A HIGH-SPEED NEURAL ANALOG CIRCUIT FOR COMPUTING THE BIT-LEVEL TRANSFORM IMAGE CODING

P. R. Chang, K. S. Hwang, and H. M. Gong
Dept. of Electrical Communication Engr. and Center for Telecommunication Engr.
National Chiao-Tung University, Hsin-Chu, Taiwan, R.O.C.

Abstract

This paper presents a Hopfield-type neural network approach which leads to an analog circuit for implementing the bit-level transform image. Different from the conventional digital approach to image coding, the analog coding system would operate at a much higher speed and requires less hardware than digital system. In order to utilize the concept of neural net, the computation of a two-dimensional DCT-based transform coding should be reformulated as minimizing a quadratic nonlinear programming problem subject to the corresponding 2's complement binary variables of 2-D DCT coefficients. A novel Hopfield-type neural net with a number of graded-response neurons designed to perform the quadratic nonlinear programming would lead to such a solution in a time determined by RC time constants, not by algorithmic time complexity. A fourth order Runge-Kutta simulation is conducted to verify the performance of the proposed analog circuit. Experiments show that the circuit is quite robust and independent of parameter variations and the computation time of an 8×8 DCT is about $1ns$ for $RC = 10^{-8}$. In practice, programmable hybrid digital-analog MOS circuits are required to implement the neural-based DCT optimizer. The circuit techniques are based on extremely simple and programmable analog parameterized MOS modules with such attractive features as reconfigurability, input/output compatibility, and unrestricted fan-in/fan-out capability.[1]

1 Introduction

The goal of transform image coding is to reduce the bit-rate so as to minimize communication channel capacity or digital storage memory requirements while maintaining the necessary fidelity of data. The discrete cosine transform (DCT) has been widely recognized as the most effective among various transform coding methods for image and video signal compression. However, it is computationally intensive and is very costly to implement using discrete components. Many investigators have explored ways and means of developing high-speed architectures [1], [2] for real-time image data coding. Up to now, all image coding techniques, without exception, haven been implemented by digital systems using digital multipliers, adders, shifters, and memories. As an alternative to the digital approach, an analog approach based on a Hopfield-type neural networks [3], [4] is presented.

Neural network models have received more and more attention in many fields where high computation rates are required. Hopfield and Tank [3], [4] showed that the neural optimization network can perform some signal-processing tasks, such as the signal decomposition/decision problem. Recently, Culhane, Peckerar, and Marrian applied their concepts to discrete Hartley and Fourier transforms. They demonstrated that the computation times for both transforms are within the RC time constants of the neural analog circuit.

In this paper, a neural-based optimization formulation is proposed to solve the two-dimensional (2-D) discrete cosine transform in real time. It is known that the direct computation of a 2-D DCT of size $L \times L$ is to perform the triple matrix product of an input image matrix and two orthonormal base matrices. After proper arrangements, the triple matrix product can be reformulated as minimizing a large-scale quadratic nonlinear programming problem subject to $L \times L$ DCT coefficient variables. However, a decomposition technique is applied to divide the large-scale optimization problem into $L \times L$ smaller-scale subproblems, each of which depends on its corresponding 2-D DCT coefficient variable only and then can be easily solved. In order to achieve the digital video applications, each 2-D DCT coefficient variable should be considered in the 2's complement binary representation. Therefore, each subproblem has been changed to be a new optimization problem subject to a number of binary variables of the corresponding 2-D DCT coefficient. Indeed, the new optimization problem is also a quadratic programming with minimization which occurs on the corners of the binary hypercube space. This is identical to the energy function involved in the Hopfield neural model [3], [4]. They showed that a neural net has associated with it an "energy function" which the net always seeks to minimize. The energy function decreases until the net reaches a steady state solution which is the desired 2-D DCT coefficient. The architecture of the neural net designed to perform the 2-D DCT would, therefore, reach a solution in a time determined by RC time constants, not by algorithmic time complexity, and would be straightforward to fabricate.

Since MOS circuits have the attractive features such as reconfigurability, input/output compatibility, and unrestricted fan-in/fan-out capability, we proposed an novel hybrid digital-analog neural network in MOS technology. This network includes compact and electrically programmable synapses and bias using the analog parameterized MOS modules. More details about the MOS realization will be discussed in section four.

2 An Optimization Formulation for The Transform Image Coding

The Discrete Cosine Transform (DCT) is an orthogonal transform consisting of a set of basis vectors that are sampled cosine functions. A normalized Lth-order DCT matrix $\mathbf{U}$ is defined by

$$u_{st} = \sqrt{\frac{2}{L}} \cos\left[\frac{\pi(2s+1)t}{2L}\right] \tag{1}$$

for $0 \leq s \leq L-1$, $1 \leq t \leq L-1$ and $u_{st} = L^{-\frac{1}{2}}$ for $t = 0$. The two-dimensional (2D) DCT of size $L \times L$ is defined as

$$\mathbf{Y} = \mathbf{U}^T \mathbf{X} \mathbf{U} \tag{2}$$

where $\mathbf{U}^T$ is the transpose of $\mathbf{U}$ and $\mathbf{X}$ is the given image data block of size $L \times L$ (typical 8×8 or 16×16).

Traditionally, the resultant matrix in the transform domain $\mathbf{Y}$ may be obtained by a direct implementation of (2) which is computationally intensive. By taking the advantage of the high-speed analog imple-

[1]This study was supported, in part, by the National Science Council, Republic of China, under contract number: NSC 80-0404-E-009-77

Manuscript received June 7, 1991

mentation of the Hopfield-type neural network [3], [4], the following formulations are required and would be described as follows:

From (2), we have

$$\begin{aligned}
\mathbf{X} &= \mathbf{U}\mathbf{Y}\mathbf{U}^T \\
&= \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_{L-1} \end{bmatrix} \cdot \\
&\quad \begin{bmatrix} y_{00} & y_{01} & \cdots & y_{0,L-1} \\ y_{10} & y_{11} & \cdots & y_{1,L-1} \\ \vdots & \vdots & & \vdots \\ y_{L-1,0} & y_{L-1,1} & \cdots & y_{L-1,L-1} \end{bmatrix} \begin{bmatrix} \mathbf{u}_0^T \\ \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_{L-1}^T \end{bmatrix} \\
&= \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} y_{ij}\mathbf{u}_i\mathbf{u}_j^T
\end{aligned} \tag{3}$$

where $\mathbf{u}_i$ is the i-th column vector of $\mathbf{U}$.

Define the distance or norm between two matrices $\mathbf{A}$ and $\mathbf{B}$ to be

$$NORM(\mathbf{A},\mathbf{B}) = tr(\mathbf{A}^T\mathbf{B}) \tag{4}$$

where $tr(\mathbf{A})$ is equal to $\sum_{i=0}^{L-1} a_{ii}$.

Let $\boldsymbol{\Delta} = \mathbf{X} - \mathbf{U}\mathbf{Y}\mathbf{U}^T$ and $\|\boldsymbol{\Delta}\|^2 = NORM(\boldsymbol{\Delta},\boldsymbol{\Delta})$. Therefore, the coefficients y_{ij} in (3) minimizes the distance function

$$\underset{\substack{y_{ij} \\ 0\le i,j\le L-1}}{\mathrm{Min}} \quad \|\boldsymbol{\Delta}\|^2 \, (= \|\mathbf{X} - \mathbf{U}\mathbf{Y}\mathbf{U}^T\|^2) \tag{5}$$

In this way, given $\mathbf{X}$, the problem of computing $\mathbf{Y}$ by (3) has been changed into the problem of finding the minimum $\mathbf{Y} = [y_{ij}]$ of the function $\|\boldsymbol{\Delta}\|^2$ in (5).

To reduce the complexity of performing the optimization problem in (5), $\|\boldsymbol{\Delta}\|^2$ can be rewritten in the following form:

$$\|\boldsymbol{\Delta}\|^2 = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} \|\mathbf{X} - y_{ij}\mathbf{u}_i\mathbf{u}_j^T\|^2 - (L^2-1)tr(\mathbf{X}^T\mathbf{X}) \tag{6}$$

Observing (6), it should be noted that the second term of the right-hand side of (2) is constant and the components involved in the summation of the first term are independent each other. Therefore, the minimization problem in (5) could be divided into L^2 subproblems as follows:

$$\underset{y_{ij}}{\mathrm{Min}} \|\Delta_{ij}\|^2 \, (= \|\mathbf{X} - y_{ij}\mathbf{u}_i\mathbf{u}_j^T\|^2), 0 \le i,j \le L-1 \tag{7}$$

Indeed, Equation (7) can be expanded and rearranged in the scalar form

$$\underset{y_{ij}}{\mathrm{Min}} \|\Delta_{ij}\|^2 \, (= \sum_{s=0}^{L-1}\sum_{t=0}^{L-1} (x_{st} - y_{ij}u_{is}u_{tj})^2) \tag{8}$$

This decomposition approach provides us with a technique to divide a large-scale optimization problem into a number of smaller-scale subproblems, each of which can be easily solved.

Due to the requirement of many digital video applications, each y_{ij} is quantized into $\widehat{y}_{ij}$ which can be represented by the 2's complement code as follows:

$$\widehat{y}_{ij} = -s_{ij}^{(m_{ij})}2^{m_{ij}} + \sum_{p=-n_{ij}}^{m_{ij}-1} s_{ij}^{(p)}2^p \tag{9}$$

where $s_{ij}^{(p)}$ is the p-th bit of $\widehat{y}_{ij}$ which has a value of either 0 or 1, $s_{ij}^{(m_{ij}-1)}$ is the most significant bit (MSB), $s_{ij}^{(-n_{ij})}$ is the least significant bit, and $s_{ij}^{(m_{ij})}$ is the sign bit.

Substituting (9) into (8), one may obtain the new minimization problem subject to the binary variables; $s_{ij}^{(p)}$, $-n_{ij} \le p \le m_{ij}$, that is,

$$\underset{\substack{s_{ij}^{(p)} \\ -n_{ij}\le p\le m_{ij}}}{\mathrm{Min}} \quad \|\Delta_{ij}\|^2 (= \sum_{s=0}^{L-1}\sum_{t=0}^{L-1} [x_{st} - (-s_{ij}^{(m_{ij})}2^{m_{ij}} + \sum_{p=-n_{ij}}^{m_{ij}-1} s_{ij}^{(p)}2^p)u_{is}u_{tj}]^2) \tag{10}$$

In the following section, a novel neural-based optimizer is proposed to solve the above minimization problem in order to meet the real-time requirement of many digital video applications.

3 A Neural-Based Optimization Approach

Artificial neural networks contain a large number of identical computing elements or neurons with specific interconnection strengths between neuron pairs [3], [4]. The massively parallel processing power of neural network in solving difficult problems lies in the cooperation of highly interconnected computing elements. It is shown that the speed and solution quality obtained when using neural networks for solving specific problems in visual perception [5] and signal processing [6] make specialized neural network implementations attractive. For instance, the Hopfield network can be used as an efficient technique for solving various combinatorial problems [7] by the programming of synaptic weights stored as a conductance matrix.

Hopfield model [3], [4] is a popular model of continuous, interconnected n nodes. Each node is assigned a potential, $u_p(t)$, $p = 1, 2, \ldots, n$ as its state variable. Each node receives external input bias $I_p(t)$, and internal inputs from other nodes in the form of a weighted sum of firing rates $\sum_q T_{pq}g_q(\lambda_q u_q)$, where $g_q(\cdot)$ is a monotonically increasing sigmoidal bounded function converting potential to firing rate. The general structure of the networks is shown in Figure 1. The equations of motion are

$$\begin{aligned}
C\frac{du_p}{dt} &= -\frac{u_p}{R} + \sum_{q=1}^{n} T_{pq}v_q + I_p \\
v_p &= g_p(\lambda_p u_p)
\end{aligned} \tag{11}$$

where λ_p's are the amplifier gains and $g_p(\lambda_p u_p)$ is typically identified as $\frac{1}{2}(1 + tanh(\lambda_p u_p))$

Electrically, $T_{pq}v_q$ might be understood to represent the electrical current input to neuron p due to the present potential of neuron q. The quantity T_{pq} represents the finite conductance between the output v_q and the body of neuron p. It would also be considered to represent the synapse efficacy. The term $-u_p/R$ is the current flow due to finite transmembrane resistance R, and it causes a decrease in u_p. I_p is any other (fixed) input bias current to neuron p. Thus, according to (11), the change in u_p is due to the changing action of all the $T_{pq}v_q$ terms, balanced by the decrease due to $-u_p/R$, with a bias set by I_p.

Hopfield and Tank [3], [4] have shown that in the case of symmetric connections ($T_{pq} = T_{qp}$), the equations of motion for this network of analog processors always lead to a convergence to stable states, in which the output voltages of all amplifiers remain constant. In addition, when the diagonal elements (T_{pp}) are 0 and the amplifier gains λ_p's are high, the stable states of a network comprised of n neurons are the minima of the computational energy or Liapunov function

$$E = -\frac{1}{2}\sum_{p=1}^{n}\sum_{q=1}^{n} T_{pq}v_p v_q - \sum_{p=1}^{n} I_p v_p \tag{12}$$

The state space over which the analog circuit operates is the n-dimensional hypercube defined by $v_p = 0$ or 1. However, it has been shown that in the high-gain limit networks with vanishing diagonal connections ($T_{pp} = 0$) have minima only at corners of this space [4]. Under these conditions the stable states of the network correspond to those locations in the discrete space consisting of the 2^n corners of this hypercube which minimize E.

To solve the minimization problem in (10) by the Hopfield-type neural network, the binary variables $s_{ij}^{(p)}$ should be assigned to their corresponding potential variables v_p with n $(= m_{ij} + n_{ij} + 1)$ neurons. Truly, the computational energy function E^{ij} of the proposed network for y_{ij} may be identified as $\|\Delta_{ij}\|^2$ in (10). However, with this simply energy function there is no guarantee that the values of $s_{ij}^{(p)}$ will be near enough to 0 or 1 to be identified as digital logic. Since (10) contains diagonal elements of the **T**-matrix are nonzero, the minimal points to the $\|\Delta_{ij}\|^2$ (10) will not necessarily lie on the corners of the hypercube, and thus represent the 2's complement digital representation. One can eliminate this problem by adding one additional term to the function $\|\Delta_{ij}\|^2$. Its form can be chosen as

$$\Delta E^{ij} = \sum_{s=0}^{L-1}\sum_{t=0}^{L-1}\left\{\left[\sum_{p=-n_{ij}}^{m_{ij}} s_{ij}^{(p)}(1-s_{ij}^{(p)})2^{2p}\right](u_{is})^2(u_{tj})^2\right\} \tag{13}$$

The structure of this term was chosen to favor digital representations. Note that this term has minimal value when, for each p, either $s_{ij}^{(p)} = 1$ or $s_{ij}^{(p)} = 0$. Although any set of (negative) coefficients will provide this bias towards a digital representation, the coefficients in (13) were chosen so as to cancel out the diagonal elements in (10). The elimination to diagonal connection strengths will generally lead to stable points only at corners of the hypercube. Thus the new total energy function E^{ij} for y_{ij} which contains the sum of the two terms in (10) and (13) has minimal value when the $s_{ij}^{(p)}$ are a digital representation close to the resultant y_{ij} in (3). After expanding and rearranging the energy function E^{ij}, we have

$$\begin{aligned} E^{ij} &= \|\Delta_{ij}\|^2 + \Delta E^{ij} \\ &= -\frac{1}{2}\sum_{p=-n_{ij}}^{m_{ij}}\sum_{q=-n_{ij}}^{m_{ij}} s_{ij}^{(p)} s_{ij}^{(q)} T_{pq}^{ij} - \sum_{p=-n_{ij}}^{m_{ij}} s_{ij}^{(p)} I_p^{ij} \end{aligned} \tag{14.a}$$

where

$$T_{pq}^{ij} = \begin{cases} 0 & \text{for } p = q \\ -2\sum_{s=0}^{L-1}\sum_{t=0}^{L-1} 2^{p+q} u_{is}^2 u_{tj}^2 & \text{for } p \neq q, p \neq m_{ij}, q \neq m_{ij} \\ 2\sum_{s=0}^{L-1}\sum_{t=0}^{L-1} 2^{p+q} u_{is}^2 u_{tj}^2 & \text{for } p \neq q, p \neq m_{ij}, q = m_{ij} \\ 2\sum_{s=0}^{L-1}\sum_{t=0}^{L-1} 2^{p+q} u_{is}^2 u_{tj}^2 & \text{for } p \neq q, p = m_{ij}, q \neq m_{ij} \end{cases} \tag{14.b}$$

and

$$I_p^{ij} = \begin{cases} \sum_{s=0}^{L-1}\sum_{t=0}^{L-1}(2^{p+1} u_{is} u_{tj} x_{st} - 2^{2p} u_{is}^2 u_{tj}^2) & \text{for } p \neq m_{ij} \\ -\sum_{s=0}^{L-1}\sum_{t=0}^{L-1}(2^{p+1} u_{is} u_{tj} x_{st} + 2^{2p} u_{is}^2 u_{tj}^2) & \text{for } p = m_{ij} \end{cases} \tag{14.c}$$

4 Programmable Hybrid Digital-Analog Neural Implementation

Observing equations (14.b) & (14.c), it is indicated that the synapse weights and bias are not fixed and depend on input analog signals x_{st}'s and the index (i,j) of their corresponding result y_{ij}. Both synapse weights and bias should be programmable in order to capture the information from the data set. Several researchers address the issues of programmable neurons. One VLSI chip from Intel [8] was implemented fully analog circuity operating in a deterministic manner, while another chip [9] was implemented with fully digital circuits operating in a stochastic manner. In this paper, an novel hybrid neural circuit including compact and electrically programming synapses and bias is described. The circuit techniques are based on extremely simple and programmable analog parameterized MOS modules with such attractive features as reconfigurability, input/output compatibility, and unrestricted fan-in/fan-out capability.

Before introducing the design of reconfigurable hybrid neural chip, several arrangements should be considered in the expressions of both synapse weigh T_{pq}^{ij} and bias I_p^{ij} as follows:

$$T_{pq}^{ij} = \begin{cases} 0 & \text{for } p = q \\ -2^{p+q-2m_{ij}}\widehat{T}_{ij} & \text{for } p \neq q, p \neq m_{ij}, q \neq m_{ij} \\ 2^{p+q-2m_{ij}}\widehat{T}_{ij} & \text{for } p \neq q, p \neq m_{ij}, q = m_{ij} \\ 2^{p+q-2m_{ij}}\widehat{T}_{ij} & \text{for } p \neq q, p = m_{ij}, q \neq m_{ij} \end{cases} \tag{15.a}$$

and

$$I_p^{ij} = \begin{cases} I1_p^{ij}\,(= 2^{p-m_{ij}}\widehat{I}_{ij}) + I2_p^{ij}\,(= (-2^{2(p-m_{ij})-1})\widehat{T}_{ij}) & \text{for } p \neq m_{ij} \\ I1_p^{ij}\,(= (-2^{p-m_{ij}})\widehat{I}_{ij}) + I2_p^{ij}\,(= (-2^{2(p-m_{ij})-1})\widehat{T}_{ij}) & \text{for } p = m_{ij} \end{cases} \quad -n_{ij} \leq p \leq m_{ij} \tag{15.b}$$

where

$$\widehat{T}_{ij} = \sum_{s=0}^{L-1}\sum_{t=0}^{L-1} 2^{2m_{ij}+1} u_{is}^2 u_{tj}^2 \tag{15.c}$$

$$\begin{aligned} \widehat{I}_{ij} &= \sum_{s=0}^{L-1}\sum_{t=0}^{L-1} 2^{m_{ij}+1} u_{is} u_{tj} x_{st} \\ &= \sum_{s=0}^{L-1}\sum_{t=0}^{L-1} u_{istj} x_{st} \end{aligned} \tag{15.d}$$

and

$$u_{istj} = 2^{m_{ij}+1} u_{is} u_{tj}\,, \quad 0 \leq i,j,s,t \leq L-1 \tag{15.e}$$

Basically, the concept of the above arrangements is conducted to categorize the expressions of both T_{pq}^{ij} and I_p^{ij} into a class of terms associated with the index (i,j) which corresponds to the result $\widehat{y}_{ij}$ and another class of terms associated with the index (p,q) which corresponds to the size of neural network (or number of bits involved in $\widehat{y}_{ij}$). In addition, (p,q)-related terms, 2^{p+q}, 2^{p+1}, and 2^{2p} are normalized by scaling factors $2^{-2m_{ij}}$, $2^{-m_{ij}}$, and $2^{-2m_{ij}-1}$, and thus the ranges of normalized terms $2^{p+q-2m_{ij}}$, $2^{p-m_{ij}}$, and $2^{2(p-m_{ij})-1}$ would lead to be within the intervals $[2^{-2(n-1)}, 1]$,$[2^{-(n-1)}, 1]$, and $[2^{-2n-3}, 2^{-1}]$, respectively, where n $(= m_{ij} + n_{ij} + 1)$ is the number of neurons. Therefore, those normalized terms are totally independent of index (i,j). Since $2^{p+q-m_{ij}} = 2^{q+p-m_{ij}}$, this allows the synapse weight T_{pq}^{ij} to hold the property of symmetry, that is, $T_{pq}^{ij} = T_{qp}^{ij}$. This also turns out that only the evaluations located on the upper (or lower) triangle area (T_{pq}^{ij} with $q > p$) of the **T**-matrix are necessary for the Hopfield neural network (note that $T_{pq}^{ij} = 0$ when $p = q$). Based on the above discussion, a proposed programmable system architecture and the function structure of each module are illustrated in Fig. 2, Fig. 3, and Fig. 4 respectively.

The parameters $\widehat{T}_{ij}$, and u_{istj} could be precomputed and are stored in the $((L^2+1) \times L^2)$ register file which is controlled by an address counter with clock Δt_1. Note that Δt_1 is defined as the sum of $\Delta t_{refresh}$ (= time for refreshing both the synaptic weights and bias) and Δt_{neural} (the computation time for the neural network). While computing a particular $\widehat{y}_{ij}$, those parameters should be pumped out from the register file. A $\widehat{T}_{ij}$ will go to the upper triangle analog multiplier array illustrated in Fig. 3 and thus compute the desired synaptic weights T_{pq}^{ij} which are used to dynamically refresh the on-chip programming Hopfield network. Since the L^2 input analog signals x_{st}'s are required in computing the vector multiplication (with $u_{istj}, 0 \leq s,t \leq L-1$) involved in each $\widehat{I}_{ij}$, $0 \leq i,j \leq L-1$, x_{st}'s should stay in the analog buffer until the L^2 $\widehat{y}_{ij}$'s have been completed. This analog buffer is controlled by a system counter with clock Δt_2 $(= L^2 \cdot \Delta t_1)$. After completing the evaluations of both $\widehat{T}_{ij}$ and $\widehat{I}_{ij}$, the synaptic weights T_{pq}^{ij} and bias I_p^{ij} could be determined according to the values $2^{p+q-2m_{ij}}$ and $2^{p-m_{ij}}$ respectively. The upper triangle array with zero diagonal used to compute T_{pq}^{ij} contains $n(n-1)/2$ analog multipliers, each of which has a prescribed operand that is independent of the index (i,j). The implementation of analog multiplier is suggested to employ the MOS modified Gilbert transconductance multiplier (or four-quadrant multiplier) [10] illustrated in Fig. 5, which has a wide range of both stabitity and linearity. Since each prescribed operand $2^{p+q-2m_{ij}}$ is power of two, another solution to implement the evaluations of **T**-matrix can use the shift registers instead of the analog multipliers. Similarly, two linear analog multiplier arrays are used to determine $I1_p^{ij}$'s and $I2_p^{ij}$'s based on the computed values $\widehat{T}_{ij}$ and $\widehat{I}_{ij}$ from the previous modules. After analog additions, the resultant $I_p^{ij} = I1_p^{ij} + I2_p^{ij}$, $-n_{ij} \leq p \leq m_{ij}$ would be then pumped into the neural network. The synaptic weights and bias from a digital register file are converted to analog form through an evaluator module and then written on the storage capacitors or analog DRAM-type storage [11] inside the Hopfield neural network selected by the address decoder. The circuit schematic of a Hopfield neural network is shown in Fig. 4. The neurons are realized by simple CMOS double inverters which are interconnected through the n $(= m_{ij} + n_{ij} + 1)$ vector multipliers. The transfer function of the double inverter is identified as the montonically increasing sigmoidal function, $g_p(\lambda_p u_p)$. Each multiplier illustrated in Fig. 6 implements the scalar vector productor of the vector of neuron outputs ($s_{ij}^{(q)}$'s) and the vector of the synaptic weights. For a network of n neurons, there are n such scalar productors. Each scalar product is achieved using only one operational amplifier and $4(n+1)$ MOS transistors for $2n$-tuple vector inputs resulting in an economic and attractive analog MOS VLSI implementation. Using depletion transistors, gates of MOS transistors can be connected to ground resulting in a special case of the vector multiplier which allows the multiplication of voltages that are referred to ground. Positive or negative grounded voltage levels can be assigned to synaptic weights, T_{pq}^{ij}. The outputs of n neurons $s_{ij}^{(q)}$ $(-n_{ij} \le q \le m_{ij})$ are fedback as inputs to the p-th multiplier $(-n_{ij} \le p \le m_{ij})$. The output of the p-th multiplier in turn is fed into the input of the p-th double inverter (neuron p). The overall output of the p-th vector multiplier, $v_0^{(p)}$ is given by

$$v_0^p = \sum_{q=-n_{ij}}^{m_{ij}} c \times T_{pq}^{ij} \times s_{ij}^{(q)} \qquad (16)$$

where c = the constant depends on the characteristics of MOS implementation.

It is interesting to note that the constant c could be compensated by absorbing the values into T_{pq}^{ij}. For example, one may precompute the new $\widehat{T}_{ij}$ as $c^{-1} \times \widehat{T}_{ij}$, where $\widehat{T}_{ij}$ is the old one. The input-output compatibility of the overall MOS implementation is of particular interest since the relatively high output impedance node of the double inverter is connected to the almost inifinite input impedance node of the MOSFET gates with almost no restrictions on the fan-in/fan-out capability. More details about the MOS vector multiplier are shown in [12].

5 Illustrated Examples

To examine the performance of the neural-based analog circuit for computing the 2-D DCT transform coding, an often used 8×8 DCT will be considered in our simulation since it represents a good compromise between coding efficiency and hardware complexity. Because of its effectiveness, the CCITT H.261 recommended standard for $p \times 64\ kb/s (p = 1, 2, \cdots 30)$ visual telephony developed by CCITT, and the still-image compression standard developed by ISO JPEG all include the use of 8×8 DCT in their algorithms.

In order to obtain the size $(= n)$ of neural network required for computing its corresponding DCT coefficient, it is necessary to calculate their respective dynamic range and to take into account the sign bit. To achieve this purpose, the range of each DCT coefficient can be determined by generating random integer pixel data values in the range 0 to 255 through the 2-D discrete cosine transform. For example, the range of y_{00} is from -1024 to 1023. Therefore, m_{00} is identified as 11, that is, 10 bits are for the magnitude of y_{00} and 1 bit is for sign. As a result, the corresponding m_{ij} for each DCT coefficient y_{ij} is illustrated in Table 1. Another important parameter required in determining the size is n_{ij} which depends on the required accuracy and the tolerable mismatch in the final representation of the reconstructed video samples. The analysis of the accuracy and mismatch involved in the finite length arithmetic DCT computation has been discussed in [1], [2]. Based on their results and the consideration of feasibe hardware implementation, the number of bits (or size of neural network) involved in each DCT cosfficient is set to be 16. Then, n_{ij} would be equal to $(15 - m_{ij})$, for example, $n_{00} = 5$. The above suggestion seems quite reasonable for improving the accuracy of a particular y_{ij} which has a small dynamic range.

We have simulated the DCT-based neural analog circuit of equation (11), using the simultaneous differential equation solver (DVERK in the IMSL). This routine solves a set of nonlinear differential equations using the fifth-order Runge-Kutta method. It is known that the converge time for neural network is within RC time constant. We used three different RC time constants, $RC = 10^{-10}$ $(R = 1k\Omega, C = 0.01pF)$, $RC = 10^{-9}$ $(R = 1k\Omega, C = 0.1pF)$, and $RC = 10^{-8}$ $(R = 1k\Omega, C = 1pF)$ and all amplifier gains λ_p's are assigned to be 100 in our experiments, and ran simulations on a SUN workstation. The test input pixel data x_{st}'s are illustrated in Table 2.(a). Figures 7.a, 7.b & 7.c show an example of the time evolution of the reduction of energy performed by a network with n $(= 16)$ neurons that represent y_{25} based on the 2's complement binary number representation for three different RC time constants. The (2, 5)-entry in Table 2.(c) shows the resulted DCT coefficient y_{25} obtained at the steady state points on the curves of Figures 7.a, 7.b & 7.c. It is shown that the result is almost independent of the RC time constants. However, each converge time will be in proportion to its corresponding RC time constant. For example, the converge times for $RC = 10^{-10}$, $RC = 10^{-9}$, and $RC = 10^{-8}$ are in proportion to the orders of time scale, 10^{-10}sec $(= 0.1ns)$, 10^{-9}sec $(= 1ns)$, and 10^{-8}sec $(= 10ns)$, respectively. But these three curves have almost the same time evolution. Starting from very high energy state, the neural network reduced its energy spontaneously by changing its state so that the 2's complement binary variables $s_{ij}^{(p)}$'s minimize the error energy function.

Considering the programmable neural MOS circuit implementation of an 8×8 DCT based on the above results, both address clock Δt_1 and system clock Δt_2 for $RC = 10^{-8}$ are estimated as $2ns$ and $128ns$ respectively with the estimated $\Delta t_{refresh} = 1ns$. Therefore the computation time for computing all DCT coefficients would be estimated as $150ns$ which includes the overhead of I/O. The real implementation of the analog MOS neural circuit will be realized in our microelectronic laboratory.

6 Conclusion

The computation of a 2-D DCT-based transform coding has been shown to solve a quadratic nonlinear programming problem subject to the corresponding 2's complement binary variables of 2-D DCT coefficients. A novel Hopfield-type neural analog circuit designed to perform the DCT-based quadratic nonlinear programming could obtain the desired coefficients of an 8×8 DCT in 2's complement code within $1ns$ with $RC = 10^{-8}$. In addition, a programmable analog MOS implementation provieds a flexible architecture to realize the DCT-based neural net.

References

[1] M. T. Sun, L. Wu, and M. L. Liou, "A concurrent architecture for VLSI implementation of discrete cosine transform," *IEEE Trans. on Circuits and Systems*, vol. CAS-34, pp. 992–994, Aug. 1987.

[2] M. L. Liou and J. A. Bellisio, "VLSI implementation of discrete cosine transform for visual communication," in *Proc. Int. Conf. on Commun. Tech.*, Bejing, China, Nov. 1987.

[3] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Natl. Acad. Sci., U.S.A.*, vol. 81, pp. 3088–3092, 1984.

[4] D. W. Tank and J. J. Hopfield, "Simple 'Neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. on Circuits and Systems*, vol. CAS-33, no. 5, pp. 533–541, May 1986.

[5] G. Bilbro, M. White, and W. Snyder, "Image segmentation with neurocomputers," in *Neural Computers*, Rolf Eckmiller and Christopher v. d. Malsburg, Eds., pp. 71–79, Springer-Verlag, 1987.

[6] A. D. Culhane, M. C. Peckerar and C. R. K. Marrian, "A neural net approach to discrete Hartley and Fourier transform," *IEEE Trans. on Circuits and Systems*, vol. 36, pp. 695–702, May 1989.

[7] J. Hopfield and D. Tank, "Neural computations of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[8] M. Holler, S. Tam, H. Castro, and R. Benson, "An electrically trainable artificial neural network (ETANN) with 10240 'Float gate' synapses," *Inter. Joint Conf. Neural Networks*, vol. 2, pp. 191–196, June 1989.

[9] David E. Van den Bout and T. K. Miller III, "A digital architecture employing stochasticism for the simulation of Hopfield neural nets," *IEEE Trans. Circuits Syst.*, vol. 36, no. 5, pp. 732–738, May 1989.

[10] C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, 1989.

[11] J. C. Lee and B. J. Sheu, "Parallel digital image restoration using adaptive VLSI neural chips," *IEEE Proc. Intl. Conf. on Computer Design: VLSI in Computers and Processors*, Cambridge, MA, Sept. 17–19, 1990.

[12] F. Salam, N. Khachab, M. Ismail, and Y. Wang, "An analog MOS implementation of the synaptic weights for feedback neural nets," *Proc. IEEE ISCAS*, pp. 1223–1225, May 1989.

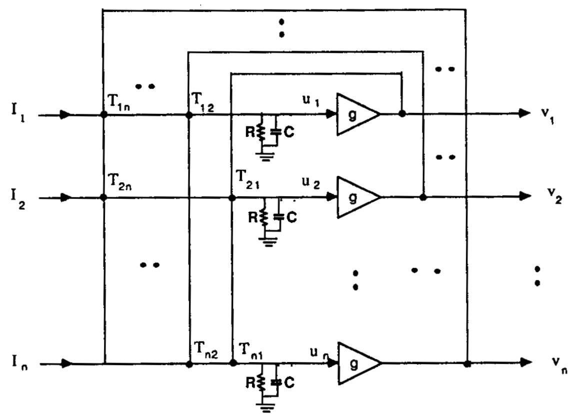


Figure 1. The Circuit Schematic of Hopfield Model

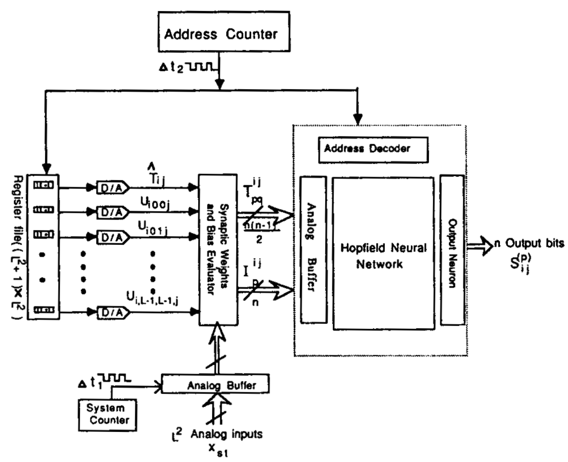


Figure.2 Architecture of Image Transform Coding Neural Chip

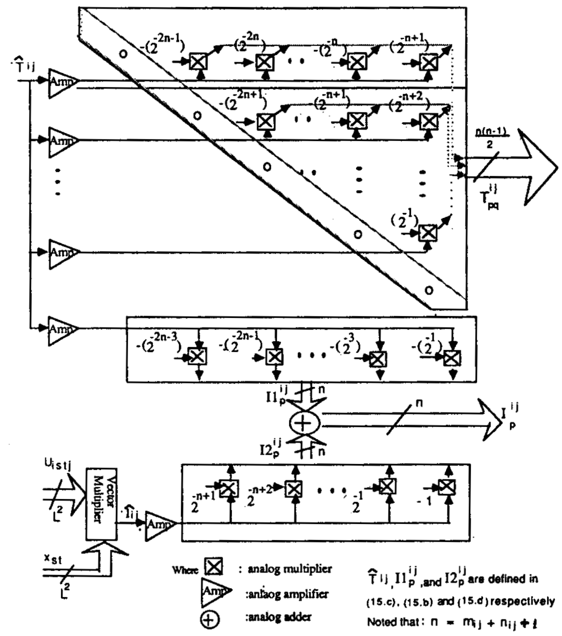


Figure.3 Synaptic Weights and Bias Evaluator

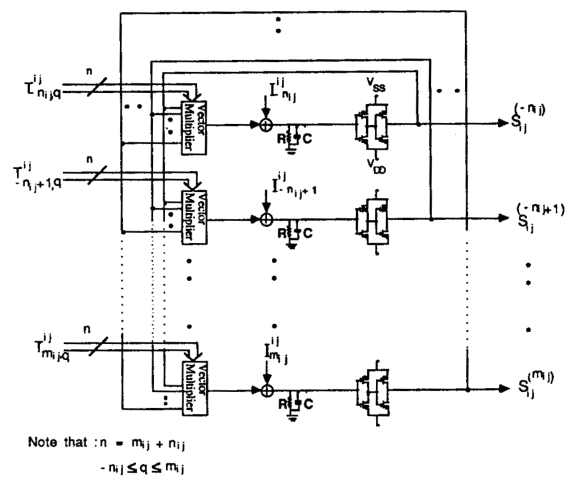
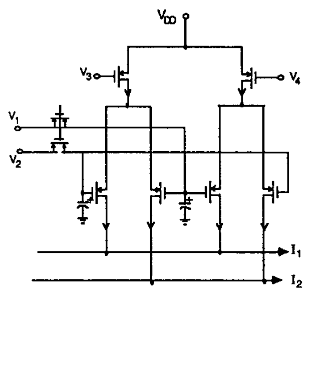
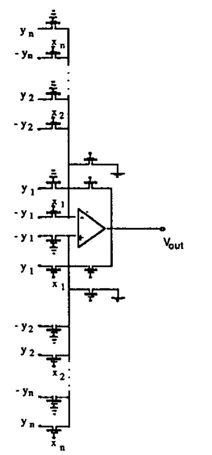


Figure.4 Analog MOS Hopfield Neural Network



Note that : $I_1 - I_2$ = Constant × $(V_1 - V_2)$ × $(V_3 - V_4)$



Note that : V_{out} = constant × $\sum_i (x_i y_i)$

Figure.5 Modified-Gilbert Programmable Analog Multiplier

Figure.6 Vector Multiplier(Analog MOS)

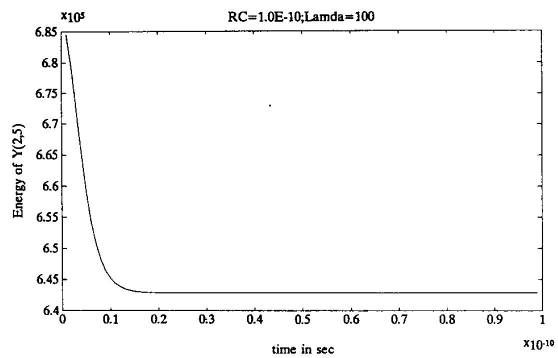


Figure.7(a)

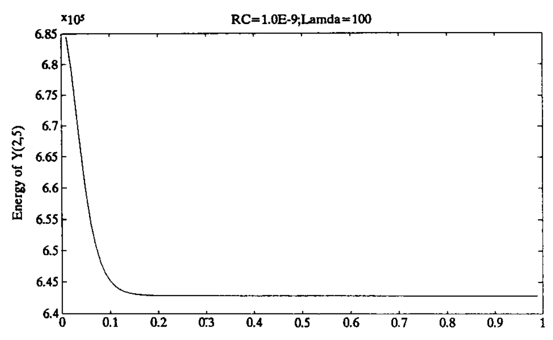


Figure.7(b)

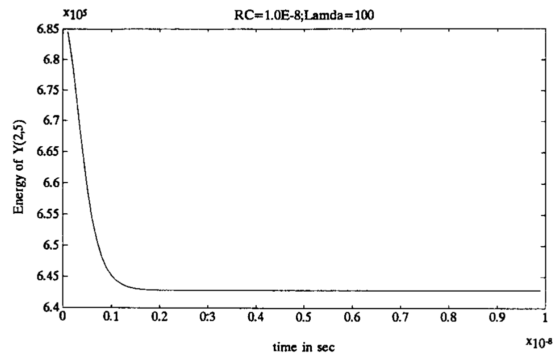


Figure.7(c)

$i \backslash j$	0	1	2	3	4	5	6	7
0	11	9	9	9	9	9	9	9
1	9	9	9	9	9	9	9	9
2	9	9	9	9	9	9	9	9
3	9	9	9	9	9	9	9	9
4	9	9	9	9	9	9	9	9
5	9	9	9	9	9	9	9	9
6	9	9	9	9	9	9	9	9
7	9	9	9	9	9	9	9	9

Table 1. m_{ij} for y_{ij}

$s \backslash t$	0	1	2	3	4	5	6	7
0	159	87	86	105	172	105	103	55
1	126	17	194	230	28	238	179	128
2	253	105	48	181	103	94	255	26
3	73	159	29	241	138	175	66	145
4	120	74	220	201	97	5	87	70
5	2	187	10	118	203	252	162	209
6	160	56	97	19	53	221	14	78
7	131	31	233	85	140	168	118	126

(a). Input pixel data x_{st}

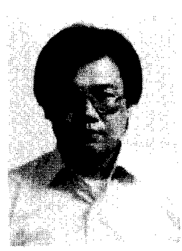
$i \backslash j$	0	1	2	3	4	5	6	7
0	981.2311	-31.9679	-73.4526	71.6891	12.4996	89.2316	60.4552	-59.9983
1	34.8142	39.8153	-10.0274	15.7371	41.3010	135.4820	-97.6490	-26.2421
2	-24.3766	-7.5722	27.8224	64.9493	-49.1428	12.9251	145.7272	93.4776
3	-40.1650	-40.7837	-24.6358	- 15.3744	-3.2812	-136.3970	-18.4978	-5.2610
4	-30.2488	83.8342	-64.5874	-101.1043	65.5008	-31.2436	-4.3674	74.8538
5	-72.6923	26.7245	42.2933	134.4323	38.6007	-117.1561	-43.0697	-64.5455
6	60.6918	-47.4648	23.9781	71.8251	13.3202	-48.0477	-165.5747	55.7453
7	-86.7412	172.0896	56.1962	7.2807	19.0053	128.7716	4.9155	137.7180

(b). The DCT coefficients

$i \backslash j$	0	1	2	3	4	5	6	7
0	981.2500	-31.9688	-73.4531	71.6875	12.5000	89.2344	60.4531	-60.0000
1	34.8125	39.8125	-10.0312	15.7344	41.2969	135.4844	-97.6562	-26.2500
2	-24.3750	-7.5781	27.8281	64.9531	-49.1406	12.9219	145.7344	93.4844
3	-40.1719	-40.7812	-24.6406	-15.3750	-3.2812	136.4062	-18.5000	-5.2656
4	-30.2500	83.8281	-64.5938	-101.1094	65.5000	-31.2500	-4.3750	74.8594
5	-72.6875	26.7188	42.2969	134.4375	38.5938	117.1562	-43.0625	-64.5469
6	60.6875	-47.4688	23.9844	71.8281	13.3202	-48.0469	-165.5781	55.7500
7	-86.7500	172.0938	56.2031	7.2812	19.0000	128.7656	4.9219	137.7188

(c). Neural-based DCT coefficients y_{ij}

Table 2.



Po-Rang Chang