BinDCT and Its Efficient VLSI Architectures for Real-Time Embedded Applications

Philip P. Dang[†]
STMicroelectronics Inc., San Diego, California, USA

Paul M. Chau and Truong Q. Nguyen
ECE Department, University of California, San Diego, La Jolla, California, USA

Trac D. Tran
ECE Department, John Hopkins University, Baltimore, Maryland, USA

In this article, we present the BinDCT algorithm, a fast approximation of the Discrete Cosine Transform, and its efficient VLSI architectures for hardware implementations. The design objective is to meet the real-time constrain in embedded systems. Two VLSI architectures are proposed. The first architecture is targeted for low complexity applications such as videophones, digital cameras, and digital camcorders. The second architecture is designed for high perform applications, which include high definition TV and digital cinema. In order to meet the real-time constrain for these applications, we decompose the structure of the BinDCT algorithm into simple matrices and map them into multi-stage pipeline architectures. For low complexity implementation, the proposed 2-D BinDCT architecture can be realized with the cost of 10 integer adders, 80 registers and 384 bytes of embedded memory. The high performance architecture can be implemented with an extra of 30 adders. These designs can calculate real-time DCT/IDCT for video applications of CIF format at 5 MHz clock rate with 1.55 volt power supply. With its high performance and low power consumption features, BinDCT coprocessor is an excellent candidate for real-time DCT-based image and video processing applications.

Journal of Imaging Science and Technology 49: 124–137 (2005)

Introduction

The Discrete Cosine Transform (DCT) is one of the most important transformations in signal processing. The DCT and its inverse transform, IDCT, are the core operations in many audio, image and video coding systems. The DCT and IDCT were adopted by several international standards included Dolby AC-3, JPEG, H.261, H.263, MPEG-1, MPEG-2, and MPEG-4.

Both DCT and IDCT are intensive computational processes. Software implementation of DCT/IDCT cannot meet real-time constraint in video applications. For mobile multimedia communications, there are several other issues related to the implementation of DCT. Mobile multimedia devices such as digital cameras, videophones, and pocket PCs have limited memory, CPU and power resources. There is a need to investigate and develop efficient DCT/IDCT coprocessors for these applications. In this article, we present the BinDCT, a multiplierless approximation of the DCT, and its efficient VLSI architectures for multimedia applications.[12] The proposed architectures provide efficient computing power for real-time processing. They, however, cost a small silicon area and require very low power consumption.

DCT Background

DCT Algorithm

Discrete Cosine Transform is an orthogonal transformation. It was first introduced by Ahmed, Natarajan, and Rao in 1974.[1] The first DCT algorithm was proposed for pattern recognition and Wiener filtering. DCT and IDCT, however, are most widely used for image and video compression due to their energy compaction property, separable transform and near optimal performance that is close to the Karhunen–Loeve transformation (KLT). The mathematical definitions of DCT and IDCT are as follow:

Original manuscript received June 21, 2004

†Corresponding Author: Philip P. Dang, philip.dang@st.com

DCT:

$$Y(m) = \frac{s(m)}{2} \sum_{n=0}^{N-1} x(n) \cos\left(\frac{(2n+1)m\pi}{2N}\right) \quad (1)$$

IDCT:

$$x(n) = \sum_{m=0}^{N-1} \frac{s(m)}{2} Y(m) \cos \frac{(2n+1)m\pi}{2N} \quad (2)$$

where
$x(n)$ are input signals in the spatial domain
$Y(m)$ are DCT coefficients in frequency domain.

$$s(m) = \begin{cases} \sqrt{\frac{1}{N}} & m = 0 \\ \sqrt{\frac{2}{N}} & m \neq 0 \end{cases} \quad (3)$$

and m = 0, 1, 2, …., N – 1. DCT can be represented in term of matrix multiplication. The 8-point DCT derived from Eq. (1) for N = 8 can be written in the matrix form as follow:

$$\begin{bmatrix} Y[0] \\ Y[1] \\ Y[2] \\ Y[3] \\ Y[4] \\ Y[5] \\ Y[6] \\ Y[7] \end{bmatrix} = \begin{bmatrix} c_4 & c_4 & c_4 & c_4 & c_4 & c_4 & c_4 & c_4 \\ c_1 & c_3 & c_5 & c_7 & -c_7 & -c_5 & -c_3 & -c_1 \\ c_2 & c_6 & -c_6 & -c_2 & -c_2 & -c_6 & c_6 & c_2 \\ c_3 & -c_7 & -c_1 & -c_5 & c_5 & c_1 & c_7 & -c_3 \\ c_4 & -c_4 & -c_4 & c_4 & c_4 & -c_4 & -c_4 & c_4 \\ c_5 & -c_1 & c_7 & c_3 & -c_3 & -c_7 & c_1 & -c_5 \\ c_6 & -c_2 & c_2 & -c_6 & c_6 & c_2 & -c_2 & -c_6 \\ c_7 & -c_5 & c_3 & -c_1 & -c_1 & c_3 & c_5 & c_7 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix} \quad (4)$$

where

$$c_k = \cos \frac{k\pi}{16} \quad (5)$$

From the matrix representation, one can realize that the direct implementation of the DCT will cost 64 multiplications and 56 additions. In their landmark study,[1] Ahmed, Natarajan, and Rao presented an N-point DCT that can be computed with a 2N-point FFT and some additional post-processing. Using even/odd decomposition, Ahmed, Natarajan, and Rao reduced the complexity of the 8 × 8 DCT matrix to 22 multiplications and 28 additions.[2]

Fast DCT Algorithms

During the last two decades, many fast DCT/IDCT algorithms have been proposed since DCT and IDCT become an essential component in numerous international standards. These works included Chen et al,[3] Wang,[4] Lee,[5] Vetterli and Nussbaunmer,[6] Suehiro and Hatori,[7] Hou,[8] Malvar,[9] and Loeffler et al.[10] The common objective of these algorithms is to reduce the number of multiplications in DCT algorithm. The basic difference among them is the way to factor the DCT matrix into simple matrices. The number of additions in these algorithms is 29. The number of multiplications is between 11 and 16.

For lossy coding applications, Arai, Agui, and Kakajima[11] showed that eight multiplications can be absorbed by quantization operations, and the computation for the scaled 8 × 8 DCT can be reduced to five multiplications and 29 additions. The computational complexity of fast 8-point 1-D DCT algorithms is summarized in the Table I.

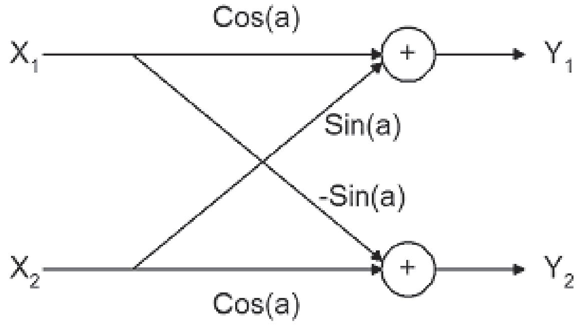


Figure 1. Plane rotation structure.

BinDCT Algorithm

Motivation

One of the disadvantages of the aforementioned fast DCT algorithms is that they still need floating point multiplications. These operations are slow in software and cost more silicon for implementation in hardware. In addition, long execution time of floating point operations also leads to more power consumption. Most of mobile multimedia devices, on the other hand, have limited resources of power, CPU and memory. There is a need to investigate and develop efficient, low complexity orthogonal transformation for these applications.

Approach

Using the lifting-based approach, Tran[12] showed that fast and accurate approximations of DCT can be obtained without using any multiplication. The proposed algorithm is called BinDCT. The basic idea of the BinDCT algorithm is that each plane rotation (Fig. 1) in the fast DCT structure, such as in Refs. [3] and [10], can be approximated by at most three dyadic lifting steps. In their article, Liang and Tran[13] showed that both forward and inverse DCT can be implemented using only shift and addition operations.

The decomposition of a plane rotation into three lifting steps is illustrated in Fig. 2(a). This operation can be represented in the matrix form

TABLE I. Computation Complexity of 8-point 1-D DCT

Algorithm	Chen[3]	Wang[4]	Lee[5]	Vetterli[6]	Suechiro[7]	Hou[8]	LLN[10]	AAN[11]	BinDCT [12]
Multiplication	16	13	12	12	12	12	11	5	0
Addition	26	29	29	29	29	29	29	29	30
Shift	0	0	0	0	0	0	0	0	13

$$\begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} = \begin{bmatrix} 1 & p \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ u & 1 \end{bmatrix}\begin{bmatrix} 1 & p \\ 0 & 1 \end{bmatrix} \tag{6}$$

where

$$p = \frac{\cos\alpha - 1}{\sin\alpha}, \text{ and } u = \sin\alpha.$$

Note that each lifting step is a biorthogonal transform, and its inverse also has a simple lifting structure, i.e.,

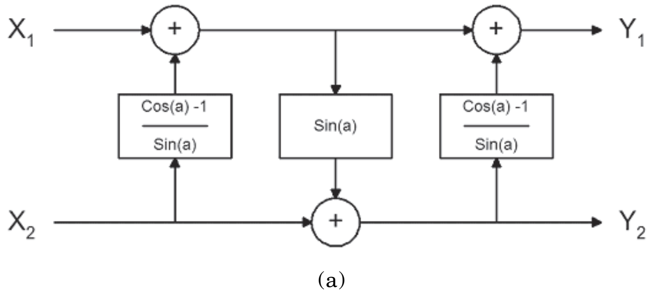
$$\begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -x \\ 0 & 1 \end{bmatrix} \tag{7}$$

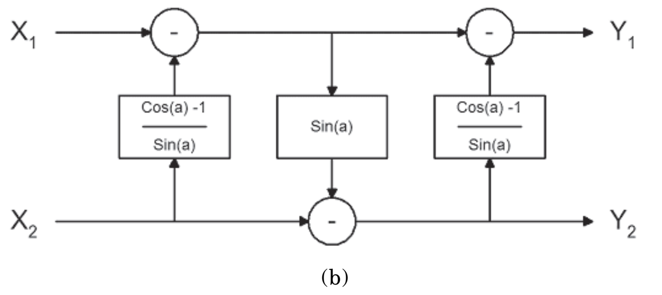
$$\begin{bmatrix} 1 & 0 \\ x & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 \\ -x & 1 \end{bmatrix} \tag{8}$$

As a result, the inverse of the plane rotation can also be represented by lifting step. Its matrix representation is:

$$\begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -p \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ -u & 1 \end{bmatrix}\begin{bmatrix} 1 & -p \\ 0 & 1 \end{bmatrix} \tag{9}$$

The structure of the inverse plane rotation using three lifting steps is shown in Fig. 2(b). To invert a lifting step we simply subtract out what was added in at the forward transform. Hence, the original signal can be perfectly reconstructed even if the floating point multiplication results in the lifting steps are rounded to integers, as long as the same procedure is applied to both the forward and inverse transform.





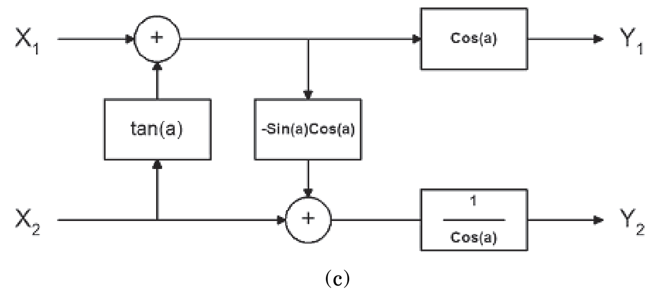


Figure 2. (a) and (b) An implementation of forward and reverse plane rotations using three lifting steps; (c) scaled lifting structure of the plane rotation.

$$BinDCT = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & 3/16 & 0 & 0 & -3/16 & -1/2 & -1/2 \\ 55/128 & 3/16 & -3/16 & -55/128 & -55/128 & -3/16 & 3/16 & 55/128 \\ 9/32 & -1/8 & -19/64 & -1/4 & 1/4 & 19/64 & 1/8 & -9/32 \\ 1/4 & -1/4 & -1/4 & 1/4 & 1/4 & -1/4 & -1/4 & 1/4 \\ 7/16 & -3/4 & 7/32 & 1/2 & -1/2 & -7/32 & 3/4 & -7/16 \\ -3/16 & 1/2 & -1/2 & 3/16 & 3/16 & -1/2 & 1/2 & -3/16 \\ -1/16 & 1/4 & -13/32 & 1/2 & -1/2 & 13/32 & -1/4 & 1/16 \end{bmatrix} \tag{10}$$

The plane rotation structure can also be implemented by the scaled lifting structure (Fig. 2(c)) for further reduce complexity. In this approach, rotation angles can be approximated with only two dyadic lifting steps, which can be implemented using shift-and-add binary operations.

BinDCT Algorithm

Figure 3(a) depicts the Chen's factorization of an 8-point forward DCT. In this picture,

$$s_n = \sin\left(\frac{n\pi}{16}\right) \text{ and } c_n = \cos\left(\frac{n\pi}{16}\right).$$

Chen's factorization can be realized into five stages. Using the lifting steps and scaled lifting structures, we derive a general lifting structure of the BinDCT family based on the Chen's factorization (Fig. 3(b)). In this design, the scaled lifting structure is utilized to reduce the computational complexity since most of scaled lifted factors can be incorporated into the quantization stage.

By varying the approximation accuracy, different versions of the BinDCT can be obtained.[12,13] Equation (10) is the version C of the forward BinDCT algorithm. Its lifting parameters and scaled factors are shown in Fig. 3(c). This particular implementation requires 30 additions and 13 shift operations. Reader can verify that

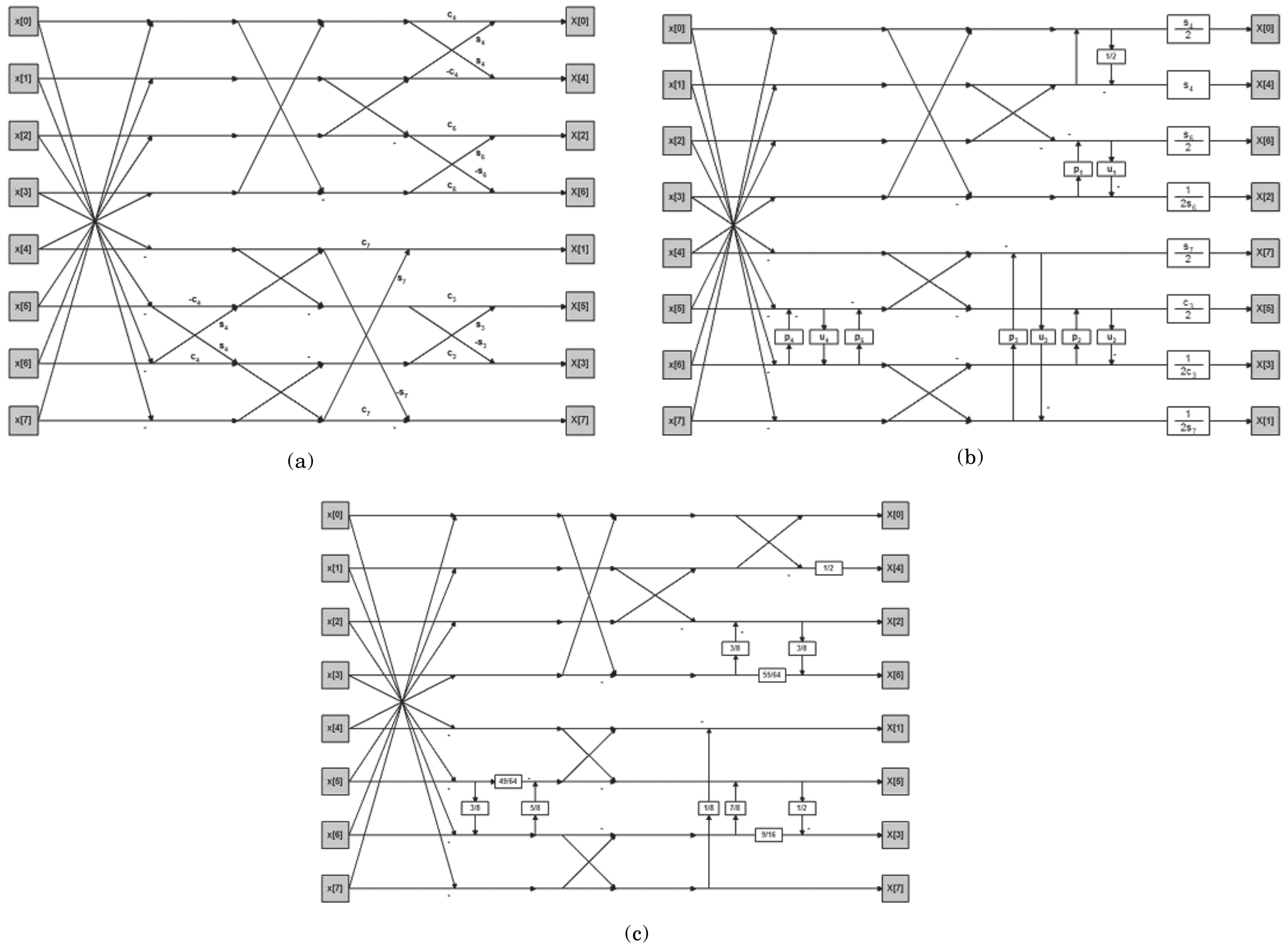


Figure 3. (a) Flow chart of Chen's facgtorization of 1-D 8-point forward DCT; (b) flow chart of the general lifting structure for 1-D 8-point forward DCT using Chen's factorization; (c) flow chart of 1-D 8-point forward BinDCT.

the inverse BinDCT has the same complexity and it is also multiplierless.

BinDCT Performance

Performance of BinDCT in JPEG

Using the source code from the Independent JPEG Group (IJG),[1] 8-point BinDCT algorithms have been implemented according to the framework of the JPEG standard.[2] Three versions of DCT implementation are provided in the IJG's code. The floating version is based on the Arai's scaled DCT algorithm with five floating multiplications and 29 additions.[11] The slow integer version is a variation of the Loeffler's algorithm[10] with 12 fixed-point multiplications and 32 additions, and the fast integer version is the Arai's algorithm[11] with five fixed-point multiplications. To measure the performance of the BinDCT, the DCT part in JPEG codec is replaced by the BinDCT algorithm, and the JPEG quantization matrix is modified to incorporate the 2-D BinDCT scaling factors. The Peak Signal-to-Noise Ratio (PSNR) between the reconstructed signal $s'_{x,y}$ and the original signal $s_{x,y}$ for each simulation was calculated for performance analysis. The PSNR is defined as

$$PSNR = 10\log\frac{255^2}{MSE}\,dB,$$

where

$$MSE = \frac{\sum\limits_{x,y\in A}\left|s_{x,y} - s'_{x,y}\right|^2}{A},$$

and A is the total number of pixels in the image.

Table II compares the compatibility of different fast DCT algorithms with respect to the floating DCT, for which the image Lena is compressed with the floating DCT and decompressed with different fast inverses. It can be seen that the differences among the BinDCT-C4, BinDCT-L3, and the IJG's fast integer DCT are negligible in most cases. However, the BinDCT-L3 has better performance when the quantization step is very small as the scaling factors of the BinDCT-L have smaller round-off errors than the BinDCT-C.

In addition, the PSNR results of the reconstructed Lena image with IJG's floating DCT, IJG's fast integer DCT, and the BinDCT-C4 are compared. Experimental results show that the performance of the BinDCT is very close to that of the floating DCT in most cases. In particular, when the quality factor is below 95, the difference between the BinDCT-C4 and the floating DCT is less than 0.1 dB. Experiments also show that even the degradation of the BinDCT-C7 is less than 0.5 dB. When the quality factor is above 90, the degradations of both fast DCTs become obvious due to the round-off errors introduced by the scaling factors. However, the result of the BinDCT is still reasonable. For example,

TABLE II. PSNR (dB) of the Reconstructed Image with Different Inverse DCT Algorithms

Quality Factor	IJG Integer DCT	IJG Fast Integer DCT	BinDCT C4	BinDCT L3
100	58.85	45.02	44.38	50.12
90	40.79	40.53	40.52	40.66
80	38.51	38.39	38.39	38.44
60	36.43	36.36	36.38	36.38
40	35.11	35.06	35.06	35.06
20	32.95	32.92	32.92	33.31
10	30.40	30.39	30.39	30.37
5	27.33	27.32	27.32	27.30

when the quality factor is 100, the BinDCT result is 10.3 dB better than that of the IJG's fastest integer DCT. In terms of the compression ratio, the compressed file size with BinDCT-C7 is about 1–3% smaller than that with the floating DCT, whereas the compressed size with BinDCT-C4 is slightly larger than the latter, but the difference is less than 0.5% in most cases.

Performance of BinDCT in H.263+

The BinDCT has also been implemented in the video coding standard H.263+, based on a public domain H.263+ software.[14] The DCT in the encoder of the selected H.263+ implementation is based on Chen's factorization with floating point multiplications, and the DCT in the decoder is the scaled version of this method with fixed-point multiplications. In H.263+, a uniform quantization step is applied to all the DCT coefficients of a block. In the BinDCT-based version, the quantization step is modified by the 2-D BinDCT scaling matrix to maintain compatibility with the standard.

In this section, four scenarios of the configuration of the encoder and the decoder are compared with the default quantization steps (40 for I frames and 26 for P frames). The average PSNRs of the reference H.263+ implementation is 30.55 dB. If the BinDCT-C4 is used in both the encoder and the decoder, the average PSNR drops to 30.46 dB. However, the compression ratio is improved to 102.67:1 from 101.03:1. If the floating DCT is used by the encoder and the BinDCT-C4 is used in the decoder, the average PSNR is 30.43 dB. On the contrary, when the sequence is encoded by the BinDCT-C4 and decoded by the default DCT, the average PSNR is 30.39 dB. These results show that the compatibility of the BinDCT with other DCT implementations is satisfactory and the overall performance of the BinDCT-C4-based H.263+ is very similar to the reference H.263+.

VLSI Architectures of BinDCT

In the previous section, we showed that the overall performance of the BinDCT is compatible to the reference DCT algorithms in the JPEG and H.263+ standards. The BinDCT is expected to have tremendous advantage in low-cost hardware implementation in terms of size, speed, and power consumption. All of these criteria are critical considerations for many hand-held devices. In this section, we present two different VLSI architectures for the BinDCT algorithm. The first architecture is designed for low power consumption applications. This architecture can be applied for mobile multimedia devices such as digital cameras, digital camcorders, pocket PCs, and videophones. The second architecture is designed for high performance applications in which the power constrain can be relaxed. Applications include MPEG decoders for digital cinema and HDTV.

Design Approach

For efficient VLSI implementation of the BinDCT algorithm, we first decompose the basic BinDCT structure in Eq. (10) into five simple matrices. Each matrix is designed as a single stage in the proposed pipeline architecture. The matrix factorization form of the BinDCT algorithm is shown in Eq. (11).

$$BinDCT = \frac{1}{2} E * D * C * B * A \tag{11}$$

where A, B, C, D, and E are defined as follows:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}, \tag{12}$$

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -49/64 & 5/8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3/8 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{13}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \tag{14}$$

$$D = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & -1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -3/8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3/8 & 55/64 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1/8 \\ 0 & 0 & 0 & 0 & 0 & 1 & 7/8 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1/2 & 9/16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (15)$$

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \quad (16)$$

The corresponding flow chart of Eq. (11) is depicted in Fig. 3(c).

In order to achieve a high system throughput, the BinDCT architecture is organized as a linear multi-stage pipeline as illustrated in Fig. 4. Each matrix A, B, C, D, and E is associated with a stage in the pipeline architecture. Stages 1 through 4 computes the forward BinDCT transformation. Stage 5 reorders output data. Let x_0–x_7 represent 8 inputs. The proposed pipeline architecture for the BinDCT algorithm is as follows:

Stage 1:
$a_0 = x_0 + x_7$; $a_1 = x_1 + x_6$; $a_2 = x_2 + x_5$; $a_3 = x_3 + x_4$;
$a_4 = x_3 - x_4$; $a_5 = x_2 - x_5$; $a_6 = x_1 - x_6$; $a_7 = x_0 - x_7$.

Stage 2:
$b_0 = a_0$; $b_1 = a_1$; $b_2 = a_2$; $b_3 = a_3$; $b_4 = a_4$; $b_7 = a_7$;

$$b_5 = \frac{5}{8}a_6 - \frac{49}{64}a_5; \quad b_6 = \frac{3}{8}a_5 + a_6.$$

Stage 3:
$c_0 = b_0 + b_3$; $c_1 = b_1 + b_2$; $c_2 = b_1 + b_2$; $c_3 = b_0 - b_3$;
$c_4 = b_4 + b_5$; $c_5 = b_4 - b_5$; $c_6 = b_7 - b_6$; $c_7 = b_6 + b_7$.

Stage 4:

$$d_0 = c_0 + c_1; d_1 = \frac{(c_0 - c_1)}{2}; \; d_2 = c_2 - \frac{3c_3}{8}; \; d_3 = \frac{3c_2}{8} + \frac{55c_3}{64};$$

$$d_4 = c_4 - \frac{c_7}{8}; \; d_5 = c_5 + \frac{7c_6}{8}; \; d_6 = \frac{9c_6}{16} - \frac{c_5}{2}; \; d_7 = c_7.$$

Stage 5:
$e_0 = d_0$; $e_1 = d_7$; $e_2 = d_3$; $e_3 = d_6$;
$b_4 = d_2$; $e_5 = d_5$; $e_6 = d_2$; $e_7 = d_5$.

Low Complexity VLSI Architecture of BinDCT

For low power design, each stage from 1 to 3 is implemented with one adder. The computation in each stage takes 8 clock cycles. To balance the latency in all five stages, two adders are used in Stage 4. Stage 5, on the other hand, does not include any arithmetic operation. The low power VLSI architecture of the 8-point 1-D BinDCT, therefore, can be realized with 5 adders and 40 registers.

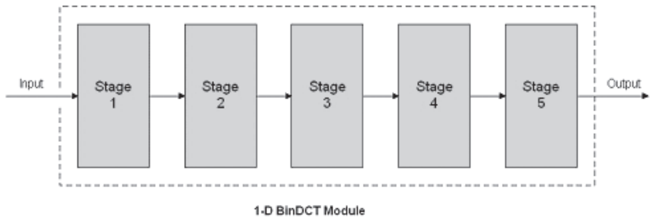


Figure 4. Five stages pipeline of 1-D BinDCT coprocessor.

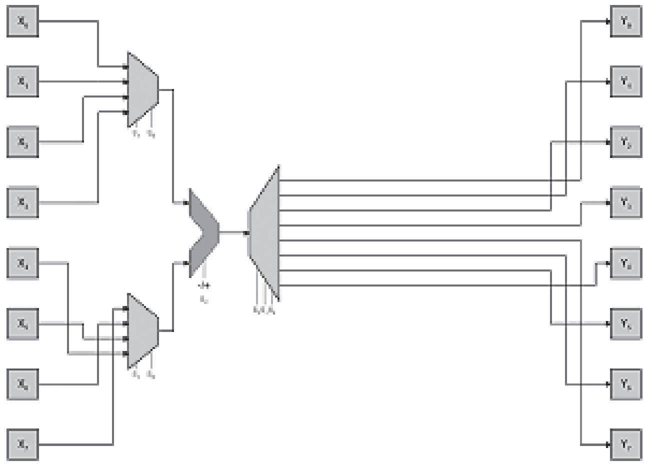


Figure 5. The architecture of Stage 1.

In the following sections, we discuss in detail the architecture of Stages 1 through 4. In general, the architecture of each pipeline stage contains two sets of registers and dedicated arithmetic units to perform the matrix multiplications. Each stage receives eight inputs and generates 8 outputs. The control signals are set locally. Since all parameters in five matrices A, B, C, D, and E are power of two numbers, the matrix multiplication can be computed by shift-add operations. In this implementation, the addition and subtraction operations are calculated in 2's complement arithmetic.

Architecture of Stage 1

Stage 1 is associated with the matrix A in Eq. (12). The architecture of Stage 1 includes two 4-to-1 multiplexers, one 8-bit integer adder, and one 1-to-8 demultiplexer. The first four inputs X_0–X_3 are connected to the first multiplexer, whereas inputs X_4–X_7 are connected to the second multiplexer. Each multiplexer has two control signals, namely, S_0 and S_1. Output of each multiplexer is tied with the inputs of the adder. The operations of the adder are controlled by the signal S_2. In the first four clock cycles, adder performs addition operations. In the last four clock cycles, subtraction operations are computed. Output from the adder is fed into the 1-to-8 demultiplexer, which distributes results to output registers in every clock cycle. The demultiplexer is controlled by signals S_0, S_1, and S_2. The truth table for Stage 1 is shown in the Table III. The architecture of Stage 1 is illustrated in Fig. 5. The synthesize circuit of Stage 1 is shown in Fig. 10.

TABLE III. The Truth Table of Stage 1

S_2	S_1	S_0	Function
0	0	0	$Y_0 = X_0 + X_7$
0	0	1	$Y_1 = X_1 + X_6$
0	1	0	$Y_2 = X_2 + X_5$
0	1	1	$Y_3 = X_3 + X_4$
1	0	0	$Y_4 = X_0 - X_7$
1	0	1	$Y_5 = X_1 - X_6$
1	1	0	$Y_6 = X_2 - X_5$
1	1	1	$Y_7 = X_3 - X_4$

TABLE IV. The Truth Table of Stage 2

S_2	S_1	S_0	Function
0	0	0	$T_0 = (1/4)\ X_5 + (1/64)\ X_5 = (17/64)\ X_5$
0	0	1	$T_1 = (17/64)\ X_5 + (32/64)\ X_5 = (49/64)\ X_5$
0	1	0	$T_2 = (1/4)\ X_5 + (1/8)\ X_5 = (3/8)\ X_5$
0	1	1	$T_3 = (1/2)\ X_6 + (1/8)\ X_6 = (5/8)\ X_6$
1	0	0	$Y_5 = (5/8)\ X_6 - (49/64)\ X_5$
1	0	1	$Y_6 = X_6 - (3/8)\ X_5$
1	1	0	–
1	1	1	–

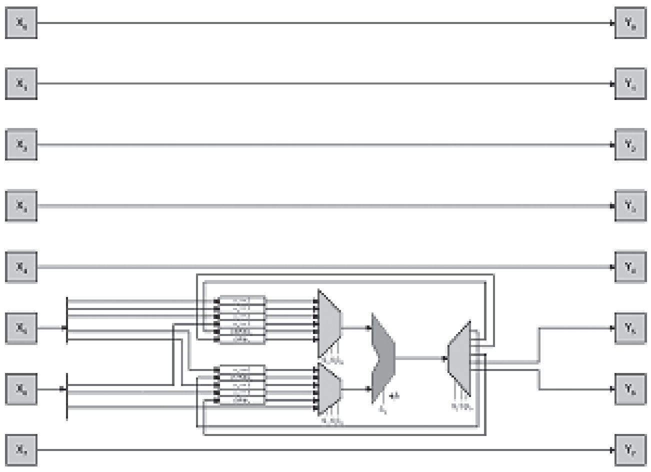


Figure 6. The architecture of Stage 2.

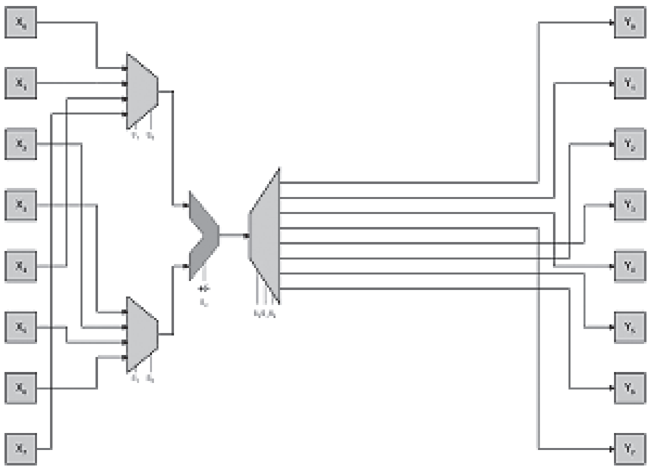


Figure 7. The architecture of Stage 3.

Architecture of Stage 2
Stage 2 performs the computations for the matrix B. The architecture of Stage 2 is similar to the architecture of Stage 1. Stage 2 includes two 8-to-1 multiplexers, one 9-bit integer adder, one 1-to-8 demultiplexer, and an internal register bank. Inputs X_0–X_4 and X_7 are connected directly to output registers. Inputs X_5 and X_6 are shifted with predefined values and are stored in the internal register bank. The multiplexers select values from the register bank and feed them into the inputs of the adder. The demultiplexer distributes output from the adder to the output registers Y_5 and Y_6 or send them back to the register file. The truth table of Stage 2 is summarized in Table IV. Note that, Stage 2 takes only 6 cycles to calculate Y_5 and Y_6. The last two cycles are don't care. The architecture of Stage 2 is illustrated in Fig. 6. The synthesize circuit of Stage 2 is shown in Fig. 11.

Architecture of Stage 3
Stage 3 performs the computations for the matrix C. The architecture of Stage 3 is almost identical with the architecture of Stage 1. The difference between the architectures of two stages is that in Stage 3 inputs X_0, X_1, X_4, and X_7 are connected to the first multiplexer. Inputs X_2, X_3, X_5, and X_6, on the other hand, are connected to the second multiplexer. The truth tables for Stage 3 are shown in the Table V. The architecture of Stage 3 is illustrated in Fig. 7. The synthesize circuit of Stage 3 is shown in Fig. 12.

Architecture of Stage 4
In order to complete the computations for Stage 4 within 8 clock cycles, Stage 4 needs two adders. Each adder connects to two 4-to-1 multiplexers and one 1-to-8 demultiplexer. The first adder computes the computations for the upper part of the matrix D, while the second adder computes the computations for the lower part. The architecture of Stage 4 is depicted in Fig. 8. The truth tables of the upper part and the lower part are summarized in the Table VI and Table VII, respectively. The synthesize circuit of Stage 4 is shown in Fig. 13.

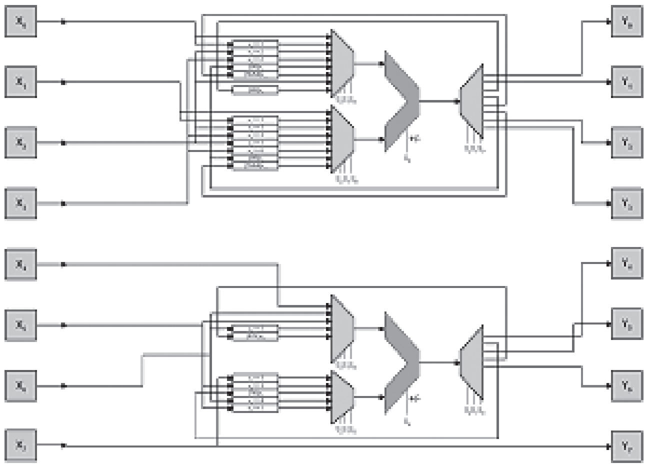


Figure 8. The architecture of Stage 4.

Architecture of Stage 5
Stage 5 is the last stage in the 1-D BinDCT pipeline architecture. It associates with the matrix E in Eq. (16).

TABLE V. The Truth Table of Stage 3

S_2	S_1	S_0	Function
0	0	0	$Y_0 = X_0 + X_3$
0	0	1	$Y_1 = X_1 + X_2$
0	1	0	$Y_4 = X_4 + X_5$
0	1	1	$Y_7 = X_6 + X_7$
1	0	0	$Y_3 = X_0 - X_3$
1	0	1	$Y_2 = X_1 - X_2$
1	1	0	$Y_5 = X_4 - X_5$
1	1	1	$Y_6 = X_7 - X_6$

TABLE VI. The Truth Table of the Upper Part of Stage 4

S_3	S_2	S_1	S_0	Function
0	0	0	0	$Y_0 = X_0 + X_1$
1	0	0	1	$Y_1 = (1/2)\ X_0 - (1/2)\ X_2$
0	0	1	0	$U_0 = (1/4)\ X_2 + (1/8)\ X_2$
0	0	1	1	$U_1 = (1/4)\ X_3 + (1/8)\ X_3$
1	1	0	0	$U_2 = X_3 - (1/8)\ X_3 = (7/8)\ X_3$
1	1	0	1	$U_3 = (7/8)\ X_3 - (1/64)\ X_3 = (55/64)\ X_3$
1	1	1	0	$Y_2 = X_2 - (3/8)\ X_3$
0	1	1	1	$Y_3 = (3/8)\ X_2 + (55/64) X_3$

TABLE VII. The Truth Table of the Lower Part of Stage 4

S_4	S_2	S_1	S_0	Function
1	0	0	0	$Y_4 = X_4 - (1/8)\ X_7$
1	0	0	1	$L_0 = X_6 - (1/8)\ X_6 = (7/8)\ X_6$
0	0	1	0	$Y_5 = X_5 + (7/8)\ X_6$
0	0	1	1	$L_1 = (1/4)\ X_3 + (1/8)\ X_3$
1	1	0	0	$Y_6 = (9/16)\ X_6 - (1/2)\ X_5$
0	1	0	1	–
0	1	1	0	–
0	1	1	1	–

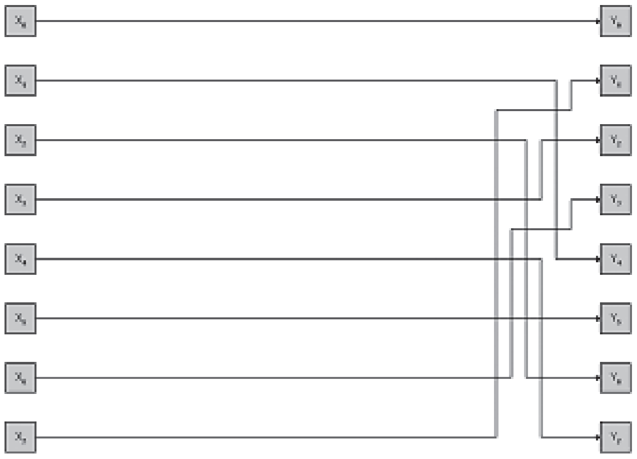


Figure 9. The architecture of Stage 5.

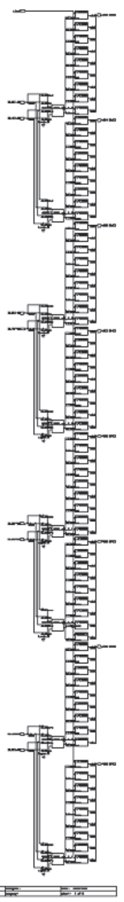


Figure 10. The synthesize circuit of Stage 1.

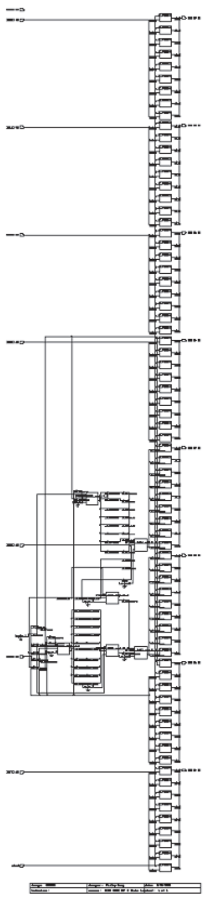


Figure 11. The synthesize circuit of Stage 2.

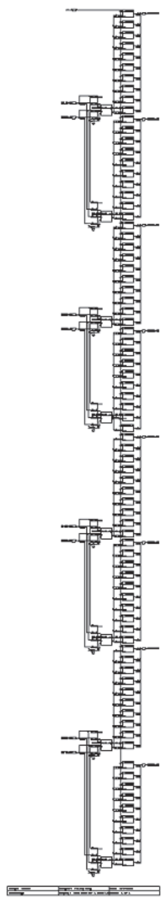


Figure 12. The synthesize circuit of Stage 3.

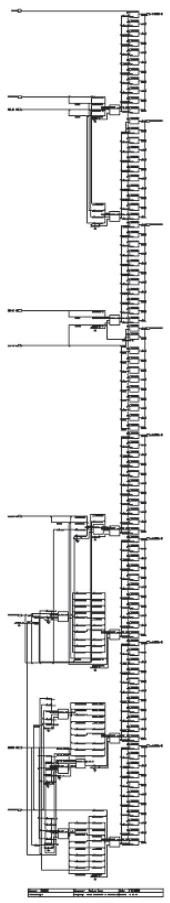


Figure 13. The synthesize circuit of Stage 4.

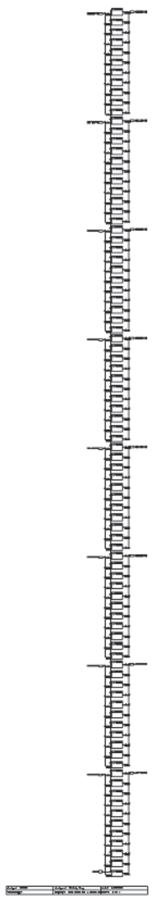


Figure 14. The synthesize circuit of Stage 5.

The basic function of Stage 5 is to re-organized output data before they are stored into buffer or memory. This matrix can be easily realized in hardware by appropriately hardwiring input registers to output registers. The architecture of Stage 5 is illustrated in Fig. 9. The synthesize circuit of Stage 5 is illustrated in Fig. 14.

Transpose Buffer

To compute the separable 2-D BinDCT, the outputs from the row decomposition are stored in a buffer. This buffer contains a 8 × 8 block of 64 12-bit coefficients. The transpose operations are managed by a control unit, which allows inputs to be written in row-wide and outputs to be read in column-wise.

To improve the throughput of the 2-D BinDCT calculation, two 8 × 8 buffers are used. When the second 1-D BinDCT circuitry reads data from the first buffer, the outputs from the first 1-D BinDCT of the next 8 × 8 block are written into the second buffer. After 64 clock cycles, the address of output port of the first dimension and the address of the input port of the second 1-D BinDCT are switched. These ping-pong operations repeat every 64 clock cycles until the transformation for the whole image is completed.

2-D BinDCT Architecture

The system architecture of the 2-D BinDCT coprocessor is depicted in Fig. 15. The 2-D BinDCT coprocessor includes two 1-D BinDCT modules and two transpose buffers. The BinDCT coprocessor receives a 8 × 8 block of inputs. It computes the first 1-D DCT, transposes data in the 8 × 8 block buffer, and then calculates the second 1-D DCT. The hardware cost of the low power architecture for 8-point 2-D BinDCT is 10 adders, 80 registers, and 384 bytes of embedded memory.

High Performance VLSI Architecture

The architecture for high performance applications is similar to the low power VLSI architecture. The basic difference is that more arithmetic units are used in each stage. For instance, four adders are used in Stages 1 through 3 and eight adders are implemented in Stage 4. The latency for each stage is reduced to only 2 clock cycles. In addition, Stage 5 is eliminated. The outputs of Stage 4 are hardwired such that 8 outputs from Stage 4 can be stored in memory column-wise and they can be read in row-wise. The hardware cost of the high performance VLSI architecture for 8-point 2-D BinDCT is 40 adders, 80 registers, and 384 bytes embedded memory.

Performance Analysis

Computational Complexity

The complexity distribution in the proposed five-stage pipeline BinDCT architecture is summarized in Table VIII.

From the algorithm presented above, Stage 1 and Stage 3 each require 8 additions. At first glance, Stage 2 seems to require 6 additions and 7 shift operations. However,

$$b_6 = a_6 + [a_5 + (a_5 << 1)] >> 3$$

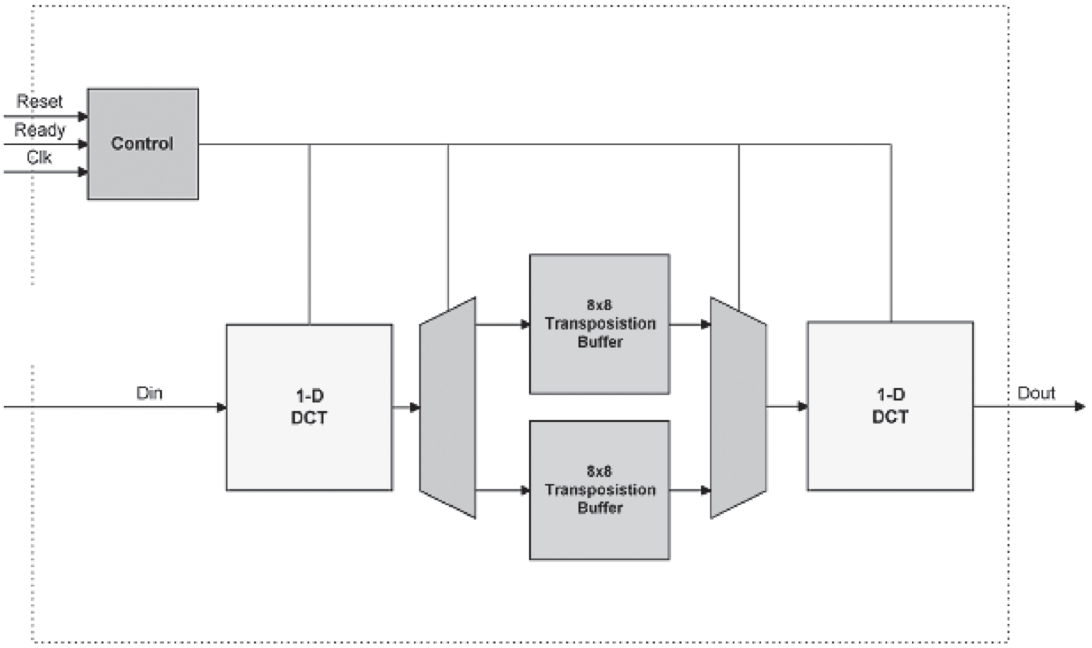


Figure 15. System architecture of 2-D BinDCT coprocessor.

TABLE VIII. Complexity Distribution in the Proposed Five-Stage Pipeline BinDCT Architecture

Stage	Additions	Shift
Stage 1	8	0
Stage 2	4	4
Stage 3	8	0
Stage 4	10	9
Stage 5	0	0
Total:	30	13

TABLE IX. The Data Bus Required for 1-D Forward BinDCT

Stage	Input Vector	Bits	Output Vector	Bits
1	f	8	A*f	9
2	A*f	9	B*A*f	10
3	B*A*f	10	C*B*A*f	11
4	C*B*A*f	11	D*C*B*A*f	12
5	D*C*B*A*f	12	E*D*C*B*A*f	12

and

$$b_5 = [b_6 + (b_6 << 2)] >> 3 - a_5 .$$

Since two additions and three shift operations are shared between b5 and b6, Stage 2 only requires a total of 4 additions and 4 shift operations. Similarly, Stage 4 would require 13 additions and 14 shifts. However,

$$d_2 = c_2 - [c_3 + (c_3 << 1)] >> 3 ,$$
$$d_3 = [d_2 + (d_2 << 1)] >> 3 - c_3,$$
$$d_5 = c_5 + [(c_6 << 3) - c_6] >> 3 ,$$
$$d_6 = c_6 - (d_5 >> 1).$$

Two additions and three shifts are shared between d2 and d3, and one addition and two shifts are shared between d5 and d6. As a result, Stage 4 requires only 10 additions and 9 shift operations. The total computational complexity of the 1-D BinDCT algorithm is 30 additions and 13 shift operations. The BinDCT algorithm is, therefore, considered as a fast transform comparing to the other DCT algorithms.[3–11]

Data Range

Besides the area and the power consumption, another issue for hardware implementation is the data range. The implementation of 2-D DCT/IDCT with finite precision arithmetic, in general, introduces truncation errors due to the finite register length. To minimize the effect of truncation errors, internal buses of processor must be increased appropriately. This approach, however, results in larger area, and it also adversely affects the arithmetic models such as adders and multipliers. As a result, one objective of the VLSI design is to choose the optimal register length. The right register length will ensure the accuracy and will lead to a small chip area.

We first consider the 1-D BinDCT module. The 8-bit input data is represented in the 2's complement format, ranging from –128 to 127. From the matrix factorization form in section 4, it is straightforward for us to determine the output data range at each stage. Since the arithmetic operations in Stages 1 through 4 are either addition or subtraction, in the worst case, the output data at each stage is 1 bit longer than the length of its input. For instance, the input of Stage 1 is 8-bit so the output at Stage 1 is 9-bit. Note that Stage 5 has no arithmetic operation; thus, the input range and output range at Stage 5 are the same. The relationship between input and output data range in the first dimension of BinDCT is summarized in Table IX.

The data range at the second dimension also increases 1-bit for each of Stages 1 through 4. In Stage 5, data range of the input and output are the same. Since the input at the second dimension is 12-bit, the output of

TABLE X. The Data Bus Required for 2-D Forward BinDCT

Stage	Input Vector	Bits	Output Vector	Bits
1	f	12	A*f	13
2	A*f	13	B*A*f	14
3	B*A*f	14	C*B*A*f	15
4	C*B*A*f	15	D*C*B*A*f	16
5	D*C*B*A*f	16	E*D*C*B*A*f	16

the 2D-BinDCT coprocessor is 16-bit data. The data ranges for the second dimension of BinDCT module are summarized in Table X.

Operation of BinDCT Coprocessor

The proposed BinDCT coprocessor includes a control unit, two 1-D BinDCT processing elements, and an embedded memory. The BinDCT coprocessor can compute both forward and inverse BinDCT. In this section, we discuss operations of the forward BinDCT. The inverse BinDCT operation is similar but in the reverse process. All input values of the BinDCT coprocessor are signed integers. The Din and Dout ports are 32-bit signed integers. The number formats in both cases are 2's complement. The BinDCT coprocessor processes data in blocks of 8×8 pixels. Input data is sampled on the Din port. The 32-bit Din port can receive four 8-bit pixels per clock cycle. As a result, it takes 2 cycles to load 8 pixels and 16 clock cycles to complete the loading of a block of 8×8 pixels.

The forward 2-D BinDCT is processed in two stages as shown schematically in Fig. 15. After the computation for the first dimension completes, a block of 64 intermediate samples is stored in the transposition buffer. Transposition buffer is a 384-byte dual port embedded memory. It is organized into two memory banks. Each memory bank can hold 64 intermediate coefficients and serves as a buffer to transpose data from column into row.

When the second 1-D BinDCT processing element starts, it receives data from the first memory bank. The intermediate results generated from the next 8×8 block by the first 1-D BinDCT module are stored in the second memory bank. On the other hand, when the second BinDCT module reads data from the second memory bank, the first BinDCT module writes outputs of the third 8×8 block into the first memory bank. These ping-pong operations repeat for the rest of 8×8 blocks. The outputs from the second processing element are sent out on the Dout port. In the low power design, the first sample of the block is available on the Dout port 132 cycles after the first sample of the block was ready on the Din port. After that, outputs are generated in every clock cycle. In the high performance design, the first sample of the block is available on the Dout port 16 cycles after the first sample of the block enters the Din port.

Performance Analysis

Hardware Implementation Results

The proposed architectures were implemented in VHDL using 0.18 μm CMOS technology. The Design Compiler of Synopsis Version 2001.08 is used to synthesize the circuits. The standard cells are from the DesignWare. The schematics are generated by the Design Analyzer. Figure 16 presents the schematic of the 2-D BinDCT chipset. The profile of the designs is summarized in the Table XI. In this table, we present the area and power consumptions of two proposed architectures. The area of high

TABLE XI. Profile of Low Power and High Performance 2-D BinDCT Coprocessors

Architecture	High PerformanceBinDCT	Low Power BinDCT
Technology	HCMOS8D – 0.18 μm	
Temperature	125 0C	
Voltage	1.55 Volt	
Frequency	5 MHz	
Combinational area	56,578.1	7,778
Non combinational area	58,355.7	58,216
Total cell area	114,933.8	65,944
Power consumption.	24 mW	12.05 mW

performance BinDCT is 2 times larger than that of in the low performance BinDCT. The throughput of each stage, however, increases 400%.

Throughput

The BinDCT architecture for low power consumption requires only 10 adders and 80 registers. Each stage needs 8 clock cycles to complete its computation. The computation time for the first dimension forward BinDCT for an 8×8 block size is:

1-D BinDCT cycle = cycle count for the first row + cycle count for the next 7 rows = (5 + 7)8 = 96 cycles.

The transposition matrix requires 64 cycles, but its execution time is also pipelined. BinDCT coprocessor can start the computation for the second-dimension BinDCT module right after the computation for the first dimension completed. As a result, the number cycles to compute 2-D BinDCT for the first 8×8 block is (12+13)8 = 200 clock cycles (Table XIIa). Note that the input data is streaming, both 1-D BinDCT modules execute concurrently. Coprocessor can calculate 2-D BinDCT for each 8×8 block in 64 cycles. The throughput of the system is 1 pixel per cycle.

For videoconferencing applications using QCIF format (176×144) at 30 frames per second, it takes 1,140,480 clock cycles to compute the forward BinDCT transformation for all Y, Cb, Cr blocks. For video in CIF format, the total computation time is 4,561,920 cycles, which is under 5 MHz. Note that the average power consumption of a CMOS gate due to the switching current is given by $P = \alpha C_L V_{dd}^2 f$, where f is the system clock frequency, V_{dd} is the supply voltage, C_L is the load capacitance, and α is the switching activity. In the proposed architecture, the system frequency is 5 MHz, the global voltage is 1.55 V. The total power consumption is relatively small. The BinDCT coprocessor is, therefore, an efficient design for low power multimedia mobile applications.

For high performance applications (Table XIIb), four adders are used in Stages 1 through 3 and eight adders are implemented in Stage 4. The latency for each stage is only 2 clock cycles. The computation time for 2-D BinDCT of the first 8×8 block is $23 \times 2 = 46$ cycles. When input data is streaming, both 1-D BinDCT modules work concurrently. As a result, it takes only 16 cycles for coprocessor to compute an 8×8 block. The throughput is 4 pixels per cycle.

Comparisons

Table XIII lists features of different DCT and/or IDCT processors, which were reported in the literatures. Each design implements different approach, which leads to a

TABLE XIIa. Timetable for the Low Complexity BinDCT Architecture

Step	Pipeline of the 1st Dimension					1-D Output (Rows)	Pipeline of the 2nd Dimension					2-D Output (Columns)
	S_1	S_2	S_3	S_3	S_5		S_1	S_2	S_3	S_4	S_5	
1	$R_{i,1}$											
2	$R_{i,2}$	$R_{i,1}$										
3	$R_{i,3}$	$R_{i,2}$	$R_{i,1}$									
4	$R_{i,4}$	$R_{i,3}$	$R_{i,2}$	$R_{i,1}$								
5	$R_{i,5}$	$R_{i,4}$	$R_{i,3}$	$R_{i,2}$	$R_{i,1}$							
6	$R_{i,6}$	$R_{i,5}$	$R_{i,4}$	$R_{i,3}$	$R_{i,2}$	$R_{i,1}$						
7	$R_{i,7}$	$R_{i,6}$	$R_{i,5}$	$R_{i,4}$	$R_{i,3}$	$R_{i,2}$						
8	$R_{i,8}$	$R_{i,7}$	$R_{i,6}$	$R_{i,5}$	$R_{i,4}$	$R_{i,3}$						
9	$R_{i+1,1}$	$R_{i,8}$	$R_{i,7}$	$R_{i,6}$	$R_{i,5}$	$R_{i,4}$						
10	$R_{i+1,2}$	$R_{i+1,1}$	$R_{i,8}$	$R_{i,7}$	$R_{i,6}$	$R_{i,5}$						
11	$R_{i+1,3}$	$R_{i+1,2}$	$R_{i+1,1}$	$R_{i,8}$	$R_{i,7}$	$R_{i,6}$						
12	$R_{i+1,4}$	$R_{i+1,3}$	$R_{i+1,2}$	$R_{i+1,1}$	$R_{i,8}$	$R_{i,7}$						
13	$R_{i+1,5}$	$R_{i+1,4}$	$R_{i+1,3}$	$R_{i+1,2}$	$R_{i+1,1}$	$R_{i,8}$	$C_{i,1}$					
14	$R_{i+1,6}$	$R_{i+1,5}$	$R_{i+1,4}$	$R_{i+1,3}$	$R_{i+1,2}$	$R_{i+1,1}$	$C_{i,2}$	$C_{i,1}$				
15	$R_{i+1,7}$	$R_{i+1,6}$	$R_{i+1,5}$	$R_{i+1,4}$	$R_{i+1,3}$	$R_{i+1,2}$	$C_{i,3}$	$C_{i,2}$	$C_{i,1}$			
16	$R_{i+1,8}$	$R_{i+1,7}$	$R_{i+1,6}$	$R_{i+1,5}$	$R_{i+1,4}$	$R_{i+1,3}$	$C_{i,4}$	$C_{i,3}$	$C_{i,2}$	$C_{i,1}$		
17	$R_{i+2,1}$	$R_{i+1,8}$	$R_{i+1,7}$	$R_{i+1,6}$	$R_{i+1,5}$	$R_{i+1,4}$	$C_{i,5}$	$C_{i,4}$	$C_{i,3}$	$C_{i,2}$	$C_{i,1}$	
18	$R_{i+2,2}$	$R_{i+2,1}$	$R_{i+1,8}$	$R_{i+1,7}$	$R_{i+1,6}$	$R_{i+1,5}$	$C_{i,6}$	$C_{i,5}$	$C_{i,4}$	$C_{i,3}$	$C_{i,2}$	$C_{i,1}$
19	$R_{i+2,3}$	$R_{i+2,2}$	$R_{i+2,1}$	$R_{i+1,8}$	$R_{i+1,7}$	$R_{i+1,6}$	$C_{i,7}$	$C_{i,6}$	$C_{i,5}$	$C_{i,4}$	$C_{i,3}$	$C_{i,2}$
20	$R_{i+2,4}$	$R_{i+2,3}$	$R_{i+2,2}$	$R_{i+2,1}$	$R_{i+1,8}$	$R_{i+1,7}$	$C_{i,8}$	$C_{i,7}$	$C_{i,6}$	$C_{i,5}$	$C_{i,4}$	$C_{i,3}$
21	$R_{i+2,5}$	$R_{i+2,4}$	$R_{i+2,3}$	$R_{i+2,2}$	$R_{i+2,1}$	$R_{i+1,8}$	$C_{i+1,1}$	$C_{i,8}$	$C_{i,7}$	$C_{i,6}$	$C_{i,5}$	$C_{i,4}$
22	$R_{i+2,6}$	$R_{i+2,5}$	$R_{i+2,4}$	$R_{i+2,3}$	$R_{i+2,2}$	$R_{i+2,1}$	$C_{i+1,2}$	$C_{i+1,1}$	$C_{i,8}$	$C_{i,7}$	$C_{i,6}$	$C_{i,5}$
23	$R_{i+2,7}$	$R_{i+2,6}$	$R_{i+2,5}$	$R_{i+2,4}$	$R_{i+2,3}$	$R_{i+2,2}$	$C_{i+1,3}$	$C_{i+1,2}$	$C_{i+1,1}$	$C_{i,8}$	$C_{i,7}$	$C_{i,6}$
24	$R_{i+2,8}$	$R_{i+2,7}$	$R_{i+2,6}$	$R_{i+2,5}$	$R_{i+2,4}$	$R_{i+2,3}$	$C_{i+1,4}$	$C_{i+1,3}$	$C_{i+1,2}$	$C_{i+1,1}$	$C_{i,8}$	$C_{i,7}$
25	Ri$_{+3,1}$	$R_{i+2,8}$	$R_{i+2,7}$	$R_{i+2,6}$	$R_{i+2,5}$	$R_{i+2,4}$	$R_{i+2,3}$	$C_{i+1,4}$	$C_{i+1,3}$	$C_{i+1,2}$	$C_{i+1,1}$	$C_{i,8}$

TABLE XIIb. Timetable for the High Performance BinDCT Architecture

Step	Pipeline of the 1st Dimension				1-D Output (Rows)	Pipeline of the 2nd Dimension				2-D Output (Columns)
	S_1	S_2	S_3	S_4		S_1	S_2	S_3	S_4	
1	$R_{i,1}$									
2	$R_{i,2}$	$R_{i,1}$								
3	$R_{i,3}$	$R_{i,2}$	$R_{i,1}$							
4	$R_{i,4}$	$R_{i,3}$	$R_{i,2}$	$R_{i,1}$						
5	$R_{i,5}$	$R_{i,4}$	$R_{i,3}$	$R_{i,2}$	$R_{i,1}$					
6	$R_{i,6}$	$R_{i,5}$	$R_{i,4}$	$R_{i,3}$	$R_{i,2}$					
7	$R_{i,7}$	$R_{i,6}$	$R_{i,5}$	$R_{i,4}$	$R_{i,3}$					
8	$R_{i,8}$	$R_{i,7}$	$R_{i,6}$	$R_{i,5}$	$R_{i,4}$					
9	$R_{i+1,1}$	$R_{i,8}$	$R_{i,7}$	$R_{i,6}$	$R_{i,5}$					
10	$R_{i+1,2}$	$R_{i+1,1}$	$R_{i,8}$	$R_{i,7}$	$R_{i,6}$					
11	$R_{i+1,3}$	$R_{i+1,2}$	$R_{i+1,1}$	$R_{i,8}$	$R_{i,7}$					
12	$R_{i+1,4}$	$R_{i+1,3}$	$R_{i+1,2}$	$R_{i+1,1}$	$R_{i,8}$	$C_{i,1}$				
13	$R_{i+1,5}$	$R_{i+1,4}$	$R_{i+1,3}$	$R_{i+1,2}$	$R_{i+1,1}$	$C_{i,2}$	$C_{i,1}$			
14	$R_{i+1,6}$	$R_{i+1,5}$	$R_{i+1,4}$	$R_{i+1,3}$	$R_{i+1,2}$	$C_{i,3}$	$C_{i,2}$	$C_{i,1}$		
15	$R_{i+1,7}$	$R_{i+1,6}$	$R_{i+1,5}$	$R_{i+1,4}$	$R_{i+1,3}$	$C_{i,4}$	$C_{i,3}$	$C_{i,2}$	$C_{i,1}$	
16	$R_{i+1,8}$	$R_{i+1,7}$	$R_{i+1,6}$	$R_{i+1,5}$	$R_{i+1,4}$	$C_{i,5}$	$C_{i,4}$	$C_{i,3}$	$C_{i,2}$	$C_{i,1}$
17	$R_{i+2,1}$	$R_{i+1,8}$	$R_{i+1,7}$	$R_{i+1,6}$	$R_{i+1,5}$	$C_{i,6}$	$C_{i,5}$	$C_{i,4}$	$C_{i,3}$	$C_{i,2}$
18	$R_{i+2,2}$	$R_{i+2,1}$	$R_{i+1,8}$	$R_{i+1,7}$	$R_{i+1,6}$	$C_{i,7}$	$C_{i,6}$	$C_{i,5}$	$C_{i,4}$	$C_{i,3}$
19	$R_{i+2,3}$	$R_{i+2,2}$	$R_{i+2,1}$	$R_{i+1,8}$	$R_{i+1,7}$	$C_{i,8}$	$C_{i,7}$	$C_{i,6}$	$C_{i,5}$	$C_{i,4}$
20	$R_{i+2,4}$	$R_{i+2,3}$	$R_{i+2,2}$	$R_{i+2,1}$	$R_{i+1,8}$	$C_{i+1,1}$	$C_{i,8}$	$C_{i,7}$	$C_{i,6}$	$C_{i,5}$
21	$R_{i+2,5}$	$R_{i+2,4}$	$R_{i+2,3}$	$R_{i+2,2}$	$R_{i+2,1}$	$C_{i+1,2}$	$C_{i+1,1}$	$C_{i,8}$	$C_{i,7}$	$C_{i,6}$
22	$R_{i+2,6}$	$R_{i+2,5}$	$R_{i+2,4}$	$R_{i+2,3}$	$R_{i+2,2}$	$C_{i+1,3}$	$C_{i+1,2}$	$C_{i+1,1}$	$C_{i,8}$	$C_{i,7}$
23	$R_{i+2,7}$	$R_{i+2,6}$	$R_{i+2,5}$	$R_{i+2,4}$	$R_{i+2,3}$	$C_{i+1,4}$	$C_{i+1,3}$	$C_{i+1,2}$	$C_{i+1,1}$	$C_{i,8}$

different architecture. As a result, it is difficult to make a fair comparison between two architectures. The IDCT processor in Ref. [21], for instance, used digit-serial construction to reduce the overall chip size. This approach, however, increases the latency. The ROM-based Distributed Arithmetic (DA) approach is used in Refs. [15], [24] and [26] to eliminate the need of multiplier. The disadvantage of this implementation is that the ROM size grows exponentially with the input precision. The common approach in many designs uses multiply-accumulate cell (MAC) for DCT/IDCT computation.[23,25] The MAC implementation, however, has to deal with several design issues including floating point operations and finite register length effect. Moreover, the implementation of multipliers requires larger chip area and high power consumption.

TABLE XIII. DCT/IDCT Processor Comparison

Authors	Technology (µm)	Area (mm^2)	Transistors	Voltage (V)	Clock rate (MHz)	Power (mW)	Through-put (Mpel/s)
Defilippis et al.[15]	2.00	22.50					
Jutand et al.[16]	0.80	41.40					
Chau et al.[17]	0.80	21.60			2200		
D. Slawecki[18]	2.00	72.68	67,929	5.00	50	1000	
T. Kuroda[19]	0.30	4.00	120,000	0.90	150	10	
Y.P. Lee[20]	0.60	50.60	152,017	2.00	100	138	
C.-Y. Hung et al.[21]	0.35		24,000		100		26.0
Katayama et al.[22]	0.35		70,700		54		27.0
Rambaldi et al.[23]	0.50		200,000		27		27.0
Okamoto et al.[24]	0.50	4.50			40		14.3
Xanthopoulos et al.[25]	0.70	20.70	160,000		14		14.0
T.-S. Chang et al.[26]	0.60		24,000		55		23.6
SGS Thompson STV3200[27]		311.52		5.00	15	500	
SGS Thompson STV3208[28]		196.00		5.00	27	750	
SGS Thompson IMSA121[29]				5.00	20	1500	
Gong el al.[31]	0.25	1.50			125		125
Low Power BinDCT	0.18	0.07		1.55	5	12	5
High Performance BinDCT	0.18	0.12		1.55	5	48.6	20

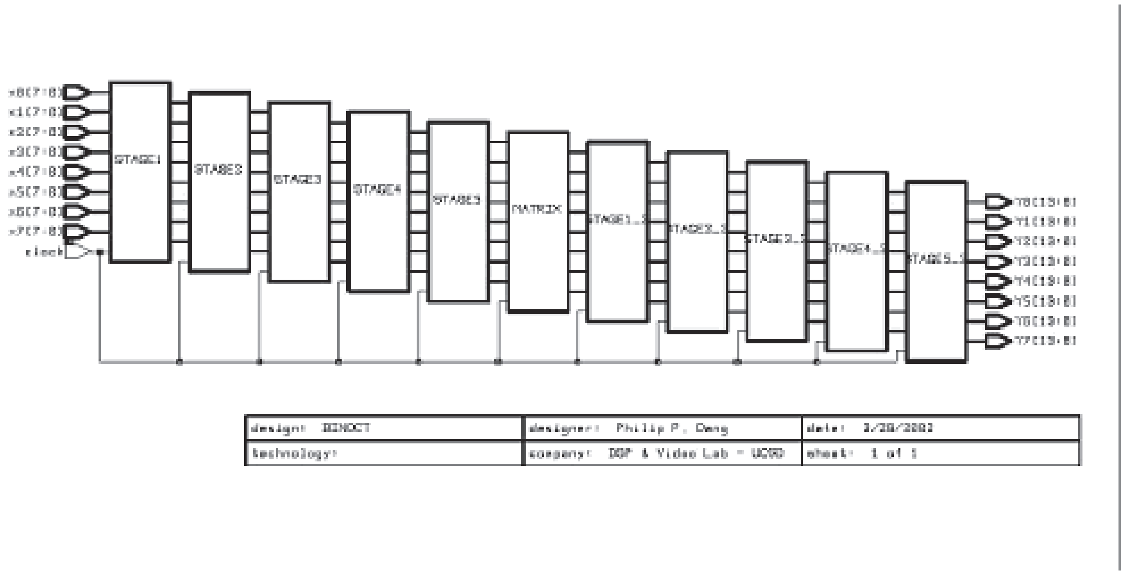


Figure 16. Synthesize schematic of 2-D BinDCT coprocessor.

Compared with the reference designs in the Table XIII, the proposed architecture is very competitive in term of performance and quality. Our designs, however, can obtain high throughput at a very low clock frequency. In fact, at the 5 MHz frequency, the proposed low power VLSI architecture can obtain 5 Mbytes/s throughput. This performance is efficient for videoconference application using CIF format. For high video bit-rate, we can scale the system clock of BinDCT coprocessor to meet the real-time requirement. In addition, the area and power consumption of our designs are significantly smaller than those of in the reference designs. The proposed low power VLSI architecture takes only 0.07 mm^2 chip area and 12 mW power consumption. These requirements are good candidates for mobile multimedia applications. The high performance VLSI architecture, on the other hand, can obtain 20 Mbytes/s throughput at 5 MHz, or 120 Mbytes/s throughput at 30 MHz. In addition, the modularity of the proposed architecture simplifies the design verification effort. Moreover, the scalability of the algorithm gives us the flexibility to meet the real-time constrain at different bit rates.

For the high performance architecture, we compare our design with the latest DCT architecture reported.[30] The proposed architecture is 12 times smaller and 25 times slower clock rate. These factors are translated into approximately 300 times lesser power consumption. If both processor run at the same clock rate 125 MHz, the throughput of proposed system will be 4 times faster than the reference design in Ref. [30].

Summary

This article presents efficient VLSI architectures and the implementation result of the BinDCT algorithm. The proposed architectures are organized as multi-stage pipeline. As a result, they can achieve high throughput, which allow them to meet the real-time of video processing. These designs, on the other hand, require very low power consumption. The hardware implementation of the low complexity BinDCT architecture requires only 10 adders and 80 registers. The high complexity implementation needs an extra 30 adders. For videoconference applications, the low complexity BinDCT coprocessor can complete real-time forward BinDCT at 5 MHz clock rate with 1.55 volt power supply. With its desirable features such as high performance and low power consumption, BinDCT coprocessor is an excellent transform candidate for hardware implementation in mobile DCT-based coding systems, which include Dolby AC-3, JPEG, H.261, H.263, MPEG-1, MPEG-2, and MPEG-4.

References

1. N. Ahmed, T. Natarajan and K. B. Rao, Discrete Cosine Transform, *IEEE Trans. Computers* **C-23,** 90-93 (1974).
2. W. B. Pennebaker and J. L. Mitchel, *JPEG Still Image Compression Standard*, Van Nostrand Reinhold, New York, 1992.
3. W. A. Chen, C. Harrison and S. C. Fralick, A fast computational algorithm transform for Discrete Cosine Transform, *IEEE Trans. Comm.* **COM-25**(9), 1004-1011 (1977).
4. Z. Wang, Fast algorithms for the Discrete W-Transform and for Discrete Fourier Transform, *IEEE Trans. Acoustic, Speech, and Signal Proc.* **ASSP-32**(4), 803-816 (1984).
5. B. Lee, A new algorithm to compute Discrete Cosine Transform, *IEEE Trans. Acoustic, Speech, and Signal Proc.* **ASSP-32**(6), 1243-1245 (1984).
6. M. Vetterli and H. Nussbaumer, Simple FFT and DCT algorithm with reduced number of operations, *Signal Proc.* (North Holland), **6**(4), 267-278 (1984).
7. N. Suehiro and M. Hatori, Fast algorithm for DFT and other Sinusoidal Transform, *IEEE Trans. Acoustic, Speech, and Signal Proc.* **ASSP-34**(3), 642-644 (1986).
8. H. S. Hou, A fast recursive algorithm for computing the Discrete Cosine Transform, *IEEE Trans. Acoustic, Speech, and Signal Proc.* **ASSP-35**(10), 1455-1461 (1987).
9. H. Malvar, Fast computation of the Discrete Cosine Transform and the Discrete Hartley Transform, *IEEE Trans. Acoustics, Speech and Signal Proc.* **ASSP-35**(10) 1484-1485 (1987).
10. C. Loeffler, A. Ligtenberg and G. S. Moschytz, Practical fast 1-D DCT algorithms with 11 multiplications, *Proc. Int'l. Conf. Acoustics, Speech, and Signal Processing*, *ICASSP '89*, IEEE Press, Los Alamitos, CA, 1989, pp. 988-991.
11. Y. Arai, T. Agui, and M. Nakajima, A Fast DCT-SQ scheme for images, *Trans. IEICEE.* **E71**(9), 1095-1097 (1988).
12. T. D. Tran, The BinDCT: Fast multiplierless approximation of the DCT, *IEEE Signal Processing Lett.* **7**, 141-145 (2000).
13. J. Liang and T. D. Tran, Fast multiplierless approximations of the DCT with the lifting scheme, *IEEE Trans. Signal Proc.* **49,** 3032-3044 (2001).
14. JPEG Image Compression Software; http://www.ijg.com; H.263+ Public Domain Code; http://spmg.ece.ubc.ca.
15. I. Defilippis, U. Sjostrom, M. Ansorge, and F. Pellandini, Optimal architecture and time scheduling of a distributed arithmetic based Discrete Cosine Transform chip, *Proc. European Signal Processing Conf., EUSIPCO-90* Barcelona, Spain, 1579 (1990).
16. K. K. Chau, I.-F. Wang and C. L. Eldrigde, VLSI implementation of a 2-D DCT in a compiler, *Proc. IEEE ICASSP-91,* **V2,** IEEE Press, Los Alamitos, CA, 1991, p. 1233.
17. L. B. Jutand, Z. J. Mou and N. Demassieux, DCT architecture for HDTV, *Proc. IEEE ISCAS-91*, **V1,** IEEE Press, Los Alamitos, CA, 1991, p. 196.
18. D. Slawecki and W. Li, DCT/IDCT processor design for high data rate image coding, *IEEE Trans. Circuits and Systems Video Technol.* **2,** 135 (1992).
19. T. Kuroda, T. Fujita, S. Mita, T. Nagamatsu, S. Yoshioka, K. Suzuki, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakumu, and T. Sakurai. A 0.9 V, 150 MHz, 10 mV, 4 mm^2, 2-D Discrete Cosine Transform core processor with variable threshold voltage (VT) scheme, *IEEE J. Solid-State Circuit* **31**(11), 1770 (1996).
20. Y. P. Lee, T. H. Chen, L. G. Chen, M. J. Chen, and C. W. Ku, A cost effective architecture for 8x8 2-D DCT/IDCT using direct method, *IEEE Trans. Circuits and Systems Video Technol.* **7,** 459 (1997).
21. C.-Y. Hung and P. Landman, Compact inverse Discrete Cosine Transform circuit for MPEG video coding, *Proc. IEEE Workshop Systems Processing Systems*, IEEE Press, Los Alamitos, CA, 1997, p. 364.
22. Y. Katayama, T. Kitsuki and Y. Ooi, A block processing unit in a single chip MPEG-2 video encoder LSI, *Proc. IEEE Workshop Systems Processing Systems*, IEEE Press, Los Alamitos, CA, 1997, p. 459.
23. R. Rambaldi, A. Ugazzoni and R. Guerrieri, A 35 mW 1.1 V gate array 8x8 IDCT processor for video telephony, *Proc. IEEE ICASSP 1998*, **5,** 2993 (1998).
24. K. Okamoto, T. Jinbo, T. Araki, Y. Iizuka, H. Nakajima, M. Takahata, H. Inoue, S. Kurohmaru, T. Yonezawa, and K. Aono, A DSP for DCT-based and wavelet-based video codecs for consumer applications, *IEEE J. Solid State Circuits* **32,** 460 (1997).
25. T. Xanthopoulos and A. Chandrakasan, A low power IDCT macrocell for MPEG-2 @ML exploiting data distribution properties for minimal activity, *Proc. IEEE VLSI Circuits*, IEEE Press, Los Alamitos, CA, 1998, p. 38.
26. T. S. Chang, C. S Kung and C. W. Jen, A simple processor core design for DCT/IDCT, *IEEE Trans. Circuits and Systems Video Technol.* **10,** 439 (2000).
27. SGS-Thompson Microelectronics, Discrete Cosine Transform, *STV3200 Technical Report*, July 1992.
28. SGS-Thompson Microelectronics, 8x8 Discrete Cosine Transform (DCT), *STV3208 Technical Report*, July 1992.
29. SGS-Thompson Microelectronics, 2-D Discrete Cosine Transform Image Processor *IMSA121 Technical Report*, July 1992.
30. D. Gong, Y. He and Z. Cao, New cost effective VLSI implementation of a 2-D Discrete Cosine Transform and its inverse, *IEEE Transactions on Circuits and Systems for Video Technol.* **14**(4), 405 (2004).