Proceedings of the 28th IEEE
EMBS Annual International Conference
New York City, USA, Aug 30-Sept 3, 2006

Integrated CMOS Electronic System-on-Chip for Medical sensors and Actuators

Fathi M Salem, *Fellow, IEEE*

Abstract— **We present an overview of an integrated low-power, lightweight, compact computing platform dedicated to addressing specific needs in sensing and actuation. The architecture includes an adaptive computing electronic design (Chip) that supersedes the capabilities of present micro-computing paradigms (micro-processors, micro-controllers, and DSPs) in the application domains of process identification, modeling, prediction, and real-time control. In particular, a domain of prominent applications is biological and medical measurements and stimulation.**

I. INTRODUCTION

In measurement and stimulation in biological experiments, there is a staggering amount of data flow. It is envisioned that incoming data/signals is processed by this platform in real-time where the outputs may represent estimates of internal states of the process producing the data, predictions about the process's future states, and/or corresponding control commands. In this view, signals from the (biological) processes, e.g., measurement of a population of neuronal firing, measurement or stimulation of cell membrane potential sites, etc., are continuously supplied to the platform, which in turn performs on-line, instantaneous, learning, adaptation, then processing-- without the traditional coding /programming!

Figure 1 depicts a 3-D micro force and force-rate sensor design based on a micro beam structure using the piezo-like PVDF material for sensing and actuation [1]. This sensor/actuator structure provides a 3-D force measurement, as well as actuations in the X, Y, and Z directions. It design and working principle is detailed in ref. [1]. This structure is intended to be used in accurate 3-D positioning in biological systems, measuring signals as well as actuation or simulation. The motivating idea of the electronic platform is that one does not have a model of the probed tissue, one desires a learning module that can estimate a model from measurements and subsequently infer control commands for positioning of the probe as well as stimulation in response to measurements.

Manuscript received April 24, 2006. This work was supported in part by the MSU IRGP award, app#03-42191, and by IC Tech Inc., Okemos, Michigan.

F. M. Salem is with the department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824 USA (phone: 517-355-7695; fax: 517-353-1980; e-mail: salem@egr.msu.edu).

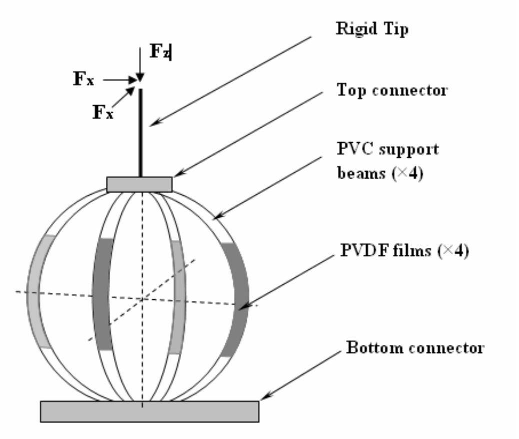


Fig. 1: 3-D PVDF-based Micro Force/Force-rate Sensor

II. OVERVIEW OF THE DESIGN

A. *Why a special architecture:*

The designed adaptive system-on-chip is geared towards applications where either an accurate mathematical model of a process cannot be explicitly developed, or is not reliable, owing to the process' complexity and/or temporal and drift changes. In such cases, models may be "learned" from measurements and data, and subsequent decisions can be executed on-line. Our focus has been on a family of applications where the sensing, decision and actuation may eventually be integrated into a self-contained device or micro-system. We mention a prominent potential application domain: smart probes used in the medical and biological fields for biological cell measurement and stimulation where no reliable model exists and where decisions have to be made on-line. Applications in this domain include, drug injections and microsurgery. In this application domain a huge amount of signals or data are generated, and would require massive processing for standard computing paradigms. Similar challenging problems do exist in pattern matching, feature extraction, and data mining, to name a few.

B. *How it works:*

The vision of self-learning computing engines (machines) is inspired by neurobiology. It, however, must be verified and tested by solid engineering methodologies [2, 3]. Several ideas, incorporated in this effort, are motivated from diverse disciplines including control systems, hardware analysis, VLSI, digital/analog circuits, adaptive learning systems, neural networks, nonlinear signal processing, optimization and optimal control.

As a by-product we present an engine capable of bypassing some of the conventional paradigms of computing, i.e., computing a given mathematical procedure or model via software programming phases, etc. In the critical applications of interest here, which include modeling, identification, prediction, and control from a stream of data/signals, presently, software can only compute off-line, and in a non-real time mode for relatively detailed models. The on-chip self-learning machine presented here would compute by virtue of receiving input-target data in the training mode and by letting the parameters (weights) settle to their steady-state values (within micro- to milli-seconds). Subsequently, the single chip machine would be ready to process new data on the fly limited only by the delay in its interconnect path from its input to its output.

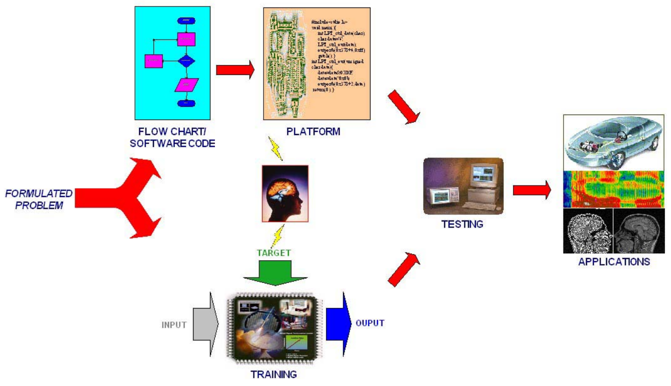


Fig. 2: Conceptual View of the neural computational paradigm

The on-line computation in real-time is achievable by way of a neural network architectural framework resulting in, among other things, solutions to challenging (NP-complete) problems in the general domain of optimization, identification, and classification.

Some of the design features include:
1) For basic implementation architecture, a forward architectural "neural" network, with optional feedback connections, was selected, given that these are the more vastly studied, and are rich enough as to be used for multiple applications. The forward network's processing and the learning module is analog, while the weight storage, control signals are digital given rise to a mixed mode circuit implementation.

2) Although the initial design was outlined for 64-bit I/O to be compatible with modern processors, during the implementation phase it was reduced to a 16-input, 16-output design. This was done as a certain number of pins were required to accommodate the intermediate internal nodes for testing, which may also be used to construct recurrent neural structures, and the supply of target signals during the learning phase. These requirements were further augmented by the addition of chip-level (global) control and program read/write signals.

3) The I/O specification is flexible and can easily be reduced/expanded in our scalable design to inputs compatible with the available packages. The expansion, however, can be achieved by using several chips in cascade and parallel combinations.

4) The chip operates in four different modes: (i) **learn**, (ii) (on-chip) **store**, (iii) **program read/write**, and (iv) **process**.

- *Learn:* It activates the learning process based on the inputs and (desired) reference targets supplied by the application or the user.
- *Store:* Once the user is satisfied with the performance of the network in the learning mode, the *store* mode saves the computed weights in on-chip static digital memory.
- *Program:* This mode was added to give the chip the capability of weight read-out or write-in. The write-in signifies programming the synapses/weights for applications where the chip has already been trained.
- *Process:* The chip is thus ready to be used in the *process* mode where the outputs are generated (i.e., computed) by the forward network.

5) There is no speed/clock specification for the *processing operations* of the chip as the speed is set by the application and the architecture's time-constant(s). In the "*learn*" mode, the speed is determined by the time the network takes to adapt itself to the input-target patterns. In digital implementations of neural networks, this stage takes the longest time, increasing with the complexity and number of training patterns. For the present chip, and based on our experience with IC chip implementations [3,6,10], the time it takes to converge to a solution can be around 100-1000 micro secs-- in conservative 2-micron technology. Storing the weights takes longer than training, but similar to the *learn* mode, it is only executed once in any training session. The *process* mode consumes a delay determined by the largest analog path and the time constant parasitics (in micro-pico seconds) as all the computation/processing is executed instantaneously and in parallel.

6) The (weight) storage memory was necessary to counter the problems of weight decay in generic analog implementations of analog neural implementations. On-chip digital memory was preferred as opposed to off-chip memory as it ensures self-contained operation and speed. After the learning phase, all the steady-state analog weights are converted to digital for block-wise storage using the on-chip ADCs to ensure efficient resource utilization and shorter store mode execution.

7) The resulting chip design would require no programming or coding. In addition to novel architectural designs, the hardware also performs the computational burden by selectively realizing programmability as on-chip self-

configuring and self-learning modules. The resulting chip would operate on 1.5v power source and would consume less than 1.5 m Watt.

8) The chip is mixed mode, mixed signal. We call it mixed mode in the sense that the learning phase is pure analog, while the store mode is analog-digital. This hierarchical view is also maintained in the layout of the chip and in routing interconnects as depicted in Figure 3 below.

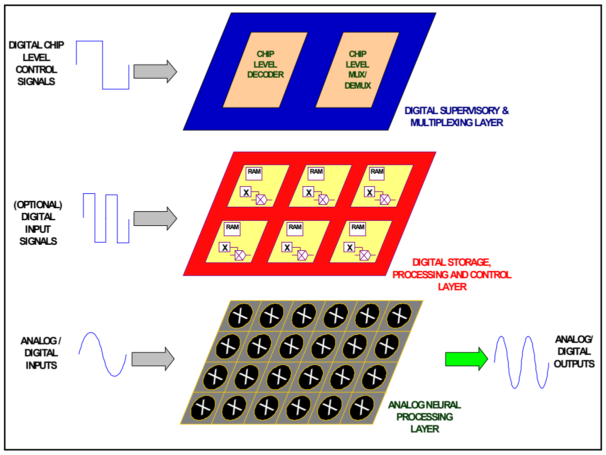

DIGITAL CHIP LEVEL CONTROL SIGNALS
CHIP LEVEL DECODER
CHIP LEVEL MUX/ DEMUX
DIGITAL SUPERVISORY & MULTIPLEXING LAYER
(OPTIONAL) DIGITAL INPUT SIGNALS
RAM
X
DIGITAL STORAGE, PROCESSING AND CONTROL LAYER
ANALOG / DIGITAL INPUTS
ANALOG/ DIGITAL OUTPUTS
ANALOG NEURAL PROCESSING LAYER

Fig. 3: Layered architecture based on signal types: processing, memory, multiplexing controls

The remaining part of the paper is organized as follows: *Section 2* focuses on the architectural designs of the *core* building blocks, which include primarily the synaptic and control cells

III. CHIP FLOORPLAN

The Chip is designed to be highly modular. The concept of this modularity was infused into the design right from the conception phase and realized into a granular structure of the chip as the design progressed.

The whole chip is composed of four cascaded building blocks and their interconnection. In addition, there are some global digital logic control elements (see Fig. 1). The first and the last of these blocks are routed to the padframe of the chip, as the input and the output layers of the neural engine.

A. *Structure of the Building Block*

The main building block of the chip comprises of a 16×18 array of building cells. The first 16×1 cells are the digital cells, while the remaining 16×17 array is formed of synaptic cells. This array of synaptic cells on the output side is padded by another column of buffers for signals to be connected to other building blocks/padframe. This stage also includes difference amplifiers used for determination of error, i.e. difference between target inputs and block outputs for tuning the local weights.

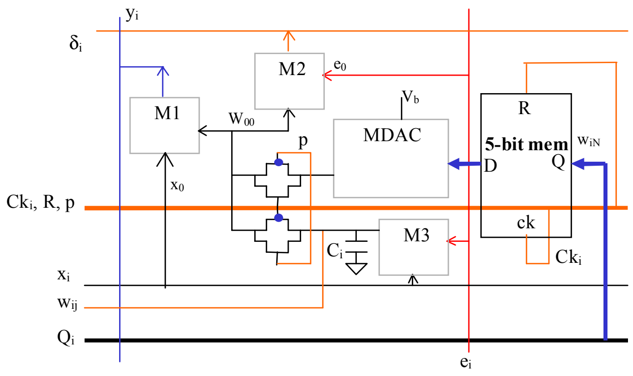
The Synaptic array can be decomposed into cascaded processing stages. Each processing stage is composed of 16 neurons built using (x17) synaptic cells and a sigmoid function. Current bus bars are used to collect output currents from each cell in a processing stage. These bus bars run horizontally and vertically for common row/column outputs. Separately designed sigmoid functions and CMOS linear resistors are used to convert these currents to voltages.

B. *The Synaptic Cell:*

Each synapse cell is composed of three analog Gilbert multipliers, a storage capacitor, a linear resistor, a set of transmission gates and 5 data flip-flops (for local memory).

FIG. 4: Synapse Cell Structure

In the *learning* stage, the processing multiplier M1 multiplies an input signal $\mathbf{x_i}$ and the synapse weight $\mathbf{W_{ij}}$, stored on the capacitor $\mathbf{C_i}$ and selected by setting **p** to high,



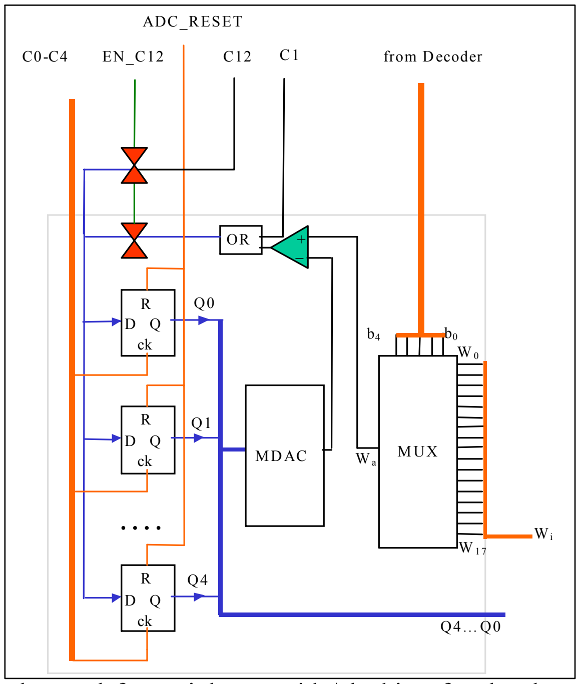
to compute the output current $\mathbf{y_i}$. The back-propagating multiplier M2 multiplies the error signal $\mathbf{e_i}$ with the weight $\mathbf{W_{ij}}$ to calculate the feedback current component $\boldsymbol{\delta_i}$. The weight update multiplier M3 multiplies the input signal $\mathbf{x_i}$ and error signal $\mathbf{e_i}$ to determine the updated weight. The current-to-voltage conversion for this multiplier is done locally using a MOS capacitor. In the *store* mode, the locally stored weight value, using capacitor C_i, is converted to its digital representation. This conversion takes place in the corresponding digital cell. The converted weights are stored locally in the 5-bit memory. In the *processing* stage, the Multiplying DAC (MDAC) converts the stored digital bits to the equivalent weight representation. At this stage the polarity of signal **p** is adjusted so that the output of the MDAC is used as the local weight instead of the charge stored on the capacitor. The **R** signal is connected to reset all local memory of data flip-flops, enabling the local memory to be reset independent of digital cell operation. The signal Ck_i is the clock to the data flip-flops for storage of converted weights.

C. *The Digital Cell*

Each digital cell contains primarily an ADC. Sharing a common ADC for all the synaptic cells in a row reduces the number of ADCs required for conversion of weight to $\sqrt{n}$. This configuration is capable of achieving satisfactory conversion time for all the weights. See Fig. 4.

FIG. 5: Digital Cell Structure

The selection of a weight in a row is controlled by chip level decoders, which executes in parallel for all rows in all the building blocks. The selected weight is *compared* with the MDAC output, this output is *ORed* with a signal **C1**, which



can be used for weight override/clocking for the data-flip-flops. The external signals **C0-C4** control the clocking of the data flip-flops. The external signal **C12** in conjunction with the **EN_C12** signal connected to the transmission gate, provide this cell the capability to program the weights. The signal **EN_C12** is generated by another set of chip-level decoders, which are supplied with the row/column selection during "programming."

IV. Design and Layout of Sub-Cells

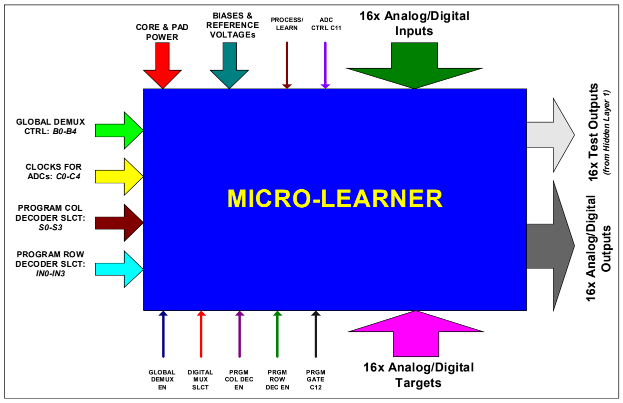
All the components for this chip were custom-designed. All the component circuits were designed on *Star-Hspice* using BSIM Level-49 models supplied by SRC/UMC. Avant's software *SUE* and *METAWAVES* was used for the schematic entry and waveform viewing, respectively. Initial layout of these sub-circuits was carried out in *Tanner Tools* but the verification and LVS were performed using *Cadence Tools*. Design details for some of these components are provided in [1,5]. In this paper we present our design of two more sub-circuits that include the Multiplying MDAC [5] and the two-stage data flip-flop. All components use Vcc at 1.5V.

By design, the neural system-on-chip (nicknamed *Micro-learner*) can accept and process signals with bandwidths greater than 100 MHz, this limit cannot be tested by the current setup. Therefore, after performing some initial validation of the chip operation, the platform for testing will be changed. The testing results for the chip will be reported in subsequent papers after its fabrication.

FIG. 6: Depiction of the I/O and Control Bits for Micro-Learner

References

[1] F. Salem, C. Radcliffe, N. XI, Y. Shen, E. Motato, Final Technical Report, IRGP grant, MSU, December 2005.
[2] K Waheed, F Salem, "MICRO-LEARNER: A self-programming chip for real-time estimation, prediction, and control," Michigan State University, Inventor, U.S. 2004, patent pending.
[3] Khurram Waheed and Fathi M. Salem, "A Mixed Mode Self-Programming Neural System-on-Chip for Real-Time Applications"; *IJCNN 2001*.
[4] Khurram Waheed and Fathi M. Salem, "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
[5] F. M. Salem, 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.
[6] MSU Team, Copper IC Design Challenge, *Phase II Report. 2001*.
[7] N. K. Bose and P. Liang, Neural Network Fundamentals with graphs, algorithms, and applications, McGraw-Hill, 1996.
[8] Simon Haykin, Neural Networks: A Comprehensive Foundation, Macmillan College Publishing Company, Inc., 1999.
[9] F. M. Salem, 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.


CORE & PAD POWER
BIASES & REFERENCE VOLTAGEs
PROCESS/ LEARN
ADC CTRL C11
16x Analog/Digital Inputs
GLOBAL DEMUX CTRL: B0-B4
CLOCKS FOR ADCs: C0-C4
PROGRAM COL DECODER SLCT: S0-S3
PROGRAM ROW DECODER SLCT: IN0-IN3
MICRO-LEARNER
16x Test Outputs (from Hidden Layer 1)
16x Analog/Digital Outputs
GLOBAL DEMUX EN
DIGITAL MUX SLCT
PRGM COL DEC EN
PRGM ROW DEC EN
PRGM GATE C12
16x Analog/Digital Targets