A HIGH PERFORMANCE VIDEO TRANSFORM ENGINE BY USING SPACE TIME SCHEDULING STRATEGY

[1]Sagar S. Chavan, [2]Prof. S. D .Joshi
[1]Student, [2]Assistant Professor
Email: [1]sagarchavan.etc@gmail.com [2]s.d.joshi@hotmail.com

Abstract— A spatial and time scheduling strategy, called the space-time scheduling (STS) strategy that achieves high image resolutions in real-time systems is proposed. The proposed spatial scheduling strategy includes the ability to choose the distributed arithmetic (DA)-precision bit length, a hardware sharing architecture that reduces the hardware cost, and the proposed time scheduling strategy arranges different dimensional computations in that it can calculate first-dimensional and second-dimensional transformations simultaneously in single 1-D discrete cosine transform (DCT) core to reach a hardware utilization of 100%. The DA-precision bit length is chosen as 9 bits instead of the traditional 12 bits based on test image simulations. In addition, the proposed hardware sharing architecture employs a binary signed-digit DA architecture that enables the arithmetic resources to be shared during the four time slots. For this reason, the proposed 2-D DCT core achieves high accuracy with a small area and a high throughput rate.

Index Terms— Binary signed-digit (BSD), discrete cosine transform (DCT), distributed arithmetic (DA)-based, space-time scheduling (STS).

I. INTRODUCTION

The development of visual media has been progressed towards high-resolution specifications, such as high definition television (HDTV). Consequently, a high-accuracy and high-throughput rate component is needed to meet future specifications. In addition, in order to reduce the manufacturing costs of the integrated circuit (IC), a low hardware cost design is also required. The 2-D DCT core design has often been implemented using either direct[1] or indirect[2] methods. The direct methods include fast algorithms that reduce the computation complexity by mapping and rotating the DCT transform into a complex number. However, the structure of the DCT is not as regular as that of the fast Fourier transform (FFT). A regular 2-D DCT core using the direct method that derives the 2-D shifted FFT (SFFT) from the 2-D DCT algorithm format, and shares the hardware in FFT/IFFT/2-D DCT computation is implemented.

On the other hand, the 2-D DCT cores using the indirect method are implemented based on transpose memory and have the following two structures:

1) Two 1-D DCT cores and one transpose memory (TMEM) and
2) A single 1-D DCT core and one TMEM.

In the first structure, the 2-D DCT core has a high throughput rate, because the two 1-D DCT cores compute the transformation simultaneously. In order to reduce the area overhead, a single 1-D DCT core is applied in the second structure, thereby saving hardware costs, present a 2-D DCT with a single 1-D DCT core and one custom transposed memory that can transpose data sequences using serial-input and parallel-output and requires only a small area, but the speed is reduced because the single DCT core performs the first-dimensional (1st-D) and

second-dimensional(2nd-D) transformations at different times. Therefore, two parallel paths to deal with the reduction of throughput rate in the second structure [3]. However, the additional input buffers are needed in order to temporarily store the input data during the 2nd-D transformation. A hardware sharing technique is used to implement the 2-D DCT core using a single 1-D DCT core. The 1-D DCT core can calculate the 1st-D and 2nd-D DCT computations simultaneously, and the throughput achieves 100 M. pixels/s. However, multiplier-based processing element requires a large amount of circuit area in. As a result, it is obvious that a tradeoff is required between the hardware cost and the speed. There is a balance point between the area required and the speed for 2-D DCT designs and present a multiplier-based pipeline fast 2-DDCT accelerator implemented using a field-programmable gate arrays (FPGA) platform [4]. Additionally, several 2-D DCT designs are implemented based on FPGA for fast verification. In this paper, an 8 X 8 2-D DCT core that consists of a single 1-D DCT core and one TMEM is proposed using a strategy known as space-time scheduling (STS). Due to the accuracy simulations in DA-based binary signed-digit (BSD) expression, 9-bit DA precision is chosen in order to meet the requirements of the peak signal to noise ratio (PSNR) outlined in previous works. Furthermore, the proposed DCT core is designed based on a hardware sharing architecture with BSD DA-based computation so as to reduce the area cost. The arithmetic's share the hardware resources during the four time slots in the DCT core design. 100% hardware utilization is also achieved by using the proposed time scheduling strategy and the 1st-D and 2nd-D DCT computations can be calculated at the same time. Therefore, a high performance transform engine with high accuracy, small area, and a high-throughput rate has been achieved in this research.

II. PROPOSED 8 X 8 2D- DCT CORE DESIGN

This section introduces the proposed 8× 8 2-D DCT core implementation. The 2-D DCT is defined as

$$Y_{u,v} = \frac{1}{4} ku \cdot kv \sum_{i=0}^{7}\sum_{j=0}^{7} x_{i,j} \cos\left(\frac{(2i+1)\mu\Pi}{16}\right) \times \cos\left(\frac{(2j+1)v\Pi}{16}\right) \quad (1)$$

where

$k_u = k_v = \frac{1}{\sqrt{2}}$ for $u = v = 0$ and $k_u = k_v = 1$ for $1 \leq u, v \geq 7$. Consider the 1-D 8-point DCT first

$$z_n = \frac{1}{2} k_n \sum_{m=0}^{7} x_m \times \cos\left(\frac{(2m+1)n\Pi}{16}\right) \quad (2)$$

By neglecting the scaling factor 1/2, the 1-D 8-point DCT in (2) can be divided into even and odd parts Ze and Zo as listed in (3) and (4) respectively.

$$\mathbf{Z}_e = \begin{bmatrix} Z_0 \\ Z_2 \\ Z_4 \\ Z_6 \end{bmatrix} = \begin{bmatrix} c_4 & c_4 & c_4 & c_4 \\ c_2 & c_6 & -c_6 & -c_2 \\ c_4 & -c_4 & -c_4 & c_4 \\ c_6 & -c_2 & c_2 & -c_6 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \mathbf{C}_e \cdot \mathbf{a} \quad (3)$$

$$\mathbf{Z}_o = \begin{bmatrix} Z_1 \\ Z_3 \\ Z_5 \\ Z_7 \end{bmatrix} = \begin{bmatrix} c_1 & c_3 & c_5 & c_7 \\ c_3 & -c_7 & -c_1 & -c_5 \\ c_5 & -c_1 & c_7 & c_3 \\ c_7 & -c_5 & c_3 & -c_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \mathbf{C}_o \cdot \mathbf{b} \quad (4)$$

where

$C_i = \cos\left(\frac{i\Pi}{16}\right)$ and

$$\mathbf{C}_e = \begin{bmatrix} c_4 & c_4 & c_4 & c_4 \\ c_2 & c_6 & -c_6 & -c_2 \\ c_4 & -c_4 & -c_4 & c_4 \\ c_6 & -c_2 & c_2 & -c_6 \end{bmatrix} \quad (5)$$

$$\mathbf{C}_o = \begin{bmatrix} c_1 & c_3 & c_5 & c_7 \\ c_3 & -c_7 & -c_1 & -c_5 \\ c_5 & -c_1 & c_7 & c_3 \\ c_7 & -c_5 & c_3 & -c_1 \end{bmatrix} \quad (6)$$

$$\mathbf{a} = \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix}. \quad (7)$$

For the 2-D DCT core implementation, the three main strategies for increasing the hardware and time utilization, called the space-time scheduling (STS) strategy, are proposed and listed as follows:

- find the DA-precision bit length for the BSD representation to achieve system accuracy requirements;
- share the hardware resource in time to reduce area cost;
- plan an 100% hardware utilization by using time scheduling strategy.

A. Analysis of the coefficient bits

The seven internal coefficients from c_1 to c_7 for the 2-D DCT transformation are expressed as BSD representations in order to save computation time and hardware cost, as well as to achieve the requirements of PSNR. The system PSNR is defined [1] as follows:

$$PSNR = 10\log_{10}\frac{255^2}{MSE_1} \quad (8)$$

Where the MSE_i is the mean-square-error between the original image and the reconstructed image for each pixel.

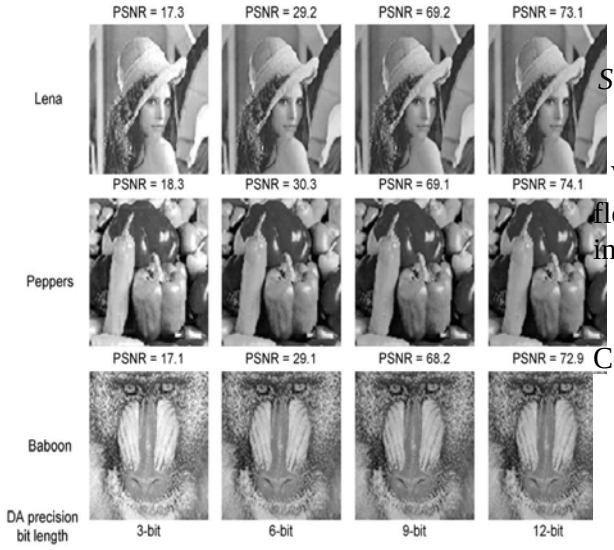

Fig.1 Reconstructed images with different BSD length

First, the BSD coefficients for different bit length expressions use the popular test images, "Lena," "Peppers," and "Baboon" to simulate the system PSNR. After inputting the original test image pixels to the forward and inverse DCT transform, the system PSNR vs. the different BSD bit length expressions is illustrated in Fig. 1, and the reconstructed images, i.e., that have passed through the forward and inverse DCT for different BSD bit lengths, are also shown.The normalized hardware (HW) cost of processing element (PE) for the different BSD bit length expressions is shown in Table I. The HW cost is synthesized in area for each BSD bit length using a Synopsys Design Compiler with the Artisan TSMC 0.18- m Standard cell library.

Table 1.Normalized H.w. Cost In Different BSD Bit Length Expressions

Bit length	1	2	3	4	5	6	7
HW cost	1	1.2	2	2.2	2.6	3	3.6
Bit length	**8**	**9**	**10**	**11**	**12**	**13**	**14**
HW cost	4	4.7	5.1	5.7	5.9	6.7	7.3

Consequently, the 9-bit BSD expression coefficient is chosen due to the tradeoff between the hardware cost and the system accuracy. The BSD expressions and the accuracy of the DCT coefficients are listed in Table 2, where the coefficient in Table 2 indicates the BSD number. The coefficient accuracy is defined as the signal to quantisation noise ratio as shown in (9)

$$SQNR(db) = 10\log_{10}\frac{S}{N} \quad (9)$$

Where S and N are the power of the desired floating-point signal and the quantization noise in this case, respectively.

Table 2. 9-Bit BSD Expressions For DCT Coefficients

Coef.	9-bit BSD expression	Value	SQNR(dB)
c_1	$1\,0\,0\,0\,0\,0\,\bar{1}\,0\,\bar{1}$	0.9805	69.8
c_2	$1\,0\,0\,0\,\bar{1}\,0\,\bar{1}\,1\,\bar{1}$	0.9258	53.7
c_3	$0\,1\,1\,0\,1\,0\,1\,0\,1$	0.8320	63.4
c_4	$0\,1\,0\,1\,1\,0\,1\,0\,1$	0.7070	79.4
c_5	$0\,1\,0\,0\,1\,0\,0\,\bar{1}\,0$	0.5547	56.0
c_6	$1\,\bar{1}\,0\,\bar{1}\,0\,0\,0\,1\,0$	0.3828	69.4
c_7	$0\,0\,0\,1\,1\,0\,0\,1\,0$	0.1953	58.9

III. HARDWARE SHARING STRATEGY

All modules in the 1-D DCT core, including the modified two-input butterfly (MBF2), the pre-reorder, the process element even (PEE), the process element odd (PEO), and the post reorder share the hardware resources in order to reduce the area cost. Moreover, the 8× 8 2-D DCT core is implemented using a single 1-D DCT core and one TMEM. The architecture of the2-D DCT core is described in this section.

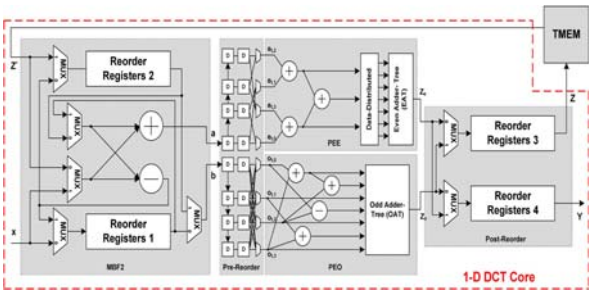

Fig.2 System Block Diagram

1. Modified Butterfly Module

Modified Butterfly Module is easily implemented using a two-input butterfly module scaled BF2. Equation (7) is easily implemented using a two-input butterfly module [8] called BF2. In general, the BF2 has a hardware utilization rate in the adder and subtracter of 50%. In order to enable the hardware resources to be shared, additional multiplexers and Reorder Registers are added to the proposed MBF2 module as illustrated in Fig. 2.The Reorder Registers consist of four word registers that use the control signals to select the input data and use enable signals to output or hold the data. Similar to BF2, the operation of the proposed MBF2 has an eight-clock-cycle period. In the first four cycles, the 1st-D input data (x_0, x_1, x_2, x_3) shift into Reorder Registers 1, and the 2nd-D input data $z^{'}$ execute the operation of addition and subtraction. In the next four cycles, the operations of 1st-D and 2nd-D will be changed. The 1st-D data (x_4, x_5, x_6, x_7) calculate a and b in (7) by using adder and subtracter. In the second four cycles, the results a $(= a_i, i = 3,2,1,0)$ are fed into next stage, and the results b $(= b, i = 3,2,1,0)$ shift to Reorder Registers 1 in each cycle. At the same time, the 2nd-D input data Z' shifts to Reorder Registers 2. Therefore, the adder and subtracter inMBF2 employ the operations of 1st-D and 2nd-D in turns to achieve 100% hardware utilization.

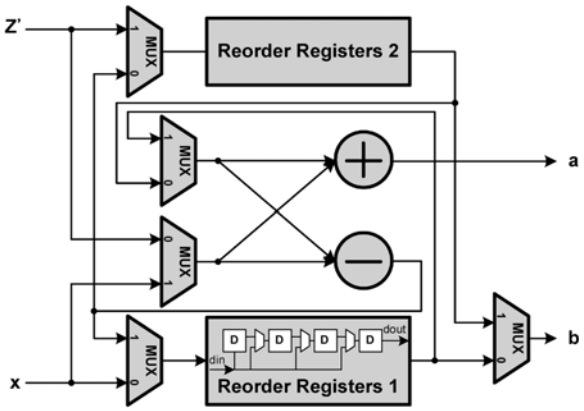

Fig.3 Proposed MBF2 architecture.

2. Processing Element

DA-based computation, the even part and odd part transformations can be implemented using PEE and PEO, respectively. The even part transformation can be expanded for the DA-based computation formats and can share the hardware resources at the bit level. The coefficient vector has two combinations of $[C_4C_4C_4C_4]$ and $[C_2C_6C_6C_2]$ for (Z_0, Z_4) and (Z_2, Z_6) transform outputs, respectively. Using given input data $e_{t,0}$, $e_{t,1}$, $e_{t,2}$, and $e_{t,3}$, the transform Output $Z_{e.}$ needs adder-tree (OAT). The EAT and OAT can be implemented using the error-compensated adder tree to improve the computation accuracy. Moreover, there are three pipeline stages (two in both the EAT and the OAT) in each PEE and PEO module that enable high speed computation to be achieved. only three adders, a Data-Distributed module, and one even part adder-tree (EAT) to obtain the result for Z_e during the four time slots and the EAT sums the different weighted values in tree-like adders to complete the transform output Z_e . Similarly, the odd part transformation it can be implemented in a DA-based format using four adders and one odd part .

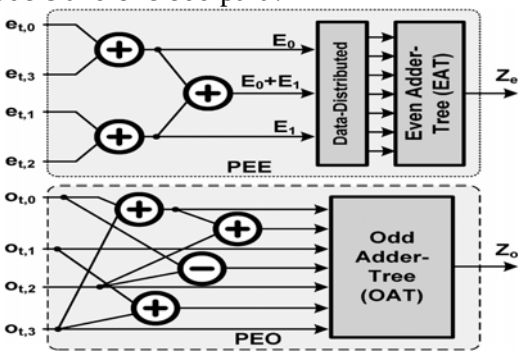

Fig.4 PEE and PEO Architecture

3. Even Adder Tree

The even part transformation can be expanded for the DA-based computation formats and can share the hardware resources at the bit level. The coefficient vector has two combinations of [C4,C4,C4,C4] and [C2,C6,C6,C2] for (Z_0,Z_4) and (Z_2,Z_6) transform outputs, respectively. Using given input data $e_{t,0}$, $e_{t,1}$, $e_{t,2}$, $e_{t,3}$ and , the transform output needs only three adders. the transform output Z_e needs only three adders (to compute $E_0=e_{t,0} + e_{t,3}$, $E_1=e_{t,1} + e_{t,2}$, and E_0+E_1), a Data-Distributed module a Data-Distributed module, and one even part adder-tree (EAT) to obtain the result for during the four time slots. The Data-Distributed module consists of seven two-input multiplexers that are used to appoint different non-zero values for each weighted input during the four separate time slots, and the EAT sums the different weighted values in tree-like adders to complete the transform output Z_e .

4. Odd Adder Tree

Similarly, the odd part transformation in can be implemented in a DA-based format using four adders and one Odd part adder-tree (OAT). The EAT and OAT can be implemented using the error-compensated adder tree to improve the computation accuracy. Moreover, there are three pipeline stages (two in both the EAT and the OAT) in each PEE and PEO module that enable high speed computation to be achieved.

5. Post reorder module

The data sequence after the PEE and PEO must be merged and re-permuted in the Post-Reorder module. The order of the original 8-point data sequence from the PEE and the PEO is {Z_0, Z_2, Z_4, Z_6, Z_1, Z_3, Z_5, and Z_7}. After the Post-Reorder module permutes the data order, the data sequence is ordered as {Z_0, Z_2, Z_4, Z_6, Z_1, Z_3, Z_5, Z_7 }. Two multiplexers select the data that is fed into the different Reorder Registers in order to permute the output order. Z is the transform output for the 1st-D DCT that will input into the TMEM. In addition, the 2nd-D DCT transform output is completed after the permutation by Reorder Registers 4

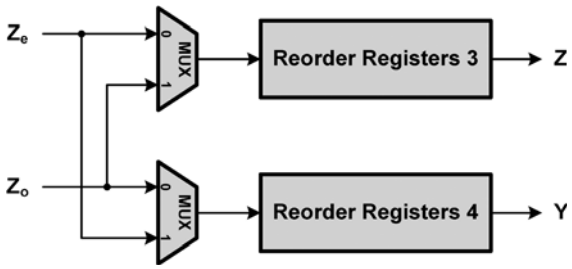

Fig.5 Post reorder module

IV. TIME SCHEDULING STRATEGY

As a result of the time scheduling strategy, the 1st-D and 2nd-D transforms are able to be computed simultaneously, which increases hardware utilization be 100%. The timing flowchart for the proposed strategy is illustrated in Fig. 7.

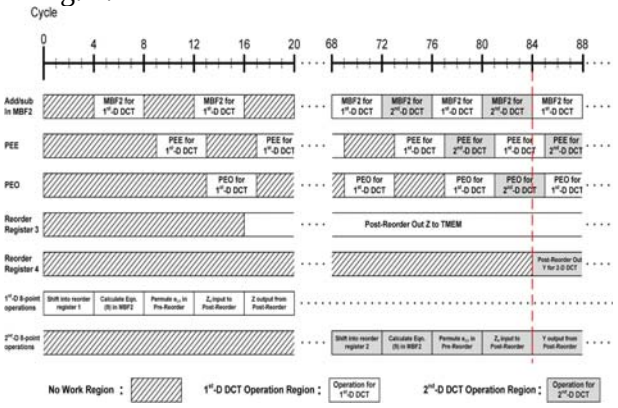

Fig.6 Timing scheduling for the proposed 2-D DCT core.

1. 1st-D Data Computation

In the first four cycles, the first four-point input data shift into the Reorder Register During cycles 5–8, the MBF2 performs additions and subtractions using the first eight point input data. In the 9th cycle, the even-part of Pre-Reorder module obtains $\{e_{t,0}, e_{t,1}, e_{t,2}, e_{t,3}\}$ and reorders the input data output to the PEE. During cycles 10–13, Z_e is calculated in the PEE module. Based on the latency of the three pipeline stages in the PEE, z_e is completed during cycles 13–16. Also z_o is performed in the PEO module during cycles 14–17 and is finished in cycles 17–20. Therefore, after the 16th cycle, the Post-Reorder module permutes the 1st-D transform results Z and inputs them into the TMEM.

2. 2nd-D Data Computation

From the 17th cycle, the 1st-Dtransform data is input into the TMEM. Due to the latency of the TMEM (= 52 cycles) the 2nd-D computation data Z' transposed by the TMEM is sent to the input of Reorder Registers 2 inMBF2 at the 69th cycle. Then, the adder and subtracter are operated during cycles 73–76 to compute the first 2nd-D 8-point data sequences, and the first 2nd-D data is permuted by Pre-Reorder module during cycles 77–80. The PEE and PEO calculate the first 2nd-D data during cycles 78–81 and 82–85, respectively. The Post-Reorder module finishes the 2nd-D transform results from the 84th cycle. In addition, the 2-D DCT transform output data is obtained at the end of 84th cycle.

3. Hardware Utilization

In Fig. 6, the adder and subtracter in MBF2 is at 50% utilization during cycles 1–68. However, after the 68th cycle, the MBF2 achieves a 100% hardware utilization as a result of the 2nd-D data being input. At the same time, the utilization in the PEE and PEO also reaches 100% from74th and 78th cycles, respectively. The Post-Reorder module does not work between cycles 1–12, but after the 16th cycle the1st-D transform output Z are completed using Reorder Registers3 in the Post-Reorder. Similarly, the 2nd-D transform result Y is finished using Reorder Registers 4 in the Post-Reorder from the 84th cycle. At the end of 84th cycle, the 1st-D and2nd-D transform outputs Z and Y are obtained simultaneously. Hence, the hardware utilization increases to 100% after the 84^{th}cycle, and the latency of the proposed 8 8 2-D DCT core is 84 clock cycles. Based on the proposed time scheduling strategy, a100% hardware utilization is achieved, and computation of the data sequence is straightforward.

V. CONCLUSION

The paper proposes a high performance video transform engine using the STS strategy. The proposed 2-D DCT core employs a single 1-D DCT core and one TMEM with a small area. Based on the test image simulations, 9-bit DA-precision is chosen in order to meet the PSNR requirements, and the hardware sharing architecture enables the arithmetic resources to be shared based on time so as to reduce area cost. Hence, the number of adders/substracters in 1-D DCT core allows a 74% saving in area over the NEDA architecture for a DA-based DCT design. Furthermore, the proposed time scheduling strategy arranges the computation time for each process element. The1-D core can calculate the 1st-D and 2nd-D transformations simultaneously, thereby achieving a high throughput rate. Therefore, a high performance 8 X 8 2-D DCT core utilizing high-accuracy, a small-area, and a high-throughput rate has been achieved using the proposed STS strategy.

REFRENCES

[1]E. Feig and S.Winograd, "Fast algorithms for the discrete cosine transform," *IEEE Trans. Signal Process.*, vol. 40, Sep.1992.

[2]M. Alam, W. Badawy, and G. Jullien, "A new time distributed DCT architecture for MPEG-4 hardware reference model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15,May 2005.

[3]J. I. Guo, R. C. Ju, and J. W. Chen, "An efficient 2-D DCT/IDCT core design using cyclic convolution and adder-based realization," *IEEETrans. Circuits Syst. Video Technol.*, vol. 14, Apr. 2004.

[4]A. Tumeo, M. Monchiero, G. Palermo, F. Ferrandi, and D. Sciuto, "A pipelined fast 2D-DCT accelerator for FPGA-based SOCs," in *Proc.IEEE Comput. Soc. Annu. Symp. VLSI.*, 2007.

[5]A. Madisetti and A. N. Willson, "A 100 MHz 2-D 8 *8 DCT/IDCT processor for HDTV applications," *IEEE Trans. Circuits Syst. VideoTechnol.*, vol. 5, Apr. 1995.

[6]A. M. Shams, A. Chidanandan, W. Pan, and M. A. Bayoumi, "NEDA: Alow-power high-performance DCT architecture," *IEEE Trans. SignalProcess.*, vol. 54, no. Mar. 2006.

[7]M. Alam, W. Badawy, and G. Jullien, "A new time distributed DCT architecture for MPEG-4 hardware reference model," *IEEE Trans. CircuitsSyst. Video Technol.*, vol. 15, no. 5, May 2005

[8]L. V. Agostini, I. S. Silva, and S. Bampi, "Pipelined fast 2D DCT architecture for JPEG image compression," in *Proc. IEEE Symp. Integr.Circuits Syst. Des.*, 2001.

[9]W. Pan, "A fast 2-D DCT algorithm via distributed arithmetic optimization,"in*Proc. IEEE Int. Conf. Image Process.*, 2000, vol. 3,

[10]S. Ghosh, S. Venigalla, and M. Bayoumi, "Design and implementation of a 2D-DCT architecture using coefficient distributed arithmetic," in*Proc. IEEE Comput. Soc. Annu. Symp. VLSI.*, 2005