean parameter. (b) Attenuating multiplier. The input to the adaptive bump is scaled by Vw using this common-mode feedback circuit. (c) This is a plot of the basis function response for different variances.

each memory cell on a second floating gate, thus locally controlling the ratio between tunneling and injection. Figure 3(c) shows the response of the memory cells after calibration. The learning rates are still different across memory cells, but are symmetric within each cell.

For our basis function, we present a novel variable-width extension to Hsu's adaptive bump [2] that allows a more accurate representation of the density of the input data. This new circuit, shown in Figure 4(a), uses the core of the bump circuit and achieves variable-width operation by adding two multipliers, shown in Figure 4(b), to scale the differential input (Vp, Vn) by the weight value Vw stored in the memory cell. The multiplier works as a common-mode feedback circuit, using feedback to minimize the difference between the average of Vin and Vout, and Vcm.

3.3 Interconnect

PLB outputs aggregate across each row of the array to compute the neuron output. PLBs configured as linear synapses sum their Gilbert multiplier differential-current outputs on common wires. PLBs configured as basis functions aggregate their outputs by computing a log-likelihood voltage using a diode-connected FET. We sum these voltages across the row using a capacitive divider whose DC level can be programmed with a floating-gate device.

We form a multi-layer network by connecting row outputs to the internal column lines via the routing blocks. Each PLB receives both the global and internal input signals and a static configuration bit controls a multiplexer that selects the appropriate input. All switches in this design are full transmission gates.

4. Experiments

Our first experiment measured the resolution of the feedback path to the memory cell when it is used as a synaptic weight in a perceptron configuration. To remove the effects of input offsets and device mismatch on our resolution measurement, we provided a fixed input and target value to a single linear synapse and adapted the synaptic weight stored in the differential memory cell using the LMS algorithm. After 400 training iterations using error pulses 10μs in width, the error in the output current converged to 0.7nA RMS on a 2μA differential output range, which corresponds to greater than 10 bits of update resolution.

Our next experiment was to implement the single-layer perceptron network in Figure 5(a) by configuring our chip as shown in Figure 5(b). We provided an input sampled uniformly over a 0.8V differential range and used an LMS update rule to train the synaptic weights. At each iteration, the circuit was shown a new input and its output error was used to generate an error pulse train. After roughly 800 iterations, the error in the output was 4nA RMS on an output range of 1.75μA, which corresponds to almost 8 bits of output accuracy.

In an additional experiment, we configured our chip as the radial basis function (RBF) network shown in Figure 6(a). The network had two inputs and two-layers, one hidden

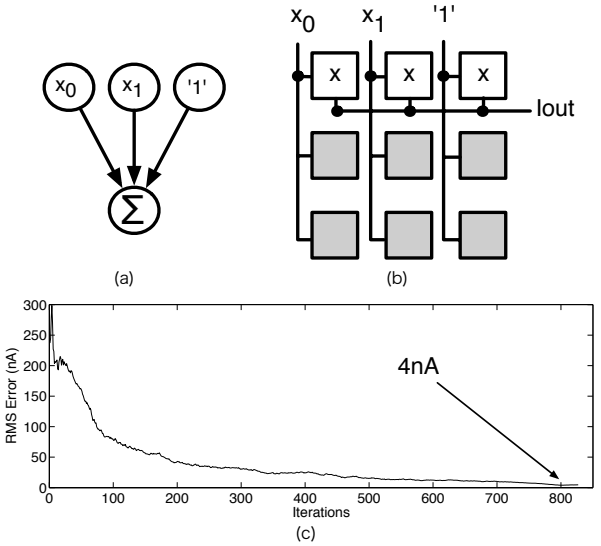

Figure 5: (a) Single-layer perceptron. We implement this three-synapse perceptron. (b) Our chip configuration for this network. We use the PLBs in the first row of our chip to implement this network while the second and third rows of PLBs are unused. (c) Training error. After training the network with LMS, the output error converged to 4nA RMS on a 1.75μA output scale which corresponds to almost 8 bits of output accuracy.

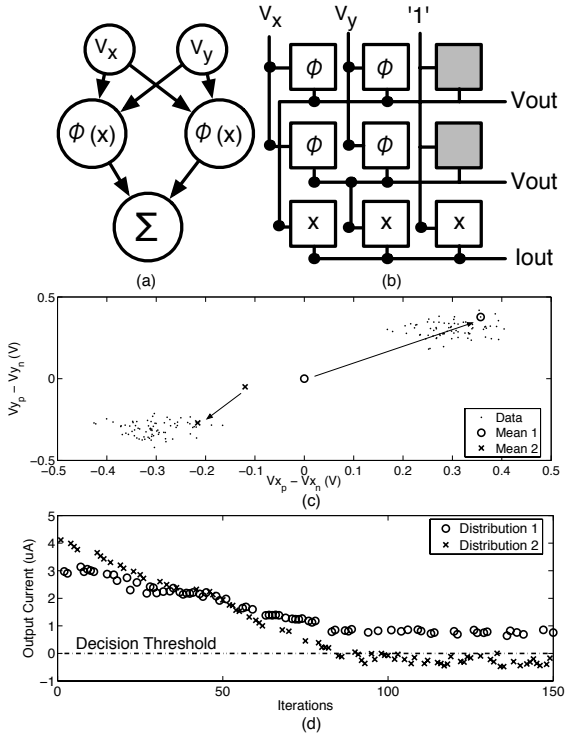

Figure 6: (a) RBF classifier. We implement this network to learn the classification of the data in (c). (b) Our chip configuration for this network. Each of the first two rows of PLBs forms a two-dimensional basis function whose outputs are fed to the third layer which implements a perceptron with an additional bias input. (c) The basis functions were trained using an online k-Means algorithm. The means of the basis functions are randomly initialized and during the experiment, adapt toward one of the data clusters. The means do not reach the true cluster centers due to asymmetry in the basis circuit operation. (d) A supervised training method was used to train the output layer to differentiate the input data.

RBF layer with two basis functions and the linear output layer with a single perceptron. Figure 6(b) shows the chip configuration for this network. We trained this network in two stages, an unsupervised stage for the hidden layer (Gaussian mixture model) and a supervised stage for the output (linear) layer. For the hidden layer, we presented synthetic data from two two-dimensional Gaussians that had diagonal covariance matrices. After the presentation of each data point, we performed a winner-take-all operation off-chip and applied update pulses to the basis function with the maximal response. After the means converged to a stationary distribution, we trained the output layer to respond only to data points from one cluster using LMS. Because the data is linearly separable, the chip was able to detect cluster membership with 100% accuracy.

5. Conclusions

In this paper, we present a low-power and compact reconfigurable array of Programmable Learning Blocks (PLB). We fabricated a 3x3 array of PLBs in a $0.35\mu m$ process and presented results for this array in a variety of configurations. Future work will focus on on-chip learning rules, design scaling, extending the set of primitive functions, and automating chip calibration.

Acknowledgments

ONR Grant #N00014-01-1-0566 supported this work. MOSIS fabricated the chips for these experiments.

References:

[1] Seth Bridges, Miguel Figueroa, David Hsu, and Chris Diorio. Field-Programmable Learning Arrays. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1155–1162. MIT Press, Cambridge, MA, 2003.

[2] Chris Diorio, David Hsu, and Miguel Figueroa. Adaptive CMOS: from Biological Inspiration to Systems-on-a-Chip. *Proceedings of the IEEE*, pages 345–357, 2002.

[3] Miguel Figueroa, Seth Bridges, and Chris Diorio. On-Chip Compensation of Device-Mismatch Effects in Analog VLSI Neural Networks. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 441–448. MIT Press, Cambridge, MA, 2005.

[4] M. Lenzlinger and E.H. Snow. Fowler-Nordheim Tunneling into Thermally Grown SiO_2. *Journal of Applied Physics*, pages 278–283, 1969.

[5] C.A. Looby and C. Lyden. Op-amp Based CMOS Field-Programmable Analogue Array. In *Circuits, Devices and Systems, IEE Proceedings*, volume 147, pages 93–99, April 2000.

[6] S. Satyanarayana, Y. Tsividis, and H. Graf. A Reconfigurable VLSI Neural Network. *IEEE Journal of Solid-State Circuits*, 27(1), January 1992.

[7] E. Takeda, C. Yang, and A. Miura-Hamada. *Hot Carrier Effects in MOS Devices*. Academic Press, 1995.

[8] Steven S. Watkins and Paul M. Chau. A Radial Basis Function Neurocomputer Implemented with Analog VLSI Circuits. In *International Joint Conference on Neural Networks*, pages 607–612, June 1992.