IOSR Journal of Electronics and Communication Engineering (IOSRJECE)
ISSN : 2278-2834 Volume 3, Issue 1 (Sep-Oct 2012), PP 39-50
www.iosrjournals.org

Optimization of ECAT through DA-DCT

S. Indumathi[1] and Dr. M. Sailaja[2]
[1]*M.Tech Student, Department of Electronics and Communication Engineering, University College of Engineering Kakinada, JNTU Kakinada, Andhra Pradesh, India*
[2]*Professor, ECE Dept., JNTUK, Kakinada, A.P, India*

ABSTRACT: *Discrete cosine transform (DCT) is a widely used tool in image and video compression applications. Recently, the high-throughput DCT designs have been adopted to fit the requirements of real-time application. Operating the shifting and addition in parallel, an error-compensated adder-tree (ECAT) is proposed to deal with the truncation errors and to achieve low-error and high-speed discrete cosine transform (DCT) design. Instead of the 12 bits used in previous works, 9-bit Distributed Arithmetic was proposed. DA-based DCT design with an error-compensated adder-tree (ECAT) is the proposed architecture in which, ECAT operates shifting and addition in parallel by unrolling all the words required to be computed. Furthermore, the Error-Compensated Circuit alleviates the truncation error for high accuracy design. Based on low-error ECAT, the DA-precision in this work is chosen to be 9 bits instead of the traditional 12 bits. Therefore, the hardware size and cost is reduced, and the speed is improved using the proposed ECAT.*

Keywords- *Adders, DCT- Discrete Cosine Transform, DA- Distributed Arithmetic, ECAT- Error-Compensated Adder-Tree.*

I. INTRODUCTION

Today we are talking about digital networks, digital representation of images, movies, video, tv, voice, digital library-all because digital representation of the signal is more robust than the analog counterpart for processing, manipulation, storage, recovery, and transmission over long distances, even across the globe through communication networks. In recent years, there have been significant advancements in processing of still image, video, graphics, speech, and audio signals through digital computers in order to accomplish different application challenges. As a result, multimedia information comprising image, video, audio, speech, text, and other data types has the potential to become just another data type. Development of efficient image compression techniques continues to be an important challenge to us, both in academia and in industry.

In [2] multiplier based DCT's were implemented, later to reduce area ROM-based DA was applied for designing DCT [3]. Then knowing the advantage of ROM-based, DA-based multipliers using ROMs were implemented to produce partial products together with adders that accumulated these partial products. By applying DA-based ROM to DCT core design we can reduce the area required. In addition, the symmetrical properties of the DCT transform and parallel DA architecture can be used in reducing the ROM size in [3], respectively. Recently, ROM-free DA architectures were presented [7]–[11]. Shams *et al*. employed a bit-level sharing scheme to construct the adder-based butterfly matrix called new DA (NEDA) [7]. Being compressed, the butterfly-adder-matrix in [7] utilized 35 adders and 8 shift-addition elements to replace the ROM.Based on NEDA architecture, the recursive form and ALU were applied in DCT design to reduce area cost [8], [9], but speed limitations exist in the operations of serial shifting and addition after the DA-computation. In DA-based computation partial products words are shifted and added in parallel [10] and [11]. However, a large truncation error occurred.

We need to reduce truncation error that error is introduced if the least significant part is directly truncated. In order to reduce truncation error effect several error compensation bias methods have been presented based on statistical analysis of relationship between partial product and multiplier-multiplicand. Hardware complexity will be reduced if truncation error minimized. In general, the truncation part (TP) is usually truncated to reduce hardware costs in parallel shifting and addition operations, known as the direct truncation (Direct-T) method. Thus, a large truncation error occurs due to the neglecting of carry propagation from the TP to Main Part (MP). Distributed arithmetic is a bit level rearrangement of a multiply accumulate to hide the multiplications. It is a powerful technique for reducing the size of a parallel hardware multiply-accumulate that is well suited to FPGA designs. The Discrete cosine transform (DCT) is widely used in digital image processing for image compression, especially in image transform coding. However, though most of them are good software solutions to the realization of DCT, only a few of them are really suitable for VLSI implementation. Cyclic convolution plays an important role in digital signal processing due to its nature of easy implementation. Specifically, there exist a number of well-developed convolution algorithms and it can be easily realized through modular and structural hardware such as distributed arithmetic and systolic array. The

way of data movement forms a significant part in the determination of the efficiency of the realization of a transform using the DA.

1.1. WHY COMPRESSION

Despite the many advantages of digital representation of signals compared to the analog counterpart, they need a very large number of bits for storage and transmission. For example, a high-quality audio signal requires approximately 1.5 megabits per second for digital representation and storage. A television-quality low-resolution color video of 30 frames per second with each frame containing 640 x 480 pixels (24 bits per color pixel) needs more than 210 megabits per second of storage. As a result, a digitized one-hour color movie would require approximately 95 gigabytes of storage. The storage requirement for upcoming high-definition television (HDTV) of resolution 1280 x 720 at 60 frames per second is far greater. A digitized one-hour color movie of HDTV-quality video will require approximately 560 gigabytes of storage. A digitized 14 x 17 square inch radiograph scanned at 70 pm occupies nearly 45 megabytes of storage. Transmission of these digital signals through limited bandwidth communication channels is even a greater challenge and sometimes impossible in its raw form. Although the cost of storage has decreased drastically over the past decade due to significant advancement in microelectronics and storage technology, the requirement of data storage and data processing applications is growing explosively to outpace this achievement.

1.2. ADVANTAGES OF DATA COMPRESSION

The main advantage of compression is that it reduces the data storage requirements. It also offers an attractive approach to reduce the communication cost in transmitting high volumes of data over long-haul links via higher effective utilization of the available bandwidth in the data links. This significantly aids in reducing the cost of communication due to the data rate reduction. Because of the data rate reduction, data compression also increases the quality of multimedia presentation through limited-bandwidth communication channels. Hence the audience can experience rich-quality signals for audio-visual data representation. For example, because of the sophisticated compression technologies we can receive toll-quality audio at the other side of the globe through the good old telecommunications channels at a much better price compared to a decade ago. Because of the significant progress in image compression techniques, a single 6 MHz broadcast television channel can carry HDTV signals to provide better quality audio and video at much higher rates and enhanced resolution without additional bandwidth requirements. The rate of input-output operations in a computing device can be greatly increased due to shorter representation of data. In systems with levels of storage hierarchy, data compression in principle makes it possible to store data at a higher and faster storage level (usually with smaller capacity), thereby reducing the load on the input-output channels. Data compression obviously reduces the cost of backup and recovery of data in computer systems by storing the backup of large database files in compressed form.

The advantages of data compression will enable more multimedia applications with reduced cost and hence aid its usage by a larger population with newer applications in the near future.

1.3. DISADVANTAGES OF DATA COMPRESSION

Although data compression offers numerous advantages and it is the most sought-after technology in most of the data application areas, it has some disadvantages too, depending on the application area and sensitivity of the data. For example, the extra overhead incurred by encoding and decoding processes is one of the most serious drawbacks of data compression, which discourages its usage in some areas (e.g., in many large database applications). This extra overhead is usually required in order to uniquely identify or interpret the compressed data. For example, the encoding/decoding tree in a Huffman coding type compression scheme is stored in the output file in addition to the encoded bit-stream. These overheads run opposite to the essence of data compression, that of reducing storage requirements. In large statistical or scientific databases where changes in the database are not very frequent, the decoding process has greater impact on the performance of the system than the encoding process. Even if we want to access and manipulate a single record in a large database, it may be necessary to decompress the whole database before we can access the desired record. After access and probably modification of the data, the database is again compressed to store. The delay incurred due to these compression and decompression processes could be prohibitive for many real-time interactive database access requirements unless extra care and complexity are added in the data arrangement in the database.

II. METHODOLOGIES

In this section elaborates the description of the algorithms.

2.1. DISCRETE COSINE TRANSFORM (DCT)

The forward and inverse 2-D DCT can be written as:

$$Z(u,v) = \frac{2}{N} C(u)C(v) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x(i,j) cos \frac{(2i+1)u\pi}{2N} cos \frac{(2j+1)v\pi}{2N}$$

.....(1)

$$x(i,j) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)Z(u,v) cos \frac{(2i+1)u\pi}{2N} cos \frac{(2j+1)v\pi}{2N}$$

.....(2)

where x(i,j) is the image pixel data, and Z(u,v) is the transport data. The 2-D DCT is an orthogonal and separable transform. It can be expressed in matrix notation as two 1-D DCT's as follows: Therefore, we can decompose (1) into two 1-D DCT's:

$$Z(u,v) = \sqrt{\frac{2}{N}} \sum_{i=0}^{N-1} C(u)Y(v,i) cos \frac{(2i+1)u\pi}{2N}$$

.....(3)

$$Y(v,i) = \sqrt{\frac{2}{N}} \sum_{j=0}^{N-1} C(v)x(i,j) cos \frac{(2j+1)v\pi}{2N}$$

.....(4)

And (2) can be decomposed into two 1-D IDCT's in similar way as above, since it is inverse 2D-DCT we need not have to compute.

2.1.1.1. COMPUTATION OF THE DCT

The 8 x 8 DCT coefficient matrix can be written as

$$C(DCT)_{8\times8} = \begin{bmatrix} a & a & a & a & a & a & a & a \\ b & d & e & g & -g & -e & -d & -b \\ c & f & -f & -c & -c & -f & f & c \\ d & -g & -b & -e & e & b & g & -d \\ a & -a & -a & a & a & -a & -a & a \\ e & -b & g & d & -d & -g & b & -e \\ f & -c & c & -f & -f & c & -c & f \\ g & -e & d & -b & b & -d & e & -g \end{bmatrix}$$

.....(5)

where $c_i = \cos\left(\frac{i\pi}{2N}\right)$ and N=8 therefore for the 8 coefficients are give below: $a = \cos\frac{\pi}{16}$,

$b = \cos\frac{2\pi}{16}$, $c = \cos\frac{3\pi}{16}$, $d = \cos\frac{4\pi}{16}$, $e = \cos\frac{5\pi}{16}$, $f = \cos\frac{6\pi}{16}$, $g = \cos\frac{7\pi}{16}$;

Even rows of C are even-symmetric and odd rows are odd-symmetric. Therefore by exploiting this symmetry in the rows of C and separating even and odd rows we can get 1D-DCT as follows,

$$Z_{even} = \begin{bmatrix} y_0 \\ y_2 \\ y_4 \\ y_6 \end{bmatrix} = \begin{bmatrix} a & a & a & a \\ c & f & -f & -c \\ a & -a & -a & a \\ f & -c & c & -f \end{bmatrix} \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix}$$

.....(6a)

$$Z_{odd} = \begin{bmatrix} y_1 \\ y_3 \\ y_5 \\ y_7 \end{bmatrix} = \begin{bmatrix} b & d & e & g \\ b & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix}$$

.....(6b)

1D-DCT is written as follows

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} a & c & a & f \\ a & f & -a & -c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} + \begin{bmatrix} b & d & e & g \\ b & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix}$$

.....(7)

$$\begin{bmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \end{bmatrix} = \begin{bmatrix} a & c & a & f \\ a & f & -a & -c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} + \begin{bmatrix} b & d & e & g \\ b & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix}$$

.....(8)

$$\text{Where } \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{bmatrix} = \sqrt{\frac{2}{N}} \begin{bmatrix} \cos\frac{\pi}{16} \\ \cos\frac{2\pi}{16} \\ \cos\frac{3\pi}{16} \\ \cos\frac{4\pi}{16} \\ \cos\frac{5\pi}{16} \\ \cos\frac{6\pi}{16} \\ \cos\frac{7\pi}{16} \end{bmatrix}$$

2.2. ROM BASED DISTRIBUTED ARITHMETIC BASED VLSI ARCHITECTURE FOR DCT

For a 2-D data X(i, j), $0 \le i \le 7$ and $0 \le j \le 7$, 8x8 2-D DCT is given by

$$Z(u,v) = \frac{2}{8} C(u)C(v) \sum_{i=0}^{7} \sum_{j=0}^{7} x(i,j) cos \frac{(2i+1)u\pi}{16} cos \frac{(2j+1)v\pi}{16}$$

.....(9)

where $0 \le u \le 7$ and $0 \le v \le 7$ and c(u),c(v) = 1/ 2 for u,v=0, C(u),C(v) =1.414 otherwise. Implementation computation is reduced by decomposing (1) in two 8x1 1-D DCT given by,

$$Z(u) = \frac{1}{2} C(u) \sum_{i=0}^{N-1} x(i) cos \frac{(2i+1)u\pi}{16}$$

.....(10)

For 2-D DCT computation of a 8x8 2-D data, first row-wise 8x1 1-D DCT is taken for all rows followed by column-wise 8x1 1-D DCT to all columns.

Distributed arithmetic is used to compute the 8 equations above where cosine terms are expressed in DA form. Implementation is realized by using shift and adds operations. Shifting operation is performed by wiring. Each value of Z is computed in parallel and hence faster speed is achieved [11]. Shifted data are represented by less number of bits and hence adder bit-width is reduced resulting in less hardware cost. For DCT computation image data is represented in signed 2's complement form range -128 to 127. Bit width of shifted data is determined by number of times shift operation is done. So different bit-width intermediate data are present which are to be added. For 2-input adder both input data width has to be equal and hence sign extension is done in smaller bit-width data. Shifting and addition with sign extension creates error. For example if initial value is -2, in 8-bit 2's complement representation this can be written 11111110. If we take 4 shifted sample of this value a=111111 (shifting 2 times), b=1111 (shifting 4 times), c=111 (shifting 5 times), d=11 (shifting 6 times) all are -1. But all these data should be zero. If we add these values in cascade, result we will have -4 (which should be zero).

2.3. NEDA: A LOW-POWER HIGH-PERFORMANCE DCT ARCHITECTURE

Distributed Arithmetic (DA) was invented about two decades ago and has since enjoyed widespread application in very large scale integration (VLSI) implementations of digital signal processing (DSP) algorithms. A significant percentage of commercial DSP chips employ the DA approach Most of these applications, for example discrete cosine transform (DCT) calculation, are arithmetic intensive with multiply/accumulate (MAC) being the predominant operation . The main advantage of DA approach is that it alters the basic assumption of using multipliers and adders for computing the DCT. The conventional DA approach speeds up the multiply process by pre-computing all the possible values and storing these values in a ROM. The input data can then be used to directly address the memory and the result . Unfortunately, the size of ROM grows exponentially when the number of inputs and internal precision increase. This is inherent to the DA mechanism where a great amount of redundancy is introduced in the ROM to accommodate all possible combinations of bit patterns exhibited by the input signals. In practice, therefore, DA often appears in the form of multiple ROMs of much smaller size coupled with conventional MAC structures.

A low-power algorithm of DCT, therefore, should favor additions to multiplication. As reducing cost metrics (area, power) is attracting more and more attention in ASIC design, there is an increasing demand for more efficient DA paradigms which can eliminate the need of using ROMs, and at the same time, are capable of meeting throughput constraints. Therefore NEDA features implementation without the need of multipliers as in conventional MAC approach, and at the same time, without the need of ROM as in DA approach[1]. Implementing NEDA with an example:

Consider the following sum of products

$$Y = \sum_{k=1}^{L} A_k \times X_k$$

.....(11)

If NEDA is used to compute, let A_k be coefficient matrix and values are 2,-1,and X_k is the input . Substituting in eqn (11)

$$Y = [2 \quad -1]\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = [010 \quad 111]\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

the DA matrix [A] = $\begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$ and thus

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} X_2 \\ X_1 + X_2 \\ X_2 \end{bmatrix}$$

.....(12)

As can be observed from this example in eqn (12), NEDA eliminates totally the encoder logic required in Booth Multiplier for two's complement manipulation. Furthermore, only one type of operations-addition, take place during the intermediate stages of computation, greatly simplifying hardware design. What needs special care is the sign output from the adder array, which is simply taking two's complement. In the above example, invert and add 1: is all one needs to convert then add 1 to "000 110"=> "111010 = YP(2)". In addition, NEDA can save hardware and the critical path is shorter in NEDA than in MAC for the same computation.

2.4. DCT IMPLEMENTATION WITH DISTRIBUTED ARITHMETIC(DA)

Distributed arithmetic is an efficient method for computing an inner product

$$Y = A^T X = \sum_{i=0}^{N-1} A_i X_i$$

.....(13)

where A =[A_0,A_1,A_2,…….A_{N-1}] is a fixed coefficient vector and X =[X_0,X_1,X_2, ……X_{N-1}] is an input vector . Suppose that X_i is represented in B-bit 2's complement form as follows:

$$X_i = -X_{i0} + \sum_{j=1}^{B-1} X_{ij} 2^{-j} \qquad 0 \le i \le N-1$$

.....(14)

Since there are 2N possible values for

$$\sum_{i=0}^{N-1} A_i X_{ij}$$

.....(15)

for each j = 0,1,2, ….B-1, these values can be pre-computed and stored in a ROM. Then, (15) can be implemented with a ROM of size 2N and an accumulator.

III. ARCHITECTURE OF PROPOSED DISTRIBUTED ARITHMETIC DCT

3.1. DISTRIBUTED ARITHMETIC

The inner product is an important tool in digital signal processing applications. It can be written as follows:

$$Y = A^T X = \sum_{i=1}^{L} A_i X_i$$

.....(16)

Where A_i – *i* th fixed coefficient

X_i – *i* th input data

L – number of inputs, respectively.

Assume that coefficient A_i Q-bit 2's complement binary fraction number. Equation (1) can be expressed as follows: from eqn (14)

$$Y = \begin{bmatrix} -2^0 & 2^{-1} & \cdots & 2^{-(Q-1)} \end{bmatrix}\begin{bmatrix} A_{1,0} & A_{2,0} & \cdots & A_{L,0} \\ A_{1,1} & A_{2,1} & \cdots & A_{L,0} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1,(Q-1)} & A_{2,(Q-1)} & \cdots & A_{L,(Q-1)} \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_L \end{bmatrix}$$

.....(17)

$$Y = \begin{bmatrix} 2^0 & 2^{-1} & \cdots & 2^{-(Q-1)} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{Q-1} \end{bmatrix} \quad \text{.....(18)}$$

Where $y_j = \sum_{i=1}^{L} A_{i,j} X_i$, and $A_{i,j} \in \{0,1\}$ for $1 \le j \le (Q-1)$ for j = 0 and $A_{i,j} \in \{-1,0\}$ for j = 0.

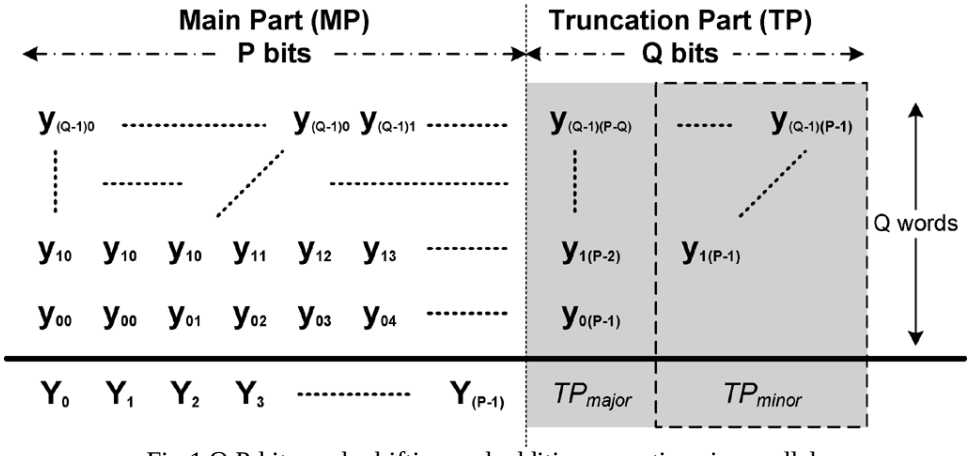

Fig.1 Q P-bit words shifting and addition operations in parallel.

3.2. Error Compensated Adder Tree

The shifting and addition computation can be written as follows:

$$Y = \sum_{j=0}^{Q-1} y_j 2^{-j} \quad \text{.....(19)}$$

In general, the shifting and addition computation uses a shift-and-add operator in VLSI implementation in order to reduce hardware cost. However, when the number of the shifting and addition words increases, the computation time will also increase. Therefore, the shift-adder-tree (SAT) presented in operates shifting and addition in parallel by unrolling all the words needed to be computed for high-speed applications. However, a large truncation error occurs in SAT, and an ECAT architecture is proposed compensate for the truncation error in high-speed applications. The main part (MP) that includes P most significant bits (MSBs) and the truncation part (TP) that has Q least significant bits (LSBs).

In this proposed design (P,Q)=(12,6) is taken .Then, the shifting and addition output can be expressed as follows:

$$Y = MP + TP \cdot 2^{-(P-2)} \quad \text{.....(20)}$$

The output Y will obtain the P-bit MSBs using a rounding operation called post truncation (Post-T), which is used for high-accuracy applications. However, hardware cost increases in the VLSI design. In general, the TP is usually truncated to reduce hardware costs in parallel shifting and addition operations, known as the direct truncation (Direct-T) method. Thus, a large truncation error occurs due to the neglecting of carry propagation from the TP to MP. In order to alleviate the truncation error effect, several error compensation bias methods have been presented [12]–[14]. All previous works were only applied in the design of a fixed-width multiplier. Because the products in a multiplier have a relationship between the input multiplier and multiplicand, the compensation methods usually use the correlation of inputs to calculate a fixed [12] or an adaptive [13], [14] compensation bias using simulation or statistical analysis. Note that the addition elements y_{qp} in the TP in Fig. 1 (where $1 \le Q \le (Q-1)$ and $P-Q-1 \le P \le (P-1)$) are independent from each other. Therefore, the previous compensation method cannot be applied in this work, and the proposed ECAT is explained as follows.

3.3. Proposed Error-Compensated Scheme

From Figure.1, eqn (4) can be approximated as

$$Y \approx MP + \sigma \cdot 2^{-(P-2)} \quad \text{.....(21)}$$

where σ is the compensated bias from the TP to the MP

$$\sigma = \text{Round}(TP_{major} + TP_{minor}) \quad \text{.....(22)}$$

$$TP_{major} = \frac{1}{2} \sum_{j=0}^{Q-1} y_{j(P-1-j)} \quad \text{.....(23)}$$

$$TP_{minor} = \frac{1}{4}\left(y_{1(P-1)} + \cdots + y_{(Q-1)(P-Q+1)}\right) + \frac{1}{8}\left(y_{2(P-1)} + \cdots + y_{(Q-1)(P-Q+2)}\right) + \cdots + \left(\frac{1}{2}\right)^{Q} y_{(Q-1)(P-1)} \quad \text{.....(24)}$$

where Round( ) is rounded to the nearest integer. The TP_{major} has more weight than TP_{minor} when contributing towards the σ. Therefore, the compensated bias σ can be calculated by obtaining TP_{major} and estimating TP_{minor}. Let the probability of Yqp=1 be 0.5, where 1<q<(Q-1) and (p-1q-1)<p<(p-1).

For a given TP_{minor},($y_{j(P-1-j)}$,0<j<(q-1)), the σ can be obtained after rounding the sum of (TP_{major} + TP_{minor}). In order to round the summation ,TP_{minor} can be divided into four parts: As K >= 1,the

TP_{minor} approximates

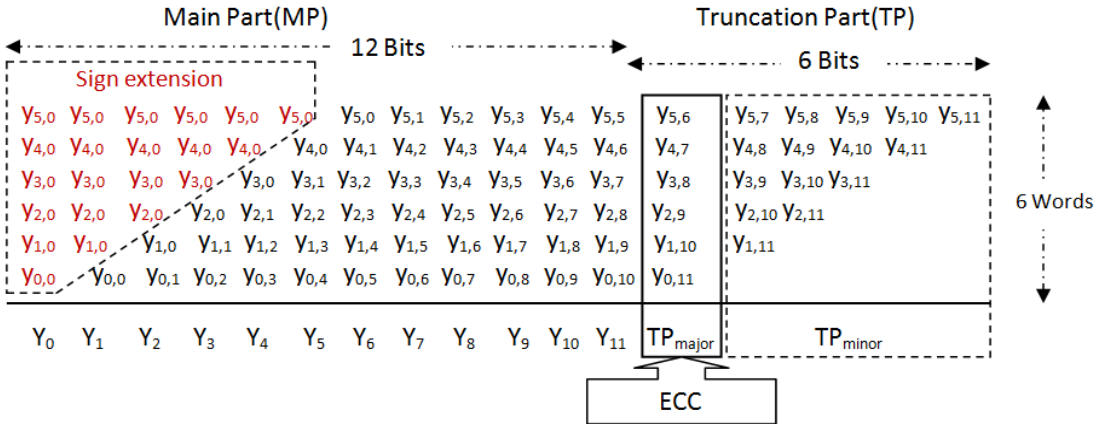

Fig.2 (P,Q)=(12,6) Shifting and Addition operations in parallel.

$$TP_{minor} \approx \begin{cases} (k-1)+\frac{1}{2}, & for\ Q = 4k \\ (k-1)+\frac{3}{4}, & for\ Q = 4k+1 \\ k, & for\ Q = 4k+2 \\ k+\frac{1}{4}, & for\ Q = 4k+3 \end{cases} \quad \text{.....(25)}$$

As the Q value is taken as 6, case: 3 is suitable for the design [9]. Hence , σ can be rewritten as

$$Q = 4k+2,4k+3 (k>1)$$

$$\sigma = k + \text{Round}(TP_{major}) \quad \text{.....(26)}$$

When coming to Performance simulation of an ECC, by measuring and comparing the absolute average error ϵ_{avg} , the maximum error ϵ_{max}, and the mean square error ϵ_{mse} for the proposed Error-Compensated Circuit with Direct-T in Table I.

$$\epsilon_{avg} = Avg\{|TP-\sigma|\} \quad \text{.....(27)}$$

$$\epsilon_{max} = Max\{|TP-\sigma|\} \quad \text{.....(28)}$$

$$\epsilon_{mse} = Avg\{(TP-\sigma)^2\} \quad \text{.....(29)}$$

The internal word-length usually uses 12 bits in a DCT design. Consequently, as said above word length P=12 is chosen with different values of Q=3, 6,9,12. Direct-T method has the largest inaccuracies of the errors shown in Table 1 for low-cost hardware design. The proposed ECAT is more accurate than Direct-T and is close to the performance of the ECAT method of [1] using an Error compensated circuit. Because the truncation part TP_{minor} is estimated using statistical analysis, the magnitude of errors also increases as the number of shift-and-add words Q increases. The eons (27),(28),(29) are calculated for (P,Q)=(12,6) ,see in Fig.2;The values in Table except proposed are taken from [1] for reference.

(P,Q)	ERROR			
	ϵ_{avg}		ϵ_{mse}	
	Direct-T	ECAT	Direct-T	ECAT
(12,3)	1.0625	0.2656	1.3516	0.1016
(12,6)	2.5078	0.3789	6.7614	0.2184
(12,9)	4.0010	0.3804	16730	0.2222
(12,12)	5.5001	0.4738	31.224	0.3472
Proposed(12,6)	0.7011	0.2988	0.5204	0.1180

TABLE 1 : Comparison of ϵ_{avg} and ϵ_{mse}

3.4. PROPOSED ECAT ARCHITECTURE

The proposed ECAT architecture is illustrated in Fig. 3 for (P, Q) = (12, 6) (*case 3*), where block FA indicates a full-adder cell with three inputs (a, b, and c) and two outputs, a sum (s) and a carry-out (co). Also, block HA indicates half-adder cell with two inputs (a and b) and two outputs, a sum (s) and a carry-out (co).

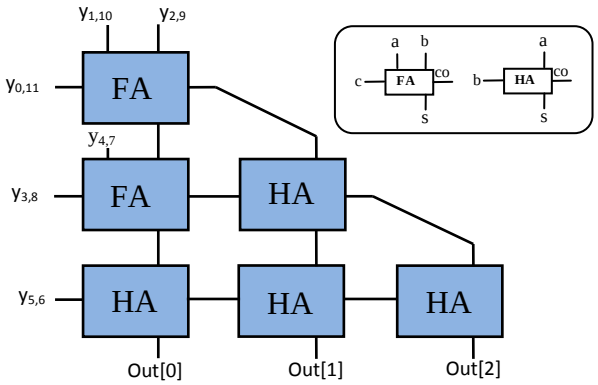

Fig.3a Proposed ECC architecture

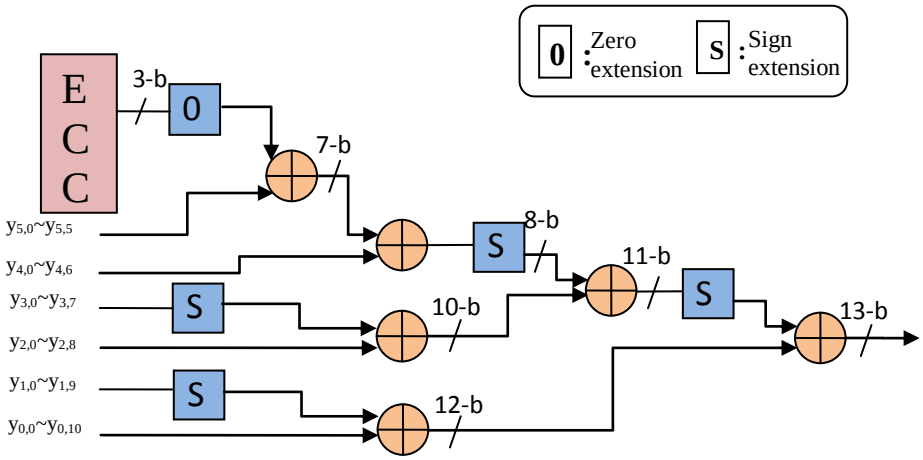

Figure.3b Proposed ECAT architecture of shifting and addition operators for the (P, Q) = (12, 6) example

3.5. PROPOSED 8×8 2-D DCT CORE DESIGN

The 1-D DCT employs the DA-based architecture and the proposed ECAT to achieve a high-speed, small area, and low-error design. The 1-D 8-point DCT eqn 3 can be expressed as follows:

$$Zn = \frac{1}{2} k_n \sum_{i=0}^{7} x_i cos \frac{(2i+1)n\pi}{16} \quad \text{.....(30)}$$

Where x_i denotes the input data; Zn denotes the transform output. By neglecting the scaling factor 1/2, the1-D 8-point DCT in above equation can be divided into even and odd parts: Ze and Zo as listed in below equations from eqn (6), respectively

$$Z_e = \begin{bmatrix} Z_0 \\ Z_2 \\ Z_4 \\ Z_6 \end{bmatrix} = \begin{bmatrix} d & d & d & d \\ b & f & -f & -b \\ d & -d & -d & d \\ -f & -b & b & -f \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = C_e \cdot a \quad \text{.....(31)}$$

$$Z_o = \begin{bmatrix} Z_1 \\ Z_3 \\ Z_5 \\ Z_7 \end{bmatrix} = \begin{bmatrix} a & c & e & g \\ c & -g & -a & -e \\ e & -a & g & c \\ g & -e & c & -a \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = C_o \cdot b \quad \text{.....(32)}$$

Where $c_i = cos \frac{i\pi}{16}$, $a_0 = x_0 + x_7$, $a_1 = x_1 + x_6$, $a_2 = x_2 + x_5$, $a_3 = x_3 + x_4$,

$b_0 = x_0 - x_7$, $b_1 = x_1 - x_6$, $b_2 = x_2 - x_5$, $b_3 = x_3 - x_4$;

$$Z_{ee} = \begin{bmatrix} Z_0 \\ Z_4 \end{bmatrix} = \begin{bmatrix} d & d \\ d & -d \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \end{bmatrix} = C_{ee} \cdot A \quad \text{.....(33)}$$

$$Z_{eo} = \begin{bmatrix} Z_2 \\ Z_6 \end{bmatrix} = \begin{bmatrix} b & f \\ f & -b \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \end{bmatrix} = C_{eo} \cdot B \quad \text{.....(34)}$$

where $A_0 = (x_0+x_7)+(x_3+x_4) = a_0+a_3$, $A_1 = (x_1+x_6)+(x_2+x_5) = a_1+a_2$,
$B_0 = (x_0+x_7)-(x_3+x_4) = a_0-a_3$, $B_1 = (x_1+x_6)-(x_2+x_5) = a_1-a_2$,

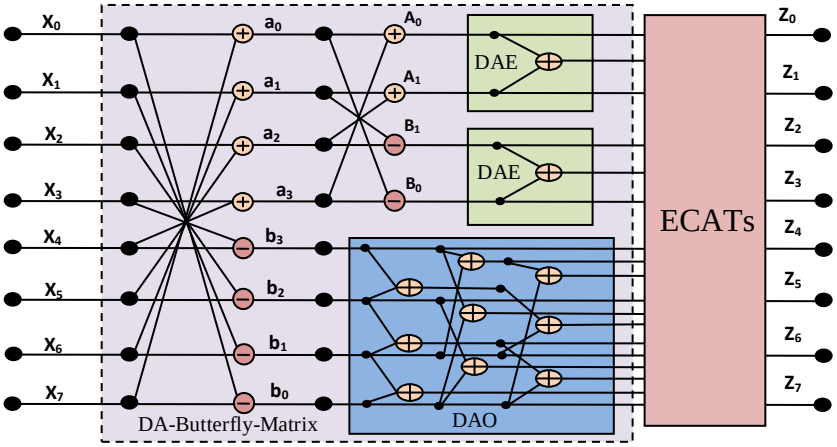

Fig.4 Architecture of the proposed 1-D 8-point DCT

Moreover, the even part Z_e can be further decomposed into even and odd parts: Z_{ee} and Z_{eo}. For the DA-based computation, the coefficient matrix C_o, C_{ee} and C_{eo} , are expressed as 9-bit binary fraction numbers.
Let us see an example for case Z_4 : $Z_4 = d\{(x_0+x_7)+(x_3+x_4)\}-d\{(x_1+x_6)+(x_2+x_5)\}$;

$d = \cos(4\pi/16) = (0.707106)_{10} = (0.10110101)_2$
$-d = (1.01001011)_2$
$= -1\times2^0+1\times2^{-2}+1\times2^{-5}+1\times2^{-7}+1\times2^{-8}$
$= -1+0.25+0.03125+0.0078125+0.003906$
$= -0.707106_{10}$

$$Z_4 = [-2^0 \quad 2^{-1} \quad 2^{-2} \quad 2^{-3} \quad 2^{-4} \quad 2^{-5} \quad 2^{-6} \quad 2^{-7} \quad 2^{-8}] \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

$$Z_4 = [-2^0 \quad 2^{-1} \quad 2^{-2} \quad 2^{-3} \quad 2^{-4} \quad 2^{-5} \quad 2^{-6} \quad 2^{-7} \quad 2^{-8}] \begin{bmatrix} A_1 \\ A_0 \\ A_1 \\ A_0 \\ A_0 \\ A_1 \\ A_0 \\ A_1 \\ A_0 + A_1 \end{bmatrix}$$

Table II expresses Z_{even} and Z_{odd} in the bit level formulation respectively

Table 2 : 9-Bit DA-based coefficient matrix for Z_{even} and Z_{odd}

Weight	Z_{even} Value				Z_{odd} Value			
	Z_0	Z_2	Z_4	Z_6	Z_1	Z_3	Z_5	Z_7
-2^0	0	0	A_1	B_1	0	$B_1+B_2+B_3$	B_1	B_1+B_3
2^{-1}	A_0+A_1	B_0	A_0	0	$B_0+B_1+B_2$	B_0+B_1	B_0+B_3	B_2
2^{-2}	0	B_0+B_1	A_1	B_0	B_0+B_1	$B_0+B_1+B_3$	B_3	B_1+B_2

2^{-3}	A_0+A_1	B_0+B_1	A_0	B_0	B_0+B_3	B_3	B_2	B_0+B_1
2^{-4}	A_0+A_1	0	A_0	B_1	$B_0+B_1+B_3$	B_0+B_3	B_2+B_3	$B_0+B_1+B_2$
2^{-5}	0	B_0	A_1	0	B_0+B_2	B_1	B_0	0
2^{-6}	A_0+A_1	B_0	A_0	0	B_1+B_2	$B_0+B_1+B_2$	$B_0+B_1+B_3$	B_2+B_3
2^{-7}	0	0	A_1	B_1	B_0+B_2	B_1	B_0	0
2^{-8}	A_0+A_1	B_1	A_0+A_1	B_0+B_1	B_0+B_3	B_2+B_3	B_1+B_2	$B_0+B_1+B_3$

Input data A_0 and A_1, the transform output Z_{ee} needs only one adder to compute $(A_0 + A_1)$ and two separated ECATs to obtain the results of Z_0 and Z_4. Similarly, the other transform outputs Z_{eo} and Z_o can be implemented in DA-based forms using 10(=1 + 9) adders and corresponding ECATs. Consequently, the proposed 1-D 8-point DCT architecture can be constructed as illustrated in Fig. 4 using a DA-Butterfly-Matrix, that includes two DA even processing elements (DAEs), a DA odd processing element (DAO) and 12 adders/subtractors, and 8 ECATs (one ECAT for each transform output Z_n). The eight separated ECATs work simultaneously, enabling high-speed applications to be achieved. After the data output from the DA-Butterfly-Matrix is completed, the transform output Z will be completed during one clock cycle by the proposed ECATs. In contrast, the traditional shift-and-add architecture requires Q clock cycles to complete the transform output Z if the DA-precision is Q-bits.With high-speed considerations in mind, the proposed 2-D DCT is designed using two 1-D DCT cores and one transpose buffer. For accuracy, the DA-precision and transpose buffer word lengths are chosen to be 9 bits and 12 bits, respectively, meaning that the system can meet the PSNR requirements outlined in previous works. Moreover, the 2-D DCT core accepts 9-bit image input and 12-bit output precision.

IV. SIMULATION RESULTS

Table 3 Device Utilization Summary of **XC3S500E** with **FG320** Package

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	591	9,312	6%
Logic Distribution			
Number of occupied Slices	344	4,656	7%
Number of Slices containing only related logic	344	344	100%
Number of Slices containing unrelated logic	0	344	0%
Total Number of 4 input LUTs	641	9,312	6%
Number used as logic	591		
Number used as a route-thru	50		
Number of bonded IOBs	170	232	73%
IOB Flip Flops	96		
Number of BUFGMUXs	1	24	4%

The device utilization summery is shown above in which its gives the details of number of devices used from the available devices and also represented in %. Hence as the result of the synthesis process, the device utilization in the used device and package is shown above.

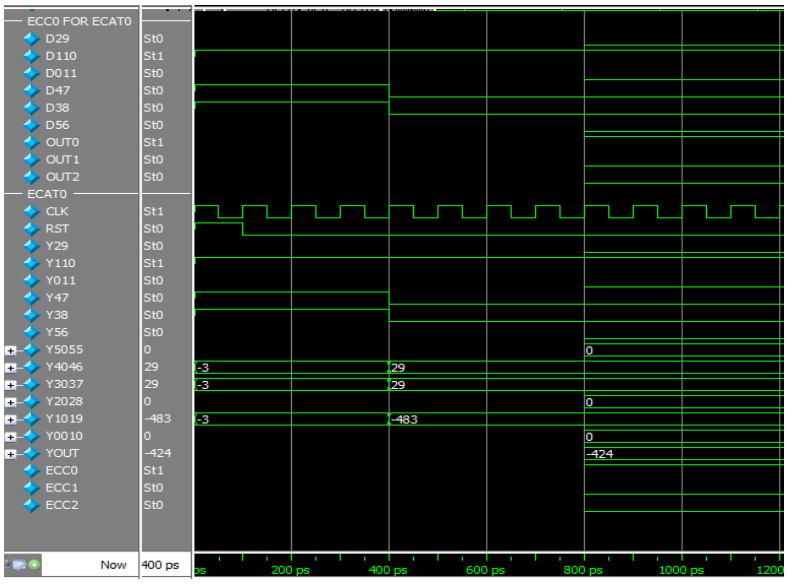

Figure.5 Simulation results of ECC & ECAT

V. CONCLUSIONS AND FUTURE SCOPE

This paper presents an efficient architecture for computing the 2-D DCT with distributed arithmetic. The proposed architecture requires less hardware than conventional architectures which use the original DCT algorithm or the even-odd frequency decomposition method. The modules of the transpose memory and parallel Distributed Arithmetic 2-D DCT architecture were designed and synthesized. The paper contributed with specific simplifications in the multiplier stage, by using shift and add method, which lead to hardware simplification and speed up over architecture.

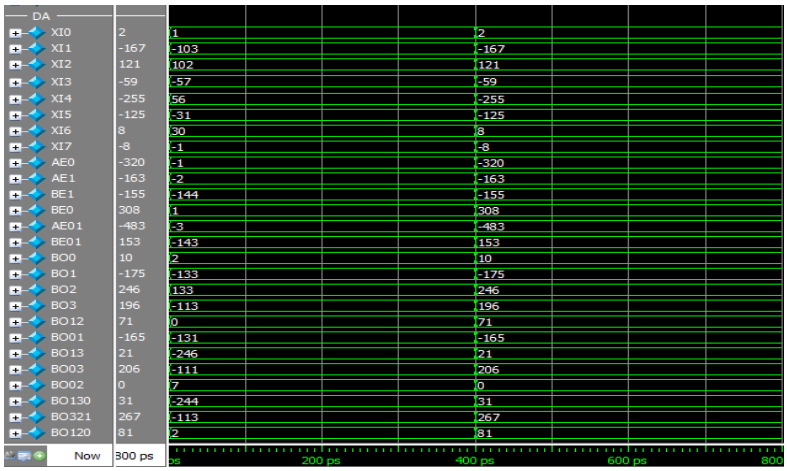

Fig.6 Simulation results of DA-BUTTERFLY MATRIX

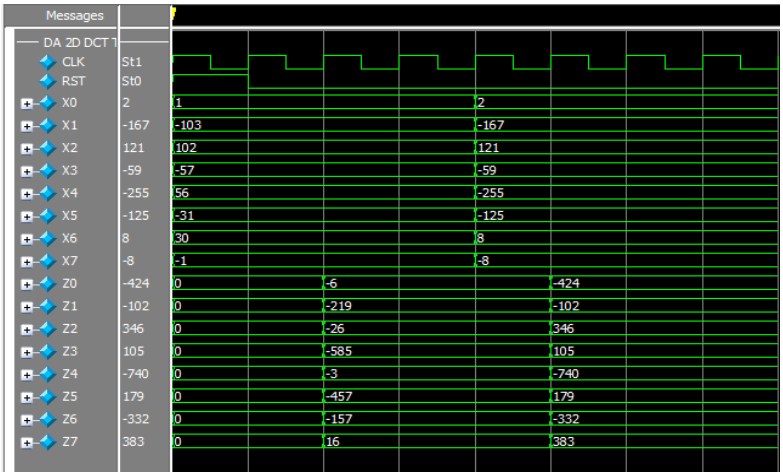

Fig.7 Simulation results of DA-2D DCT

As future work,

- This work can improve by implementing JPEG 2000 Image compression standard in which ordinary DCT transformation part can be replaced by our Distributed Arithmetic Discrete Cosine Transform (DA DCT) Design.
- The 8-Point Distributed Arithmetic Discrete Cosine Transform (DA DCT) Design can be made to 16, 32-point Distributed Arithmetic Discrete Cosine Transform (DA DCT) by making minor modifications to the code.
- This work can be extended in order to increase the accuracy by increasing the level of transformations.
- This can be used as a part of the block in the full fledged application, i.e., by using these DA DCT, the applications can be developed such as compression, watermarking, etc.

REFERENCES

[1] Yuan-Ho Chen; Tsin-Yuan Chang; Chung-Yi Li;, *"High Throughput DA-Based DCT With High Accuracy Error-Compensated Adder Tree," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol.19, no.4, pp.709-714, April 2011 doi: 10.1109 / TVLSI.2009.2037968.

[2] Y. Chang and C.Wang, "*New systolic array implementation of the 2-D discrete cosine transform and its inverse*," IEEE Trans. Circuits Syst.Video Technol., vol. 5, no. 2, pp. 150–157, Apr. 1995.

[3] S. Ghosh, S. Venigalla, and M. Bayoumi, *"Design and implementation of a 2D-DCT architecture using coefficient distributed arithmetic,"* in Proc. IEEE Compute. Soc. Ann. Symp. VLSI, 2005, pp. 162– 166.

[4] S. Uramoto, Y. Inoue, A. Takabatake, J. Takeda, Y. Yamashita, H.Yerane, and M. Yoshimoto, *"A 100-MHz 2-D discrete cosine transform core processor,"* IEEE J. Solid-State Circuits, vol. 27, no. 4, pp492–499, Apr. 1992.

[5] S. Yu and E. E. S. , Jr., *"DCT implementation with distributed arithmetic,"*IEEE Trans. Comput., vol. 50, no. 9, pp. 985–991, Sep. 2001.

[6] P. K. Meher, *"Unified systolic-like architecture for DCT and DST using distributed arithmetic,"* IEEE Trans. Circuits Syst. I, Reg. Papers, vol.53, no. 12, pp. 2656–2663, Dec. 2006.

[7] A. M. Shams, A. Chidanandan, W. Pan, and M. A. Bayoumi, *"NEDA:A low-power high-performance DCT architecture,"* IEEE Trans. SignalProcess., vol. 54, no. 3, pp. 955–964, Mar. 2006.

[8] M. R. M. Rizk and M. Ammar, *"Low power small area high performance 2D-DCT architecture,"* in Proc. Int. Design Test Workshop,2007, pp. 120–125.

[9] Y. Chen, X. Cao, Q. Xie, and C. Peng, *"An area efficient high performanceDCT distributed architecture for video compression,"* in Proc.Int. Conf. Adv. Comm. Technol., 2007, pp. 238–241.

[10] C. Peng, X. Cao, D. Yu, and X. Zhang, *"A 250 MHz optimized distributed architecture of 2D 8×8 DCT*," in Proc. Int. Conf. ASIC, 2007,pp. 189–192.

[11] C. Y. Huang, L. F. Chen, and Y. K. Lai, *"A high-speed 2-D transform architecture with unique kernel for multi-standard video applications,"* in Proc. IEEE Int. Symp. Circuits Syst., 2008, pp. 21–24.

[12] S. S. Kidambi, F. E. Guibaly, and A. Antonious, *"Area-efficient multipliers for digital signal processing applications,"* IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 43, no. 2, pp. 90–95, Feb. 1996.

[13] K. J. Cho, K. C. Lee, J. G. Chung, and K. K. Parhi, *"Design of low-error fixed-width modified booth multiplier,"* IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 5, pp. 522–531, May 2004.

[14] L. D. Van and C. C. Yang, *"Generalized low-error area-efficient fixedwidth multipliers,"* IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 52, no. 8, pp. 1608–1619, Aug. 2005.

[15] C. C. Sun, P. Donner, and J. Gotze, "Low-complexity multi-purpose IP core for quantized discrete cosine and integer transform," in *Proc.IEEE Int. Symp. Circuits Syst.*, 2009, pp. 3014–3017.