SIGNAL SPECTROGRAM USING NEURAL NETWORKS

El-Adawy M.I, and I. I. Ibrahim
Faculty Of Engineering And Technology,
Helwan, Cairo, Egypt

استخدام الشبكات العصبية كمحلل طيفي زمني للإشارات

ملخص البحث:

المحلل الطيفي الزمني للإشارات هو علاقة بين الزمن والتردد ومقياس الإشارة. وهذا المحلل الطيفي الزمني له الكثير من التطبيقات خاصة في تدريب الصم والتعرف على المتكلمين من خلال أصواتهم. ولذا يقدم هذا البحث شبكة عصبية من طبقة واحدة تم تدريبها لتعطي في الخرج كل تردد من ترددات الإشارة ومقياسه وتغيره مع الزمن. وتم إختبار هذه الشبكة على إشارات مختلفة إبتداء من الإشارات المركبة من تردد واحد ثم الإشارات معدلة تعديل سعوي (AM) والإشارات معدلة تعديل ترددي (FM) وفي النهاية تم اختبار الدائرة على الترددات المنخفضة لإشارات صوتية . وفي جميع هذه الحالات حصلنا على الخرج المتوقع والمطلوب من الشبكة كما هو موضح في نهاية البحث. والمشكلة الوحيدة مع هذه الشبكة هي أنها مناسبة جدا في الترددات المنخفضة أما في الترددات العالية فإن الشبكة تصبح معقدة ويصعب تدريبها.

ABSTRACT

The spectrogram of a signal is a three dimensional relation between time, frequency and amplitude. The spectrogram has many applications specially in the speaker identification and the training aids of the deaf where the determination of the pitch is a key factor. A neural network that will give this spectrogram is introduced in this paper. The proposed network is tested under different test conditions to verify its operation.

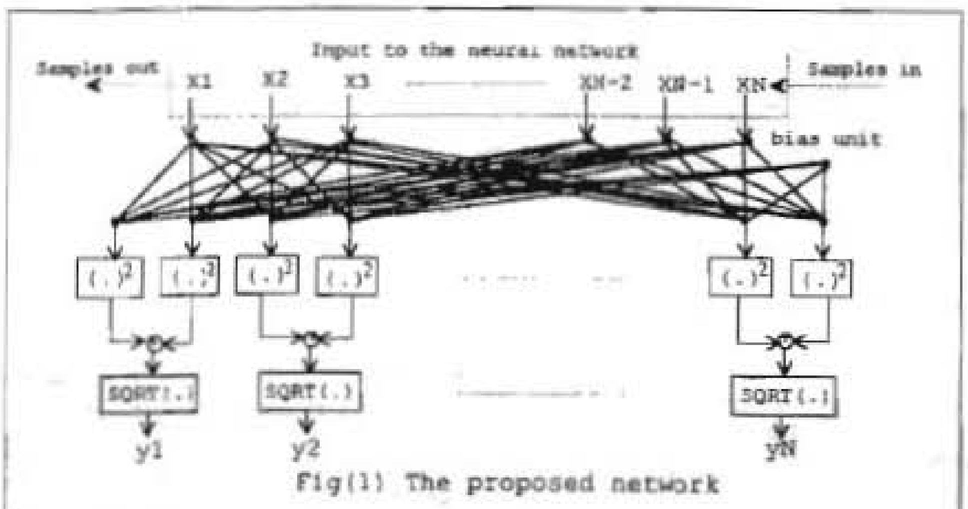
I. INTRODUCTION

A graph that shows the relation between time, frequency and magnitude of a signal is very useful in a lot of applications. Most of these applications are in the sound processing fields where such relation can play a key factor in the determination of the pitch, speaker identification and verification, and in sound recognition. Many of the instruments that are used in treating and training the deaf people are based on the spectrogram.

It has been shown in previous works that a neural network can be trained to do some important frequency related matters. For example a neural network has been trained to behave like a filter with a good filtering characteristics that can only be obtained from a complicated digital filters [1]. In another article a simple perceptron has been trained to detect the frequency contents of a short time signal [2]. In this previous work the signal was assumed to be stationary. In this paper we will extend the work to include the non stationary signals like sound. So the dimension of time has to be taken into account and the relation between the frequency and amplitude will be monitored and recorded continuously with time. The proposed network will be tested on known signals like amplitude modulation, AM, and frequency modulated, FM signals to verify the operation of the network. Then the network will be tested on a sound signal.

2. NETWORK DESCRIPTION

Fig. (1) shows the one layer neural network that we proposed. This network is composed of an input, and an output layer. The input layer represents a window that is opened on the stream of samples of the input data that are continuously shifted sample by sample. These samples inter the input window from one end and exit from the other end as shown in fig. (1). With each shift of the input data from right to left, the output of the network is calculated. This output represents the instantaneous frequency contents of input window. Each output will be proportional to the amplitude of a certain frequency if this frequency exists in the input at that instant. If this frequency does not exist, then the corresponding output will be zero.



Fig(1) The proposed network

As we see in fig. (1) the input layer is fully connected to the output layer. The number of neurons in the input layer is related to the maximum frequency to be detected in the signal by the Nyquest role. So, the number of neurons in the input layer should be higher than double the frequency to be detected in the input signal in Hertz. The number of neurons in the output layer depends on the frequency components to be detected where it will be double the number of those frequency components as shown in fig. (1) and as we will discuss in the next section. A bias component is added to the input of each neuron where it was found that these biases also speed up the convergence of the network. All the neurons that we used in this paper are from the simple type that are characterized by the equation [3-5]:

$$y = \Sigma(x_i w_{ij} - \theta_i) \tag{1}$$

where x_i is the inputs, w_{ij} is the weighting factor of the connection between neuron i and neuron j, and θ_i is a threshold factor. A linear activation function is assumed in all the neurons that we used in this paper. Nonlinear activation functions have been tried but they were found to have a longer training time before they reach a converge. This result is expected due to the nature of the network where the output is composed of several analog components.

Because the input signal is always complex, i.e. composed of real and imaginary components as in equation 2, phase will be a crucial factor and must be taken into consideration. The input signal in its general form can be written as

$$\begin{aligned} x_i &= Ae^{j(2\pi i f T + \Phi)} \\ &= A\cos(2\pi i f T + \Phi) + jA\sin(2\pi i f T + \Phi) \end{aligned} \tag{2}$$

where A is the amplitude, f is the frequency, Φ is the phase, and T is the sampling period that is related to the time t by the relation $t = iT$.

As we see in fig. (2) the output layer is divided into neuron pairs, each pair is associated with a certain frequency component. One neuron of each pair will detect the real component, and the other neuron of the pair will detect the imaginary component of that output. So the net output of each neuron pair will be a combination of the outputs from each neuron in the pair according to the following equation:

$$\begin{aligned} y_i &= \sqrt{(A\cos(2\pi i f T + \Phi))^2 + (A\sin(2\pi i f T + \Phi))^2} \\ &= A \end{aligned} \tag{3}$$

So from equation 3 we can say that when we train the first neuron of any output neuron pair to detect the real component of the signal, and the other neuron of the pair to detect the imaginary component, we are actually training the pair to detect the amplitude of that frequency component. This training is such that if the frequency exists in the signal the output of its corresponding neuron pair will equal to the amplitude A of that frequency, and if this frequency does not exists then the output of this pair will be zero. The training algorithm for the network will be as follows:

3. TRAINING ALGORITHM

1. Random initial values for the weights and biasing coefficients will be considered.
2. Samples of the input signal will be introduced to the input layer and a corresponding output will be calculated. The input signal is assumed to have certain frequency, amplitude and phase.
3. If the value of the corresponding output to that frequency does not equal to the amplitude of that frequency A, an error will be calculated and used to correct all the weights of the

network.

4. Step 3 will be repeated until the output reaches the desired value A.
5. The phase of the input signal will be incremented and steps 2, 3, and 4 will be repeated until the output will equal to the desired value A at any value of the signal phase.
6. The amplitude of the input signal will be incremented and steps 2, 3, 4, and 5 will be repeated until the output will equal to the desired value at all different values of the phase and amplitude.
7. The frequency of the input signal will be incremented and steps 2, 3, 4, 5 and 6 will be repeated until the output will be equal to the desired value at all different values of the phase, amplitude, and frequency.
8. The input signal will be shifted by one sample and steps 2, 3, 4, 5, 6, and 7 will be repeated until the output will be equal to the desired value at all values of phase, amplitude, frequency, and time.

We used the simple perceptron training technique in the training of the proposed network. Literature's are full of the discussion of this technique, so there is no need to repeat its discussion in here [3-5].

4. RESULTS

The first step was the testing of the proposed network on simple signals that must have a nonvarying spectrum with time. Fig. (2) shows the output of the network when the input was a simple sinusoidal wave given by the equation:

$$x = 0.9\cos(2\pi i 20T) \tag{4}$$

where the output is a peak at f=20Hz, and this peak equal to 0.9 and the peak is constant with respect to time. Fig. (3) shows the testing of the network on a more complicated signal that is given by the equation.

$$x = 0.5\cos(2\pi i 20T + 0.3\pi) + 0.7\cos(2\pi i 10T) \tag{5}$$

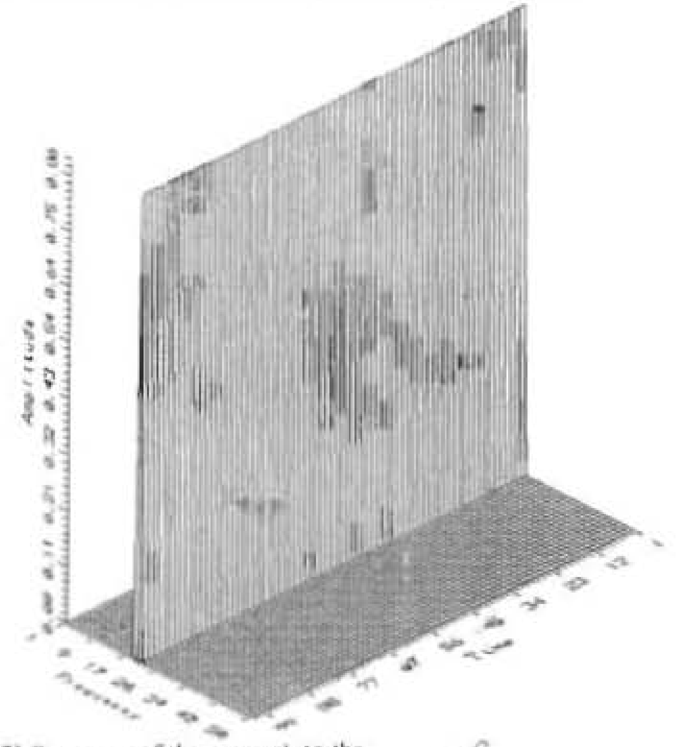
Notice from this figure that the output equal to the amplitude and independent of the phase of the signal. A 30Hz carrier was amplitude modulated with a 20Hz modulating signal at a modulation index of 0.5 and the compound signal was applied to the network. This AM signal is given by equation 6 and its response is shown in fig. (4).

$$x = 0.7(1 + 0.5\cos(2\pi i 20T))\cos(2\pi i 30T) \tag{6}$$

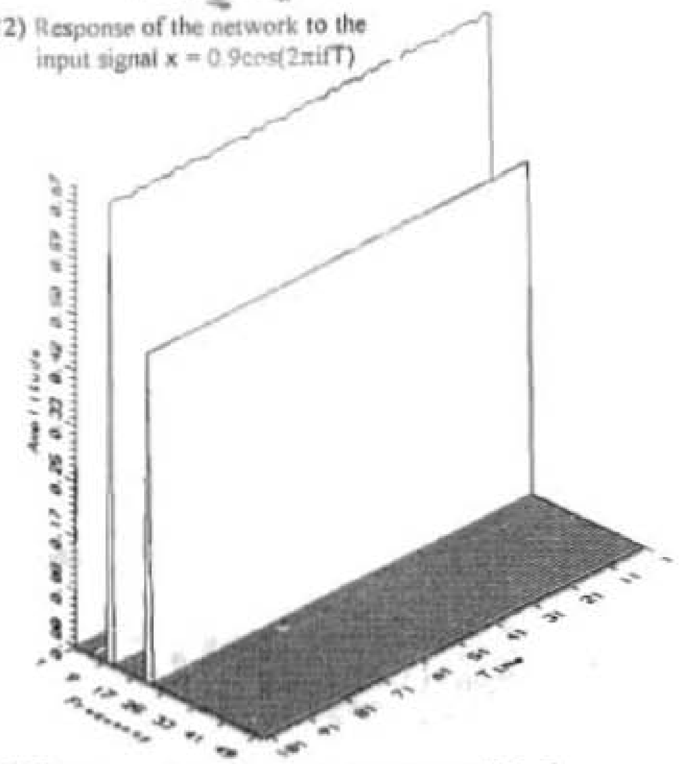
Fig. (5) shows the response of the network to a frequency modulated signal with a carrier 30Hz and a modulating signal 10Hz at a modulation index of 0.4 so that the response is the carrier and four side bands. This equation is as follows:

$$x = 5\cos(2\pi i 30T + 0.4\sin(2\pi i 10T)) \tag{7}$$

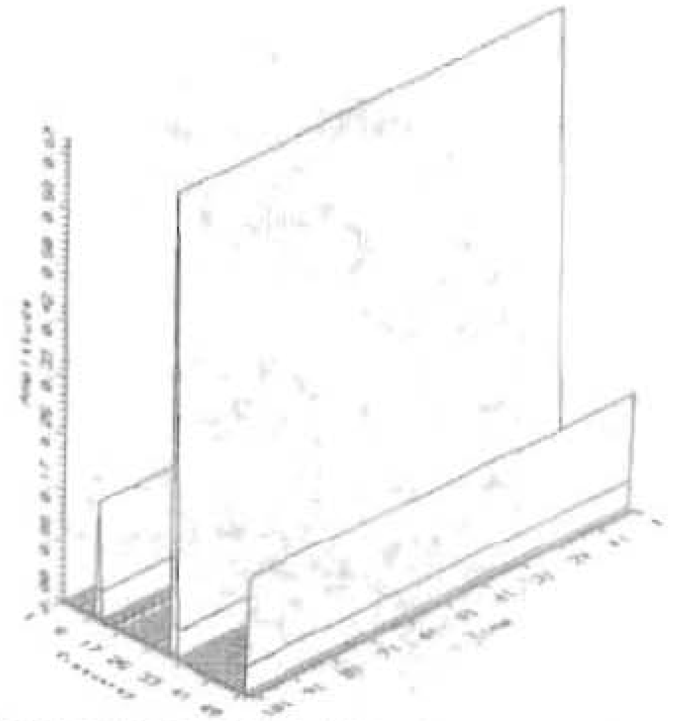
Fig. (6) shows the response of the network to a speech signal for an adult male sampled at only 102Hz. The sampling frequency was such low due to the RAM limitations of the computer we used because as we saw earlier that increasing the maximum frequency to be

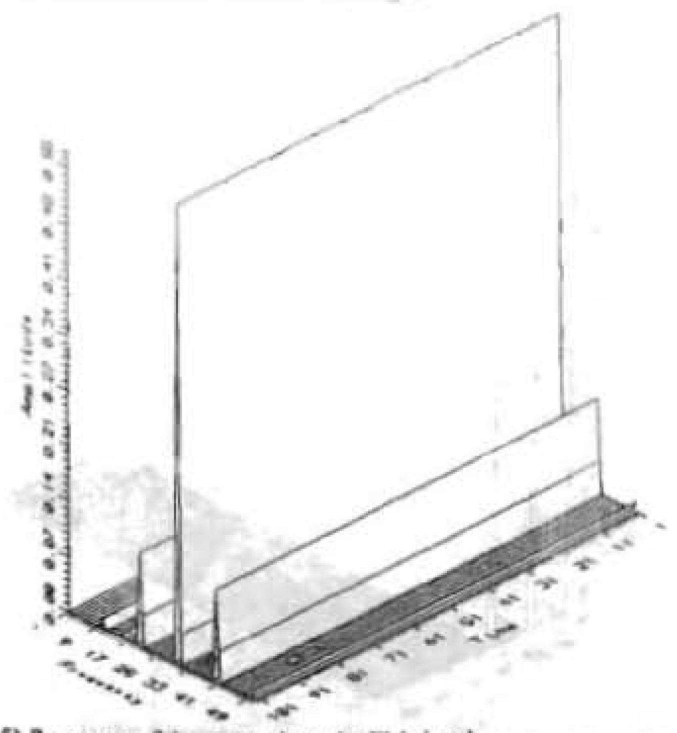


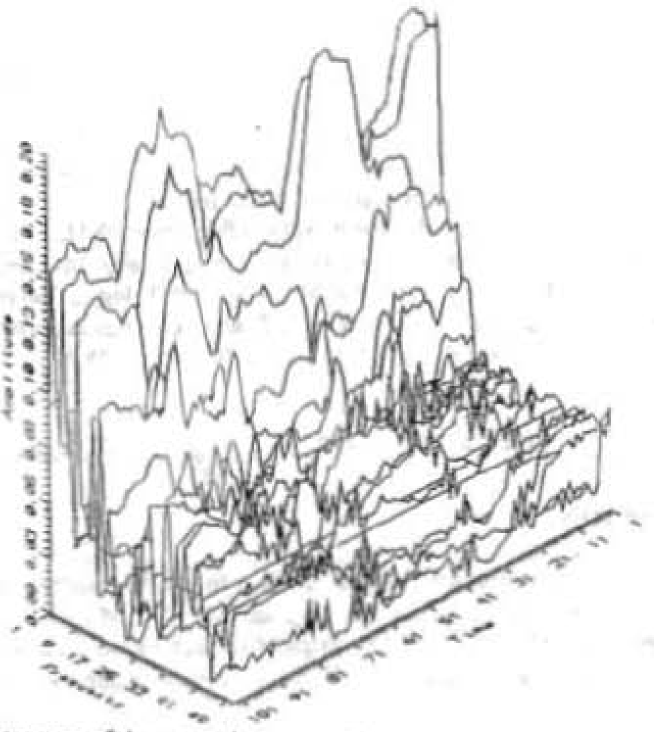
Fig(2) Response of the network to the input signal x = 0.9cos(2πifT)



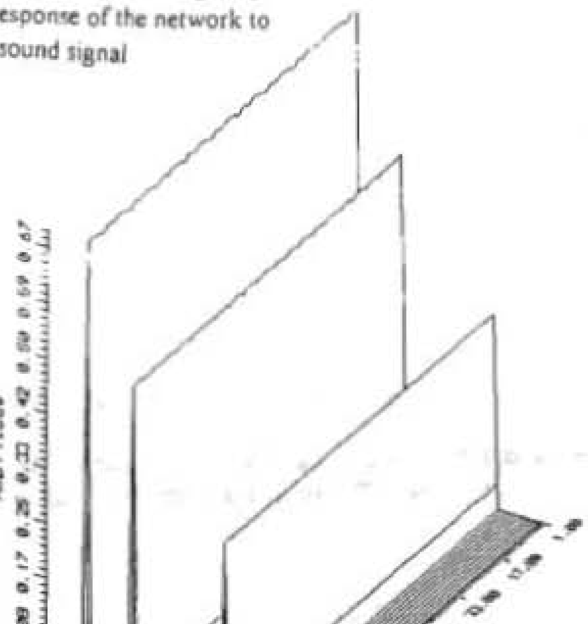
Fig(3) Response of the network to the compound signal $x = 0.5\cos(2\pi\, i20T + 0.3\pi\,) + 0.7\cos(2\pi\, i10T)$


Fig(4) Response of the network to the AM signal


Fig(5) Response of the network to the FM signal



Fig(6) Response of the network to a sound signal



Fig(7) Response of the net at sampling freq. 100 times that used during training phase.

detected will require the increase of the number of neurons in the input and the output layers which will increase the required RAM for such calculations. In reference [2] it has been demonstrated that working at a sampling frequency different from that defined by the Nyquest role during the training of the network will multiply the output detected frequencies by the ratio of the new sampling frequency to the sampling frequency used during the training. For example, if we train the network to detect the frequencies 1, 2, 3, ..., to 100Hz with the sampling frequency of 200Hz (double the maximum frequency 100Hz), then this network will detect the frequencies 5, 10, 15, ..., to 500Hz if the input signal is sampled at 1000Hz (5 times the 200 Hz used during the training of the network). Fig. (7) demonstrates this fact where the complex signal x of the frequencies 1, 2, and 4kHz that is given by equation 8 was introduced to the same network and the sampling frequency was 10200Hz (100 times that during training).

$$x = 0.5\cos(2\pi i2000T + 0.3\pi) + 0.7\cos(2\pi i1000T) + 0.3\cos(2\pi i4000T) \qquad (8)$$

5. CONCLUSION

This network has introduced an easy method to get the spectrogram of a signal. The problem with this method is that it needs a large amount of RAM during the training session. So, such network is useful only at lower frequencies such as the medical applications. This research can be extended in the future to find an easy method to apply such networks at higher frequencies

REFERENCES

1. El-Adawy M. I. "Filtering effect of the simple perceptron" Eleventh National Radio Science conference, NRSC'94, Military Technical College, Cairo, Egypt, C32, pp. 1-8, March 1994.
2. El-Adawy M. I. "Neural network for the detection of the frequency contents of a short time signal", Eng. Research Bulletin, Faculty of Eng. and technology, Mataria, Helwan University, Cairo, Vol. 4, pp. 328-336, August 1994.
3. Philip D. Wasserman, "Neural computing, theory and practice" Van Nostrand Reinhold, 1989.
4. David P. Morgan, Christopher L. Scofield, "Neural network and speech processing" Kluwer Publishers, 1989
5. Jacek M. Zurada, "Introduction to artificial neural systems", West Publishing Company, 1992.
6. Alan V, Oppenhiem, Ronald W. Schafer, "Digital signal processing" Prentice-Hall 1975.
7. Lawrence R. Rabiner, Bernard Gold, "Theory and application of digital signal processing", Printice-Hall 1975.

A New Systolic Array for Two-Dimensional Discrete Cosine Transform

هيكل انقباضي جديد للتحول الجيبي التمامي المتقطع ثنائي الأبعاد

Samia A. Mashali
Computer & Systems Dept., Electronics Research Institute
Cairo, Egypt.
E-mail : ERI @ FRCU.EUN.EG

Abstract -An efficient systolic implementation of the two-dimensional discrete cosine transform (2-D DCT) is developed. It offers a highly regular structure ideally suited for VLSI implementation. The computation complexity of the proposed implementation is O(N) as compared to the two-dimensional fast cosine transform algorithm of O(N^2) [1].

ملخص البحث:

يعرض هذا البحث هيكل انقباضي للتحول الجيبي التمامي المتقطع ثنائي الابعاد ذو كفاءة عالية ويوفر هذا الهيكل تركيبة منتظمة تلائم تنفيذ الدوائر المتكاملة شديدة الاتساع.
وبالحساب يتبين ان درجة تعقيد الخوارزم تتناسب مع N وهذه تعتبر افضل من خوارزم التحويل الجيبي التمامي السريع الثنائي الأبعاد والذي تصل درجة تعقيده الى N^2

1 - Introduction

The discrete cosine transform (DCT) is a better approximation to the statistically optimal Karhunen - loeve transform (KLT) than most other orthogonal transforms [2],[3]. Therefore, it plays a very important role in image and speech coding.

The DCT transform requires massive and complicated data manipulation that lead to consideration and implementations employing parallelism and pipelining. The systolic array has the advantages of pipelinability, regularity, locality, and scalability, thus making it very suitable for VLSI signal processing [4]-[6].

Recently, Anew recursive algorithm for computing the 1-D DCT [7],[8] and the 2-D DCT [1] is introduced. The algorithm is similar to a decimation-in-frequency Cooley - Tukey FFT algorithm. Its computation complexity is O(N^2). In this paper a new systolic array for the 2-D DCT is developed. It reduces the computation complexity to O(N).

The approach for mapping algorithms onto Systolic hardware can be described as follows :

- Given the algorithm, a set of recursive expressions is derived.

• Each iteration in the system recursive expression is assigned a processing element (PE) to perform it.

• From the recursive expressions, the structures of the PE and the systolic array are obtained.

• The number of PEs is determined by the number of iterations required to produce the final output.

In section II, a derivation of the recurrence formulae is presented. The proposed systolic array is described in section III while its complexity is discussed in section IV.

II. Derivation of Recurrence Formulae

The 2-D DCT of the real sequence $x(n_1,n_2)$ can be defined as

$$X(k_1,k_2)=\frac{4\,\epsilon_{k1}\epsilon_{k2}}{N^2}\sum_{n1=0}^{N_1-1}\sum_{n2=0}^{N_2-1}x(n_1,n_2)\cos\{\frac{\pi(2n_1+1)k_1}{2N_1}\}\cos\{\frac{\pi(2n_2-1)k_2}{2N_2}\} \quad (1)$$

where

$$\epsilon_{ki}=\begin{cases}\frac{1}{\sqrt{2}} & \text{for } k_i=0\\ 1 & \text{otherwise}\end{cases}\ ;\ i=1,2$$

For simplicity, we define $C_{k_1}^{2n_1+1}$, $C_{k_2}^{2n_2+1}$ and $X(k_1,k_2)$ such that

$$C_{k_1}^{2n_1+1}=\cos\{\frac{\pi(2n_1+1)k_1}{2N_1}\} \quad (2)$$

$$C_{k_2}^{2n_2+1}=\cos\{\frac{\pi(2n_2+1)k_2}{2N_2}\} \quad (3)$$

$$\tilde{X}(k_1,k_2)=\sum_{n1=0}^{N_1-1}\sum_{n2=0}^{N_2-1}x(n_1,n_2)\,C_{k_1}^{2n_1+1}\,C_{k_2}^{2n_2+1} \quad (4)$$

Substitution of (4) into (1) yields

$$X(k_1,k_2)=\frac{4\,\epsilon_{k1}\epsilon_{k2}}{N^2}\,\tilde{X}(k_1,k_2)$$

Since the leading constant $\frac{4\,\epsilon_{k1}\epsilon_{k2}}{N^2}$ is a scale factor, we can consider $\bar{X}(k_1,k_2)$ instead of $X(k_1,k_2)$ in the following -By trigonometric function difference relations, we have the following equations

for $C_{k_1}^{2n_2+1}$

$$C_{k_1}^{2n_1+1} - C_{k_1}^{2n_1-1} = -2\sin\{\frac{\pi n_1 k_1}{N_1}\} \sin\{\frac{\pi k_1}{2N_1}\} \qquad (5)$$

$$C_{k_1}^{2n_1+1} - C_{k_1}^{2n_1-3} = -2\sin\{\frac{p\,(n_1-1)k_1}{N_1}\} \sin\{\frac{p\,k_1}{2N_1}\} \qquad (6)$$

&

$$\sin\{\frac{\pi n_1 k_1}{N_1}\} - \sin\{\frac{\pi (n_1-1)}{N_1}\} = 2\cos\{\frac{\pi (2n_1+1)k_1}{2N_1}\} \sin\{\frac{\pi k_1}{2N_1}\}$$

$$= 2\,C_{k_1}^{2n_1-1} \sin\{\frac{\pi k_1}{2N_1}\} \qquad (7)$$

Taking the difference between (5) and (6), we find

$$C_{k_1}^{2n_1+1} = 2C_{k_1}^{2n_1-1} - 2\sin\{\frac{p\,k_1}{2N_1}\}\,[\sin\{\frac{p\,n_1 k_1}{2N_1}\}$$

$$-\sin\{\frac{\pi (n_1-1)k_1}{N_1}\}] - C_{k_1}^{2n_1-3} \qquad (8)$$

with the substitution of (7) into (8), we have

$$C_{k_1}^{2n_1+1} = [-2^2\sin^2\{\frac{p\,k_1}{2N_1}\} + 2]\, C_{k_1}^{2n_1-1} - C_{k_1}^{2n_1-3} C_{k_1}$$

$$= [(2C_{k_1}^{1})^2 - 2]\, C_{k_1}^{2n_1-1} - C_{k_1}^{2n_1-3} \qquad (9)$$

Similarly

$$C_{k_2}^{2n_2+1} = [(2C_{k_2}^{1})^2 - 2]\, C_{k_2}^{2n_2-1} - C_{k_2}^{2n_2-3} \qquad (10)$$

It is evident that we can derive one recurrence relation among coefficients of three successive data points when transforming for a fixed n. This leads to the introduction of systolic array schemes in hardware design.

III- Proposed Systolic Array

If we consider the DCT using (4),(9) & (10) simultaneously, the complete recurrence formulae are :

$$C_{k_1}^{2n_1+1} = A(C_{k_1}^{1})\, C_{k_1}^{2n_1-1} - C_{k_1}^{2n_1-3} \qquad (11)$$

$$C_{k_2}^{2n_2+1} = A(C_{k_2}^{1})\, C_{k_2}^{2n_2-1} - C_{k_2}^{2n_2-3} \qquad (12)$$

$$Z_{k1,k2}(n_1,n_2) = x(n_1,n_2)\, C_{k_1}^{2n_1+1} C_{k_2}^{2n_2+1} + Z_{k1,k2}(n_1-1,n_2) \qquad (13)$$

Where

&

$$A(C_{k_2}^{1}) = [(2\, C_{k_2}^{1})^2 - 2]$$

The systolic array for realizing the above recurrence formulae is shown in Fig 1. It requires $(N_1 N_2)$ processing elements. Four types of processing elements are used. The first row of the array consists of the basic elements of Fig. 2a (PE1). The left most processor in each row (PE1') is shown in Fig. 2b. The processing elements shown in Fig. 2 (a) computes equation (12) while the elements in Fig. 2 (b) computes equation (11). The basic element stored with the value for one point of the data sequence computes the cosine values for the next element and accumulates the partial transform result. These processes are done in parallel, thereby consuming only one addition and one multiplication time equivalently. The other processing elements (PE2) are shown in Fig. 3 . They use the cosine values already computed in the previous processors PE1 or PE1'. PE2 is more simple (one multiplier + one adder) than PE1 but consumes the same computation time. This reduces the hardware complexity of the overall array. The upper left processing element (PE) is the only PE that computes equation (11) & (12) simultaneously. Although the number of multipliers and adders differ in each type of processing element, they still consume the same computation time (one addition and one multiplication).

To obtain the final value $\hat{X}(k_1,k_2)$ an additional row of PE2 s with zero data sequence is introduced. The reason for choosing this type processors is to maintain the regularity of the array. $\hat{X}(1,1), \hat{X}(2,1), \ldots\ldots\ldots$ are sequentially produced from the right most processing element.

IV- Computation Complexity.

$\hat{X}(k_1,k_2)$can be obtained after (N_1+N_2) computation steps. if we let M and A be the time for performing one multiplication and one addition, respectively, then M+A elapses in one processing element, i.e one computation step. Thus, $\hat{X}(k_1,k_2)$ is available after $(N_1+N_2)(M+A)$ time units. For the sake of pipelining, the computation of the whole transformed sequence needs $\{(N_1+N_2)+(k_1+k_2)\}(M+A)$ units of time. The number of additions $A(N,N)$ and multiplications $M(N,N)$ needed to compute the $(N \times N)$ - point DCT using the proposed systolic array are

$$A(N \times N) = 4N$$
$$M(N \times N) = 4N$$

Thus the number of multiplications & additions for the proposed algorithm is O(N), while for the fast cosine transform algorithm {equation (27) in [1] } it is $O(N^2)$.

V-Conclusion:

For real - time processing, the hardware cost paid for the recursive computation is worth compared to the simple prestorage schemes for transform kernel values. The computations of the kernel values and the intermediate results are executed in parallel in each basic element, thus consuming only one multiplication and one addition at a time. This algorithm results in a significant reduction in computation time relative to the fast cosine transform implementation [1].

References

[1] Sc. Chan and K.L.HO, "A New Two - Dimensional Fast Cosine Transform Algorithm", IEEE Signal Processing, vol. 39, No. 2, pp. 481-485 Feb. 1991.

[2] N. Ahmed, T. Natarajan, and K.R. Rao, "Discrete cosine Cosine and Fourier Transforms ", IEEE Trans Comput., vol. C-23 , pp 90-94, Jan. 1974.

[3] M Hamidi and J. Pearl, "Comparison of the cosine and Fourier Transforms of Markov - I Signals, " IEEE Trans Acoust., Speech, Signal Processing vol. ASSSP-24, pp. 428-429, Oct. 1976.

[4] H.T. Kung, "Why systolic architectures" Computer, pp. 37-46, Jan. 1992.

[5] S. Y. Kung, VLSI Array Processors, Prentice Hall, 1988.

[6] S. Manohar and G. Baudet "Pragmatic Approach to Systolic Design", IEE Proceeding, vol. 137, PL. E.No. 4, pp. 277 - 282 July 1990.

[7] H. S. Hou, "A Fast Recursive Algorithm For Computing the Discrete Cosine Transform, "IEEE Trans. Acoust., Speech , Signal Processing, vol. ASSP - 35, pp. 1445 - 1461, Oct. 1987.

[8] B.G.Lee "A New Algorithm to compute the Discrete Cosine Transform" IEEE Trans. Acoust, Speech , Signal Processing, vol. ASSP - 32, pp. 1243-1245, Dec. 1984.

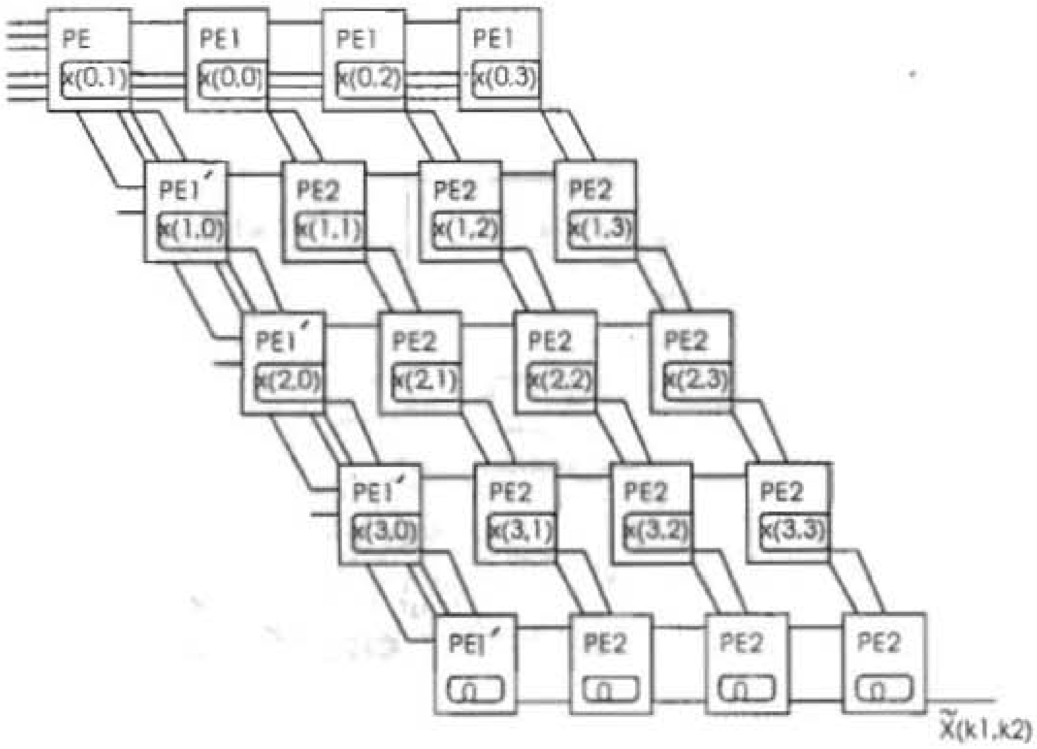
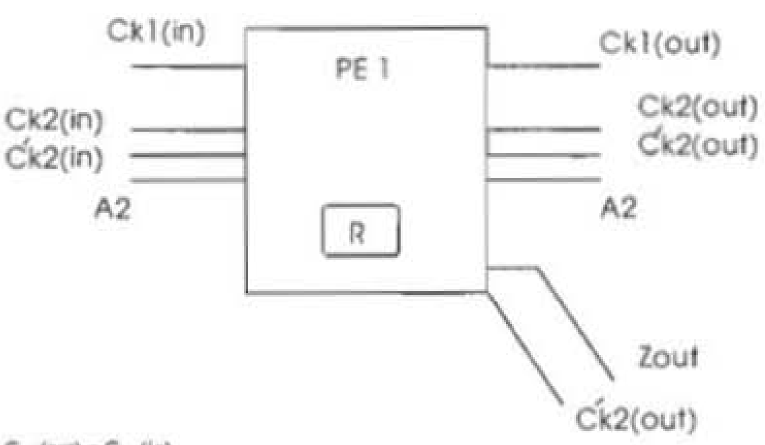
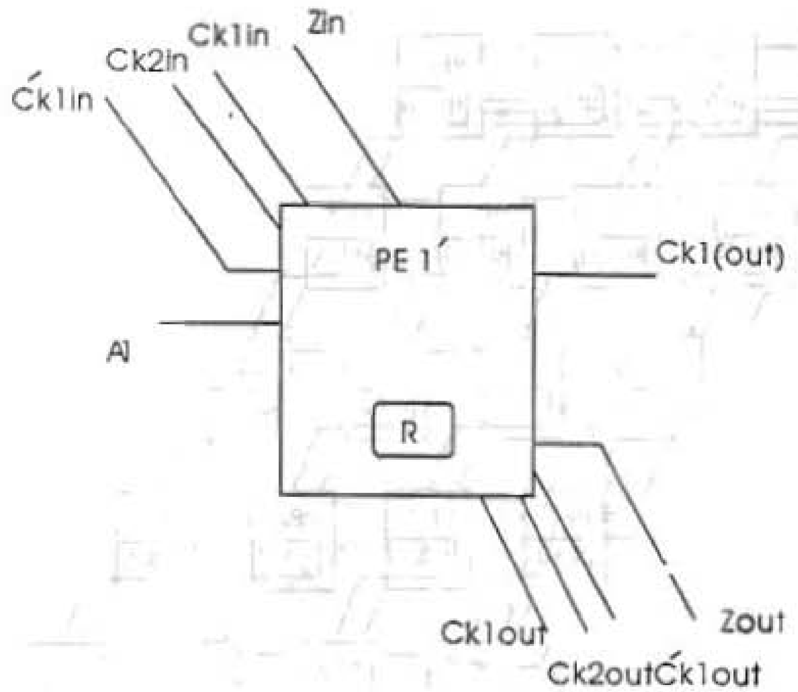


Fig. 1 The proposed systolic array for 2-D DCT ($N_1=N_2=4$)



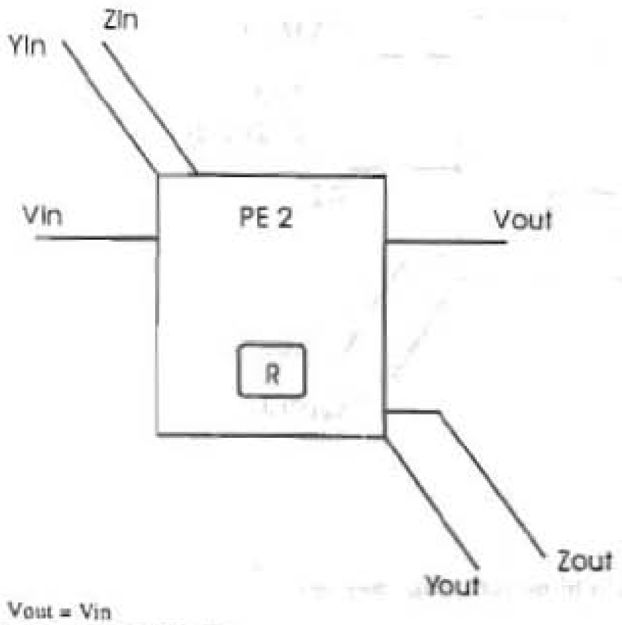
$C_{k1}(out) = C_{k1}(in)$
$C_{k2}(out) = A_2 * C_{k2}(in) - C'_{k2}(in)$
$C'_{k2}(out) = C_{k2}(in)$
$Zout = R * C_{k1}(in)\ C_{k2}(in)$

Fig. 2(a) PE 1 (in the 1st row of the array) computes (12)



$C_{k1}out = C_{k1}in$
$C_{k2}out = C_{k2}in$
$C'_{k1}out = A_1 C_{k1}in \cdot C'_{k1}in$
$Zout = Zin + R^* C_{k1}in C_{k2}in$

Fig. 2 (b) PE 1´ (the left most processing elements) computes(11)



Vout = Vin
Zout = Zin + R*Vin Yin
Yout =Yin

Fig. 3 PE 2