VLSI Implementation of a Functional Neural Network

Dimokritos A. Panagiotopoulos[%], Sanjeev K. Singh[+] and Robert W. Newcomb[#]

[%] Automation Department
School of Technology Applications
Technological Institute of Thessaloniki
Mail Address: P.O. Box 10403,
541 10 Thessaloniki, GREECE
E-mail: dpanag@filippos.techpath.gr

[+, #] Microsystems Laboratory, Electrical Engineering Dept.,
University of Maryland, College Park, MD 20742, USA
[#] EE Dept., POSTECH, Pohang, Korea
[+] E-mail: sksingh@eng.umd.edu
[#] E-mail: newcomb@eng.umd.edu Phone: [1]-(301)-4053662
[#] URL: http://www.ee.umd.edu/newcomb/mslab.html

Abstract

Focus is placed upon a modular current-mode VLSI implementation of the Functional Artificial Neural Network. The modules used are multipliers, exponential amplifiers, and analog memory cells.

1. Introduction

This article focuses on the integration of a discretized analog functional artificial neural network (FANN) [1], [2], using current-mode circuitry. The FANN under consideration, implements an operator that is an optimal L_2 interpolation map [2] of a function $\mathbf{u}(\cdot)$ into another function $\mathbf{y}(\cdot)$ over a time-interval I, according to the formula:

$$\mathbf{y}(t) = V(\mathbf{u}(\cdot)) = \sum_{j=1}^{N} \mathbf{c}_j(t) \cdot \exp\left(\frac{1}{r} \cdot \sum_{k=1}^{N_S} \mathbf{u}^{j^T}(k)\mathbf{u}(k)\Delta t\right) \quad (1)$$

for input over discretized time of N_S samples. In general, $\mathbf{u}(t)$ and $\mathbf{y}(t)$ are vectors of real values at time t, i.e. $\mathbf{u}(\cdot)$ consists of n elements being functions of t, and $\mathbf{y}(\cdot)$ consists of m elements being functions of t, resulting in a multiple-input multiple-output functional artificial neural network. The patterns $\{\{\mathbf{u}^j(t),\ t \in I\},\ 1 \le j \le N\}$ are exemplar inputs for training the network under supervision [1], [2]. In the implementation of a FANN, the exemplar training inputs $\mathbf{u}^j(\cdot)$ are represented by N_S samples in the time-interval I, and treated as weight vectors for the first of the two layers of the FANN. The $\mathbf{c}_j(t)$ is a vector of m synaptic weights at t in the second layer of the FANN; during training the $\mathbf{c}_j(\cdot)$ is calculated for discrete time instances, or else it is modelled as a continuous vector function. Given an unknown pattern $\{\mathbf{u}(t),\ t \in I\}$, the FANN evaluates the similarity of this pattern to each of the exemplars $\{\{\mathbf{u}^j(t),\ t \in I\},\ 1 \le j \le N\}$, and estimates $\mathbf{y}(\cdot)$ through weighted averaging of high selectivity over the corresponding exemplar actions $\mathbf{y}^j(\cdot)$, where the degree of selectivity may be adjusted through r. Thus, if $\mathbf{u}(.)$ is very similar to $\mathbf{u}^k(.)$ and quite dissimilar from all other exemplars $\mathbf{u}^l(.)$, then $\mathbf{y}(\cdot)$ is approximated by $\mathbf{y}^k(\cdot)$.

The FANN may learn, under supervision, mappings of functions to functions rather that just function mappings of points to points as with conventional multilayer perceptrons. Subsequently, a FANN may be used in applications such as *planning* [3] where a sequence of input events map to a sequence of output actions, or *multimode signal processing* where the transformation applied on a portion of signal may depend on that signal segment's shape, thus allowing to select the transformation kernel according to the signal's properties.

The various signals in a FANN's implementation are represented here either as currents, or current differences. Current-mode allows for more efficient manipulation of summation and more compact multiplication circuitry.

2. The Block Structure of FANN

The FANN may be realized as a two-hidden layer network [1], where the synaptic weight associated with the connection from $\mathbf{u}(k)$ to the jth neuron of the first layer represents the value $\mathbf{u}^j(k)$ of the jth exemplar input. The output of each neuron in the first layer being exponential over the sum of weight-input products. The second hidden layer consists of a single linear neuron, with synaptic weights $\mathbf{c}_j(t)$ and output equal to $\mathbf{y}(t)$. The block structure of the FANN realization is shown in figure 1, for the case of single input - single output.

3. Definition of Current-mode Building Blocks

3.1. Introduction

The FANN realization is based on the following current-mode analog modules: *multiplier*, *exponential amplifier* allowing for signed input [5], *summation* (being trivial as wire connections), and *analog memory cells* [6]. Also, voltage-to-current converters may be used in order to allow a FANN to receive input from conventional voltage-mode external circuitry. An introduction to each of these modules follows, with their I/O block diagrams being defined in figure 2. The *filled* arrows in those diagrams should be interpreted as follows: a bi-directional vector means that current can flow in any direction (therefore it has a signed value) while a unidirectional vector means that current can flow in that direction only.

3.2. The Multiplier

This module is an analog four quadrant current-controlled current-divider multiplier circuit [4] which implements the following transfer function:

$$I_{OUT} = K \cdot \frac{I_X \cdot I_Y}{I_b} \qquad (2)$$

where $K \approx 1$. The I_b is internal biasing current; having $I_b = 1 \times (units)$, where '*units*' may be chosen to be '10^{-3}', or '10^{-6}', etc. (Amperes implied for currents), and if $I_X = x \times (units)$, $I_Y = y \times (units)$, then $I_{OUT} = x \cdot y \times (units)$. The implementation of this circuit requires that I_X, I_Y, be expressed as differences of currents, where $I_X = I_X^+ - I_X^-$ and $I_Y = |I_Y^+| - |I_Y^-|$. The superscripts, +, -, denote the subtrahend and minuend components respectively in the difference, and not polarity of currents; e.g. the currents, I_X^+, I_X^-, can be of either sign. There is an additional requirement about I_X : the specification in [4] is $I_X = -(I_X^+ - I_X^-)$, also assuming a minus sign in front of K in Eq (2), and the requirement is that $I_b = I_X^+ + I_X^-$; in our implementation, this input is specified as $I_X = I_X^+ - I_X^-$ and the requirement is that $I_X^+ + I_X^- = 0$; under this constraint, we have $I_X^+ = \frac{I_X}{2}$ and $I_X^- = -\frac{I_X}{2}$. This modification facilitates the design of the output stages of those modules that connect to a multiplier, and it results in reduced power consumption when the summed current from more than one module enters the multiplier. There are two I/O models available for this multiplier module, which differ in their output stage, and are shown in figures 2(a), 2(b).

3.3. Exponential Circuit

This module is an analog four quadrant current-divider exponential amplifier circuit with current-controlled amplitude modulation [5] which implements the following transfer function:

$$I_{OUT} = I_X \cdot \exp(\frac{R_a}{V_T} \cdot I_Y) \qquad (3)$$

where $V_T \approx 26\times10^{-3}$ Volt and R_a is the value of an internal resistor (expressed in Ohms). If $I_X = x \times (units)$, $I_Y = y \times (units)$, then $I_{OUT} = x \cdot \exp(K \cdot y) \times (units)$, where $K = \frac{R_a}{V_T} \times (units)$. The implementation of this circuit requires that I_X is expressed as a difference of currents, i.e. $I_X = |I_X^+| - |I_X^-|$, while I_Y is expressed directly. There are two I/O models available for this exponential module, which differ in their output stage, and are shown in figures 2(c), 2(d).

3.4. Summation

There is no special circuitry for this module. Since the quantities to be summed are currents, this module represents a direct connection of wires.

3.5. Analog Memory Cell

A Read/Write analog memory is needed in order to store the synaptic weights in the neural chip. An appropriate analog memory cell has been developed in [6] and is used in the implementation of the analog memory here. There are two I/O models available for this memory cell, which differ in their output stage, depending on whether the memory cell connects to the I_X input of an exponential module, or the I_Y input of a multiplier. Their I/O block diagrams are shown in figures 2(e), 2(f), respectively. The output current is expressed as $I_{OUT} = |I_{OUT}^+| - |I_{OUT}^-|$. This cell is dynamically refreshed and allows the continuous value of a current to be stored. Part of the refreshing mechanism is in the cell, allowing such cells to be refreshed independently of each other and asynchronously as needed, and part of the refreshing mechanism is common to all such cells, interfacing with them through the control input REFR. The input $\overline{\text{WE}}$ controls writing the input current I_{IN} in the cell.

3.6. Considerations on the Representation of Math Variables with Currents

The elements of **u**(t), **c**(t), **y**(t), and r, in Eq (1), being real variables could take on values in the interval [$-\infty$, $+\infty$]. In a physical implementation, these mathematical variables are implemented through physical quantities, such as currents in this case, which may only be defined in a subinterval [$-(I_{max} \times (units))$, $+(I_{max} \times (units))$] of the real line $\Re$, due to practical limitations (power consumption, overheating, size). This forces confinement of the mathematical variables to a finite subinterval, [$-X_{max}$, $+X_{max}$], of $\Re$, where these variables are evaluated in all practical applications. In this case the exponential term is the most critical for overflows, and we would like to choose

$$X_{\max} = \max\{\max_j\{\exp(\frac{1}{r} \cdot \int_{\tau \in I} \mathbf{u}^{j^T}(\tau)\mathbf{u}(\tau)d\tau)\}, \max_j\{|\mathbf{c}_j(t)| \cdot \exp(\frac{1}{r} \cdot \int_{\tau \in I} \mathbf{u}^{j^T}(\tau)\mathbf{u}(\tau)d\tau)\}\} \qquad (4)$$

Then, the value i_x, of the current $i_x \times (units)$, that represents the value of the mathematical variable x in the implementation, may be defined as

$$i_x = x \cdot \lambda \ , \quad \text{where} \quad \lambda = \frac{I_{\max}}{X_{\max}} \ , \qquad (5)$$

and Eq (1) is implemented by replacing the mathematical quantities $\mathbf{y}(t)$, $\mathbf{u}(t)$, $\mathbf{c}_j(t)$, $\mathbf{u}^j(t)$, with $\mathbf{i}_y(t)$, $\mathbf{i}_u(t)$, $\mathbf{i}_{c_j}(t)$, $\mathbf{i}_{u^j}(t)$, respectively. Since $\exp\left(\frac{i_x}{r}\right) = \exp\left(\frac{x \cdot \lambda}{r}\right) = \exp\left(\frac{x}{r_{effective}}\right)$, the effective value of r in Eq (1) becomes

$$r_{effective} = \frac{r}{\lambda} \ . \qquad (6)$$

The r may either be implemented by choosing a proper value of R_a in Eq (3), such that

$$\frac{1}{r} = K = \frac{R_a}{V_T} \times (units), \qquad (7)$$

or it may be implemented as

$$\frac{1}{r} = \left(\frac{R_a}{V_T} \times (units)\right) \cdot I_r \ , \qquad (8)$$

having a constant value for R_a and a variable I_r that allows more flexibility in the definition of r.

In this BiCMOS implementation of modules we use MOSFETs in saturation, with relatively large currents, assuming that $(units) = 10E\text{-}3$ and $I_{\max} = 0.5$, in order to have a large dynamic range for experimentation and good chances for success in the IC implementation. Since large currents force relatively large size circuitry, also reducing the number of components that can be integrated, and thus the size of an integrated FANN in terms of neurons, the final implementation should assume $(units) < 10E\text{-}3$, probably using CMOS in subthreshold.

4. Integration of FANN using Current-mode Building Blocks

A typical path from input to output in the FANN, is shown in figure 3(a). Implementations of this path, using the modules that were presented in section 3, are shown in figures 3(b) and 3(c). In figure 3(b) the exponent's coefficient r is implemented as in Eq (7), while in figure 3(c) the r is implemented as in Eq (8). The multiplication stage that follows the exponential stage in figure 3(c), and shown with dotted lines, may be omitted in a single output FANN and becomes as in figure 3(b). If y(t) is an m-element vector, then the FANN has m outputs and the dotted stage must be used, since each output has its own set of *c(t)* weights. The voltage-to-current converters, labelled "V-I" in figure 3, are optional and only needed in the case that voltage input is required, and should be provided in a separate chip.

A full implementation of the FANN, is demonstrated in figure 4. Three types of separate integrated circuits (chips) are identified as 'A', 'B', 'C', in that figure. The chip 'A' is a complete FANN having N paths, like those shown in figure 3, that can accept a single function input sampled over N_s points at maximum, and being capable of supporting a single output. If multiple outputs are required, then the chip 'B' may be added, as it is shown in figure 4. This chip contains $m \cdot N$ multipliers with associated memory cells, and the resulting FANN can have up to m outputs; the number of outputs may further be increased by adding more chips of this type. If the number of input samples is more than N_s, then the chip 'C' may be added, as it is shown in figure 4. This chip contains $n \cdot N \cdot N_S$ multipliers with associate memory cells, and the resulting FANN can accept a single function input sampled over $(n + 1) \cdot N_S$ points at maximum. Alternatively, this addition of chip 'C' allows FANN to accept an (n+1)-element vector function input sampled over N_s points at maximum. More chips of type 'C' may be added in this manner, in order to further extend both the allowable number of elements in the input vector and the allowable number of sample points.

The memory cells that connect to the input of the exponential amplifiers in the last stage of chip 'A', or to the input of the multipliers in chip 'B', can hold values of the element functions *c(t)* for a single instance of time. All of this assumes that somewhere *c(·)* is available for which a storage module is needed.

5. Conclusions

The VLSI implementation of a two-hidden layer discretized Functional Artificial Neural Network (FANN) has been demonstrated. A chip-set has been defined that implements the FANN, while it allows for expandability in the number of its inputs and outputs, as well as the number of neurons and connections in each hidden layer. Use of current-mode circuitry has resulted in compact multiplication circuitry and efficient manipulation of summation. The building components (multiplier, exponential amplifier, analog memory cell) of the FANN have been implemented successfully, through MOSIS, using BiCMOS technology. The next step is to fabricate the three chips that are shown in figure 4, and build a prototype FANN.

References

[1] R. W. Newcomb and R. J. P. de Figueiredo, "A Multi-Input Multi-Output Functional Artificial Neural Network," to appear in *Int'l Journal of Intelligent & Fuzzy Systems.*

[2] R. J. P. de Figueiredo, "The OI, OS, OMNI, and OSMAN Networks as Best Approximations of Nonlinear Systems Under Training Data Constraints," *Proceedings of IEEE ISCAS '96,* Atlanta, May 1996, pp. 349-352.

[3] D. A. Panagiotopoulos, R. W. Newcomb and S. K. Singh, "Planning with a Functional Neural Network Architecture," *manuscript in preparation.*

[4] M. Herpy, *Analog Integrated Circuits,* John Wiley & Sons, Chichester, 1980.

[5] D. A. Panagiotopoulos and R. W. Newcomb, "A Current-mode Exponential Amplifier," *manuscript in preparation.*

[6] D. A. Panagiotopoulos and R. W. Newcomb, "A Current-mode Analog Memory Cell," *manuscript in preparation.*

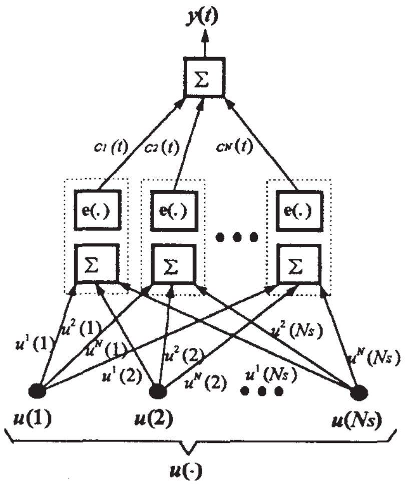

Figure 1: Discrete input FANN for single input - single output.

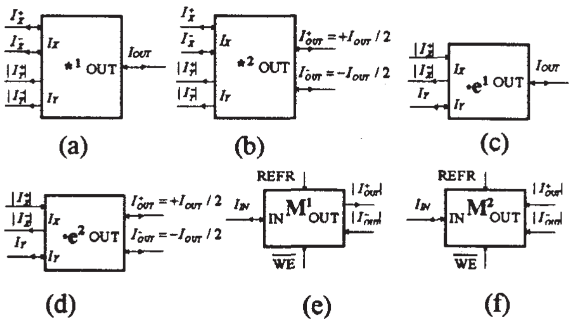

Figure 2: Black box definition of the multiplier, exponential, and memory cell modules.

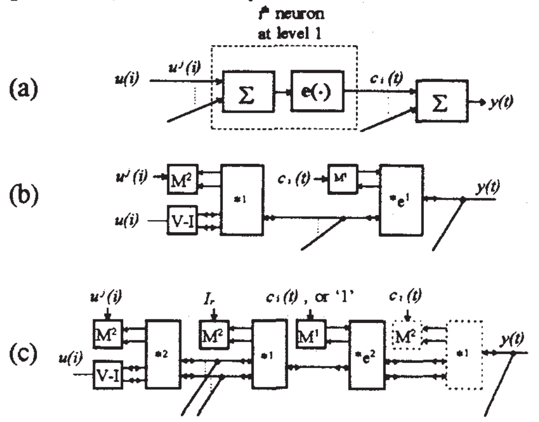

Figure 3: A path from input to output in the FANN: (a) logical block diagram; (b), (c) physical block diagram realizations.

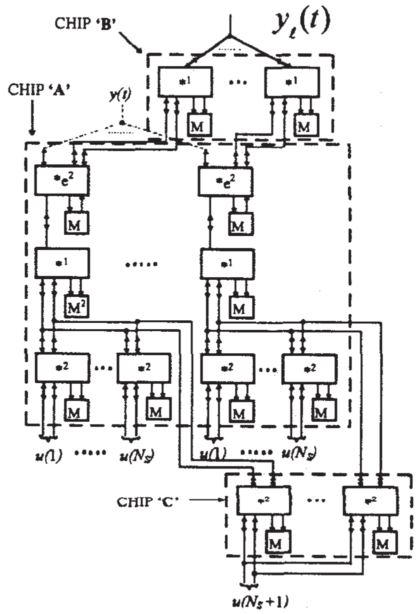

Figure 4: Definition of 3 types ('A', 'B', 'C') of chips and their interconnectivity for implementation of large FANNs.