A BIT-SERIAL ARCHITECTURE FOR A MULTIPLIERLESS DCT

S. Choomchuay, *Member,* ECTI and S.Timakul

Department of Electronics, Faculty of Engineering, and
Research Center for Communications and Information Technology (ReCCIT)
King Mongkut's Institute of Technology Ladkrabang (KMITL)
Chalongkrung Road, Ladkrabang, Bangkok 10520, Thailand

Tel: +66 2326 4222 Ext. 114, Fax: +66 2739 2398, Email: kchsomsa@kmitl.ac.th

ABSTRACT

This paper describes the implementation of a multiplierless 1-D DCT which is also applicable to 2-D computation. The key idea is based on the binary DCT of which multipliers are replaced by lifting parameters that essentially are shifts and adds. For low power applications and smaller hardware size, a bit-serial architecture was invoked in the implementation of such an algorithm. Varying data word length, MSE obtained from our approach and some similar algorithms are also investigated and reported.

Key words: DCT, Multiplierless DCT, BinDCT, VLSI, Image Coding

1.0 INTRODUCTION

The Discrete Cosine Transform (DCT) is significantly of interest in the area of image compression according to its high compaction energy. It has become the core of many standards, JPEG, MPEG, and the H.26x family, such as ITU-81 (1992), ISO-1772 (1990). Those are well discussed by Westwater and Furht (1997). DCT is also playing an important role and digital watermarking (Hartang and Girod, 1998; Suhail and Obaidat, 2001; Pereira *et al.*, 2000). In both software and hardware implementations, there are many fast algorithms to speed up the computation of a DCT. A 2-D DCT can be easily computed by recursive use of a 1-D DCT computing scheme. The direct implementation of a 2-D DCT is also possible but generally requires more effort.

Direct computation of a DCT requires floating point multiplications, which is indeed slow and complicated. Such a mathematical manipulation can be avoided by using integer implementations based on distributed arithmetic (DA) proposed by Peled and Lui (1974) and used later by many others in hardware DCTs and low rate video codec designs (Sun *et al.*,1989; Fujiwara *et al.*, 1992). The approach still has some drawbacks since those fixed-point multiplications need rather wide data bus (32-bit, for instance). The implementation can be a limitation when the constraints are chip area and power consumption.

Based on Chen's factorisation of the DCT matrix (Chen *et al.*, 1977), Tran (2000) and Liang and Tran (2001) have proposed an approximation computation of DCT by introducing the lifting scheme. The multiplication basis is approximated by rationals of the form $k/2^m$, which can be implemented efficiently by binary shifts. This multiplierless type DCT is also known as a binary DCT or Bin DCT. Both the forward and the inverse transforms are implemented in a similar manner. The implementation can be made further less complicated and more regular by making use of scaled DCT and in-place computation.

Our work concerns the implementation of a 1-D DCT based on Liang and Tran's work (2001). For very low bit rate applications with quite high compression gain, we treated our design by making use of bit-serial computation scheme. The resultant design is compact and has low power consumption.

2.0 DCT COMPUTATION

2.1 Fast DCT Technique

The one-dimensional (1-D) N-point DCT, forward and inverse transforms are defined as (1) and (2) below. These are also recommended by ITU-81 (1992) for its 8-point DCT.

$$X(k) = \alpha(k) \cdot \sqrt{\frac{2}{N}} \cdot \sum_{n=0}^{N-1} x(n) \cos\left\{\frac{(2n+1)k\pi}{2N}\right\} \qquad (1)$$

$$x(n) = \sqrt{\frac{2}{N}} \cdot \sum_{k=0}^{N-1} X(k) \cdot \alpha(k) \cos\left\{\frac{(2n+1)k\pi}{2N}\right\} \qquad (2)$$

for $0 \le k \le N-1$ and $0 \le n \le N-1$. Where

$$\alpha(i) = \begin{cases} \sqrt{\frac{1}{2}} & i = 0 \\ 1 & otherwise \end{cases} \qquad (3)$$

Direct computation of (1) is computation intensive, order of N^2. Many fast computations were implemented by matrix-matrix multiplication (MMM) or fast algorithm (FCT) or a combination of both. Chen *et al.* (1977) proposed a method that vastly reduces the computations. Their implementation takes $(N\log_2 N - \frac{3}{2}N + 4)$ real multiplications. To reorder the matrix elements, the sine and cosine butterfly matrices were alternated by introducing binary matrices. Because of the fewer operations, not only the speed can be increased but the hardware complexity also can be reduced.

The resultant Chen's factorisation of an 8-point DCT is depicted in Fig. 1. The computation requires 13 multiplications and 29 additions (Chen *et al.*, 1977). The butterfly-like computing scheme and plane rotation enables an in-place computation. It should be noted that the scaling factor of 1/2 has to be applied at the end of the transformer to obtain the true DCT coefficients.

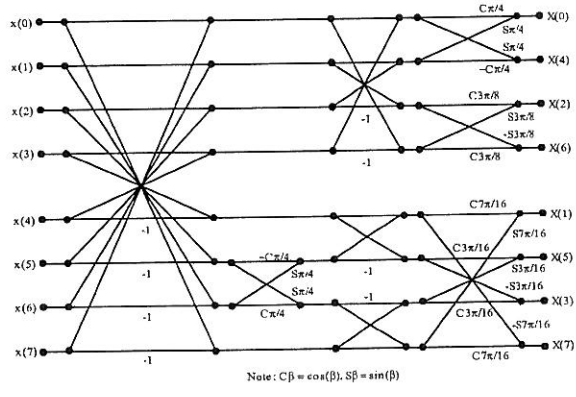


Fig. 1: Chen's Factorisation 1-D Fast DCT Flow Diagram

2.2 Lifting Structures and Binary DCT

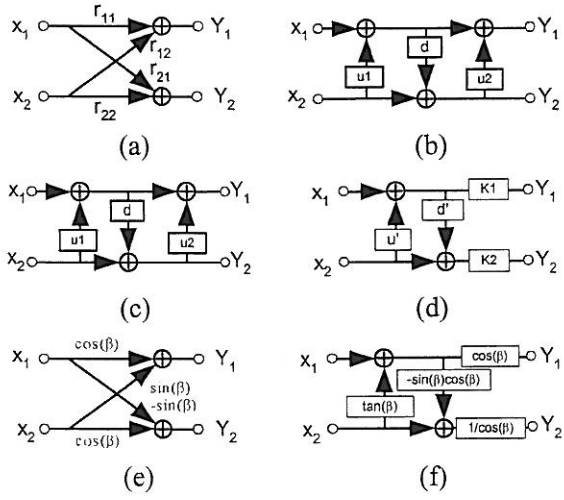
A butterfly with rotation plane can be viewed as a matrix operation, i.e. $Y_1 = r_{11}x_1 + r_{12}x_2$ and $Y_2 = r_{21}x_1 + r_{22}x_2$. In Fig. 2(a), consider the case: $r_{11} = r_{22} = \cos(\beta)$ and $r_{12} = -r_{21} = \sin(\beta)$, with 3-lift lifting structure shown in Fig. 2(b). It can be computed that

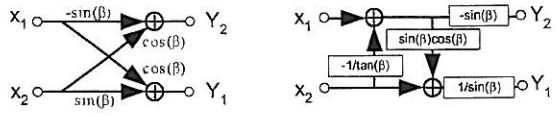
$$
\begin{aligned}
d &= r_{21} = -\sin(\beta) \\
u_1 &= \frac{r_{22}-1}{r_{21}} = \frac{\cos(\beta)-1}{-\sin(\beta)} \\
u_2 &= \frac{r_{11}-1}{r_{21}} = \frac{\cos(\beta)-1}{-\sin(\beta)}
\end{aligned} \qquad (5)
$$

The same method can be applied to other butterflies and rotating angles throughout the DCT. With the idea of a scaled DCT, it is possible to reduce 3-lifting steps to 2-lifting steps with scaling factors. According to the butterfly shown in Fig. 2(a), if $r_{11} \neq 0$ and $r_{11}r_{22} - r_{21}r_{12} \neq 0$, then, as shown in Fig. 2(d), u', d', K_1 and K_2 can be computed from

$$
\begin{aligned}
u' &= \frac{r_{12}}{r_{11}}, \qquad d' = \frac{r_{11}r_{21}}{r_{11}r_{22} - r_{21}r_{12}} \\
K_1 &= r_{11}, \qquad K_2 = \frac{r_{11}r_{22} - r_{21}r_{12}}{r_{11}}
\end{aligned} \qquad (6)
$$

In Fig. 2(e) where $r_{11} = r_{22} = \cos(\beta)$ and $r_{12} = -r_{21} = \sin(\beta)$, then, as shown in Fig. 2(f), $u' = \tan(\beta)$, $K_1 = \cos(\beta)$, $d' = -\sin(\beta)\cos(\beta)$ and $K_2 = 1/\cos(\beta)$. For some particular rotating angles such as $\beta \to j\pi + \pi/2$ for any integer j, the very small value of $\cos^2(\beta)$ can have high sensitivity to round-off and truncation errors. At the same time, tan( ) becomes bigger and the dynamic range increases. To avoid such undesirable conditions, the computation in Fig. 2(e), 2(f) can be permutated, as shown in Fig. 2(i) and 2(j).





(i) (j)

Fig. 2: (a) General butterfly, (b,c) 3-lifting steps structure, (d,e,f) Lifting steps with scaling, (i,j) Scaled lifting structure for permutation plane rotation

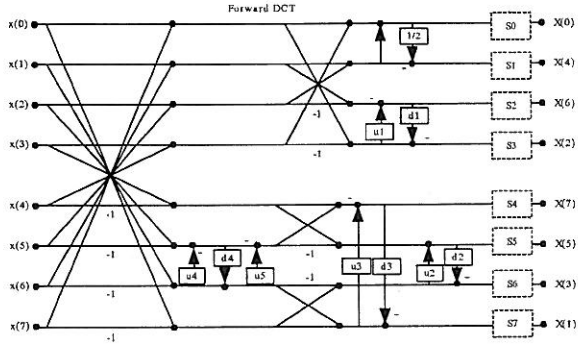


Fig. 3: Structure of a Lifted DCT (Forward Transform) Based on Chen's Factorisation Shown in Fig. 1

In summary, if $\cos^2(\beta) > \sin^2(\beta)$, the original form (Fig. 2(e), 2(f)) is used, and if $\cos^2(\beta) < \sin^2(\beta)$, then the permutation form (Fig. 2(i), 2(j)) is used instead. Applying lifting scheme to the original DCT shown in Fig. 1, the data flow diagram of the lifted forward DCT can be drawn as shown in Fig. 3. Scaling factors S0→S7 are $\frac{Sin\ (\pi\ /\ 4)}{2}$, $\frac{Sin(\pi/4)}{2}$, $\sin(\pi/4)$, $\frac{\sin(3\pi/8)}{2}$, $\frac{1}{2\sin(3\pi/8)}$, $\frac{\sin(7\pi/16)}{2}$, $\frac{\cos(3\pi/16)}{2}$, $\frac{1}{2\cos(3\pi/16)}$, $\frac{1}{2\sin(7\pi/16)}$, respectively.

Each lifting step is a biorthogonal transform. Hence it is reversible. The inverse lifting step is also simple, as what is added in the forward lifting has to be subtracted out in the inverse lifting. In the matrix form the invert lifting matrix can be written as $[M^{-1}]$ where $[M]$ is the forward lifting matrix. Hence,

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \left[M^{-1}\right] \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \qquad (7)$$

For 2-lifting step,

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} -1 & u \\ d & 1-du \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (8a)$$

and

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1+du & u \\ d & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \quad (8b)$$

And for 3-lifting step,

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1-du_2 & u1+u2-u_1u_2d \\ -d & 1-du_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (9a)$$

and

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1-du_1 & -u1-u2+u_1u_2d \\ d & 1-du_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \quad (9b)$$

The invert transform of the 8-point DCT can then be drawn as shown in Fig. 4. S'0→S'7 are the inverts of S0→S7 respectively.

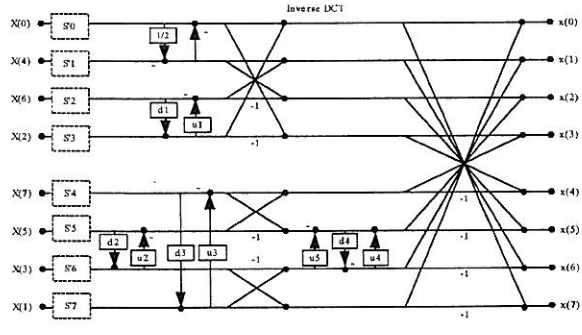


Fig. 4: Structure of a Lifted DCT (Inverse Transform) Based on Chen's Factorisation

As discussed above, using the floating-point number format, computation is still resource intensive. To reduce such a huge workload, with resolution trade-off, the fixed-point system is an alternative choice. The fixed-point system can be implemented using either distributed arithmetic or just a binary shift-and-add approach (Bin DCT). The lifting parameters elaborated in the last section (u and d) can be approximated by rationals of the form $k/2^m$, where k and m are integers. The implementation then involves binary shifts and adds as the name binary DCT suggest. The complexity of the computation of this BinDCT relies upon how close the approximation is made. The bigger the number of 2^m the better the resolutions but the operation requires longer shifts and larger number of bits for storing the

intermediate results. The scaling values appear in the last stage can be omitted. In most cases, there is no a major problem using this type of transform. However, if the compatibility with the true DCT is needed, the scaling relationship has to be studied. A disadvantage of using a fixed-point number representation is the precision loss introduced by the finite word length. We are going to investigate the effects of the number of decimal bits on Mean Square Error (MSE) in the next section. MSE is widely used in measuring the transformation quality. It is defined as

$$MSE = \frac{1}{(N)^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (F_{ij} - G_{ij})^2 \qquad (10)$$

Where F_{ij} and G_{ij} are input and output pixel value respectively.

3.0 A BIT-SERIAL MULTIPLIERLESS DCT

The lifting parameters of the computing topology shown in Fig. 3 and 4 can be approximated to many design sets as given in Liang and Tran (2001). Ours are different. They are particularly based on the small number of shifts and adds as the MSE is taken care of. To minimise the number of delays and to reduce signal latency we used the maximum $2^m = 8$. The resulting design is given in Table 1 below. Number of shift and adds are counted according to the actual operations; for instance 5/8 = 1/2 +1/8 is counted as 3 shifts and 1 add since 1/8 contains 3 shifts and 1/2 contains a single shift. We have 2 approximations, namely, algorithm CA and algorithm CB. Both are limited to a maximum of 3 shifts/lift. The CA contains more shifts. Its lifting approximations are closer to their corresponding floating-point values. The CB on the other hand, comprises less shifts compared to CA. Despite the hardware complexity, both yield a similar performance. With less hardware complexity, the CB holds only 17 shifts and 30 adds in its forward transform scheme.

Table 1: Designed Lifting Parameters

Lifters	Values (Function)	Floating-point	CA	CB
u1	1/tan(3□/8)	0.414213586	3/8	1/2
d1	sin(3□/8)cos(3□/8)	0.353553405	3/8	3/8
u2	tan(3□/16)	0.668178623	5/8	5/8
d2	sin(3□/16)cos(3□/16)	0.461939762	1/2	1/2

u3	1/tan(7□/16)	0.198912392	1/4	1/4
d3	sin(7□/16)cos(7□/16)	0.191341738	1/4	1/4
u4	-(cos(□/4)-1)/sin(□/4)	0.414213545	3/8	1/2
d4	sin(□/4)	0.707106717	3/4	3/4
u5	-(cos(□/4)-1)/sin(□/4)	0.414213545	3/8	1/2

As the number of shifts has been defined in quite a different manner compared to Liang and Tran (2001), our count reflects the latency according to delay insertions. Even of though the fixed-point number representation cannot yield the lossless transformation, we will investigate the number of decimal bits that provide the acceptable resolution for any specific applications. To do this, we examined the number of decimal bits from 0 to 24 bits. This implies the data bus width of 8 to 32 bits. According to our simulation, when the number of decimal bit is greater than 15 the MSE becomes very small (less than 1e-8). We thus ignore the details in the graph illustrated in Fig. 9. We also compared our approximations to C5 and C6 of Liang and Tran (2001). The resulting performances are given in Table 2.

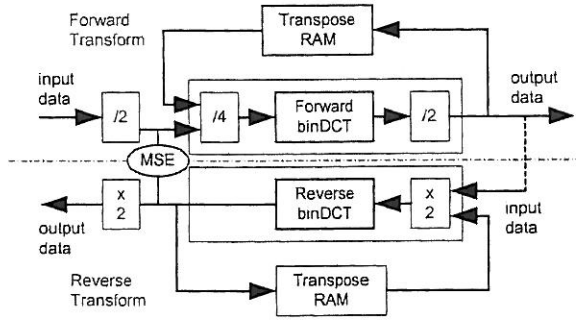


Fig. 5: Data Arrangement for a 2-D DCT

Upon the MSE investigation results, we concentrated on a 16-bit fixed-point 2's compliment number system (8 bits for decimals) in the implementation of this multiplierless DCT. Input signal is assumed to be grey scale, with a pixel value of -128 to +127. We scaled this signal down by 2, before entering the BinDCT module. Before any computation, the signal needs to be further scaled down by 4. This is because the 8-point DCT comprises 3 butterfly states and at each state the magnitude may grow twice. We also scaled the results of the final stage by 2 before feeding them into a transpose RAM. This is to avoid the overflow. The arrangement is demonstrated as shown in Fig. 5. We ignored the scaling values appearing at the end states of the DCT. It is assumed that the quantisation table is to be altered.

appearing at the end states of the DCT. It is assumed that the quantisation table is to be altered.

4.0 A BIT SERIAL ARCHITECTURE

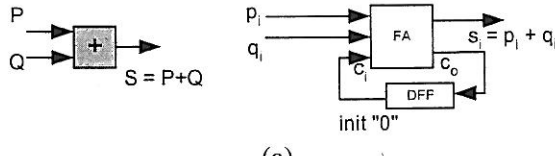
Despite its slow operation, a bit-serial system is fairly simple, has low gate counts and requires low bandwidth. It therefore is very well suited to low power applications where the speed is less crucial.

4.1 A Bit-Serial Adder/Subtractor

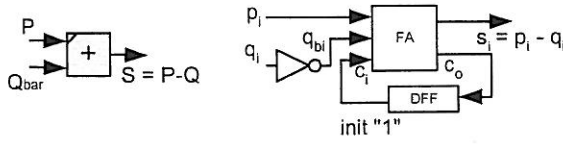
A butterfly as well as a lifting scheme computation requires operations like P+Q and P-Q. A bit-serial adder is shown in Fig. 6(a). The corresponding LSBs of P and Q are summed first, the carry out (if any) is kept in a flip-flop, preparing to add to the sum of the next higher weight bits. The flip-flop should be initialised with "0" since there is no carry before LSBs are added. A single adder comprises a full adder and a delay flip-flop. In terms of computation time, a bit-serial adder requires n clock cycles, where n is the number of bits in a word. Subtraction is also as simple as addition, since $P-Q=P+\overline{Q}+1$ in the 2's complementary number system. An inverter is required to invert Q and the delay flip-flop is initialised with "1". A bit-serial subtractor is shown in Fig. 6(b). It is also easy to combine an adder and a subtractor into a single circuit.

4.2 Scaling Parameters

The algorithm CB holds the following lifters: 3/8, 5/8, 3/4, 1/4 and 1/2, which can be generalised to $kP/2^m$. The computation of $kP/2^m$ is shown in Fig. 7(a) where m can be 2 or 3. The computation of $Q \pm kP/2^m$ or $Q \pm kP/2^m$ for $m \geq 3$ is also fairly simple as shown in Fig. 7(b). For m = 1 and 2 the arrangement can be made simpler as shown in Fig. 7(d).

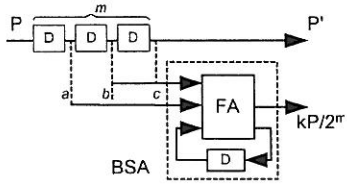


(a)



(b)

Fig. 6: (a) A Bit-Serial Adder (b) A Bit-Serial Subtractor



$kP/2^m$	m	
3P/4	*2*	*a+b*
3P/8	*3*	*b+c*
5P/8	*3*	*a+c*

(a)

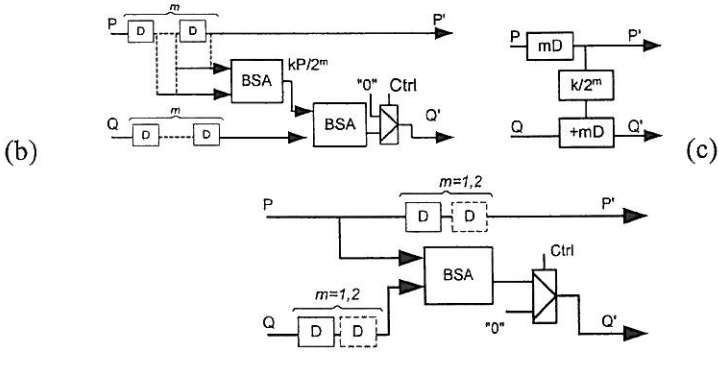


Fig. 7: (a) $kP/2^m$ Scaling Circuit (b) Circuit for Computing $Q' = Q + kP/2^m$ for k=3, 5 and $m \geq 2$ (c) Simplified Diagram of (b); (d) Circuit for k=1 and, m=1 and 2

4.3 Bit Slicing Technique

To smooth the data flow, all bits have to be aligned. As the maximum number of shifts is 3, we have assumed three zeros are to be inserted between each data block. The control signal (Ctrl) shown in Fig. 7(b) have to discard the sum of the first three bits and replace that with zeroes. The simplified version of Fig. 7(b) is shown in Fig. 7(c). In addition, we also have to insert delays in paths that holds no or

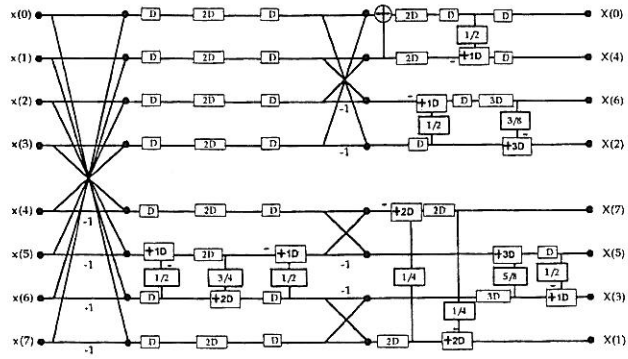


Fig. 8: An Architecture With Bit Slicing Technique

The latency of the above arrangement is 10 clock cycles. Data are flowing in a bit-serial manner, 8 words in parallel. Result is obtained at every clock cycle. As such, the data rate is 8 bits/clock or 16×8 = 128 bits/19 clocks for the useful bits. Regardless of the data transposition, a 8×8 DCT computation takes 38 clock cycles. For the image size of 256×256 pixels, the transformation can take 38×32×32 = 38912 clock cycles or 9.7 ms, if the clock speed of 4 MHz is assumed. Although the rate of the shorter word length (for 12 bits word length: rate = 12×8 = 96 bits/15 clocks) is similar to that of the example above, the system rate can be a little bit better. This can be 30×32×32 = 30720 clock cycles or just 7.7 ms for the same clocking speed.

5.0 RESULTS

Of the arrangement shown in Fig. 5 and Fig. 8, MSE of different algorithms are observed via C programming simulation results. The MSE obtained from different decimal bits length are graphically shown in Fig. 9. When the bus-width is greater than 24 bits, the MSE can be made very small and that may claim a near lossless transformation. The performance of CA and CB are comparable to that given by C5 and C6 of Liang and Tran (2001). Numerical results obtained from the mentioned algorithms are given in Table 2. Compared to others, it is obvious that CB has less complexity: - about 60% of C5. This is confirmed by its number of shifts and adds. The resulting portion of images of different intermediate word lengths are illustrated in Fig. 10. The 12-bit word length can offer a fairly good image quality. Because of its lower hardware complexity compared to the 16-bit system, this can be a good alternative for some applications.

Table 2: Performance of BinDCT with Different Approximations (8 bits decimal)

	C5†	C6†	CA	CB
No. of Shifts	30	24	23	17
No. of Adds	36	33	33	30
MSE*	0.001966	0.001837	0.001800	0.001400
MSE**	0.002373	0.002239	0.002137	0.001589

† Based on Liang and Tran (2001) * Lena 512×512 pixels ** Lena 128×128 pixels

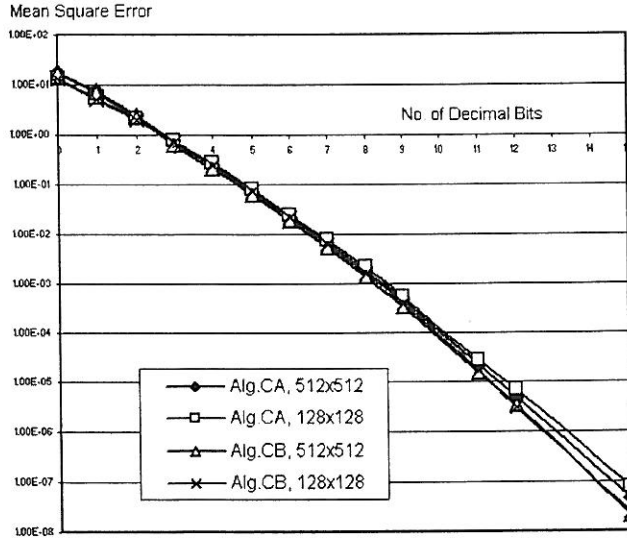


Fig. 9: MSE Performance of Algorithms CA and CB

(a) Original Image

(b) 8-bit Word Length

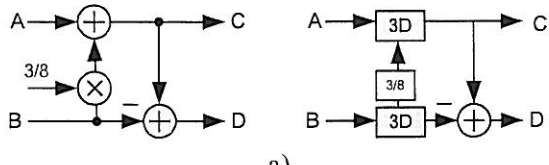
(c) 12-bit Word Length

(d) 16-bit Word

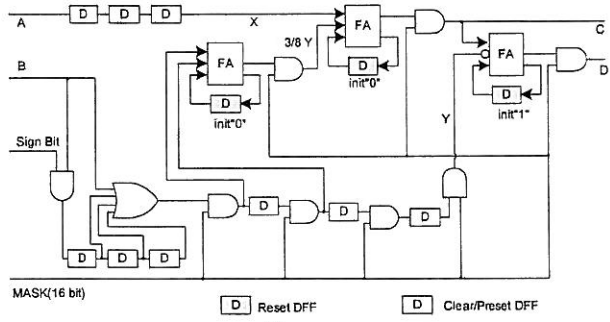
Fig. 10: Images at Different Intermediate Word Length (Algorithm CB, Full image size = 512×512 pixels)

Architecture shown in Fig. 8 was realized using a bit-serial approach of which the actual implementation can be a gate array or full custom design. In our implementation, only the main portion the architecture was verified by using Xilinx foundation kit. As an example case: let A = +31.1796875 and B = -20.875 and we want to compute $C = A + \frac{3}{8}B$ and $D = C - B$. The results C = +23.3515625 and D = +44.2265625 are obtained. Arrangement and simulation results are illustrated in Fig. 11 and Fig. 12 below.

A: +31.1796875 = 00011111.00101110
B: -20.8750000 = 11101011.00100000
C: +23.3515625 = 00010111.01011010
D: +44.2265625 = 00101100.00111010



a)



b)

Fig. 11: Arrangement for addition, scaling and subtraction a) Computing scheme b) Hardware realization

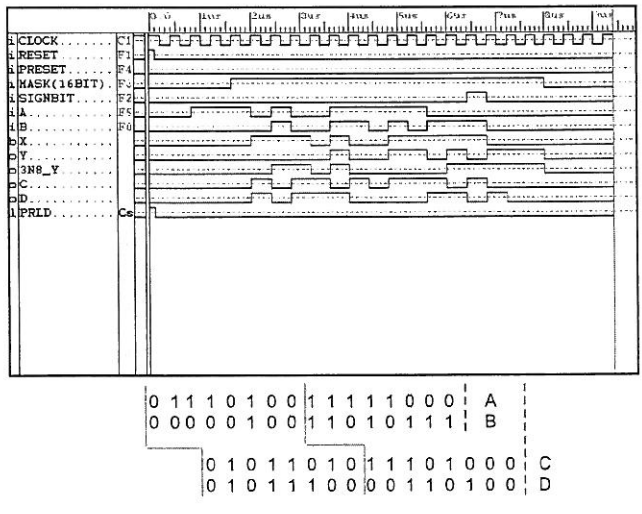


Fig.12: Simulated Waveform of A, B, C and D

6.0 CONCLUSIONS

A BinDCT can lead to a simple implementation since multipliers can be avoided and the lifting schemes can be realised by using binary shift and add operations. It tends to be a lossy transformer of which errors are introduced by both lift parameter approximations and finite word length. To reduce the hardware

complexity or to increase the computation speed, the lifting parameters have to be as simple as x, x/2, x/4 or x/8. Different sets of approximation can affect the value of MSE obtained. For a particular arrangement, a fine- tuning can be made to obtain the best result. Lengthening the word length can of course improve both MSE and data rate, but such an effort leads to the increasing of hardware complexity, signal latency and the whole processing speed.

A bit-serial implementation of the proposed architecture does require minimum hardware. The word length can be easily truncated or extended according to the precision needed. In our case, a 16 bits fixed-point (with 8 decimal bits) system was simulated and designed. Regardless of the complexity of the control circuit which is assumed to be fairly small, about 40 bit-serial adders and about 80 latches are used. With the latency of 10 clock cycles, our design yields the pixel rate of 1.68-bit/clock cycle (pixel rate = 8×8/38). For a fairly low system clock speed, it is possible to meet the specification of low bit rate video (H.261 recommendation), i.e., n × 64 kbit/sec. For example: n = 30, the system has to run at the clock speed of $(30\times64\times10^3)\times\frac{38}{64}=1.14$ MHz.

REFERENCES

Chen, W. H., Smith, C. H., & Fralick, S. C. (1977, September). A Fast Computational Algorithm for the Discrete Cosine Transform. *IEEE Trans. Commun.*. Vol. COM-25, pp. 1004-1009.

Committee Draft of Standard ISO 1172 (1990, December). Coding of moving pictures and associated audio. *ISO/MPEG 90/176.*

Fujiwara, H., Liou, M. L., Sun, M. T., Yang, K. M., Shoumura, K., & Ohyama, K. (1992, June). An All ASIC Implementation of a Low Bit-Rate Video Codec. *IEEE Trans. Circuit Syst. Video Tech.* Vol. 2, No. 2, pp. 123-134.

Hartang, F., & Girod, B. (1998, May). Watermarking of Uncompressed and Compressed Video. *Signal Processing,* Vol. 66, No. 3, pp. 283-301.

ITU-T recommendation T-81 (1992, September). *Digital Compression of Continuous Tone and Still Images.*

Liang, J., & Tran, T. D. (2001, December). Fast Multiplierless Approximations of the DCT With the Lifting Scheme. *IEEE Trans. Signal Processing.* Vol. 49, No. 12, pp. 3032-3044.

Peled, A. & Lui, B. (1974, December). A New Hardware Realization of Digital Filters. *IEEE Trans. Acoust. Speech and Signal Processing*, Vol. ASSP-22, No. 6, pp. 456-462.

Pereira, S., Voloshynovskiy, S., & Pun, T. (2000). Effective channel coding for DCT watermark. *Proc. of 2000 International Conference on Image Processing,* Vol. 3, pp. 671-673.

Raymond W. & Borko F. (1997). *Real-Time Video Compression; Techniques and Algorithms.* Kluwer Academic Publishers.

Suhail, M.A., & Obaidat, M.S. (2001). On the digital watermarking in JPEG 2000. *Proc. of The 8th IEEE International Conference on Electronics, Circuits and Systems (ICECS).* Vol. 2, pp. 871-874.

Sun, M. T., Chen, T. C., & Gottlieb, A. M. (1989, April). VLSI Implementation of a 16×16 Discrete Cosine Transform. *IEEE Trans. Circuit Syst.*. Vol. 36, No. 4, pp. 610-617.

Tran, T. D. (2000, June). The BinDCT: Fast Multiplierless Approximation of the DCT. *IEEE Signal Processing Letters.* Vol. 7, No. 6, pp. 141-144.