ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 9, September 2014

An efficient Video Transform Engine by Using Space-Time Scheduling Strategy

K.E. RAJIVA RANJAN [1], S.P.SURESH NAIK [2]

Student[1], Assistant Professor[2]

GATES Institute of Technology, Ananthapur, AP, India, Ananthapur, AP, India[1,2]

Abstract: In the proposed system different optimization techniques are explored to improve the performance of the DCT. The latency of clock cycles of two dimensional DCT has been reduced by using pipelined and parallel processing of architecture. In the proposed system 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.

Keywords: Discrete cosine transform (DCT), Binary signed-digit (BSD), space-time scheduling (STS, VHDL).

I. INTRODUCTION

1.1 Definition of DCT

DISCRETE COSINE TRANSFORM (DCT) is a widely used transform engine for image and video compression applications. In recent years, 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. Therefore, a high performance video transform engine that utilizes high accuracy, a small area, and a high-throughput rate is desired for VLSI Designs.

The 2-D DCT core design has often been implemented using either direct [2]–[4] or indirect 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. . However, the additional input buffers are needed in order to temporarily store the input data during the 2nd-D transformation. Madisetti *et al.* present a hardware sharing technique 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 Mpels/s.

As a result, it is obvious that a tradeoff is required between the hardware cost and the speed. Tumeo *et al.* find a balance point between the area required and the speed for 2-D DCT designs and present a multiplier-based pipeline fast 2-D DCT accelerator implemented using a field-programmable gate arrays (FPGA) platform. Additionally, several 2-D DCT designs are implemented based on FPGA for fast verification.

In this paper, an 8x8 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. A 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. This paper is organized as follows; the mathematical derivation of the BSD format distributed arithmetic is given. The proposed 8x8 2-D DCT architecture that includes an analysis of the coefficient bits, the hardware sharing architecture, and the proposed timing scheduling strategy

2. METHODOLOGY

MATHEMATICAL DERIVATION OF BSD FORMAT DISTRIBUTED ARITHMETIC

The inner product for a general matrix multiplication-and accumulation can be written as follows:

where A_i is a fixed coefficient and X_i is the input data. The matrix is called the DA coefficient matrix. In (2), Y can be calculated by adding or subtracting x. With A not equal to 0, and then the transform output Y can be obtained by shifting and adding every non-zero . Thus, the inner product computation can be implemented by using shifters and adders instead of multipliers. Therefore, a smaller area can be achieved by using BSD DA-based architecture.

3. PROPOSED 8x8 2-D DCT CORE DESIGN

This section introduces the proposed 8x8 2-D DCT core implementation. The 2-D DCT is defined as

$$Y_{u,v} = \frac{1}{4}k_u k_v \sum_{i=0}^{7}\sum_{j=0}^{7} x_{i,j} \cos\left(\frac{(2i+1)u\pi}{16}\right) \times \cos\left(\frac{(2j+1)v\pi}{16}\right) \quad (3)$$

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

$$Z_n = \frac{1}{2}k_n \sum_{m=0}^{7} x_m \times \cos\left(\frac{(2m+1)n\pi}{16}\right) \quad (4)$$

where $k_n = 1/\sqrt{2}$ for $n= 0, k_n = 1$, for non-zero n, and x_m and Z_n ($0 \leq n \leq 7$) denote the input data and the transform output, respectively. By neglecting the scaling factor 1/2, the 1-D 8-point DCT in (4) can be divided into even and odd parts, Z_c and Z_o , as listed in (5) and (6), respectively

$$\mathbf{Z}_e = \begin{bmatrix} Z_0 \\ Z_2 \\ Z_4 \\ Z_6 \end{bmatrix} = \begin{bmatrix} c_4 & c_4 & c_4 & c_4 \\ c_2 & c_6 & -c_6 & -c_2 \\ c_4 & -c_4 & -c_4 & c_4 \\ c_6 & -c_2 & c_2 & -c_6 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \mathbf{C}_e \cdot \mathbf{a} \quad (5)$$

$$\mathbf{Z}_o = \begin{bmatrix} Z_1 \\ Z_3 \\ Z_5 \\ Z_7 \end{bmatrix} = \begin{bmatrix} c_1 & c_3 & c_5 & c_7 \\ c_3 & -c_7 & -c_1 & -c_5 \\ c_5 & -c_1 & c_7 & c_3 \\ c_7 & -c_5 & c_3 & -c_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \mathbf{C}_o \cdot \mathbf{b} \quad (6)$$

Where Ci = cos (iπ/16) and

$$\mathbf{C}_e = \begin{bmatrix} c_4 & c_4 & c_4 & c_4 \\ c_2 & c_6 & -c_6 & -c_2 \\ c_4 & -c_4 & -c_4 & c_4 \\ c_6 & -c_2 & c_2 & -c_6 \end{bmatrix} \quad (7)$$

$$\mathbf{C}_o = \begin{bmatrix} c_1 & c_3 & c_5 & c_7 \\ c_3 & -c_7 & -c_1 & -c_5 \\ c_5 & -c_1 & c_7 & c_3 \\ c_7 & -c_5 & c_3 & -c_1 \end{bmatrix} \quad (8)$$

$$\mathbf{a} = \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix}. \quad (9)$$

$$Y = \begin{bmatrix} 2^0 & 2^{-1} & \cdots & 2^{-(N-1)} \end{bmatrix} \cdot \begin{bmatrix} A_{1,0} & A_{2,0} & \cdots & A_{L,0} \\ A_{1,1} & A_{2,1} & \cdots & A_{L,1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1,(N-1)} & A_{2,(N-1)} & \cdots & A_{L,(N-1)} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_L \end{bmatrix} = \begin{bmatrix} 2^0 & 2^{-1} & \cdots & 2^{-(N-1)} \end{bmatrix} \begin{bmatrix} Y_0 \\ Y_1 \\ \vdots \\ Y_{(N-1)} \end{bmatrix} \quad (2)$$

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.

3.1 Analysis of the Coefficient Bits

The seven internal coefficients from c1 to c7 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:

$$\mathrm{PSNR} = 10\log_{10}\frac{255^2}{\mathrm{MSE}_I} \quad (10)$$

where the MSE_I is the mean-square-error between the original image and the reconstructed image for each pixel.

3.2. 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 the 2-D DCT core is described in the following sections.

1) Modified Butterfly Module: Equation (9) is easily implemented using a two-input butterfly module [18] called BF2. In general, the BF2 has a hardware utilization rate in the adder and subtracted 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(X0,X1,X2,X3) shift into Reorder Registers 1, and the 2nd-D input data 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 (X4,X5,X6,X7) calculate a and b in (9) by using adder and subtracter. In the second four cycles, the results a (=ai, i = 3,2,1,0) are fed into next stage, and the results b (= bi, i= 3,2,1,0) shift to Reorder Registers 1 in each cycle. At the same time, the 2nd-D input data Z' shift to Reorder Registers 2. Therefore, the adder and subtracter in MBF2 employ the operations of 1st-D and 2nd-D in turns to achieve 100% hardware utilization.

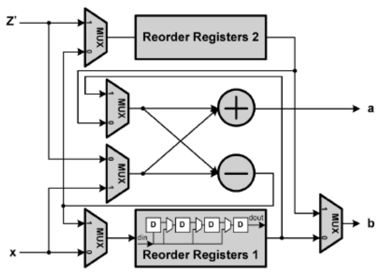


Fig. 2. Proposed MBF2 architecture.

2) Pre-Reorder Module: In (5), the even part transform output $\mathbf{Z}_c$ can be modified as follows:

$$\begin{bmatrix} Z_0 \\ Z_4 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_0 & -a_1 & -a_2 & a_3 \end{bmatrix} \begin{bmatrix} c_4 \\ c_4 \\ c_4 \\ c_4 \end{bmatrix}$$

$$\begin{bmatrix} Z_2 \\ Z_6 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & -a_2 & -a_3 \\ -a_1 & a_0 & -a_3 & a_2 \end{bmatrix} \begin{bmatrix} c_2 \\ c_6 \\ c_6 \\ c_2 \end{bmatrix}. \qquad (12)$$

Also, the odd part transform **Zo** output in (6) can be rewritten as

$$\begin{bmatrix} Z_1 \\ Z_3 \\ Z_5 \\ Z_7 \end{bmatrix} = \begin{bmatrix} b_2 & -b_0 & b_3 & b_1 \\ -b_3 & b_2 & -b_1 & b_0 \\ b_0 & b_1 & b_2 & b_3 \\ -b_1 & b_3 & b_0 & b_2 \end{bmatrix} \begin{bmatrix} c_5 \\ -c_1 \\ c_7 \\ c_3 \end{bmatrix}. \qquad (13)$$

From (12) and (13), the transform output **Z** can be calculated using four separate time slots in order to share the hardware resources. Hence, (12) and (13) can be expressed as (14) and (15), respectively

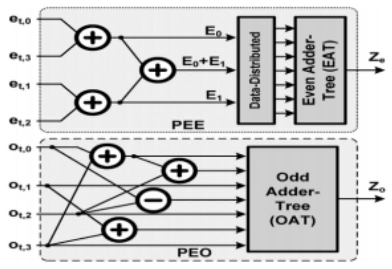
$$Z_{2t} = [e_{t,0} \quad e_{t,1} \quad e_{t,2} \quad e_{t,3}] \begin{bmatrix} ce_{t,0} \\ ce_{t,1} \\ ce_{t,2} \\ ce_{t,3} \end{bmatrix} = \sum_{n=0}^{3} e_{t,n} \cdot ce_{t,n} \qquad (14)$$

$$Z_{2t+1} = [o_{t,0} \quad o_{t,1} \quad o_{t,2} \quad o_{t,3}] \begin{bmatrix} co_{t,0} \\ co_{t,1} \\ co_{t,2} \\ co_{t,3} \end{bmatrix} = \sum_{n=0}^{3} o_{t,n} \cdot co_{t,n} \qquad (15)$$

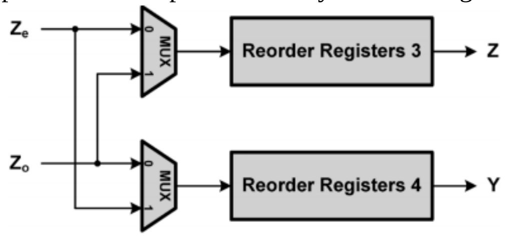
where the time slot $0 \leq t \leq 3$, the coefficient elements $ce_{t,n}$ $\epsilon\{c2,c4,c6\}$, $co_{t,n}$ $\epsilon\{c1,c3,c5,c7\}$ and the transform inputs from the MBF2 stage and $e_{t,n}$ $\epsilon\{a0,a1,a2,a3\}$ and $o_{t,n}$ $\epsilon\{b0,b1,b2,b3\}$

3) 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 [C4C4C4C4] and [C2C6C6C2] for (Z0, Z4) and (Z2, Z6) transform outputs, respectively. Using given input data et,0, et,1, et,2 , and et,3 , the transform Output Ze needs only three adders, a Data-Distributed module, and one even part adder-tree (EAT) to obtain the result for Ze during the four time slots and the EAT sums the different weighted values in tree-like adders to complete the transform output Ze . Similarly, the odd part transformation it 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. Inputs enter into the processing element from the upper left. The first step is for each of these inputs to be multiplied by their respective weighting factor (w(n)). Then these modified inputs are fed into the summing function, which usually just sums these products. Yet, many different types of operations can be selected. These operations could produce a number of different values which are then propagated forward; values such as the average, the largest, the smallest, the OR -ed values, the AND -ed values, etc. Furthermore, most commercial development products allow software engineers to create their own summing functions via routines coded in a higher level language (C is commonly supported). Sometimes the summing function is further complicated by the addition of an activation function which enables the summing function to operate in a time sensitive way. Either way, the output of the summing function is then sent into a transfer function. This function then turns this number into a real output via some algorithm. It is this algorithm that takes the input and turns it into a zero or a one, a

minus one or a one, or some other number. The transfer functions that are commonly supported are sigmoid, sine, hyperbolic tangent, etc. This transfer function also can scale the output or control its value via thresholds. The result of the transfer function is usually the direct output of the processing element.



4) Post Reorder: 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 {Z0, Z2, Z4, Z6, Z1, Z3, Z5, and Z7}. After the Post-Reorder module permutes the data order, the data sequence is ordered as {Z0, Z2, Z4, Z6, Z1, Z3, Z5, Z7 }. 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.

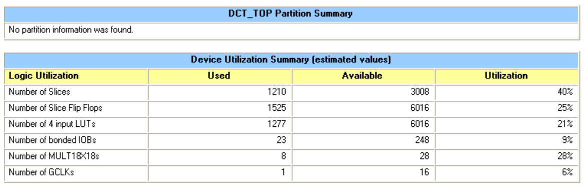


5) TMEM: The TMEM is implemented using 64-word 12-bit dual-port registers and has a latency of 52 cycles. Based on the time scheduling strategy and result of the time scheduling strategy, the 1st-D and 2nd-D transforms are able to be computed simultaneously.

2-D DCT Core Architecture: To save hardware costs, the proposed 2-D DCT core, is implemented using a single 1-D DCT core and one TMEM. The 1-D DCT core includes an MBF2, a Pre-Reorder module, a PEE, a PEO, a Post-Reorder module, and one TMEM. The TMEM is implemented using 64-word 12-bit dual-port registers and has a latency of 52 cycles. Based on the time scheduling strategy, a hardware utilization of 100% can be achieved.

4. RESULTS AND ANALYSIS

In the proposed system, the input images are all 256x256 pixels in size, with each pixel being represented by 8-bit 256 gray level data. After the original test image pixels are fed into the proposed 2-D DCT core, the transform output data is captured and passed to MATLAB tool in order to compute the inverse DCT using 64-bit double-precision operations. Furthermore, the proposed 2-D DCT core was introduced to enable utilization of JPEG compression flow to evaluate the proposed 2-D DCT applied in real compression systems. The quality factor (QF) dominates the compression quality and data size, and therefore QF affects the system PSNR of the JPEG compression. The figure illustrates both the original and the compressed versions of the test image using the JPEG compression flow.
The proposed 2-D DCT core simulated using Modelsim PE 6.4 and synthesized using Xilinx ISE9.1i and Xilinx XC2VP4 FPGA can be operated at a clock frequency of 170MHz. Table I shows a synthesis results of the proposed 2-D DCT core.The parallel architecture and instead of using two reorder registers, only one reorder register is used in this 2- DCT core and the proposed 2-D DCT core employs a single 1-D core and one TMEM in order to reduce the core area. Also, the 1-D core utilizes 9 adders and 2 adder-trees (AT) using the proposed hardware sharing architecture in order to reduce the hardware cost.



DCT_TOP Partition Summary
No partition information was found.

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	1210	3008	40%
Number of Slice Flip Flops	1525	6016	25%
Number of 4 input LUTs	1277	6016	21%
Number of bonded IOBs	23	248	9%
Number of MULT18X18s	8	28	28%
Number of GCLKs	1	16	6%

Table : synthesis results

5. CONCLUSION

The proposed system was implemented in VHDL language and simulated using Modelsim Tool and target for XC2VP4 FPGA device. The proposed spatial scheduling strategy includes the capacity to select 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 1-D and 2-D transformations at the same time in single 1-D discrete cosine transform (DCT) core. The structure of Re-order Register is customized and designed and its output is obtained. The DA-precision bit length is preferred as 9 bits as an alternative of the usual 12 bits based on test image simulations. the reorder register is implemented in the 2-D DCT architecture and the area is reduced.

ACKNOWLEDGMENT

I express my sincere thanks to Founder President and Chancellor of Engineering College, for their motivation and support which enabled me to do this project. My sincere thanks to principal for their encouragement, support and motivation in all my endeavor's. I express my gratitude to the Head of the Department and Project Co-coordinator, for his/her valuable guidance. I express my profound sense of gratitude to my supervisor Associate Professor, Department of Electronics and Communication Engineering, who guided me with profound reference, high regards and keen interest in guidance. Her vision, enthusiasm and spirit will always inspire and enlighten me. My heartfelt gratitude to my parents, all my friends and supporters for their encouragement and prayers.

REFERENCES

[1] A. M. Shams, A. Chidanandan, W. Pan, and M. A.Bayoumi, “NEDA: Alow-power high-performance DCT architecture,” IEEE Trans. Signal Process., vol. 54, no. 3, pp. 955–964, Mar. 2006.
[2] A. Madisetti and A. N. Willson, “A 100 MHz 2-D 8 X 8 DCT/IDCT processor for HDTV applications,” IEEE Trans. Circuits Syst. Video Technol., vol. 5, no. 2, pp. 158–165, Apr. 1995.
[3] A. Tumeo, M. Monchiero, G. Palermo, F. Ferrandi, and D. Sciuto, “A pipelined fast DDCT accelerator for FPGA-based SOCs,” in Proc. IEEE Comput. Soc. Annu. Symp. VLSI., 2007, pp. 331– 336.
[4] C. C. Sun, P. Donner, and J. Gotze, “Low-complexity multi-purpose IP core for quantized discrete cosine and integer transform,” in Proc. IEEE Int. Symp. Circuits Syst.,2009, pp. 3014–3017.
[5] C. Peng, X. Cao, D. Yu, and X. Zhang, “A 250 MHz optimized distributed architecture of 2D 8 x 8 DCT,” in Proc. Int. Conf. ASIC, 2007, pp. 189–192.
[6] C. T. Lin, Y. C. Yu, and L. D. Van, “Cost-effectivetriple-mode reconfigurable pipeline FFT/IFFT/2-D DCT processor,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, no. 8, pp. 1058–1071, Aug. 2008.
[7] 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.*, pp. 162–166, 2005.
[8] C. Peng, X. Cao, D. Yu, and X. Zhang, “A 250 MHz optimized distributed architecture of 2D 8x8 DCT,” in *Proc. Int. Conf. ASIC*, pp. 189–192, 2007.
[9] Yuan-Ho Chen and Tsin-Yuan Chang,*“*A High Performance Video Transform Engine by Using Space-Time Scheduling Strategy,*” IEEE Trans. Very Large Scale Integration (VLSI) systems,* vol. 20, no. 4, pp.655-664, April 2012.
[10] C. Y. Huang, L. F. Chen, and Y. K. Lai, “A high-speed 2-D transform architecture with unique kernel for multi-standard video applications, in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 21–24, 2008.