American Journal of Applied Sciences 11 (2): 180-188, 2014
ISSN: 1546-9239
©2014 Science Publication
doi:10.3844/ajassp.2014.180.188 Published Online 11 (2) 2014 (http://www.thescipub.com/ajas.toc)

AREA EFFICIENT DISTRIBUTED ARITHMETIC DISCRETE COSINE TRANSFORM USING MODIFIED WALLACE TREE MULTIPLIER

[1]Meenaakshi Sundari, R.P., [2]R. Anita and [1]V. Venkateshwaran

[1]Department of Electronics and Communication Engineering,
Sasurie College of Engineering, Vijayamangalam, Tamilnadu, India
[2]Department of Electrical and Electronics Engineering,
Institute of Road and Transport Technology, Erode, Tamilnadu, India

Received 2013-03-15; Revised 2013-10-29; Accepted 2013-12-19

ABSTRACT

In this study by using the modified Wallace tree multiplier, an error compensated adder tree is constructed in order to round off truncation errors and to obtain high through put discrete cosine transform design. Peak Signal to Noise Ratio (PSNR) is met efficiently since modified Wallace Tree method is an efficient, hardware implementable digital circuit that multiplies two integers resulting an output with reduced delays and errors. Nearly 6% of delays and around 1% of gate counts are reduced. The number of look up tables consumed is 2% lesser than that of the previous multipliers. Thus an area efficient discrete cosine transform is built to achieve high throughput with minimum gate counts and delays for the required Peak Signal to Noise Ratio when compared to the existing DCT's.

Keywords: 2-D Discrete Cosine Transform, Error Compensated Adder Tree (ECAT), Read Only Memory (ROM), New Distributed Arithmetic (NEDA)

1. INTRODUCTION

Discrete Cosine Transform is a broadly used tool in image and Video Compression. High throughput DCT designs are adopted to fit the requirements of real time applications. Video Processing and Communications, used DCT,s for the compression purposes. The multipliers multiplier based DCT's are implemented. ROM based logics are very effective in reducing area. The ROM based logic is shown in the **Fig. 1**. The DA based ROM can be adapted in DCT's to reduce area. The DA based multipliers use ROMs to produce the partial products along with adders which are capable of accumulating these partial products. The symmetrical properties of the 2-D DCT transform and the parallel DA architectures are used in reducing the ROM size (Chang and Wang, 1995). Moreover, the retrenched multipliers are adopted to decrease the multiplier cost and ROM size. To further decrease the ROM cost, a finite wordlength analysis is provided to indicate the architecture with 13-bit internal wordlength to achieve 40 dB signal to noise ratio in 256 point 2D-DCT mode (Lin *et al*., 2008). ROM free DA architectures were constructed. Shams *et al.* (2006) employed a bit level sharing technique to implement an adder based butterfly adder matrix which uses 35 adders and 8 Shift and addition elements to replace the ROM. According to NEDA method, the recursive form and the Arithmetic Logic Unit were used in DCT design to reduce area and cost. The NEDA architecture is the smallest and an efficient architecture for DA based DCT core designs. There exist some speed limitations in the operations of serial shifting and addition after the DA computation. The high throughput Shift Adder Tree and Adder Tree unrolls the number of shifting and addition operation in simultaneously for DA based computation. A large truncation error occurs.

Corresponding Author: Meenaakshi Sundari, Department of Electronics and Communication Engineering, Sasurie College of Engineering, Vijayamangalam, Tamilnadu, India

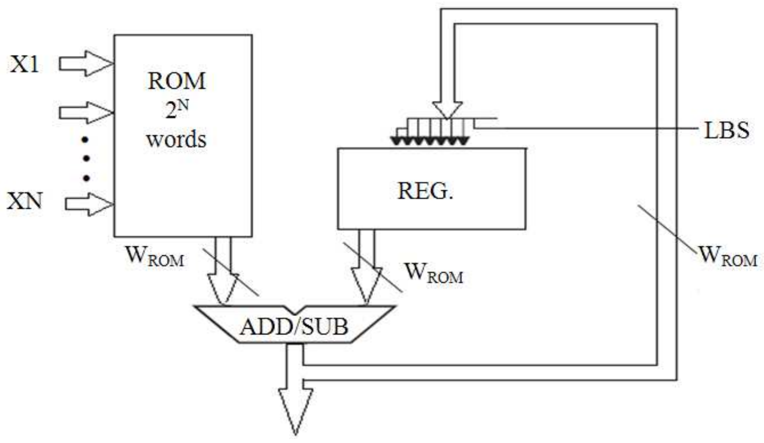


Fig. 1. ROM based logic

To reduces the truncation errors, multiple error compensation bias methods have been adapted based on the relationship between the partial products and the multiplier and multiplicand. The elements of the truncation part explained in this study are independent. Hence previously described compensation methods cannot be applied. These results in an DA based Error Compensated Adder Tree (ECAT). The ECAT operates shifting and addition in parallel by unrolling all the words to be computed. The ECAT circuit removes the truncation error for high accuracy.

An excellent architecture utilizes a fast DCT algorithm and multipliers accumulators based on distributed arithmetic have contributed to reducing hardware amount and to enhance the speed performance (Yu and Swartziander, 2001). Furthermore, the mean values of errors generated in the core were minimized to enhance the computational accuracy with the wordlength constraints (Uramoto *et al*., 1992). The speed of the DCT lies in the Speed of the multiplier that was used to compute the bits after the completion of DA computation. The Shift and Add multiplier consumes much time for every computation. So we are moving for the fast multiplier. Here we have chosen the Wallace tree multiplier and modified Wallace tree multiplier.

1.1. Distributed Arithmetic

Distributed Arithmetic is highly an efficient technique for computing inner products between a fixed and a variable data. Croisier and Peled designed this method. Liu presented a similar method. The Equation for DA can be written as:

$$Y = a^T.x = \sum^N{}_{i=1} a_i x_i$$

where, a = 1, 2, 3…N are the coefficients that are fixed. The scaled two's complement representation is used for the data components. Inputs are shifted bit serially out from the shift registers initialize with the least significant bit. Bits are used to be address for the ROM storing Look up Table. The word length in the ROM, WROM depends on the value of the magnitude and the coefficient of the word length. The parallel implementation of ROM Based Logic is shown in **Fig. 1**. The Inner products that contains many terms can be partitioned into a number of smaller inner products that can be computed and summed as per distributed arithmetic or by using an adder tree. Distributed Arithmetic can also use two or more bits concurrently. Here the distributed arithmetic is chosen as key to reduce the area by Yu and Swartziander (2001). "DCT implementation with distributed arithmetic".

1.2. Shift Accumulator Logic

The Shift Accumulator Logic is shown in **Fig. 2**. For F0, the clock cycle corresponds to the signed bit of data, F0 should be subtracted. This is performed by adding-F0 and inverting all the bits in F0 using the XOR gates. After-F0 has been added, the most significant part of the inner product is shifted out of the register accumulator. This can be done by accumulating the Zeros. The clock cycle corresponding the inner ROM is WD+WROM. The carry save adders in the accumulator is let free efficiently by loading the sum and carry bits of the carry save adder into two shift registers. Then the out puts from them are again added by a single carry save adder as shown in the **Fig. 1**. This way of computing helps to achieve the double throughput but still the delay and gate counts are major constraints.

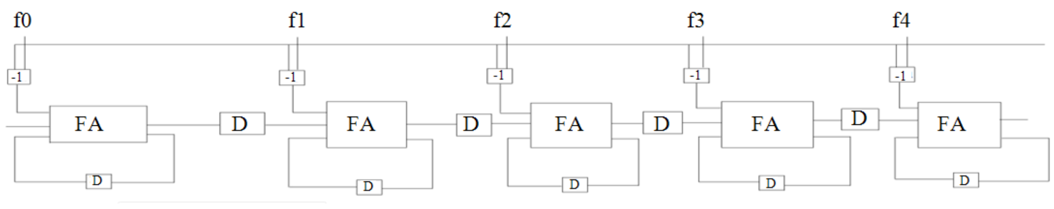


Fig. 2. Shift and add logic

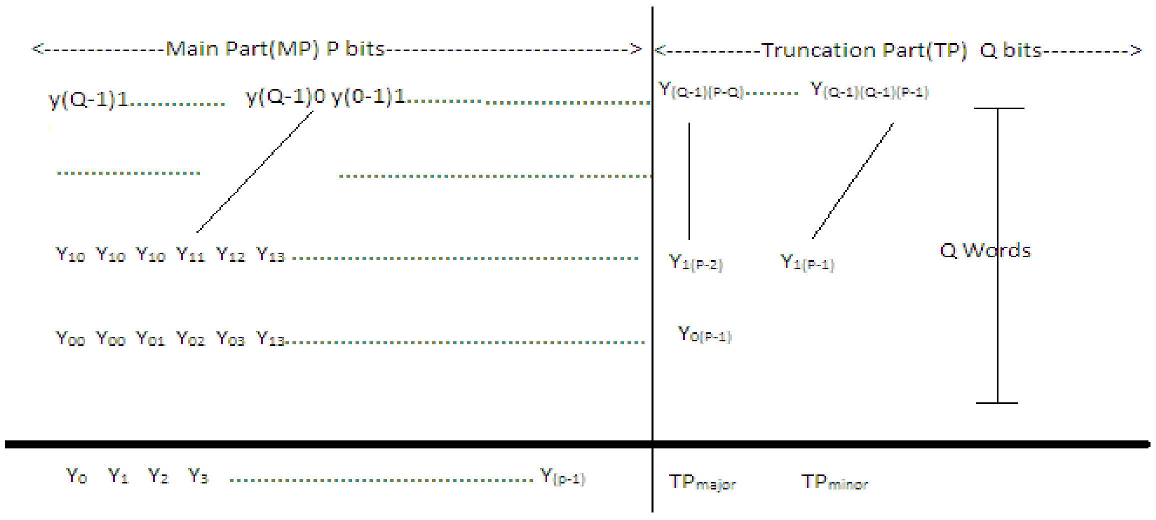

<-------------Main Part(MP) P bits------------------------->
<----------Truncation Part(TP) Q bits--------->
y(Q-1)1............. y(Q-1)0 y(0-1)1..........
Y(Q-1)(P-Q)........ Y(Q-1)(Q-1)(P-1)
Y10 Y10 Y10 Y11 Y12 Y13 ....
Y1(P-2)
Y1(P-1)
Q Words
Y00 Y00 Y01 Y02 Y03 Y13........
Y0(P-1)
Y0 Y1 Y2 Y3 .... Y(p-1)
TPmajor
TPminor

Fig. 3. Error compensation technique

1.3. Error Compensation

Generally the shifting and addition computation uses a shift and add operator for reduction of hardware in VLSI technology. The computation increases when the number of the shifting and addition increases. Thus the Shift Adder Tree operates by shifting and adding in parallel. A large truncation error occurs in SAT and hence ECAT architecture is proposed in this study to reduce errors. In the **Fig. 3** shown below the QP bit word operates the shifting and addition in parallel. The operation can be divided into two parts, The Main Part, that contains the Most Significant Bits and the Truncation Part that consists of the Least Significant Bits. The shifting and addition output can be written as:

$$Y = MP+TP.2^{-(P-2)}$$

The output Y will gain the P-bit MSBs applying a rounding operation called post truncation which is represented as (Post-T). The hardware cost increases in the VLSI design anyway. Commonly the TP is truncated to reduce the hardware cost in parallel shifting and adding operations known as the Direct Truncation method. A large truncation error occurs due to the negation of carry propagation from the truncation part to most significant part. In order to remove the truncation errors many error compensation bias methods have been presented. All the efforts result in a fixed width multiplier. The products in the multiplier have a relationship between the input multiplier and the multiplicand.

This type of compensation method uses the correlation of inputs to calculate a fixed or an adaptive compensation bias using simulation or the statistical analysis. The error compensation by shifting and adding operation is given in the following **Fig. 4**. The figure shows about the sequence of shifting and additions. It consists of sign extension bits and zero extension bits. Every sign extension is represented with ‘S’ and zero extension bit with ‘0’. The bits are serially shifted and then added with the next four coming bits from the registers. The comparisons of the Absolute average error, the Maximum error and the mean square error will be discussed in the simulation environment.

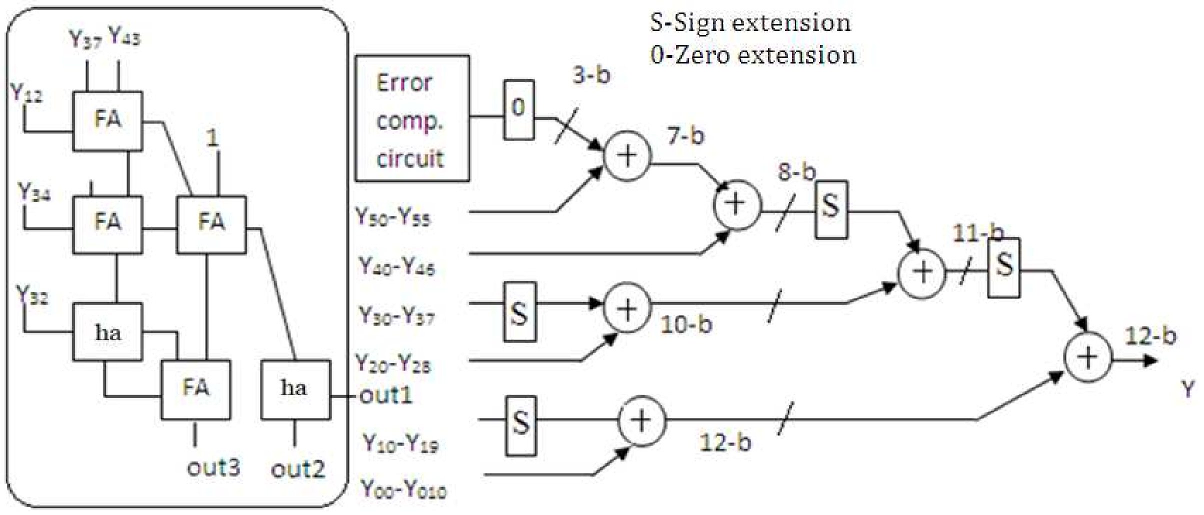


Fig. 4. Error compensation using shift and add logic

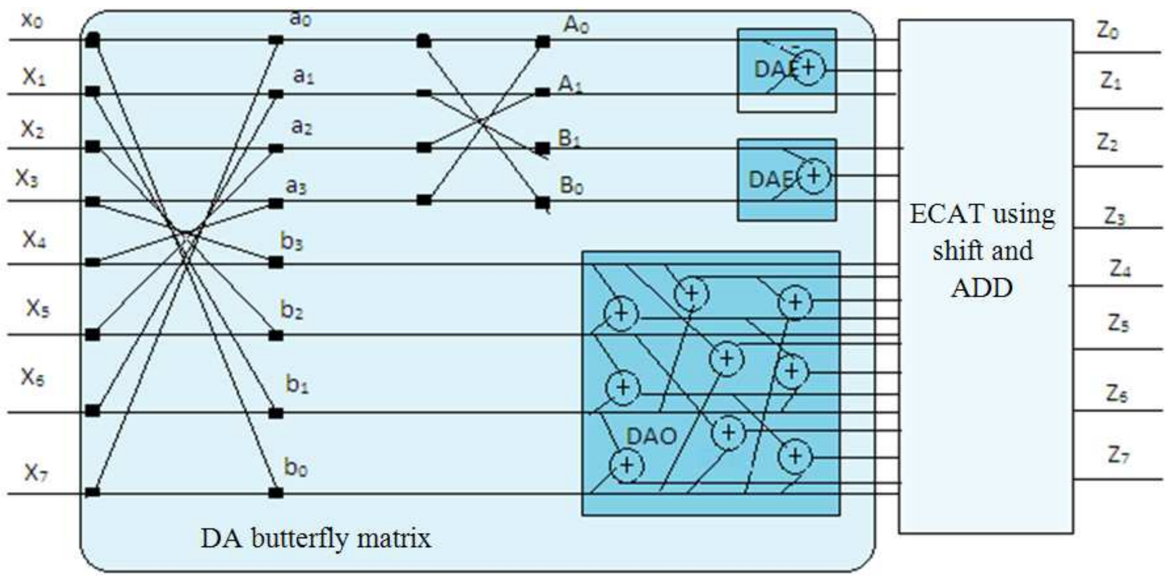


Fig. 5. DCT Using Shift and Add

The adders reduce the chip area of DCT core. A speed limitation occurs in the shift and adds multiplier. The following **Fig. 5** shows the implementation of the DCT using an ECAT with shift and add logic. The error compensation is performed at the end of the DCT. The DCT is freed of error at its output with the help of an ECAT with Shift and Add Logic.

1.4. Wallace Tree Multiplier

A Wallace tree is an efficient hardware technique which can be implemented in the form of a digital circuit that multiplies two integers. The multiplier was introduced in 1964 by an Australian computer scientist Chris Wallace. Basically a Wallace has three steps:

- Multiply each and every bit of the arguments with AND, producing n^2 results. With the position of the multiplied bits, the wires always carry varying weights
- Reduce the number of partial products to two with the layers of full and half adders
- Group the wires into two numbers and add those wires with a conventional adder

The Wallace is considerably faster than a simple array multiplier since its height is logarithmic in word size and not a linear one. A large number of adders are required and also its wiring is much irregular and more complicated. So it is rejected by designers where its design constraints plays a major role. It uses a logarithmic style for tree network reduction. The Wallace is a very high speed multiplier. The addition of the partial product bits in parallel using a tree of carry save adders constitutes a Wallace multiplier as shown in the **Fig. 6**. Initially the produce partial products are feed into an array of parallel adders. The adders used may be half adders and full adders depending on the number of partial products that is fed. The carry in one stage will be provided to the other stage in order to save carry. The final stage of the multiplier always propogates in a ripple manner and hence constitutes ripple carry propagation. The adders plays a major role in the multiplication. The way in which the adders are used constitutes a simple and efficient multiplication; in case of Wallace tree multiplier. Parallel addition results in less time delay which in turn proves Wallace to be better than shift and add logic multipliers.

1.5. Modified Wallace Tree Multiplier

The modified Wallace tree uses compressors. Compressors are arithmetic components that are similar to parallel counters but differs in two cases:

- They have explicit carry in and carry out bits
- There may be some redundancy among the ranks of the sum and carry output bit

1.6. Compressor

Figure 7 and 8 shows the 4:2 compressor i/o diagram and the compressor architecture. The compressor has 4 input bits and produces 2 sum output bits 'out0' and 'out1'. This also has a carry in 'Cin' and a carry out 'Cout' bit. The total number of the input and output bits are 5 and 3.The input bits including 'Cin' have rank 0 while the two output bits have ranks 0 and 1. The 'Cout' has rank 1. Thus the output of the 4:2 compressors is a redundant number. **Figure 9** shows the compressor logic used in four bit Wallace tree multiplier. The light green shades indicates the partial products that can be reduced with the help of compressor. They were four bits and hence they are reduced by 4:2 compressors. The partial bits 3 and 2 are reduced with full adders and half adders.

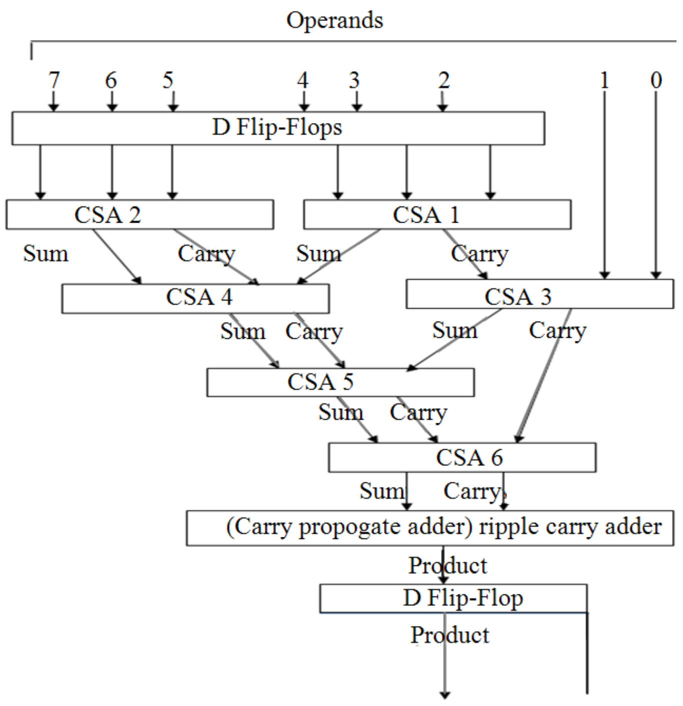


Fig. 6. Wallace tree multiplier

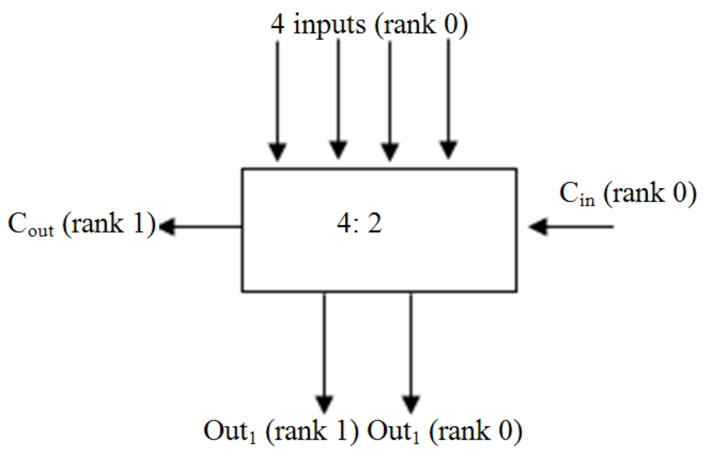


Fig. 7. 4: 2 Compressors I/O diagram

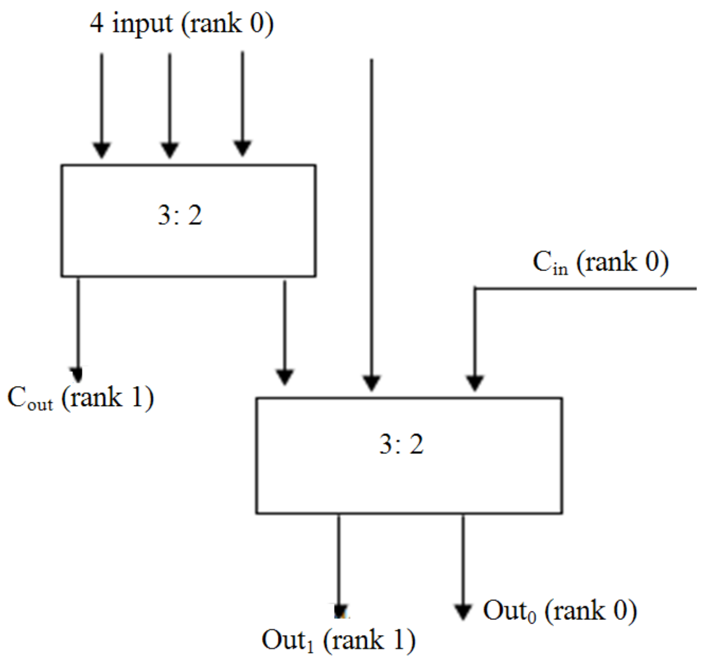


Fig. 8. 3: 2 Compressor architecture

2. RESULTS

Thus an error compensated adder tree was constructed using Modified wallace multiplier in order to reduce the number of delays which results in high throughput. Nearly 6% of delays were reduced when compared to the conventional shift and add multipliers which have limitations in high speed applications as shown in the **Table 1**. Around 1% of gate counts are also reduced as shown in **Fig. 11**. The number of 4 input LUT’s consumed was 2% less than that of the shift and adds logic as shown in **Fig. 10**. Comparatively Modified Wallace is better than Shift and Add logic in many ways. The **Fig. 13** shows the Model sim output of the wallace tree multiplier based DCT. The model sim output of the three multipliers are the same in simulation results but they differ in the compilation time and gate counts as shown in **Fig. 11 and 12**.

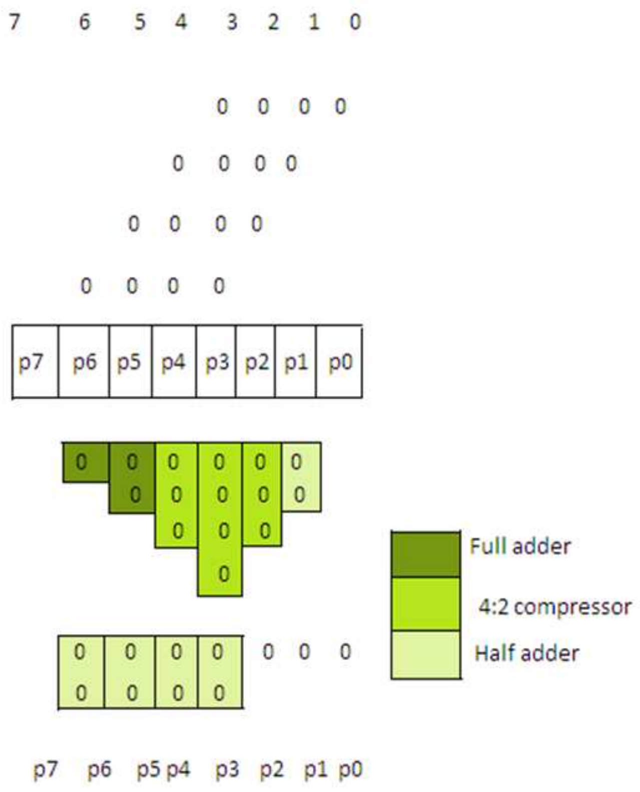


Fig. 9. 4: 2 Compressor logic used in 4 bit wallace

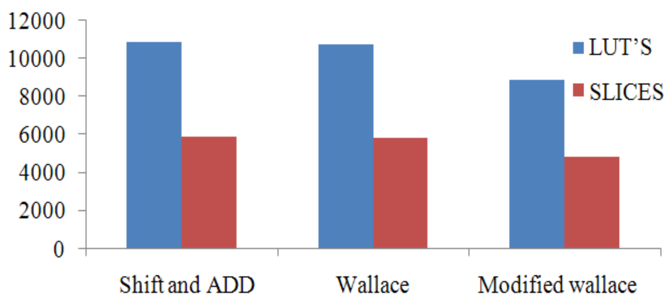


Fig. 10. Comparison of three multipliers performance (LUT's and SLICES)

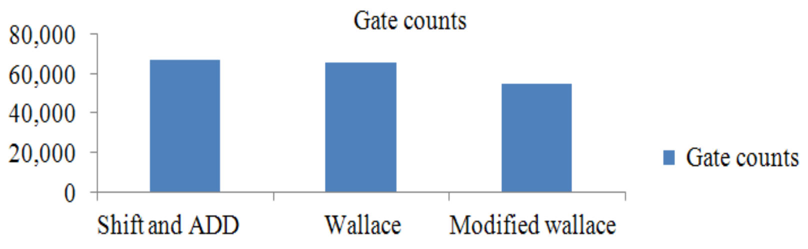


Fig. 11. Comparison of three multipliers performance (Gate counts)

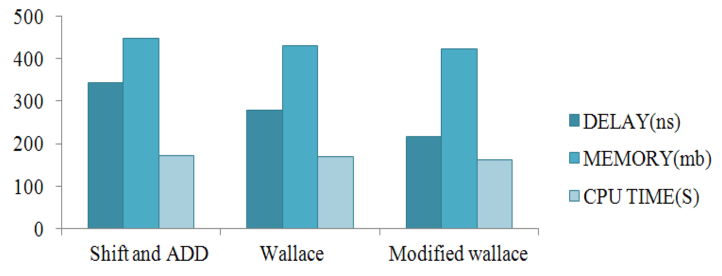
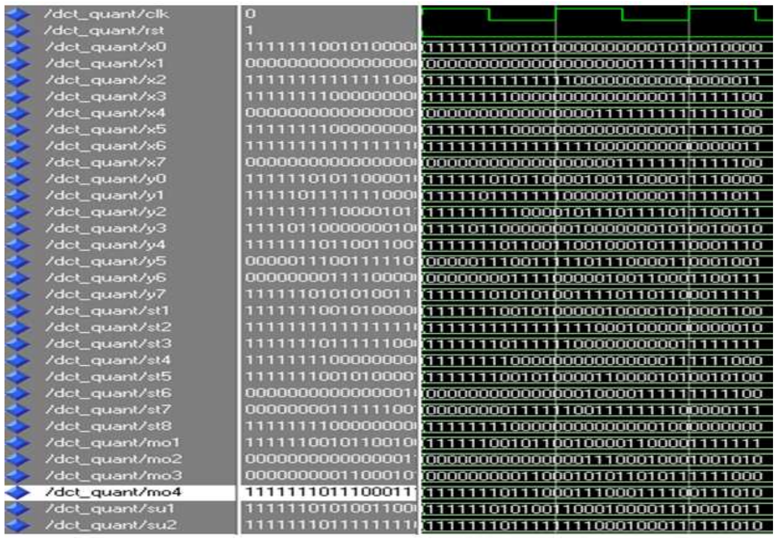


Fig. 12. Comparison of three multipliers performance (DELAY, MEMORY and CPU time)



```
/dct_quant/clk    0
/dct_quant/rst    1
/dct_quant/x0     1111111100101000
/dct_quant/x1     0000000000000000
/dct_quant/x2     1111111111111100
/dct_quant/x3     1111111110000000
/dct_quant/x4     0000000000000000
/dct_quant/x5     1111111110000000
/dct_quant/x6     1111111111111111
/dct_quant/x7     0000000000000000
/dct_quant/y0     1111110101100001
/dct_quant/y1     1111101111111000
/dct_quant/y2     1111111110000101
/dct_quant/y3     1111011000000010
/dct_quant/y4     1111111011001100
/dct_quant/y5     0000011100111110
/dct_quant/y6     0000000011110000
/dct_quant/y7     1111110101010011
/dct_quant/st1    1111111100101000
/dct_quant/st2    1111111111111111
/dct_quant/st3    1111111011111100
/dct_quant/st4    1111111110000000
/dct_quant/st5    1111111100101000
/dct_quant/st6    0000000000000001
/dct_quant/st7    0000000011111100
/dct_quant/st8    1111111110000000
/dct_quant/mo1    1111110010110010
/dct_quant/mo2    0000000000000001
/dct_quant/mo3    0000000001100010
/dct_quant/mo4    1111111011100011
/dct_quant/su1    1111110101001100
/dct_quant/su2    1111111011111111
```

Fig. 13. Model sim output of Wallace based DCT

Table 1. Comparison of three different multiplier results

Parameters	Shift and ADD	Wallace	Modified wallace
No. of 4 I/P LUT's	10,849	10698	8838
No. of occupied slices	5852	5798	4793
Gate counts	66489	65484	54222
Delay (ns)	343	279	217
Memory usage (kb)	459540	441044	433436
CPU time (sec)	171	169	161

3. CONCLUSION

The Proposed DCT core has the highest hardware efficiency than those in previous works or the same PSNR requirements. The model sim output of the three multipliers are the same in simulation results but they differ in the compilation time and gate counts. The improved version multipliers may be used in DCT's in future for further reduction of delays. The various multiplier based DCT's may be applied in video compression as like in delay-power performance comparison of multipliers in VLSI circuit design. In summary, the proposed architecture is suitable for high compression rate applications in VLSI designs.

4. FUTURE ENHANCEMENT

Furthermore, the proposed 2-d DCT core synthesized by using Xilinx 13.1 and the Xilinx XC2VP30 FPGA can achieve 792 megapixels per second. The high speed modified Wallace DCT's can be applied in many applications in future since it is significant for high speed and less number of delays. Any other advanced multiplier can be used in future for further reduction in delays.

5. REFERENCES

Chang, Y.T. and C.L. Wang, 1995. New systolic array implementation of the 2-D discrete cosine transform and its inverse. IEEE Trans. Circ. Syst. Video Technol., 5: 150-157. DOI: 10.1109/76.388063

Lin, C.T., Y.C. Yu and L.D. Van, 2008. Cost-effective triple-mode reconfigurable pipeline FFT/IFFT/2-D DCT processor. IEEE Trans. Very Large Scale Integr. Syst., 16: 1058-1071. DOI: 10.1109/TVLSI.2008.2000676

Shams, A.M., A. Chidanandan, W. Pan and M.A. Bayoumi, 2006. NEDA: A low-power high-performance DCT architecture. IEEE Trans. Signal Process., 54: 955-964. DOI: 10.1109/TSP.2005.862755

Uramoto, S., Y. Inoue, A. Takabatake, J. Takeda and Y. Yamashita *et al*., 1992. A 100-MHz 2-D discrete cosine transform core processor. IEEE J. Solid-State Circ., 27: 492-499. DOI: 10.1109/4.126536

Yu, S. and E.E.S. Swartziander, 2001. DCT implementation with distributed arithmetic. IEEE Trans. Comput., 50: 985-991. DOI: 10.1109/12.954513