ABSTRACT
In this project we have proposed a low-cost high throughput multistandard transform (MST) core, which can support MPEG 1/2/4 (8 × 8), H.264 (8 × 8 & 4 × 4), and Video Codecs VC-1 (8 × 8, 8 × 4, 4×8 & 4×4) transforms. Common sharing distributed arithmetic (CSDA) algorithm combines factor sharing and distributed arithmetic sharing techniques, to exploit the available resources on FPGAs, which incorporates pipelining and parallel processing of the input samples. With the help of this architecture the throughput of the design is increased. The main strategy is to reduce the nonzero elements using CSDA algorithm. The reduction in adders in the proposed MST is achieved up to 44.5%, compared with the direct implementation method. The proposed MST core has an eightfold operation frequency throughput rate with eight parallel computation paths.

KEYWORDS: Common sharing distributed arithmetic (CSDA), discrete cosine transforms (DCT), FPGA, Multistandard transform (MST)

I. INTRODUCTION
Different transforms are widely used in image and video compression applications. Various groups, such as The International Organization for Standardization (ISO), International Telecommunication Union Telecommunication Standardization Sector (ITU-T), and Microsoft Corporation, have developed different Transform dimensions and coefficients, corresponding to several applications.

Discrete Cosine Transform represents the finite sequences of data points in terms of sum of cosine term functions oscillating at different frequencies. 2D DCT is often used in signal and image processing because of its strong energy compaction property.

Number of researchers have worked on various transform core designs, that includes discrete cosine transform (DCT) as well as integer transform, which uses distributed arithmetic (DA), factor sharing (FS), and matrix decomposition methods for reducing hardware cost. The inner product can be implemented using ROMs and accumulators at place of multipliers that reduces the area cost. Yu and Swartzlander present very efficient method for reducing size of ROMs with recursive DCT algorithms. Delta matrix is used to share hardware resources using the FS method. They generates matrices for multi standards as linear combinations obtained from the same matrix and delta matrix, and show that by factorization the same matrix coefficients can share the same hardware resources.

Recently, reconfigurable architectures have been proposed as a solution that achieves very good flexibility of processors in field-programmable gate array (FPGA) platform or application-specific integrated circuit (ASIC).

II. LITERATURE REVIEW

Yuan-Ho Chen, Jyun-Neng Chen, Tsin-Yuan Chang, and Chih-Wen Lu.
Researchers have worked on transform core designs, including discrete cosine transform (DCT) and integer transform, using distributed arithmetic (DA), factor sharing (FS) and matrix decomposition methods to reduce hardware cost.

A. M. Shams, Peng C et al., 2007; Huang C Y et al., 2008; Chen Y H et al., 2011; Chen Y H et. NEDA
In this paper the researchers have employed a bit-level sharing scheme to construct the adder-based butterfly matrix, called new DA (NEDA). To improve the throughput rate of the NEDA method, high throughput adder trees are introduced.


H. Chang, S. Kim, S. Lee, and K. Cho.,
This paper proposes use of delta matrix to share hardware resources using the FS method. They derive matrices for multi standards as linear combinations from the same matrix and delta matrix, and show that the coefficients in the same matrix can share the same hardware resources by factorization. Matrices for VC-1 transformation can be decomposed into several small matrices, a number of which are identical for different points transforms. Hardware resources can be shared.

4. An architecture and programming model for a resource efficient coarse grained reconfigurable processor.

S. R. Chalamalasetti, S. Purohit, M. Margala, and W. Vanderbauwhede,
This paper proposes the recently reconfigurable architectures have been presented as a solution to achieve a good flexibility of processors in field programmable gate array (FPGA) platform or application-specific integrated circuit (ASIC), such as AP, Ambric, MORA, and Smart Cell. Although these reconfigurable architectures have the feature of flexibility, the pure ASIC design can be recommended for a fixed customer application suitably.

III. MATERIALS AND METHODS
The proposed CSDA combines the FS and DA methods to gain better resource sharing for inner-product operation. The FS and DA method are described in the following.

Fig 1 Block diagram

1. Mathematical Derivation of Factor Sharing
In Factor Sharing method the same factor is shared in different coefficients among the same input. Consider two different elements \( S_1 \) and \( S_2 \) having the same input \( X \) as an example

\[
S_1 = C_1X, \quad S_2 = C_2X.
\]

(1)

Assuming that the coefficients \( C_1 \) and \( C_2 \) having same factor \( F_s \) can be found in the, (1) can be rewritten as follows:

\[
S_1 = (F_s 2k_1 + F_d 1)X \\
S_2 = (F_s 2k_2 + F_d 2)X
\]

(2)

where \( k_1 \) and \( k_2 \) specifies the weight position of the shared factor \( F_s \) in coefficients \( C_1 \) and \( C_2 \), respectively. The remainder coefficients are \( F_d 1 \) and \( F_d 2 \) after extracting the shared factor \( F_s \) forcoefficients \( C_1 \) and \( C_2 \), respectively

\[
F_d 1 = C_1 - F_s 2k_1
\]
2. Mathematical Derivation of CSD Format Distributed Arithmetic

For a general matrix multiplication-and accumulation the inner product can be written as follows:

\[ Y = A^T X = \sum_{i=1}^{L} A_i X_i \]  

(4)

where \( A_i \) is an \( N \)-bit CSD coefficient, and the input data is \( X_i \). Equation (4) can be given as listed below

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

(5)

where \( Y_i = \sum_j A_{i,j} X_i, A_{i,j} \in \{-1, 0, 1\} \), and \( j = 0, \ldots, (N-1) \).

In (5), \( Y \) can be computed by adding or subtracting \( X_i \) with \( A_{i,j} \neq 0 \). The resultant product \( Y \) can then be obtained by shifting and adding every nonzero \( Y_j \). In this manner, the inner product computation in (4) and can be implemented using shifters and adders instead of multipliers. Therefore, CSD DA-based architecture can achieve a more efficient area optimization.

3. Proposed Common Sharing Distributed Arithmetic (CSDA) Algorithm

The proposed CSDA algorithm combines the Factor Sharing (FS) and Distributed Arithmetic (DA) methods. The coefficients matrix is expanded at the bit level then in each coefficient the same factors are shared by the FS method; after that the DA method is applied to share the same input combination for each coefficient position. The proposed CSDA algorithm example in a matrix inner product is given as follows:

\[
\begin{bmatrix}
Y_1 \\
Y_2
\end{bmatrix} = \begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix} \cdot \begin{bmatrix}
X_1 \\
X_2
\end{bmatrix}
\]

(6)

Where \( C_{11} \) and \( C_{22} \) are the coefficients of five-bit CSD numbers

\[
C_{11} = \begin{bmatrix}
1 & -1 & 1 & 0 & 0
\end{bmatrix}
\]

\[
C_{12} = \begin{bmatrix}
1 & -1 & 0 & 0 & 1
\end{bmatrix}
\]

\[
C_{21} = \begin{bmatrix}
1 & 1 & -1 & 0 & 0
\end{bmatrix}
\]

\[
C_{22} = \begin{bmatrix}
0 & 1 & -1 & 0 & 0
\end{bmatrix}
\]

(7)

Fig. 1 shows the flow of proposed CSDA algorithm. In these four coefficients the same shared factor \( F_s \) is \([1 \ -1 \ ]\), and \( C_{11} = C_{22} \) can use \( F_s \) instead of \([1 \ -1 \ ]\), with the corresponding position under the FS method. To share the
same position for the input, and the DA shared coefficient $DA_1$ which is equal to $(X_1 + X_2) F_s$ the DA method is subsequently applied. Finally, the matrix inner products in (6) can be implemented by shifting and adding every nonzero weight position. Fig. 2 shows the coefficient searching flow of the proposed CSDA algorithm. To obtain better resource sharing for inner product operation, the proposed CSDA combines the Factor Sharing and Distributed Arithmetic methods. The FS method is adopted first to find the factors that have higher hardware resource sharing capability, where the hardware resource in this paper is defined as the use of number of adders. Next, the DA method is used to search the shared coefficient based on the result of the Factor Sharing method. The adder-tree circuits will be followed by the designed CSDA circuit. Thus, the CSDA method aims to reduce the nonzero elements. For estimating the number of adders in the CSDA loop the CSDA shared coefficient is used. Therefore, the iteration searching loop requires a large number of iteration loops to determine the smallest hardware resource with the help of these steps, and the CSDA shared coefficient can be generated. Notice that the optimal factor or coefficient in only FS or DA methods is not conducted for the smallest resource in the proposed CSDA method. Thus, a number of iteration loops are needed for determining the better CSDA shared coefficient. An example of the performance in FS, DA, and CSDA methods is shown in a later section. Although the joint DA and sub-expression sharing method, which adopts DA method first for eliminating multipliers and then to reduce the number of adders by the sub-expression sharing method for sharing the same adder operations, is presented in the proposed CSDA method adopts the FS method first to share the same adder operations and then adopts the DA method to reduce the nonzero term, which can reduce the operation of the adder tree. Therefore, the proposed CSDA improves the sharing capacity and is further adopted to implement multiple block sizes and coefficients.

![Fig 2 Example of a CSDA algorithm](image-url)
Fig. 3 Architecture of the proposed 1-D CSDA-MST

The 1-D eight-point MST core architecture is shown in Fig. 3, which consists of a selected butterfly (SBF) module, an even part CSDA (CSDA_E) and an odd part CSDA (CSDA_O), eight error-compensated error trees (ECATs) and a permutation module.

Fig. 4 Architecture of the proposed 1-D CSDA-MST

The designed 2-D CSDA-MST core consists of two 1-D CSDA-MST core with Core-1 and Core-2 having a transposed memory (TMEM), as shown in Fig. 4. Core-1 and Core-2 have different word length for each arithmetic, register, and MUX. The TMEM is designed using sixty-four 12-bit registers, where the output data from Core-1 is transposed and fed into Core-2. Both cores have four pipeline stages: two in the even and odd part CSDA circuit, and two in ECATs. Therefore, the proposed 2-D CSDA-MST core has a latency of 16 clock cycles (= 4 + 8 + 4), and the TMEM executes transposed operation after 12 clock cycles (= 4 + 8) when 8 pixels are input.

IV. RESULTS AND DISCUSSION

In this paper we have proposed a MST core that supports MPEG-1/2/4 (8 × 8), H.264 (8 × 8 & 4 × 4), and Video CodesVC-1 (8 × 8, 8 × 4, 4 × 8 & 4 × 4) transforms. However, the proposed MST core includes DA and FS schemes as common sharing distributed arithmetic (CSDA) that reduces the hardware cost. The main strategy is to reduce the nonzero elements using CSDA algorithm; which results in, few requirement of adders in the adder-tree circuit. According to the proposed designed strategy, the selected canonic signed digital (CSD) coefficients can achieve excellent sharing capability for hardware resources.
V. CONCLUSION

The CSDA-MST core can achieve high performance, with a high throughput rate and low-cost VLSI design, supporting MPEG-1/2/4, H.264, and VC-1 MSTs. By using the proposed CSDA method, the number of adders and MUXs in the MST core can be saved efficiently. Measured results show the CSDA-MST core with a throughput rate of 1.28 G-pels/s, which can support (4928 × 2048@24 Hz) digital cinema format with only 30 k logic gates. Because visual media technology has advanced rapidly, this approach will help meet the rising high-resolution specifications and future needs.

VI. REFERENCES

[1] Kuo-Huang Chang1, Yi-Cheng Chen2, Chung-Cheng Hsieh1, Chi-Wu Huang2 and Chi-Jeng Chang1

[2] *1 Institute of Applied Electronics Technology *2 Department of Industrial Education


"Comparison of Hardware Implementation of Stream Ciphers”


[9] "Image Compression Using Discrete Wavelet Transform” M.Mozammel Hoque Chowdhury and Amina Khatun Department of Computer Science and Engineering Jahangirnagar University


CITE AN ARTICLE