APPLICATION OF MULTI-ZERO ARTIFICIAL NEURAL NETWORK TO THE DESIGN OF AN M-VALUED DIGITAL MULTIPLIER

Chia-Lun J. Ru, Professor
Electrical Engineering Department
Southern Illinois University at Carbondale
Carbondale IL 62901

ABSTRACT
The multi-zero neural network is a feedback artificial neural system consisting of N neurons. Each neuron is a nonlinear amplifier with input-output response function equal to a polynomial function containing 2M+1 real zeros. A very unique property possessed by this nonlinear feedback system is that if the connection matrix is programmed correctly, any N-bit ANALOG input vector will always be converged to an N-bit M-valued DIGITAL vector at the output. This output will be locked-in in place (or it can be MEMORIZED) even when the input is removed.

Because of these unique properties, the network can be used in the design of a fast, M-valued computing system. This paper reports the origin of this multi-zero system, the analysis of its properties, and the design of a fast, M-ary (or M-valued) digital multiplier using this system.

I. INTRODUCTION
The study of multi-valued logic (MVL) systems grows rapidly in recent years at the demands for higher speed, less circuit complexity, and more compact space in the modern digital system designs. The traditional two-valued or binary systems appear to reach a ceiling at the present time facing these application demands. MVL systems, on the other hand, seem to have the advantage of alleviating these pressing problems. In MVL designs, the development of new mathematical rules and new digital systems are mostly based on the study of properties of some MVL building blocks. These building blocks include units such as successor, predecessor, cycler, literal (input-gating) operator, complement operator, Max operator, Min operator, and T-gate (or MVL-controlled multiplexer) [1, 2]. Any general M-valued, N-variable function can then be synthesized using these building blocks [1, 3]. Parallel to this general design approach, practical M-valued devices using MOS units have been studied. Higuchi, al. et. designed a parallel quarternary signed-digit adder and a parallel quarternary signed-digit multiplier using current-mode MOS modules such as current mirrors, threshold detectors, and bi-directional current input units [4, 5].

Summarizing all these MVL arithmatic studies, one could see that designs of most of these systems seem to follow the SAME philosophy as that of a binary system. That is, one first synthesizes a required system response (an M-valued function) and then minimizes it and implements it with some M-valued building blocks. This article, on the other hand, is intended to look into the MVL design problem from another point of view: the design of an M-valued digital...
arithmetic system using an M-zero
ARTIFICIAL NEURAL NETWORK [6-8].

II. MULTI-ZERO FEEDBACK NEURAL SYSTEM

Fig. 1 is the schematic of an N-neuron feedback artificial neural system. Each triangle there represents a neuron (or a nonlinear amplifier) with its input, output voltages $x_i, y_i$ related by the following dynamic transmission equation:

\[
\frac{dy_i}{dt} = f(x_i), \quad i = 1 \text{ to } N \tag{1}
\]

$f$ in (1) is the nonlinear amplifier (neuron) response function and the time derivative in (1) is due to the passage time delay for the signal to travel through the amplifier. As shown in Fig. 1, all $y_i$'s are fed back to the inputs of the neurons through a linear network $[a_{ij}]$. Each $a_{ij}$ is a BIPOLAR resistor with its resistance adjustable to any positive or negative values. Bipolar resistor can be realized by using an inverted amplifier. $[a_{ij}]$ matrix has the following function on the $x, y$ voltages.

\[
x_i = \sum_{j=1}^{N} a_{ij} y_j \tag{2}
\]

If one writes (1) and (2) in matrix forms, the following is obtained.

\[
d\bar{y}/dt = f(\bar{x}) \tag{3}
\]

\[
\bar{x} = \bar{A} \bar{y} \tag{4}
\]

where $f(\bar{x})$ is an $N \times 1$ column matrix with each element equal to $f(x_i)$. $\bar{A}$ is an $N \times N$ square matrix representing $[a_{ij}]$. Eliminating $\bar{y}$ from (3) and (4), we have:

\[
d\bar{x}/dt = \bar{A} f(\bar{x}), \quad \text{or,}
\]

\[
dx_i/dt = \sum_{j=1}^{N} a_{ij} f(x_j), \quad i = 1 \text{ to } N
\]

..............................................(5)

This is the control equation of the artificial neural system shown in Fig. 1.

(5) is a set of coupled nonlinear ordinary differential equations. The solution can be (asymptotically) stable if $\bar{A}$, $f$ are properly chosen. The steady-state output vector may be taken either as $\bar{x}_f$ or $\bar{y}_f$, which is the asymptotic solution of (5) and (4). This output state will be locked in even when the input $\bar{x}_0$ (see Fig. 1) is removed because $\bar{x}_0$ only changes the initial state of the system. This output will change only when the system receives another $\bar{x}_0$. The new $\bar{x}_0$ will introduce a new initial condition to (5) and the system will then search for a new stable output state to settle down. The solution of (5), the input-output mapping relations, and the criteria for stable output to exist were analyzed and reported [9]. The important result is that if the connection matrix of this network is programmed correctly, a very unique property is seen to exist in this network:

INTEGER CONVERGENCE PROPERTY — Any N-bit analog input $\bar{x}_0$ will
always converge to an N-bit, M-ary digital output $X_f$ whose components are equal to any of the integers (0,1,...,M).

Because of this property, the circuit is very attractive to be used in fast M-ary (or M-valued) computing system designs. Our attention will be focused to one particular example discussed in the next section — the design of a fast digital multiplier using this novel neural circuit.

III. M-ARY DIGITAL MULTIPLICATION USING MULTI-ZERO NEURAL NETWORKS

In the following, a scheme similar but not equal to the carry-save multiple-addition in binary multiplication [10] is described.

PART 1. Un-quantized Analog to M-ary Encoder (UAME)

In the following we will discuss an unquantized analog to M-ary conversion scheme that will convert any analog voltage $S$ to a 4-ary number (ABC) with $A \leq 2$, $B,C \leq 3$. We assume here that $S \leq 47$. The reason for this limitation is seen at the end of PART 3 below.

Fig. 2-1 is a comparator circuit that will yield two outputs $V_{11}$ and $V_{12}$ under the input $S$. The comparator supply voltage is 5 volt. The input-output relation of this circuit is shown in Table 1 below.

<table>
<thead>
<tr>
<th>$S$</th>
<th>$V_{11}$</th>
<th>$V_{12}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S &lt; 16$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16$ - $32$</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>32$ - $S$</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

If we connect these outputs to an analog adder (Fig. 2-b) such that the output is $(V_{11} + V_{12})/5$, then this output is the digit $A$ in the 4-ary representation of $S$, because $S = (ABC)_4 = (16A + 4B + C)_{10}$ and $A^c = 2$.

Now if we connect $S$ and $A$ to another analog adder (Fig. 2-c) such that the output $S'$ is $S = 3 - 16A$, then $S' = 4B + C$. We can then design a similar comparator circuit as shown in Fig. 2-d to Fig. 2-f to read the last two 4-ary digits $B$ and $C$.

Notice that we used volts here to implement the digital numbers, i.e, n volt = digit n, which is, of course, not necessary. In general, one can use any voltage level to implement a digit "one" in the 4-ary representation by using appropriate voltage scaling in the design.

The AME designed in this way is purely analog and it is subject to accuracy degradation after a few stages of operations. This is very common in analog circuits and it is the main reason that analog arithmetic processors are not used in digital computations. The following is a scheme used to recovered the digital accuracy by means of the M-zero neural networks.

PART 2. Quantized Analog to M-ary Encoder (QAME)

If we connect the 3-bit output of the UAME discussed above to a 3-bit 4-zero neural system discussed in section II, then as shown in Fig. 3-a, the output will be an ACCURATE 3-bit 4-ary number because of the domain-attraction or integer-convergence property of the M-zero neural system. We can also adjust the range of this convergence by
adding a bias voltage $V_b$ to the input $S$ as shown in Fig. 3-b.

**PART 3. M-ary Digital Multiplier**

The multiplier discussed here is similar (but not equal) to the conventional carry-save-addition used in binary multiplications. The crucial difference is that we use analog multipliers and analog adders to get all partial products and "partial sums" in analog form. Then we use neural network quantized analog-to-M-ary encoder's (QAME's) to convert these analog results to M-ary digital numbers before carry operation is executed. The inaccuracy introduced by analog processors will be eliminated because of the unique integer convergence property of the multi-zero neural network. The following example may serve the purpose for demonstrating the principal idea.

Suppose we wish to multiply two 3-bit, 4-ary numbers, 333 and 333. (We use two maximum 3-bit numbers here to demonstrate the maximum lengths of carry-propagation.) As shown in Fig. 4-a, we first get all partial products $(9, 9, . . . .)$ and "partial sums" $(S_1, S_2, . . . S_5) = (9, 18, 27, 18, 9)_{10}$ by ANALOG multipliers and ANALOG adders. These $S$'s expressed in 4-ary digital forms are shown in the bottom line of Fig. 4-a. It is to be noted that the maximum carry-feed-forward is two digits as shown by the central three brackets in the last line of Fig. 4-a. To implement this carry feed-forward operation and to obtain the final result, we refer to Fig. 4-b. The circuit in Fig. 4-b accepts the five analog partial sums $(S_1, . . . S_5)$ and each is converted into a 4-ary number by a QAME. The carry digits are connected forwardly to the next one channel or next two channels on the left according to the carry digits shown in the last line of Fig. 4-a. These carry digits are added to proper $S$'s by analog adders as shown in Fig. 4-b. The digits $(P_1, P_2, . . . P_6)$ taken out from the bottom of the circuit shown in Fig. 4-b are then the digits of the final product in the 4-ary (quarternary) form.

It is seen that the carry operation should be quite accurate here because all partial sums are re-digitized and any analog inaccuracies are rounded off by the M-zero neural network before the carry operation is executed. The overall accuracy of this multiplier should be quite good (close to conventional binary digital multiplier) because it is a DIGITAL OPERATION. On the other hand the speed of multiplication should be very high because it is NOT a sequential process. It is a combinatorial or FREE-RUNNING feed-forward amplification process. The time for reaching the steady state output is mainly the transient time for any analog amplifier to reach steady state. The latter is generally much shorter than the total clock time required for completing the sequential process. In addition to this, the fact that the system is mostly parallel and the system is an M-ary (M-valued) operation will also contribute significantly to the enhancement of the speed of the total operation.

Notice that we only use a 3-bit by 3-bit 4-ary multiplication here as an example to demonstrate the principle. Actually, with the UAME depicted in Fig. 2 which converts any $S$ to $(ABC)_4$ with $A<=2$, the maximum $S$ that can be converted is
From Fig. 4-a, we see that the maximum partial sum for a k-bit by k-bit 4-ary multiplication is 9k. Therefore the optimum k to satisfy 9k < 47 is k = 5. Consequently, for an optimum design using minimum number of neural circuits, the 4-ary multiplication should be 5-bit by 5-bit, which is equivalent to a 10-bit by 10-bit binary multiplication.

IV. CONCLUSION
An M-ary digital multiplier employing artificial multi-zero neural networks and elementary analog arithmetic units has been derived. This multiplier should be accurate becuase its main arithmetic process is digital while the speed should be very high because it is a free-running, parallel, and M-ary operation.

ACKNOWLEDGEMENT
The author would like to acknowledge the Illinois State Technology Challenge Fund Office for providing a grant to the SIUC Electrical Engineering Department. Part of the work reported in this article was supported by this grant. The author is also grateful to Dr. M. Manzoul for constructive discussions.

REFERENCES


