A New Convolution Structure for the Realisation of the Discrete Cosine Transform

Yuk-Hee Chan and Wan-Chi Siu

Department of Electronic Engineering
Hong Kong Polytechnic
Hung Hom, Kowloon
Hong Kong

ABSTRACT

In this paper, we present a new formulation for converting a length-N($=2^m$) Discrete Cosine Transform into two length-N/2 correlations. This formulation enables us to realise the Discrete Cosine Transform with a reduced number of operations compared to conventional approaches and it also results in extremely regular structure which is most suitable for the realisation using distributed arithmetic.

Introduction

The discrete cosine transform (DCT) has been widely used as a tool for digital signal processing applications, such as image coding. Many algorithms for the computation of the DCT have been proposed since the introduction of the DCT in 1974 by Ahmed et al.[1]. These algorithms can broadly be classified into two groups: 1) indirect computation through fast Discrete Fourier Transform and Walsh Hadamard transform[2-5] and 2) direct computation of DCT through matrix decomposition or recursive computation[6-15].

Among them, Lee[6]'s Algorithm and Vetterli[15]'s algorithm meet the minimum known number of multiplications to implement a length 2^m DCT. Lee[6]'s algorithm decomposes an N-point DCT into the sum of two N/2 point DCT's repeatedly to achieve the number of real multiplications as $(N/2)\log_2 N$ for an N-point DCT, with $N=2^m$. Vetterli[15]'s algorithm is special as it is a recursive algorithm which uses the property that DCT, DFT, sin-DFT and cos-DFT can be decomposed into two half-length of the individual transforms. Hence, it is difficult to classify it critically. Narasimha[5]'s algorithm is a typical example of group one. It uses an N-point discrete Fourier transform (DFT) algorithm to evaluate a DCT by a simple rearrangement of the input data, which requires $N\log_2 N-N+2$ real multiplications.

In this paper, we present a new formulation for converting a length-N($=2^m$) Discrete Cosine Transform into two length-N/2 correlations. This formulation enables us to realise the Discrete Cosine Transform with a reduced number of operations compared to conventional approaches and it also results in extremely regular structure which is most suitable for the realisation using distributed arithmetic.

The Algorithm Derivation

The DCT[1] of a real data sequence $\{x(i):i=0,1,...N-1\}$, where $N=2^m$ and m is an integer, is defined by

$$X(k)=\sum_{i=0}^{N-1} x(i)\cos[2\pi(2i+1)k/4N] \qquad \text{for } k=0,1,...N-1 \quad (1)$$

Let $y(i) = x(2i)$

$$y(N-i-1) = x(2i+1) \qquad \text{for } i=0,1,...N/2-1 \quad (2)$$

then,
$$X(k)=\sum_{i=0}^{N-1} y(i)\cos[2\pi(4i+1)k/4N] \qquad \text{for } k=0,1...N-1 \quad (3)$$

Now let us split X(k) into odd and even sequences, say X(2k+1) and X(2k), and look for a formulation of the form $\cos[(4i+1)(4k+1)2\pi/4N]$ for cosine terms. The reason for such an arrangement will be clear at a latter stage.

For odd terms of X(k):

$$X(2k+1)=\sum_{i=0}^{N-1} y(i)\cos[2\pi(4i+1)(2k+1)/4N] \qquad \text{for } k=0,1,...N/2-1 \quad (4)$$

We define
$$X'(k)=\sum_{i=0}^{N-1} y(i)\cos[2\pi(4i+1)(4k+1)/4N] \qquad \text{for } k=0,1,...N-1 \quad (5)$$

then it can be shown that

$$X(4k+1)=X'(k) \qquad \text{for } k=0,1,...N/4-1 \quad (6)$$

$$X(2N-4k-1) = -X'(k) \qquad \text{for } k=N/4,N/4+1,...N/2-1 \quad (7)$$

For even terms of X(k):

$$X(2k)=\sum_{i=0}^{N-1} y(i)\cos[2\pi(4i+1)(2k)/4N] \qquad \text{for } k=0,1,...N/2-1 \quad (8)$$

If we define
$$X^*(k)=\sum_{i=0}^{N-1} z(i)\cos[2\pi(4i+1)(2k+1)/4N] \qquad \text{for } k=0,1,...N-1 \quad (9)$$

where $z(i)=2y(i)\cos[(2\pi/4N)(4i+1)]$ for $i=0,1,...N-1$ (10)

then we have $X^*(N/2-1)=X(N-2)$
and $X^*(k)=X(2k)+X(2k+2)$ for $k=0,1,...N/2-2$ (11)

In order to have the required form, let us define again

$$F(k)=\sum_{i=0}^{N-1} z(i)\cos[2\pi(4i+1)(4k+1)/4N] \qquad \text{for } k=0,1,...N-1 \quad (12)$$

We can obtain $X^*(k)$ through the realisation of eqn. 12 since it is readily shown that

$$X^*(2k) = F(k) \qquad \text{for } k=0,1,...,N/4-1 \quad (13)$$

$$X^*(N-2k-1) = -F(k) \qquad \text{for } k=N/4,N/4+1,...N/2-1 \quad (14)$$

Hence, the even terms of X(k) can be determined by the sequence $\{F(k):k=0,1..N/2-1\}$ through eqn.11, 13 and 14.

Let us summarize what we have done so far. X(k) can be computed through two steps:

Step 1: Compute X'(k) and F(k) for $k=0,1,...N/2-1$ and

Step 2: Compute X(k) by the following set of equations:

$$\begin{aligned} X(4k+3) &= -X'(N/2-k-1) \\ X(4k+2) &= -F(N/2-k-1) - X(4k+4) \\ X(4k+1) &= X'(k) \\ X(4k) &= F(k) - X(4k+2) \qquad \text{for } k=N/4-1,...1,0 \quad (15) \end{aligned}$$

where X(N)is defined as zero.

Now let us define a bijective mapping[14] on the set $\text{INDEX} = (i: i=0,1,...N-1)$, where $N=2^m$, to itself:

$$<5^v>_{4N} = 4u+1 \qquad ,\text{where } u,v \in 0,1,...N-1 \quad (16)$$

then from eqn. 5 and 12,

$$\text{we have } X''(k) = \sum_{i=0}^{N-1} y''(i)\cos[(2\pi/4N) <5^{i+k}>_{4N}] \qquad \text{for } k=0,1...N-1 \quad (17)$$

$$\text{and} \quad F''(k) = \sum_{i=0}^{N-1} z''(i)\cos[(2\pi/4N) <5^{i+k}>_{4N}] \qquad \text{for } k=0,1...N-1 \quad (18)$$

$$\begin{aligned} \text{where } X''(k) &= X'(\ (<5^k>_{4N} - 1)/4\ ) \\ F''(k) &= F(\ (<5^k>_{4N} - 1)/4\ ) \\ y''(i) &= y(\ (<5^i>_{4N} - 1)/4\ ) \\ z''(i) &= z(\ (<5^i>_{4N} - 1)/4\ ) \qquad \text{for } i,k \in \text{INDEX} \quad (19) \end{aligned}$$

We note that both of X"(k) and F"(k) are in correlation form which can be computed easily as there exists a number of fast and easy- implemented algorithms[16] for the realisation of such a structure.

Further Simplification and Realisation

Consider the correlation F"(k) in eqn.(18). We have

$$\begin{bmatrix} F''(0) \\ F''(1) \\ .. \\ F''(N-1) \end{bmatrix} = \begin{bmatrix} C(0) & C(1) .. & C(N-1) \\ C(1) & C(2) .. & C(0) \\ .. & .. \quad .. .. & .. \\ C(N-1) & .. & C(N-2) \end{bmatrix} \begin{bmatrix} z''(0) \\ z''(1) \\ .. \\ z''(N-1) \end{bmatrix} \qquad (20)$$

$$\text{where } C(n) = \cos[(2\pi/4N) <5^n>_{4N}] \qquad \text{for any integer } n \quad (21)$$

As $F''(N/2+n) = -F''(n)$ for $n=0,1...N/2-1$, only F"(0),..F"(N/2-1) are required to compute. One may observe that $C(N/2+n) = -C(n)$ for $n=0, 1...N/2-1$. Hence we have

$$\begin{bmatrix} F''(0) \\ F''(1) \\ .. \\ F''(N/2-1) \end{bmatrix} = \begin{bmatrix} C(0) & C(1) .. & C(N/2-1) \\ C(1) & C(2) .. & -C(0) \\ .. & .. \quad .. .. & .. \\ C(N/2-1) & .. & -C(N/2-2) \end{bmatrix} \begin{bmatrix} g(0) \\ g(1) \\ .. \\ g(N/2-1) \end{bmatrix} \qquad (22)$$

$$\text{where } g(n) = z''(n) - z''(N/2+n) \qquad \text{for } n=0,1...N/2-1$$

This saves almost half of the number of operations required to realise F"(k) by eqn.(20). Furthermore, as $g(n) = 2\{y''(n)+y''(N/2+n)\}C(n)$ for $n=0,1..N/2-1$, we need not compute z(n) from y(n) term by term as shown in eqn.(10) to compute the sequence $\{g(n)\}$. This further saves N/2 multiplications.

Let us clarify our proposal with a length 8 DCT with input sequence $\{x(i):i=0,1,...7\}$. Rearranging the data according eqn.(2) and from eqn.(10), we have

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \\ y(4) \\ y(5) \\ y(6) \\ y(7) \end{bmatrix} = \begin{bmatrix} x(0) \\ x(2) \\ x(4) \\ x(6) \\ x(7) \\ x(5) \\ x(3) \\ x(1) \end{bmatrix}, \quad \begin{bmatrix} z(0) \\ z(1) \\ z(2) \\ z(3) \\ z(4) \\ z(5) \\ z(6) \\ z(7) \end{bmatrix} = \begin{bmatrix} 2y(0)\cos\alpha \\ 2y(1)\cos5\alpha \\ 2y(2)\cos9\alpha \\ 2y(3)\cos13\alpha \\ -2y(4)\cos\alpha \\ -2y(5)\cos5\alpha \\ -2y(6)\cos9\alpha \\ -2y(7)\cos13\alpha \end{bmatrix}$$

where $\alpha = \pi/16$.

From eqns. (18), (20) and (22), we have

$$\begin{bmatrix} F(0) \\ F(1) \\ -F(2) \\ -F(3) \end{bmatrix} = \begin{bmatrix} \cos\alpha & \cos5\alpha & \cos25\alpha & \cos29\alpha \\ \cos5\alpha & \cos25\alpha & \cos29\alpha & -\cos\alpha \\ \cos25\alpha & \cos29\alpha & -\cos\alpha & -\cos5\alpha \\ \cos29\alpha & -\cos\alpha & -\cos5\alpha & -\cos25\alpha \end{bmatrix} \begin{bmatrix} 2(y(0)+y4))\cos\alpha \\ 2(y(1)+y(5))\cos5\alpha \\ 2(y(6)+y(2))\cos25\alpha \\ 2(y(7)+y(3))\cos29\alpha \end{bmatrix}$$

Similarly, from eqn.(17),

$$\begin{bmatrix} X'(0) \\ X'(1) \\ -X'(2) \\ -X'(3) \end{bmatrix} = \begin{bmatrix} \cos\alpha & \cos5\alpha & \cos25\alpha & \cos29\alpha \\ \cos5\alpha & \cos25\alpha & \cos29\alpha & -\cos\alpha \\ \cos25\alpha & \cos29\alpha & -\cos\alpha & -\cos5\alpha \\ \cos29\alpha & -\cos\alpha & -\cos5\alpha & -\cos25\alpha \end{bmatrix} \begin{bmatrix} y(0)-y(4) \\ y(1)-y(5) \\ y(6)-y(2) \\ y(7)-y(3) \end{bmatrix}$$

Hence,

$$\begin{aligned} X(7) &= -X'(2), & X(6) &= -F(2), \\ X(5) &= X'(1), & X(4) &= F(1)-X(6), \\ X(3) &= -X'(3), & X(2) &= -F(3)-X(4), \\ X(1) &= X'(0), & X(0) &= F(0)-X(2) \end{aligned}$$

Note that values of F"(N/2-n-1)'s of eqn.(22) are exactly equal to the coefficients of z^n's of the polynomial F(z):

$$F(z) = C(z)G(z)\bmod(z^{N/2}+1) \qquad (23)$$

$$\begin{aligned} \text{where } G(z) &= g(N/2-1)z^{N/2-1} + ... + g(1)z + g(0), \text{ and} \\ C(z) &= C(0)z^{N/2-1} + ... + C(N/2-2)z + C(N/2-1) \end{aligned}$$

Recall that an Nth order polynomial can be interpolated exactly through N+1 points using Lagrange's interpolation formula. This suggests a method for determining the polynomial F(z). Firstly, the (N-2)th order polynomial C(z)G(z) is interpolated by using Lagrange's interpolation formula with $G(z_0)C(z_0)$, $G(z_1)C(z_1)$, ... and $G(z_{N-2})C(z_{N-2})$, which requires N-1 multiplications. Then F(z) can be determined by equation (23). Hence, values of F"(k)'s in eqn.(22) can be determined with N-1 multiplications only. Values of X"(k)'s can also be computed by the same method. This method reduces the number of real multiplications of the new algorithm to 5N/2-2.

Table 1 shows a comparison of the multiplicative complexity among the new algorithm, Lee[6]'s algorithm, Vetterli[15]'s algorithm and Narasimha[5]'s algorithm. The new algorithm shows its superiority in multiplicative complexity compared to other algorithms when N is larger than 16.

Table 1. Comparision of number of multiplications between algorithms.

	New al.	Algorithm[6]	Algorithm[15]	Algorithm[5]
N	5N/2-2	(N/2)log2N	(N/2)log2N	Nlog2N-N+2
8	18	12	12	18
16	38	32	32	50
32	78	80	80	130
64	158	192	192	322
128	318	448	448	770
256	638	1024	1024	1794
512	1278	2304	2304	4098

Hardware Realisation

The proposed algorithm is very structural. Hence it is most suitable for VLSI implementation. Consider Fig.1 which shows a block diagram of the implementation of the new algorithm. Only three simple permutation networks and two correlation hardware modules are required. Besides, nearly half of hardware cost can be saved as hardware modules A and B can be realised serially using a single unit. In this case the realisation time is unavoidably increased.

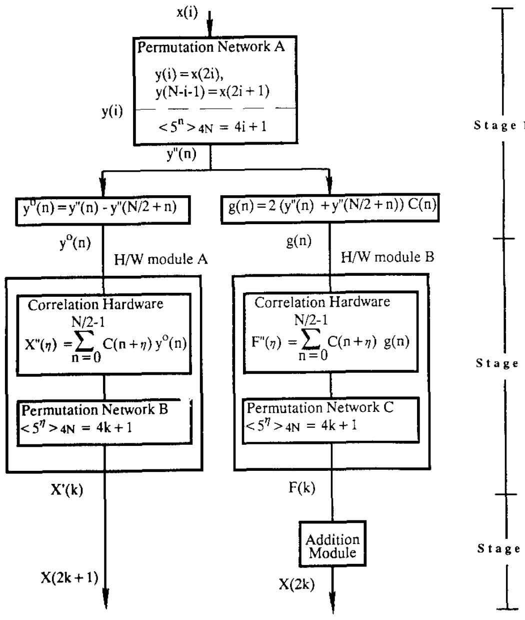

Figure 1. Hardware implementation of the proposed algorithm

There are a number of implementation methods to build a cyclic correlation hardware module. One of the possibilities is to realise it by distributed arithmetic[17]. The distributed arithmetic is so regular and structural that it is very suitable for VLSI realisation. The advantages of this architecture are: (1) no actual multiplication involved as multipliers are replaced by memory look-up tables, (2) high accuracy as it suffers fewer rounding/truncation error than the other structures, (3) possible for modular circuit design as the structure is extremely regular and (4) simple structure which leads to a saving of gate count and makes routing easy. These features allow a high speed circuit design composed of memories, adders and registers only.

Consider eqn.(22), if g(-n) is defined as -g(N/2-n), we have

$$F''(k) = \sum_{\eta=0}^{N/2-1} g(\eta-k)\, C(\eta) \qquad (24)$$

As $g(\eta\text{-}k)$ can be expressed in the way

$$g(\eta-k) = -g(\eta-k)_0 + \sum_{j=1}^{M-1} g(\eta-k)_j\, 2^{-j} \qquad (25)$$

where M, $g(\eta\text{-}k)_j$ and $g(\eta\text{-}k)_0$ are the word length, the jth most significant bit and the sign bit respectively. After scaling to 2's-complement fractional number, equation (24) becomes

$$F''(k) = \sum_{j=1}^{M-1} \{ \sum_{\eta=0}^{N/2-1} g(\eta-k)_j\, C(\eta)\}\, 2^{-j} - \sum_{\eta=0}^{N/2-1} g(\eta-k)_0\, C(\eta) \qquad (26)$$

Value of $\sum_{\eta=0}^{N/2-1} g(\eta-k)_j\, C(\eta)$ can be pre-calculated and stored in a ROM with ROM size = $2^{N/2}$ words. Then F"(k) can be obtained by M ROM accesses and M-1 shift-additions after g(n)'s are available. The implementation of the convolution by distributed arithmetic is as shown in Fig.2. X"(k) can also be computed by the same approach.

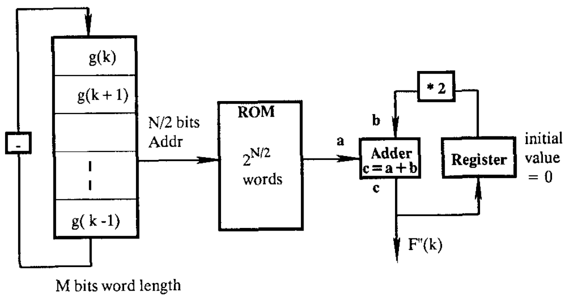

Figure 2. Implementation of convolution with Distributed Arithmatic

To speed up the whole process, we introduce the concept of pipelining. As shown in Fig.1, the whole process is divided into three stages. Stage 1 includes permutation Network A and hardware modules for realising $y^o(n)$ and g(n). Fig.3 is a typical approach to realise $y^o(n)$ and g(n). Note that no address generation is required to obtain the values for C(n). Actually, values of C(n) are stored in a circular buffer such that it is automatically sent out one by one sequentially. The circular buffer can be constructed with either ROM or RAM. It would be more flexible if RAM is used. One multiplier is required in this stage.

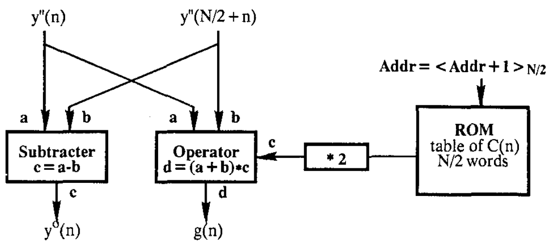

Figure 3. Hardware module for realising $y^o(n)$ and g(n) from y"(n).

Stage 2 includes the correlation hardware and the permutation network for realising $<5^\eta>_{4N} = 4k+1$. The correlation hardware is realised by the distributed arithmetic as mentioned above. Fig. 4 shows a possible approach to realise the permutation network. Typically, permuted data can be obtained by using the technique of table look-up or a switch network. The use of switch network increases the hardware complexity and hence the hardware cost while the use of table look-up involves address generation. However, we can use ROM to store up the address generation table such that permuted data can be retrieved efficiently within two memory accesses. In a practical case, such as image compression, we only deal with short lengths, for instance,

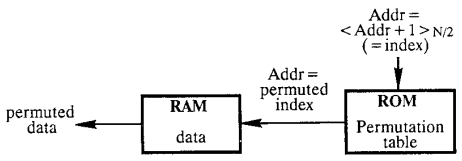

Figure 4. Permutation Network by using Table Look-Up.

a length-16 DCT. Hence, this approach is effective and efficient as only a table of small size is required. Instead of dynamic approaches mentioned above, we can use static approach by wiring up inputs to appropriate outputs of the permutation network. This is not flexible but can speed up the permutation. It may not be impractical as only a particular short length DCT is required for some special usage. The permutation network A in stage 1 has also the same type of constraints.

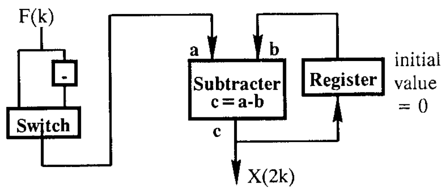

Figure 5. Addition Matrix in Stage 3.

Stage 3 can be implemented as shown in Fig.5. Input data should be negated alternatively before going into the subtracter. This can be done by a dedicated hardware buffer which negated input data alternatively before sending them out or by using a multiplexer which acts as an switch as shown in Fig.5. The output of the subtracter is fed back to the input of the subtracter for the following subtraction such that final results can come out recursively.

According to the derivation from eqn.(24) to (26), the time required for computing the two correlations is N(M-1) $T_A/2$, where T_A is the time required for an addition, if they are computed in parallel as shown in Fig.1. The time required for stage 1 and stage 3 are roughly (N/2) ($T_A + T_M$) and (N/2-1) T_A respectively, where T_M is the time required for an multiplication. Therefore, stage two will dominate the timing of the pipeline process if a fast multiplier exists. This gives the lower bound of the total time required for computing all DCT coefficients. Actually, we can make some modifications to increase the speed of the correlation further. The simplest one is to partition the input words into the most significant half and least significant half and so on. Then we can introduce parallelism in the computation by increasing the number of adders as we can deal with additions in parallel. Theoretically speaking, we can speed up the whole process to the upper bound that requires the least computation time, say (N/2-1) T_A, by introducing a sufficient number of adders and multipliers when the hardware cost is not a problem. This upper bound is limited by the recursive nature of stage 3.

In short, the proposed algorithm can be realised efficiently and easily by dedicated hardware or gate array technology. The structure of the hardware required is so simple that it involves a small memory size, a few adders, registers and one multiplier only. This can achieve a high performance DCT processor at a minimum cost and development time.

Conclusion

In this paper, a new algorithm is presented such that an N- length DCT can be directly computed by correlation. This algorithm involves no DFT computation and can be realised through very simple hardware structures and is very suitable for VLSI implementation.

Reference

[1] N.Ahmed, T.Natarajan and K.R.Rao, "Discrete cosine transform," IEEE trans., Vol.C-23, pp.90-94, Jan. 1974.

[2] M.R.Haralick, "A storage efficient way to implement the discrete cosine transform," IEEE Trans., Vol.C-25, pp.764-765, July 1976.

[3] B.D.Tseng and W.C.Miller, "On computing discrete cosine transform," IEEE Trans., Vol. C-27, pp.966-968, Oct. 1978.

[4] J.Makoul, "A fast cosine transform in one and two dimensions," IEEE Trans., Vol.ASSP-28, pp.27-34, Feb. 1980.

[5] M.J.Narasimha and A.M.Peterson, "On the computation of the discrete cosine transform," IEEE Trans., Vol.COM-26, pp.934-946, June 1978.

[6] B.G.Lee, "A new algorithm to compute the discrete cosine transform," IEEE Trans., Vol.ASSP-32, pp.1243-1245, Dec. 1984.

[7] W.H.Chen, C.H.Smith and S.C.Fralick, "A fast computational algorithm for the discrete cosine transform," IEEE Trans., Vol. COM-25, pp.1004-1009, Sept. 1977.

[8] H.S.Hou, "A fast recursive algorithm for computing the discrete cosine transform," IEEE Trans., Vol.ASSP-35, pp.1455-1461, Oct. 1987.

[9] M.L.Haque, " A two dimensional fast cosine transform," IEEE Trans., Vol.ASSP-33, pp.1532-1539, Dec. 1985.

[10] H.Kitajima, "A symmetric cosine transform," IEEE Trans., Vol.C-29, pp.317-323, Apr. 1980.

[11] Z.Wang, "Fast algorithm for the discrete W transform and for the discrete Fourier transform," IEEE Trans., Vol.ASSP-32, pp.803- 816, Aug. 1984.

[12] N.Suehiro and M.Hatori, "Fast algorithms for the DFT and other sinusoidal transforms," IEEE Trans., Vol.ASSP-34, pp.642-644, June 1986.

[13] Z.Wang, "On computing the discrete Fourier and cosine transforms," IEEE Trans., Vol.ASSP-33, pp.1341-1344, Oct. 1985.

[14] P.Duhamel and H. H'mida, " New 2^n DCT algorithms suitable for VLSI implementation," ICASSP-85, Tampa, March 1985, pp.780-783.

[15] M. Vetterli and H. Nussbaumer, " A simple FFT and DCT algorithms with reduced number of operation," Signal Processing, Vol. 6, No. 4, August 1984, pp.267-278.

[16] H.J.Nussbaumer, "Fast Fourier Transform and Convolution Algorithms," 2nd corrected and Updated Edition, Springer-Verlag, 1982. pp.32-78.

[17] S.A. White, " Applications of distributed arithmetic to digital signal processing: A tutorial review," IEEE ASSP magazine, Vol. 6, No. 3, July 1989, pp.4-19.