ISSN 2319 – 2518 www.ijeetc.com
Special Issue, Vol. 1, No. 1, March 2015
National Level Technical Conference P&E- BiDD-2015
© 2015 IJEETC. All Rights Reserved

Research Paper

DESIGN OF MULTISTANDARD TRANSFORM CORE USING COMMON SHARING DISTRIBUTED ARITHMETIC

A Bharathi[1]*, M Jayabharathi[1], S Keerthana[1] and P Nivetha[1]

Corresponding Author: **A Bharathi, ✉Bharathi15892@gmail.com*

This paper presents a design of low cost and high throughput multistandard transform (MST) core using combination of Factor Sharing and Distributed Arithmetic called Common Sharing Distributed Arithmetic (CSDA) method. CSDA technique reduces the number of adders efficiently compared to the direct implementation by 44.5%.This architecture shares the available hardware resources, so the hardware cost gets reduced. With eight parallel computation paths, the proposed MST core has an eightfold operation frequency throughput rate. CSDA-MST core achieves a high-throughput rate, supporting the (4928 × 2048@24 Hz) digital cinema or ultrahigh resolution format. It supports MPEG-1/2/4 (8 × 8), H.264 (8 × 8, 4 × 4), and VC-1 (8 × 8, 8 × 4, 4 × 8, 4 × 4) transforms. Reduction of gate counts from 28,606 to 23,799 can be achieved here.

Keywords: Common Sharing Distributed Arithmetic (CSDA), Discrete Cosine Transform (DCT), Integer transform, Multistandard transform (MST)

INTRODUCTION

Several video compression standards, e.g., MPEG-2, MPEG-4, H.264 and VC-1 (Windows Media Video 9), are widely used in video codec products, such as digital TV, mobile video, video conference, and so on. The intercommunications between these devices using different standards are so inconvenient, thus video codec that supports multiple standards are more useful and more attractive.

Many researchers have worked on transform core designs that include integer Transform and discrete cosine transform, distributed arithmetic factor sharing technique and matrix decomposition methods to reduce hardware cost. This system includes ROM and accumulators instead of multipliers for reducing area cost. Represents a method for implementing Discrete Cosine Transform (DCT) with distributed arithmetic with dramatic reduction in size of ROM Accumulators (RAC).

[1] Department of Electronics and Communication Engineering, Panimalar Engineering College.

In Factor Sharing (FS) method matrices are derived from same matrix and delta matrix where coefficients in same matrix can share the same hardware resources by factorization, for different standards. This paper proposes a MST core that supports MPEG-1/2/4 (8 × 8), H.264 (8 × 8, 4 × 4), and VC-1 (8 × 8, 8 × 4, 4 × 8, 4 ×4) transforms. The proposed MST core employs DA and FS schemes as Common Sharing Distributed Arithmetic (CSDA) to reduce hardware resources. The main goal is to reduce the nonzero elements using CSDA; thus few adders are required in the adder-tree circuit. According to the mapping strategy, the chosen Canonic Signed Digital (CSD) coefficients achieve excellent sharing capability for hardware resources.

The rest of the paper is organized as follows. The Section II shows the mathematical derivation of the proposed CSDA. Section III addresses the proposed 2-D CSDA-MST architecture, which include derivation of eight-point transform, choice of coefficients, and the proposed 2-D CSDA architecture. Section IV presents comparisons and discussions. Finally, Section V offers a detailed conclusion.

DERIVATION OF COMMON SHARING DISTRIBUTED ARITHMETIC

To achieve resource sharing for inner-product operation effectively, the proposed CSDA combines the FS and DA methods. The FS and DA method are described as follows.

Mathematical Derivation of Factor Sharing

The FS method shares the same factor in different coefficients among the same input.

Consider two separate elements S1 and S2 with the same input X as an example.

S1 = C1X, S2 = C2X.

Assuming that the same factor Fs can be found in the coefficients C1 and C2 can be rewritten as follows:

$$S1 = (Fs2^{k1} + Fd1)X$$

$$S2 = (Fs2^{k2} + Fd2)X$$

where k1 and k2 indicate the weight respectively. Fd1 and Fd2 denote the Remainder coefficients after extracting the shared factor Fs for C1 and C2, respectively.

Fd1 = C1–Fs2^{k1}, Fd2 = C2 – Fs2^{k1}.

Mathematical Derivation of CSD Format Distributed Arithmetic

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

$$Y = \sum_{i=1}^{l} AiXi$$

where Ai is an N-bit CSD coefûcient, and Xi is the input data.

The above equation is written as

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

The inner product computation in the above equation can be implemented using adders and shifters instead of multipliers. CSDA based architecture can achieve a smaller area.

Proposed Common Sharing Distributed Arithmetic (CSDA)

The proposed CSDA algorithm combines the FS and DA methods. By expanding the coefficients matrix at the bit level, the FS method first shares the same factor in each coefficient; the DA method is then applied to share the same combination of the input among each coefficient position. The FS method is adopted first to identify the factors that can achieve higher capability in hardware resource sharing, where the hardware resource in defined as the number of adder usage. Next, the DA method is used to find the shared coefficient based on the results of the FS method. The adder-tree circuits will be followed by the proposed CSDA circuit. Thus, the CSDA method aims to reduce the nonzero elements to as few as possible. The CSDA shared coefficient is used for estimating and comparing thereafter the number of adders in the CSDA loop.

PROPOSED 2-D CSDA-MST CORE DESIGN

Mathematical Derivation of Eight-Point Transform

This section introduces the proposed 2-D CSDA-MST core implementation. Neglecting the scaling factor, the one dimensional (1-D) eight-point transform can be defined as:

$$\begin{bmatrix} Z_0 \\ Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \\ Z_7 \end{bmatrix} = \mathbf{C}\cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

where

$$\mathbf{C} = \begin{bmatrix} c_4 & c_4 & c_4 & c_4 & c_4 & c_4 & c_4 & c_4 \\ c_1 & c_3 & c_5 & c_7 & -c_7 & -c_5 & -c_3 & -c_1 \\ c_2 & c_6 & -c_6 & -c_2 & -c_2 & -c_6 & c_6 & c_2 \\ c_3 & -c_7 & -c_1 & -c_5 & c_5 & c_1 & c_7 & -c_3 \\ c_4 & -c_4 & -c_4 & c_4 & c_4 & -c_4 & -c_4 & c_4 \\ c_5 & -c_1 & c_7 & c_3 & -c_3 & -c_7 & c_1 & -c_5 \\ c_6 & -c_2 & c_2 & -c_6 & -c_6 & c_2 & -c_2 & c_6 \\ c_7 & -c_5 & c_3 & -c_1 & c_1 & -c_3 & c_5 & -c_7 \end{bmatrix}$$

Because the eight-point coefficient structures in MPEG-1/2/4, H.264, and VC-1 standards are the same, the eight-point transform for these standards can use the same mathematic derivation.

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

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

where

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

Moreover, the even part *Ze* can be further decomposed into even and odd parts: *Zee* and *Zeo*

$$\mathbf{Z}_{ee} = \begin{bmatrix} Z_0 \\ Z_4 \end{bmatrix} = \begin{bmatrix} c_4 & c_4 \\ c_4 & -c_4 \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \end{bmatrix} = \mathbf{C}_{ee} \cdot \mathbf{A}$$

$$\mathbf{Z}_{eo} = \begin{bmatrix} Z_2 \\ Z_6 \end{bmatrix} = \begin{bmatrix} c_2 & c_6 \\ c_6 & -c_2 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \end{bmatrix} = \mathbf{C}_{eo} \cdot \mathbf{B}$$

where

$$\mathbf{A} = \begin{bmatrix} a_0 + a_3 \\ a_1 + a_2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} a_0 - a_3 \\ a_1 - a_2 \end{bmatrix}$$

Architecture of the Proposed 2-D CSDA-MST Core

The architecture of the 1-D eight-point MST core is shownin Figure 1, which consists of a selected butterfly (SBF) module,an even part CSDA (CSDA_E), an odd part CSDA (CSDA_O),eight error-compensated error trees (ECATs), and a permutationmodule. Figure 2, shows the proposed 2-D MST core that consists of two 1-D CSDA core and a TMEM(Transposed Memory).The TMEM can be referred in Lai and Lai (2010).

The Figure 3 shows the even part CSDA and the odd part CSDA is shown in Figure 4.

Figure 1: Architecture of 1-D CSDA-MST

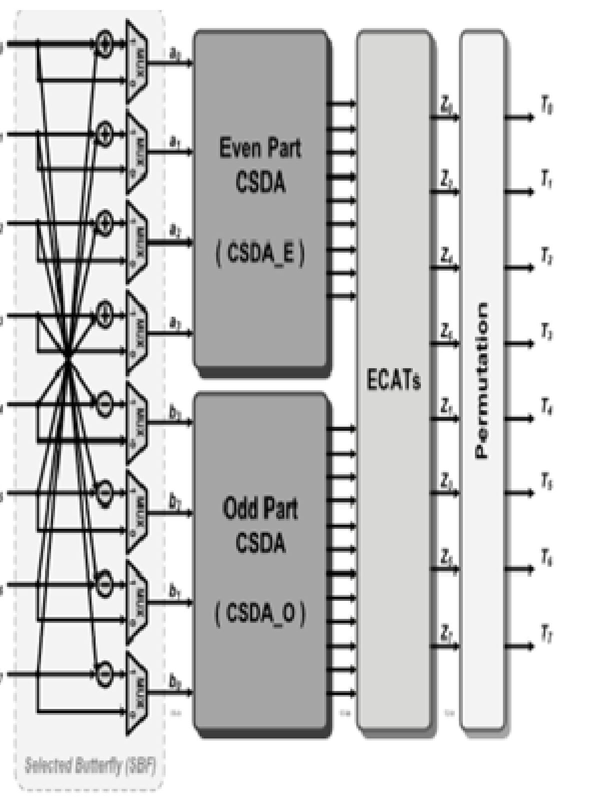


Figure 2: Architecture of 2-D CSDA-MST

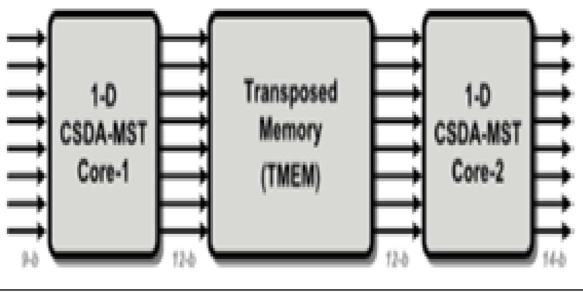


Figure 3: Even Part CSDA

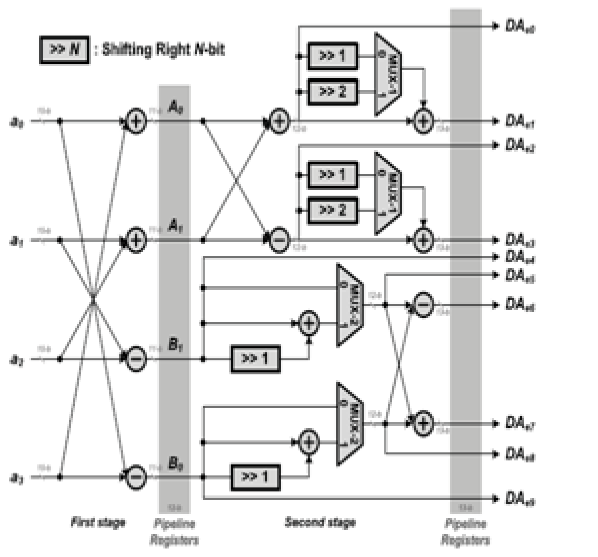
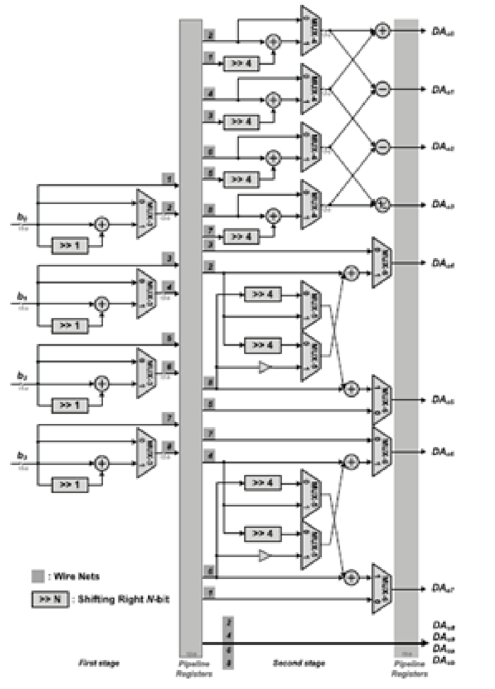


Figure 4: Odd Part CSDA



The transform output *Z*0 can be expressed in distributed arithmetic as follows:

$$\begin{aligned} Z_0 &= c_4 A_0 + c_4 A_1 \\ &= \left(M_1 2^{-2} + M_2 2^{-5} + M_1 2^{-7}\right) A_0 \\ &\quad + \left(M_1 2^{-2} + M_2 2^{-5} + M_1 2^{-7}\right) A_1 \\ &= D_{ee0} 2^{-2} + D_{ee1} 2^{-5} + D_{ee0} 2^{-7} \\ &= DA_{e1} 2^{-2} + DA_{e0} 2^{-5} + DA_{e1} 2^{-7} \end{aligned}$$

where

$$\begin{aligned} DA_{e0} &= D_{ee1} = (A_0 + A_1) M_2 \\ DA_{e1} &= D_{ee0} = (A_0 + A_1) M_1 . \end{aligned}$$

Before the 1-D transform output, the permutation module repermutes the transform outputs *Z* to *T* between eightpoint and four-point transformations.

As with the eight-point transform, the output data *T* are

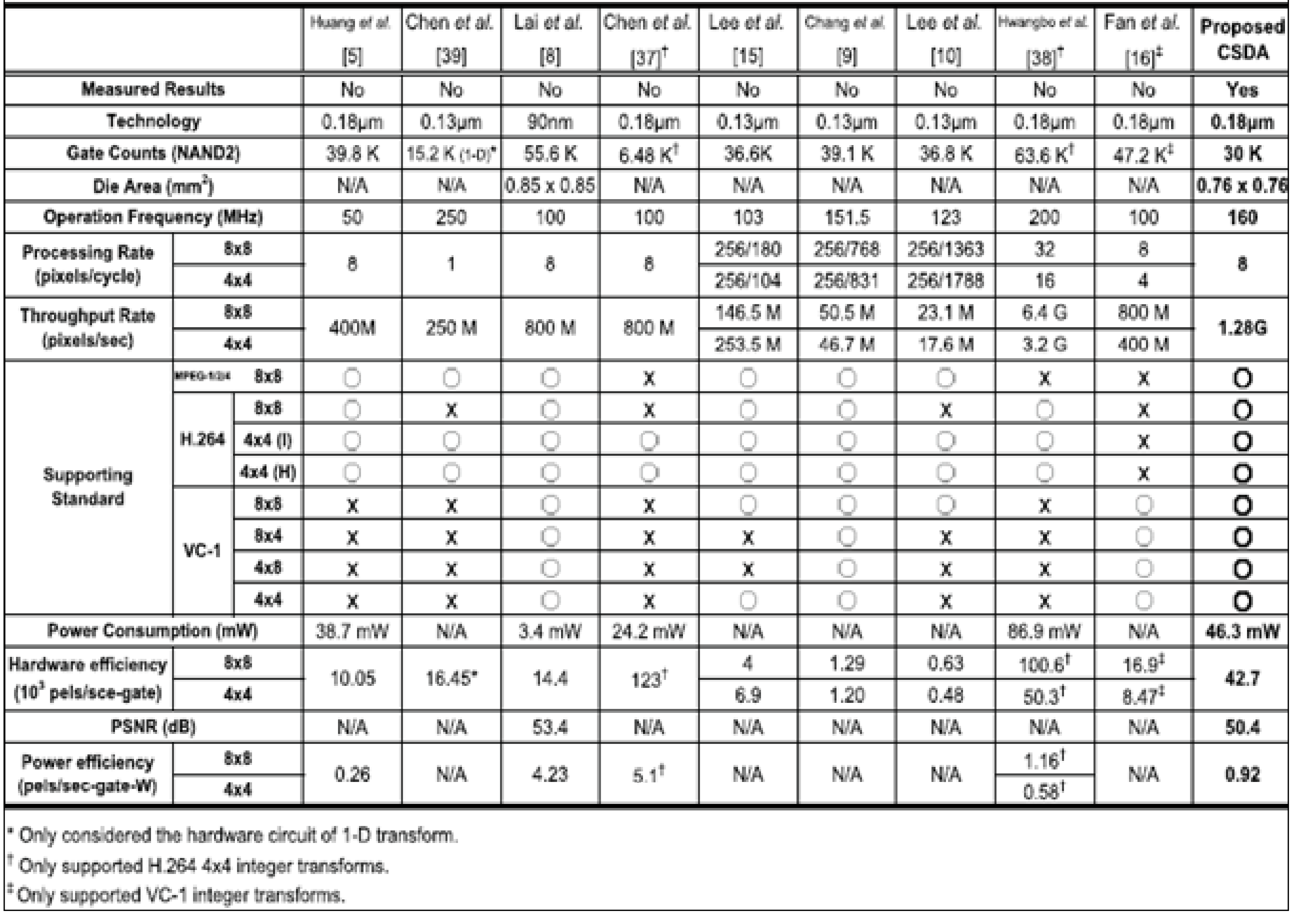
$$[\, T_0 \; T_1 \; \cdots \; T_7 \,] = [\, Z_0 \; Z_1 \; \cdots \; Z_7 \,]$$

And the transform output data *T* for the four-point transform are

$$\begin{aligned} [\, T_0 \; T_1 \; T_2 \; T_3 \,] &= [\, Z_0 \; Z_2 \; Z_4 \; Z_6 \,] \\ [\, T_4 \; T_5 \; T_6 \; T_7 \,] &= [\, Z_1 \; Z_3 \; Z_5 \; Z_7 \,] \end{aligned}$$

Core-1 and Core-2 are different in word length for each arithmetic, MUX, and register, and the TMEM is designed using sixty-four 12-bit registers, where the output data from Core-1 can be transposed and fed into Core-2.

Table 1: Comparisons of Different 2-D Transform Cores with the Proposed Architecture



			Huang *et al.* [5]	Chen *et al.* [39]	Lai *et al.* [8]	Chen *et al.* [37]†	Lee *et al.* [15]	Chang *et al.* [9]	Lee *et al.* [10]	Hwangbo *et al.* [38]†	Fan *et al.* [16]‡	Proposed CSDA
Measured Results			No	No	No	No	No	No	No	No	No	Yes
Technology			0.18μm	0.13μm	90nm	0.18μm	0.13μm	0.13μm	0.13μm	0.18μm	0.18μm	0.18μm
Gate Counts (NAND2)			39.8 K	15.2 K (1-D)*	55.6 K	6.48 K†	36.6K	39.1 K	36.8 K	63.6 K†	47.2 K‡	30 K
Die Area (mm^2)			N/A	N/A	0.85 x 0.85	N/A	N/A	N/A	N/A	N/A	N/A	0.76 x 0.76
Operation Frequency (MHz)			50	250	100	100	103	151.5	123	200	100	160
Processing Rate (pixels/cycle)	8x8		8	1	8	8	256/180	256/768	256/1363	32	8	8
	4x4						256/104	256/831	256/1788	16	4	
Throughput Rate (pixels/sec)	8x8		400M	250 M	800 M	800 M	146.5 M	50.5 M	23.1 M	6.4 G	800 M	1.28G
	4x4						253.5 M	46.7 M	17.6 M	3.2 G	400 M	
Supporting Standard	MPEG-1/2/4	8x8	○	○	○	X	○	○	○	X	X	**O**
	H.264	8x8	○	X	○	X	○	○	X	○	X	**O**
		4x4 (I)	○	○	○	○	○	○	○	○	X	**O**
		4x4 (H)	○	○	○	○	○	○	○	○	X	**O**
	VC-1	8x8	X	X	○	X	○	○	○	X	○	**O**
		8x4	X	X	○	X	X	○	X	X	○	**O**
		4x8	X	X	○	X	X	○	X	X	○	**O**
		4x4	X	X	○	X	○	○	X	X	○	**O**
Power Consumption (mW)			38.7 mW	N/A	3.4 mW	24.2 mW	N/A	N/A	N/A	86.9 mW	N/A	46.3 mW
Hardware efficiency (10^3 pels/sce-gate)	8x8		10.05	16.45*	14.4	123†	4	1.29	0.63	100.6†	16.9‡	42.7
	4x4						6.9	1.20	0.48	50.3†	8.47‡	
PSNR (dB)			N/A	N/A	53.4	N/A	N/A	N/A	N/A	N/A	N/A	50.4
Power efficiency (pels/sec-gate-W)	8x8		0.26	N/A	4.23	5.1†	N/A	N/A	N/A	1.16†	N/A	0.92
	4x4									0.58†		

* Only considered the hardware circuit of 1-D transform.

† Only supported H.264 4x4 integer transforms.

‡ Only supported VC-1 integer transforms.

COMPARISON AND DISCUSSION

Table below shows a comparison of the proposed 2-D CSDAMST core with previous works. In the DA-based row–column decomposition methods, eight parallel computation paths cause the transform core to achieve a throughput rate of 400 M-pels/s when operated at 50-MHz operation frequency by using the DA method to achieve a high throughput rate.

The power consumption is another issue in the circuit designs. In order to have a fair comparison, the power efficiency is defined as throughput rate per power and gate. The proposed CSDA-MST core has medium power consumption between pervious works, and the power efficiency is:

$$\text{Power Efficiency (pels/sec-gate-W)} = \frac{\text{Throughput Rate}}{\text{Power} \times \text{Gate Counts}}$$

The proposed CSDA-MST core can support MPEG-1/2/4 (8 × 8), H.264 (8 × 8, 4 × 4), and VC-1 (8 × 8, 8 × 4, 4 × 8, 4 × 4) multiple transforms by using the CSDA algorithm to reduce area cost. Furthermore, by employing eight parallel computation paths, the CSDA-MST core can achieve a high throughput rate and the four-point transform can achieve the same throughput rate because the CSDA_O can execute four-point transform. Hence the CSDA-MST core has high hardware efficiency when supporting MPEG-1/2/4,H.264, and VC-1 transformations.

CONCLUSION

By using the proposed CSDA method, the number of adders and MUXs in the MST core can be saved efficiently. The CSDA-MST core has good performance, with a high throughput rate and low-cost VLSI design. So far visual media technology has advanced rapidly; this approach will help to meet the rising high-resolution specifications.

REFERENCES

1. Chang H, Kim S, Lee S and Cho K (2009), "Design of Area-Efficient Unified Transform Circuit for Multi-Standard Video Decoder", in Proc. IEEE Int.SoC Design Conf., November, pp. 369-372.
2. Chen Y H, Chang T Y and Li C Y (2011), "High Throughput DA-Based DCT with High Accuracy Error-Compensated Adder Tree", *IEEE Trans. Very Large Scale Integr (VLSI) Syst.*, Vol. 19, No. 4, pp. 709-714.
3. Lai Y K and Lai Y F (2010), "A Reconfigurable IDCT Architecture for Universal Video Decoders", *IEEE Trans. Consum. Electron.*, Vol. 56, No. 3, pp. 1872-1879.
4. Yu S and Swartzlander E E (2001), "DCT Implementation with Distributed Arithmetic", *IEEE Trans. Comput.*, Vol. 50, No. 9, pp. 985-991.