A Mixed Mode Self-Programming Neural System-on-Chip for Real-Time Applications

Khurram Waheed and Fathi M. Salam

Department of Electrical and Computer Engineering
Michigan State University
East Lansing, MI 48824-1226

Abstract

The paper provides an overview of the development of a self-learning computing chip in the new 0.18 micron copper technology. The chip realizes an architecture that achieves the task of self-learning execution times in the micro to milli seconds. The core consists of basic building blocks of 4-quadrant multipliers, trans-conductance amplifiers, and active load resistances, for analog (forward-) network processing and learning modules. Super-imposed on the processing network are digital memory and control modules composed of D-Flip-flops, ADC, Multiplying D/A Converter (MDAC), and comparators for parameter (weight) storage, logical control and analog/digital conversions.

The single System-on-Chip design impacts several domains of critical applications that include nano-scale biotechnology, automotive sensing, control and actuation, wireless communications, image feature extraction and pattern matching etc

1. Introduction

The core of the chip is a neurally inspired scalable (re-configurable) array network for compatibility with VLSI. The chip is endowed with tested auto-learning capability, realized in hardware, to achieve global task auto-learning execution times in the micro to milli seconds.

The architectural forward network (and learning modules) process in analog continuous-time mode, while the (converged, steady state) weights/parameters can be stored on chip in digital form. The overall architectural design adopts engineering methods from adaptive networks and optimization principles [1,2].

The designed chip can handle 16 inputs, 16 outputs. In addition there are inputs for control interface, synchronization and stand-alone programmability of the chip resulting in an approximate die area of 4000 X 6000 μm^2 in a QFP-208L package

The 6 layer Copper (Cu) interconnect, single poly, 0.18 micron process enables dense connectivity and dense die area of this highly interconnected network resulting in a compact powerful engine. Moreover, the special low resistance and low capacitance electrical properties of copper permit the design to achieve the high connectivity while still managing precise distributions of resistive and capacitive loads. These properties enable one to predict performance and limit signal time-delays along the interconnect. The small feature size and the electrical interconnect properties for copper are enablers to the realization of such a powerful chip with dense interconnectivity.

Figure 1: Architectural Overview of the Chip

The chip operates in four different modes: (i) *learn*, (ii) (on-chip) *store*, (iii) *program read/write*, and (iv) *process* which selectively combine its intrinsic analog and digital building blocks in a novel manner (see fig. 1).

Initially the system-level chip design was simulated and verified using SIMULINK/MATLAB. All the building blocks were custom designed and extensively simulated using HSPICE (incorporating the UMC level 49 models). The design was implemented in the 6 level Copper (Cu) interconnect, single poly, 0.18 micron process which is an enabler for the dense connectivity and dense die area of this highly interconnected network. The design was laid out and verified using Cadence Tools. More details of

high-level design, circuit design of major blocks and chip-layout will be provided in the subsequent sections [8,9].

2. Architectural Design

- A *high-level design*, specifying the characteristics of the neural network to be implemented and the definition of its basic building blocks.
- A *circuit level design*, describing each of these blocks based on the copper technology with their corresponding simulations.
- Finally a *layout level design*, where the actual chip layout is created and verified

We present our stepwise approach for tailoring the BP algorithm so that it becomes suitable for VLSI implementation. For illustration in each case, we present the simulated results of the XOR problem, highlighting the effect of each modification on the performance of the algorithm.

Figure 2: Simulink Model for Chip Simulations

simulation we are illustrating, there are two input neurons and one output neuron.

$\dot{W}^{(3)}$ - represents the time derivative of the weight for output layer

3.1 Modified BP algorithm with linear multiplier.

$$\dot{W} = -\eta \cdot \frac{\partial E}{\partial W} - \alpha \cdot W \quad \text{where} \quad NET^{(i)} = W^{(i)} \cdot X^{(i-1)}$$

3.2 Modified BP algorithm with nonlinear multiplier.

$$\dot{W} = -\eta \cdot \frac{\partial E}{\partial W} - \alpha \cdot W, \text{ where } NET^{(i)} = \tanh(W^{(i)}) \cdot \tanh(X^{(i-1)})$$

The update laws are

$$\begin{aligned}\dot{W}^{(3)} &= \eta \cdot \tanh(D - Y) \cdot \tanh(X^{(3)}) - \alpha \cdot W^{(3)} \\ \dot{W}^{(2)} &= \eta \cdot \tanh(\tanh(\psi'(NET^{(2)})) \cdot \tanh(\tanh(D - Y) \cdot \\ &\quad \tanh(W^{(3)}))) \cdot \tanh(X^{(2)}) - \alpha \cdot W^{(2)} \\ \dot{W}^{(1)} &= \eta \cdot \tanh(\tanh(\psi'(NET^{(1)})) \cdot \\ &\quad \tanh(\tanh(\tanh(\psi'(NET^{(2)})) \cdot \tanh(\tanh(D - Y) \\ &\quad \cdot \tanh(W^{(3)}))) \cdot \tanh(W^{(2)}))) \cdot \tanh(X^{(1)}) - \alpha \cdot W^{(1)}\end{aligned}$$

3.3 The Modified BP algorithm with nonlinear multipliers and the removal of the derivative function in all the hidden layers.

The update laws are

$$\begin{aligned}\dot{W}^{(3)} &= \eta \cdot \tanh(D - Y) \cdot \tanh(X^{(3)}) - \alpha \cdot W^{(3)} \\ \dot{W}^{(2)} &= \eta \cdot \tanh(\tanh(D - Y) \cdot \tanh(W^{(3)})) \cdot \tanh(X^{(2)}) \\ &\quad - \alpha \cdot W^{(2)} \\ \dot{W}^{(1)} &= \eta \cdot \tanh(\tanh(\tanh(D - Y) \cdot \tanh(W^{(3)})) \cdot \tanh(W^{(2)})) \\ &\quad \cdot \tanh(X^{(1)}) - \alpha \cdot W^{(1)}\end{aligned}$$

Figure 5: Training Error and convergence for modified BP Algorithm with removal of derivative function in all hidden layers

Presented in fig. 6 are the input/output waveforms for the network trained using the above final update rule

Figure 6: Input /Output waveforms for the final network

3.4 Summary of the High-level Simulation Results

From the high-level Matlab/Simulink simulations, one can draw the following conclusions:

- 1) In replacing the ideal linear multiplier model by the realistic nonlinear multiplier model, the neural network converges.
- 2) Removing the derivative function of the second hidden layer, the neural network could still converge. In fact, this can be easily verified mathematically.
- 3) When the derivative function in all hidden layers are removed, the neural network could converge, but in this case the training error is not zero but attains a small constant mean value. The update law derived in this case is still a gradient type law [7].

For all presented simulation results, we used the same set of initial conditions for all the models i.e all initial conditions at zero. In general for different nonlinear systems, the same initial conditions may bring different training result. An initial weight set which results in a good training result for one nonlinear system will not necessarily ensure that it will result in a good training result for another nonlinear system.

4. Conceptual Chip Design

Presented below is a higher block level design of the chip. For more details see [1,9]

4.1 The Synaptic Cell:

The initial and highest level design of each neural cell shows the central idea of the processing network and the learning network (Fig. 7).

The processing stage is composed of 16 neurons built using vector multipliers and a sigmoid function. The multipliers use as operands an input vector and a weight vector. The input is common to all processing units, and the weights belong to each neuron. The scalar product is then applied to the non-linear function, resulting in the output of a neuron.

The learning stage works in a similar manner as the processing stage, but using different sources for the product. It receives the signals from the next stage, creates a new signal to be sent to any previous cell, and updates the weights according to the update law. Both the stages of learning and the processing networks are merged locally. To accomplish this, it was necessary to decompose the 17-D multipliers that constituted each network node into a set of 1-D multipliers.

On-chip memory is designed as local digital memory. It is therefore necessary to add a stage where the current analog

Figure 10: Gilbert Multiplier – DC Characteristics

5.2 Wide Transconductance Amplifier

The wide-transconductance amplifier is used in multiple modes in this chip. It is used as the sigmoid non-linearity at the end of each row, and as a buffer for in-chip signal buffering.

For the transconductance amplifier, the differential-in, differential-out transconductance is given by the equation

$$g_{md} = \frac{\partial I_{out}}{\partial V_{ID}} = \sqrt{\beta I_{D3}} = \sqrt{\frac{K_1 I_{D3} W_1}{L_1}} \quad (\text{at } V_{ID} = 0)$$

Wide transconductance amplifier was preferred over the simple transconductance amplifier for its better characteristics against

- transistor size/current mismatch and hence common-mode gain
- input common-mode voltage range
- better input/output voltage swing

The designed amplifier was achieved by adding two extra current mirrors to the simple transconductance amplifier. By reflecting the currents of M1 and M2 to the upper current mirrors, the output current is just the difference between I_1 and I_2 , with the advantage that both input and output voltages can run almost up to V_{dd} and almost down to V_{ss} , without affecting the operation of the circuit.

The output current, I_{out} , in the schematic (fig. 11), is converted to a voltage value using a 2-CMOS active load (not shown in the schematics). The complete layout is shown in fig. 12 with the simulated transfer characteristics shown in fig. 13.

Figure 11: Wide Transamp - Schematics

Figure 12: Wide Transamp - Layout

Figure 13: Wide Transamp - DC Characteristics

5.3 Comparator

The comparator is used in ADC to compare the Multiplying DAC output voltage with the actual input voltage that is to be converted. The input stage of the comparator is a differential amplifier and the next stage is a decision circuit. The last stage is nothing but an inverter used as a thresholding/polarity correction circuit. The schematic is shown in fig. 14, the layout in fig. 15 and the characteristics waveforms are shown in fig. 16.

Figure 14: Threshold Comparator- Schematics

Figure 15: Threshold Comparator- Layout

Figure 16: Threshold Comparator- Characteristics

5.4 ADC Operation

The simulation results below illustrate a simulation of conversion of an analog voltage of 0.85V to its digital equivalent by the successive approximation ADC internal to the chip. In this simulation the conversion process takes approximately $3\mu\text{s}$ to convert, implying that all the converged analog weights will be converted to their digital equivalent in approximately $3\mu\text{s} \times 17 = 51\mu\text{s}$.

Figure 17: ADC Convergence Characteristics

5.5 Vector Multiplications

Presented below are the transient characteristics of multiplication of two time domain sinusoids. The third waveform presents the result, which is the cascaded result of the current output of seventeen multipliers collected in current busbars and converted to the voltage domain at the end using a 4-CMOS active load.

Figure 18: Vector Signal Multiplication- Transient

6. Overall Chip Layout & Interconnects

The figure below shows the array structure of the implemented chip, best possible fit of the building sub-cells were tried in order to achieve a highly dense building block structure. A hierarchical routing assignment was made for the available 6 metal layers to achieve the required dense connectivity. For more details see [1].

Figure 19: Layout for the Building Block

7. References

- [1] Khurram Waheed and Fathi M. Salam, "A Mixed-Mode Design for a Self-programming Chip For real-time estimation, prediction, and control"; Proc. Of 43rd IEEE Midwest Symposium on Circuits and Systems, Aug 8-11, 2000, pp. 810-813
- [2] Gert Cauwenberghs and M. Bayoumi, (editors) Learning on Silicon, adaptive VLSI neural systems, Kluwer Academic Publishers, July 1999.
- [3] F. M. Salam, H-J. Oh, "Design of a Temporal Learning Chip for Signal Generation and Classification," Analog Integrated Circuits and Signal Processing, an international journal, Kluwer Academic Publishers, Vol. 18, No. 2/3, February 1999, pp. 229-242.
- [4] M. Ahmadi and F. Salam, Special Issue on Digital and Analog Arrays, International Journal on Circuits, Systems, and Computers, October/December 1998 (Issue published in December 1999).
- [5] F. M. Salam, M. R. Choi."An All-MOS Analog Feedforward Neural Circuit with Learning," IEEE Int' Symp. on Circuits and Systems (ISCAS), May 1990, pp. 2508-2511.
- [6] MSU Team, Copper IC Design Challenge, Phase I Report, January 2000.
- [7] MSU Team, Copper IC Design Challenge, Phase II Report, August 2000.
- [8] Website:http://www.egr.msu.edu/annweb/cu_contest/