IMPLEMENTATION OF HIGH EFFICIENCY DA-BASED DCT USING ERROR-COMPENSATED ADDER TREE

[1]B.Naresh Reddy, [2] B.Narasimha Rao, [3] B.Kiran Kumar

[1]Kodada Institute of Technology & Science for Women, Kodada, India
[2]SVEC, Suryapeta, India
[3]GIST, Jagayyapeta, 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-throughput discrete cosine transform (DCT) design. Instead of the 12 bits used in previous works, 9-bit distributed arithmetic. DA-based DCT core with an error-compensated adder-tree (ECAT). The proposed 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 cost is reduced, and the speed is improved using the proposed ECAT.

Keywords Distributed arithmetic (DA)-based, error-compensated adder-tree (ECAT), 2-D discrete cosine transform (DCT).

I. INTRODUCTION

Discrete cosine transform (DCT) is a widely used tool in image and video compression applications[1]. Recently, the high-throughput DCT designs have been adopted to fit the requirements of real-time application [2]-[11]. The multiplier-based DCTs were presented and implemented in [2] and [3].To reduce area ,ROM-based distributed arithmetic (DA)was applied in DCT cores[4]–[6].Uramoto*et al.*[4]implemented the DA-based multipliers usingROMsto produce partial products together with adders that accumulated these partial products. In this way, instead of multipliers, the DA-based ROM can be applied in a DCT core design to 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 [5] and [6], 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 arithmetic logic unit (ALU) were applied in DCT design to reduce area cost [8], [9]. Hence the NEDA architecture is the smallest architecture for DA-based DCT core designs, but speed limitations exist in the operations of serial shifting and addition after the DA-computation. The high-throughput shift-adder-tree (SAT) and adder-tree (AT), those unroll the number of shifting and addition words in parallel forDA-based computation, were introduced in [10] and [11], respectively.However, a large truncation error occurred. In order to reduce the truncation error effect, several error compensation bias methods have been presented [12]–[14] based on statistical analysis of the relationship between partial products and multiplier-multiplicand. However, the elements of the truncation part outlined in this work are independent so that the previously described compensation methods cannot be applied.

Data compression is the technique to reduce the redundancies in data representation in order to decrease data storage requirements and hence communication costs. Reducing the storage requirement is equivalent to increasing the capacity of the storage medium and hence communication bandwidth. Thus the development of efficient compression techniques will continue to be a design challenge for future communication systems and advanced multimedia applications.

A. Classification of Compression Algorithms:

In an abstract sense, we can describe “data compression” as a method that takes an input data Dand generates a shorter representation of the data

c(D) with a fewer number of bits compared to that of D. The reverse process is called "decompression", which takes the compressed data c(D) and generates or reconstructs the data D' as shown in Figure 1.1. Sometimes the compression (coding) and decompression (decoding) systems together are called a "CODEC," as shown in the Fig 1.

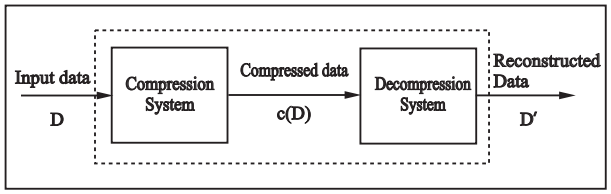


Fig. 1. Codec

The reconstructed data D' could be identical to the original data *D* or it could be an approximation of the original data D, depending on the reconstruction requirements. If the reconstructed data D' is an exact replica of the original data D, we call the algorithm applied to compress D and decompress c(D) to be "lossless". On the other hand, we say the algorithms are "lossy" when D' is not an exact replica of D. Hence as far as the reversibility of the original data is concerned, the data compression algorithms can be broadly classified in two categories – "lossless" and "lossy".

Loss less compression techniques work by removing redundant information as well as removing or reducing information that can be recreated during decompression. Loss less compression is ideal, as source data will be recreated without error. However, this leads to small compression ratios and will most likely not meet the needs of many applications. Compression ratios are highly dependent on input data, thus loss less compression will not meet the requirements of applications requiring a constant data rate or data size.

This brief addresses a DA-based DCT core with an error-compensated adder-tree (ECAT). The proposed 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 so as to achieve the peak-signal-to-noise-ratio (PSNR) [1] requirements. Therefore, the hardware cost is reduced, and the speed is improved using the proposed ECAT.

II. ARCHITECTURE OF PROPOSED DISTRIBUTED ARITHMETIC

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

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

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 two's complement binary fraction number. Equation (1) can be expressed as follows:

$$Y = \left[\, 2^0\ 2^{-1} \ldots 2^{-(Q-1)} \,\right] \begin{bmatrix} A_{1.0} & A_{2.0} & \cdots & A_{1.,0} \\ A_{1.1} & A_{2.1} & \cdots & A_{1.,1} \\ . & . & \cdots & . \\ . & . & \cdots & . \\ . & . & \cdots & . \\ A_{1,(Q-1)} & A_{2.(Q-1)} & \cdots & A_{1.,(Q-1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ X_L \end{bmatrix}$$

$$= \left[\, 2^0\ 2^{-1} \ldots 2^{-(Q-1)} \,\right] \begin{bmatrix} y_0 \\ y_1 \\ . \\ . \\ . \\ y(Q-1) \end{bmatrix} \qquad ...(2)$$

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

Note that y_0 may be 0 or a negative number due to two's complement representation. In (2), y_0 can be

calculated by adding all X*i* values when $A_{i,j}$=1 and then the transform output Y can be obtained by shifting and adding all nonzero yi values. Thus the inner product computation in (1) can be implemented by using shifting and adders instead of multipliers. Therefore, low hardware cost can be achieved by using DA-based architecture.

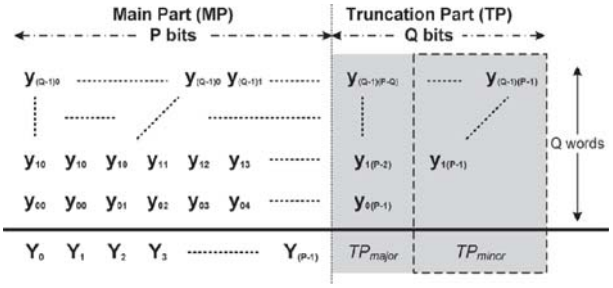


Fig .2 Q P-bit words shifting and addition operations in parallel

III. ECAT ARCHITECTURE

From equation (2), the shifting and addition computation can be written as follows:

$$Y = \sum_{j=0}^{Q-1} yj \cdot 2^{-j} \qquad ...(3)$$

In general, the shifting and addition computation uses a shift-and-add operator[7] 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[10] 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 in this brief to compensate for the truncation error in high-speed applications.

In Fig 2, the Q P-bit words operate the shifting and addition in parallel by unrolling all computations. Furthermore, the operation in Fig.2 can be divided into two parts: the main part (MP) that includes P most significant bits (MSBs) and the truncation part (TP) that has Q least significant bits (LSBs). Then, the shifting and addition output can be expressed as follows:

$$Y = MP + TP \cdot 2^{-(P-2)} \qquad ...(4)$$

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 2 (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.

A. Error-Compensated Scheme

From Fig. 2, (4) can be approximated as

$$Y \approx MP + \sigma \cdot 2^{-(P-2)} \qquad ...(5)$$

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

$$\sigma = \text{Round}\left(TP_{\text{major}} + TP_{\text{minor}} \right) \qquad ...(6)$$

$$TP_{\text{major}} = \frac{1}{2} \sum_{j=0}^{Q-1} y\,j\,(P-1-j) \qquad ...(7)$$

$$TP_{\text{minor}} = \frac{1}{4}\left(y_1\,(P-1) + \ldots + y\,(Q-1)\,(P-Q+1) \right) + \frac{1}{8}\left(y_2\,(P-1) + \ldots + y\,(Q-1)\,(P-Q+2) \right) + \ldots + \left(\frac{1}{2} \right)^{Q} y\,(Q-1)\,(P-1) \qquad ...(8)$$

For a given TP_{major}, ($Yj(p-1-j), 0 \le j \le (Q-1)$), the can σ 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:

$$TP_{minor} = k - \frac{1}{2} + \left(\frac{1}{2}\right)^{4k+1} \text{ for } Q = 4k$$

$$= k - \frac{1}{4} + \left(\frac{1}{2}\right)^{4k+2} \text{ for } Q = 4k+1$$

$$= k - \frac{1}{2} + \left(\frac{1}{2}\right)^{4k+3} \text{ for } Q = 4k+2 \quad \ldots. (9)$$

$$= k - \frac{1}{4} + \left(\frac{1}{2}\right)^{4k+4} \text{ for } Q = 4k+1$$

As $k \ge 1$,the TP_{minor} approximates

$$\left.\begin{aligned} TP_{minor} &= (k-1) + \frac{1}{2} \text{ for } Q = 4k \\ &= (k-1) + \frac{1}{4} \text{ for } Q = 4k+1 \\ &= k \text{ for } Q = 4k+2 \\ &= k + \frac{1}{2} \text{ for } Q = 4k+3 \end{aligned}\right\} \quad \ldots(10)$$

Hence, σ can be rewritten as three cases

Case 1. $Q = 0, 1, 2, 3\ \sigma =$ Round (TP_{major}) ...(11)

Case 2. $Q = 4k, 4k+1\ (k = 1)$

$$\sigma = (k-1) + \text{Round}(TP_{major} + 0.5) \quad \ldots(12)$$

Case 3. $Q = 4k+2, 4k+3\ (k = 1)$

$$\sigma = (k-1) + \text{ Round } (TP_{major}) \quad \ldots(13)$$

Table 1 Comparisons of absolute average error ε maximum absolute error ε_{max} and ε_{mse}

Error	(P,Q)	(12,3) Case1	(12,6) Case3	(12,9) Case2	(12,12) Case1
ε	Direct-T	1.0625	2.5078	4.0010	5.5001
	Proposed	0.2656	0.3789	0.3804	0.4738
	Post-T	**0.2500**	**0.2500**	**0.2500**	**0.2500**
ε_{max}	Direct-T	2.1250	5.0156	8.0020	11.000
	Proposed	0.6250	1.5000	2.0020	3.0000
	Post-T	**0.5000**	**0.5000**	**0.5000**	**0.5000**
ε_{mse}	Direct-T	1.3516	6.7614	16.730	31.224
	Proposed	0.1016	0.2184	0.2222	0.3472
	Post-T	**0.0859**	**0.0834**	**0.0833**	**0.0833**

comparisons of the absolute average error ε the maximum error ε_{max} and the mean square error ε_{mse} for the proposed error-compensated circuit with Direct-T and Post-T are listed in Table I. The internal word-length usually uses 12 bits in a DCT design. Consequently, word-length $P = 12$ is chosen together with different Q values of 3, 6, 9, and 12, which are listed in Table I. The Post-T method provides the most accurate values for fixed-width computation nowadays. In addition, the Direct-T method has the largest inaccuracies of the errors shown in Table I for low-cost hardware design. The proposed ECAT is more accurate than Direct-T and is close to the performance of the Post-T method using a 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.

B. Proposed ECAT Architecture:

The proposed ECAT architecture is illustrated in Fig 3 for $(P, Q) = (12, 6)$, 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).

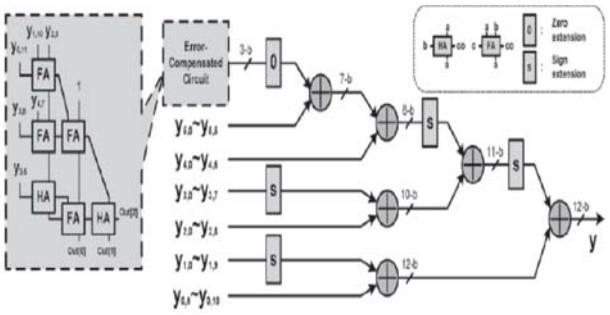


Fig. 3 Proposed ECAT architecture of shifting and addition operators for the $(P, Q) = (12,6)$ example.

Table 2. Comparisons of the proposed ecat with other architectures for a six 8-bit words example

	Shift-and-add	SAT	Proposed ECAT
Area(gates)	236	406	463
Delay(ns)	10.8	3.72	3.89
Area x delay	100%	59.3%	70.7%
ε_{mse}	0.326	6.761	0.218

The comparisons of area, delay, area-delay product, and accuracy for the proposed ECAT with other architectures are listed in Table II. The area and delay are synthesized using a Synopsys Designcompiler with the Artisan TSMC 0.18-%m Standard cell library. The proposed ECAT has the highest accuracy with a moderate areadelay product. The shift - and - add [7] method has the smallest area, but the overall computation time is equal to 10.8 ($= 1.8 \times 6$) ns that is the longest. Similarly, the SAT [10], which truncates the TP and computes in parallel, takes 3.72 ns to complete the computation and uses 406 gates, which is the best area-delay product performance. However, for system accuracy, the SAT is the worst option shown in Table II. Therefore, the ECAT is suitable for high-speed and low-error applications.

IV. 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 can be expressed as follows:

$$Z_n = \frac{1}{2} \text{kn} \sum_{m=0}^{7} x_m \times \cos\left(\frac{(2\,\text{m}+1)\, n\pi}{16}\right)$$

Where x_m denotes the input data; Z_n denotes the transform output. By neglecting the scaling factor 1/2, the 1-D 8-point DCT in aboue equation can be divided into even and odd parts: Z_e and Z_o as listed in below equations, respectively.

$$Z = \begin{bmatrix} Z_0 \\ Z_2 \\ Z_4 \\ Z_6 \end{bmatrix} = \begin{bmatrix} C_4 & C_4 & C_4 & C_4 \\ C_2 & C_6 & -C_6 & -C_2 \\ C_4 & -C_4 & -C_4 & C_4 \\ C_6 & -C_2 & C_2 & -C_6 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = C_{a \cdot a}$$

$$Z = \begin{bmatrix} Z_1 \\ Z_3 \\ Z_5 \\ Z_7 \end{bmatrix} = \begin{bmatrix} C_1 & C_3 & C_3 & C_7 \\ C_3 & -C_7 & -C_1 & -C_5 \\ C_5 & -C_1 & C_7 & C_3 \\ C_7 & -C_5 & C_3 & -C_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = C_{a \cdot b}$$

Where $C_i = \cos(i\,\Pi/16)$. 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. Table 1 expresses Z_{ee} ($Z_0$and Z_4) in the bit level formulation.

$$Z_{ee} = \begin{bmatrix} Z_0 \\ Z_4 \end{bmatrix} = \begin{bmatrix} C_4 & C_4 \\ C_4 & -C_4 \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \end{bmatrix} = C_{ee} \cdot A$$

$$Z_{eo} = \begin{bmatrix} Z_2 \\ Z_6 \end{bmatrix} = \begin{bmatrix} C_2 & C_6 \\ C_6 & -C_2 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \end{bmatrix} = C_{eo} \cdot B$$

Table 3. 9-BIT DA-BASED COEFFICIENT MATRIX

Z_0		Z_4	
Weight	**Value**	**Weight**	**Value**
-2°	0	-2°	A_1
2^{-1}	$A_0 + A_1$	2^{-1}	A_0
2^{-2}	0	2^{-2}	A_1
2^{-3}	$A_0 + A_1$	2^{-3}	A_0
2^{-4}	$A_0 + A_1$	2^{-4}	A_0
2^{-5}	0	2^{-5}	A_1

Z_0		Z_4	
Weight	**Value**	**Weight**	**Value**
2^{-6}	$A_0 + A_1$	2^{-6}	A_0
2^{-7}	0	2^{-7}	A_1
2^{-8}	$A_0 + A_1$	2^{-8}	$A_0 + A_1$

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 3.4 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.

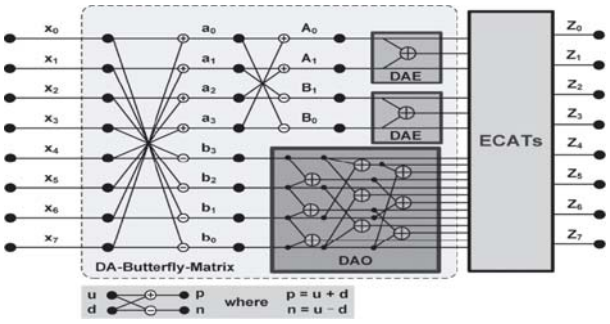
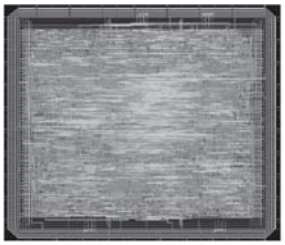


Fig. 4 Architecture of the proposed 1-D 8-point DCT.

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.

For the proposed 2-D DCT, the Synopsys Design Compiler was applied to synthesize the RTL design of the proposed core, and the Cadence SOC Encounter was adopted for placement and routing (P&R). Implemented in a 1.8-V TSMC $0.18-\mu m$ 1P6 M CMOS process, the proposed $8\times 82-$ D DCT core has a latency of 10 clock cycles and is operated at 125 MHz. As a result of the 8 parallel outputs, the proposed 2-D DCT core can achieve a throughput rate of 1 Gpixels per second $(=8\times 125$ MHz), meeting the 1080 p $(1920\times 1080\times 60$ pixels/s) high definition television (HDTV) specifications for 200 MHz based on low power operations. The core layout and simulated characteristics are shown in Fig. 5.



Characteristics	
Technology	0.18 µm
Supply Power	1.8 V
Die Size	690 µm x 683 µm
Gate Count	22.2 K
Max Freq.	125 MHz
Power	39 mW @125MHz (Max. Freq.) 8.6 mW @ 25MHz (HDTV Spec.)

Fig. 5. Core layout and characteristics.

V. RESULTS AND DISCUSSION

The 8-point 1-D DCT architecture and the implementation were discussed in the previous chapters. Now this chapter deals with the simulation and synthesis results of the implemented 1-D 8-point DCT. Here ModelSim tool is used in order to simulate the design and checks the functionality of the design. Once the functional verification is done, the design will be taken to the Xilinx tool for Synthesis process and the net-list generation.

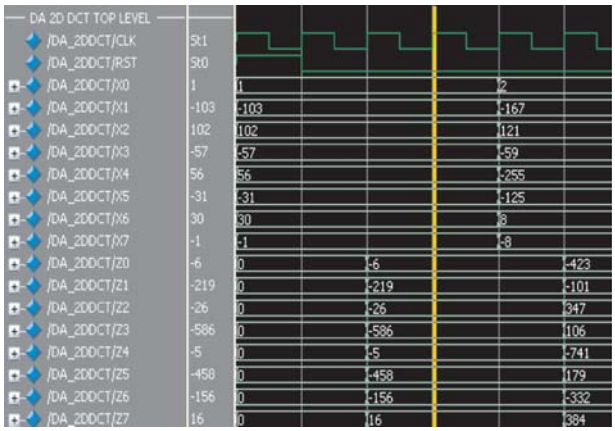


Fig. 6 Simulation results of Distributed Arithmatic DCT

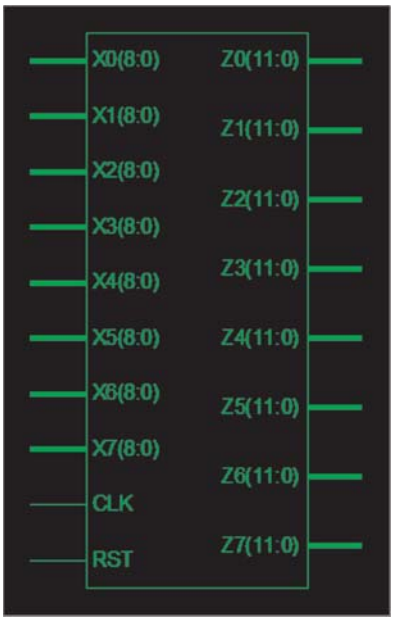


Fig. 7 RTL Schematic View

Table 4. Device Utilization Summary

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
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 developed 8-point 1-D DCT are simulated and verified their functionality. Once the functional verification is done, the RTL model is taken to the synthesis process using the Xilinx ISE tool. In synthesis process, the RTL model will be converted to the gate level netlist mapped to a specific technology library. This AES algorithm design can be implemented on FPGA (Field Programmable Gate Array) family of Virtex-5. Here in this Virtex-5 family, many different devices were available in the Xilinx ISE tool. In order to implement this AES design the device named as "XC3S500E" has been chosen and the package as "FG320" with the device speed as "-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.

Timing Summary: In timing summery, details regarding time period and frequency is shown are approximate while synthesize. After place and routing is over, we get the exact timing summery. Hence the maximum operating frequency of this synthesized design is not found and the minimum period as 5.037ns. OFFSET IN is the minimum input arrival time before clock and OFFSET OUT is maximum output required time after clock.

Table 5. Comparison of different 2-D DCT architectures with the proposed architecture

	Lin et al. [3]	Uramoto et al. [4]	Shams et al. [7]	Chungan et al. [10]	Huang et al. [11]	Proposed
Architecture	Multiplier-based	ROM-based	NEDA	DA-based	DA-based	**DA-based**
Technology	0.13μm	0.8μm	0.18μm	0.18μm	0.18μm	**0.18μm**
Multipliers/ROMs	1/0	0/256	0/0	0/0	0/0	**0/0**
Adders	26	16	92	12+18ALU*+16SAT	50+16AT	**46+16 ECAT***
DA-precision	–	–	12 bits	13 bits	13 bits	**9 bits**
Throughput Rate (pels/sec)	100 M	100 M	77 M†	250 M	400 M	**1 G**
Gate Counts(NAND2)‡	60 K	25.5 K	22.5 K	27.8 K	39.8 K	**22.2 K**
Hardware Efficiency	1.6	3.92	3.42	9	10.05	**45**
Accuracy (CCITT° Compatible)	Yes	Yes	Yes	N/A	N/A	**Yes**

Table 5 compares the proposed $8 \times 82-$D DCT core with previous 2-D DCT cores. In [3], a multiplier-based DCT core based on pipeline radix-4^2 single delay feedback path (R4^2SDF) architecture to achieve high-speed design. The ROM-based DCT core is presented in [4] to reduce hardware cost. In [7], a NEDA architecture is presented by using adders to reduce the chip area of DCT core. Nevertheless, a speed limitation for shift-and-add is in NEDA design. In [10] and [11], the SAT and AT architectures for DA-based DCTs improve the throughput rate of the NEDA method. However, DA-precision must be chosen as 13 bits to meet the system accuracy with more area overhead. The proposed DCT core uses low-error ECAT to achieve a high-speed design, and the DA-precision can be chosen as 9 bits to meet the PSNR requirements for reducing hardware costs. The proposed DCT core has the highest hardware efficiency, defined as follows (based on the accuracy required by the presented standards)

$$\text{Hardware efficiency} = \frac{\text{Throughput Rate}}{\text{Gate Counts}}$$

Table 6. COMPARISONS OF 2-D DCT ARCHITECTURES IN FPGAS

FPGA-Chip	XC2VP30			XC3S200	
Architecture	[15]	[16]	Proposed	[17]	Proposed
# of 4 input LUTs	10310	2618	2990	2271	2847
# of Slices	5729	2823	1872	1221	1585
# of Slice Flip Flops	3736	3431	1837	616	1817
Clock Freq. (MHz)	149	107	99	50	61
Throughput (M-pels/s)	149	107	792	400	488
Power (mW)	N/A	N/A	166.8	281	91

VI. CONCLUSION

A high-speed and low-error 8×82-D DCT design with ECAT is proposed to improve the throughput rate significantly up to about 13 folds at high compression rates by operating the shifting and addition in parallel. Furthermore, the proposed error-Compensated circuit alleviates the truncation error in ECAT. In this way, the DA-precision can be chosen as 9 bits instead of 12 bits so as to meet the PSNR requirements. Thus, the proposed DCT core has the highest hardware efficiency than those in previous works for the same PSNR requirements.

REFERENCES

[1] Wang Y., Ostermann J., and Zhang Y., 2002 Video Processing and Communications, 1st ed. Englewood Cliffs, NJ: Prentice-Hall,.

[2] Chang Y. and Wang C., 1995 "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,.

[3] Lin C. T., Yu Y. C., and Van L. D., 2008 "Cost-effective triple-mode reconfigurable pipeline FFT/IFFT/2-D DCT processor," IEEE Trans. Very Large Scale Integr. Syst., vol. 16, no. 8, pp. 1058–1071,.

[4] Uramoto S., Inoue Y., Takabatake A., Takeda J., Yamashita Y., Yerane H., and Yoshimoto M., 1992 "A 100-MHz 2-D discrete cosine transform core processor," IEEE J. Solid-State Circuits, vol. 27, no. 4, pp. 492–499,.

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

[6] Meher P. K., 2006. "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,

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

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

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

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

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

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

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

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

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

[16] Tumeo A., Monchiero M., Palermo G., Ferrandi F., and Sciuto D., 2007, "A pipelined fast 2D-DCT accelerator for FPGA-based SoCs," in Proc. IEEE Comput. Soc. Annu. Symp. VLSI, pp. 331–336.

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