This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2016.2576061, IEEE Transactions on Circuits and Systems II: Express Briefs

A Bitplane Decomposition Matrix Based VLSI Integer Transform Architecture for HEVC

Honggang Qi, *Member, IEEE,* Qingming Huang, *Senior Member, IEEE* and Wen Gao *Fellow, IEEE*

***Abstract*—In this paper, a new VLSI integer transform architecture is proposed for high efficient video coding (HEVC) encoder. The architecture is designed based on the signed bitplane transform (SBT) matrices which are derived from the bitplane decompositions of the integer transform matrices in HEVC. Mathematically, an integer transform matrix can be equally expressed by the binary weighted sum of several SBT matrices which are only composed of binary 0 or ± 1. The SBT matrices are very simple and have lower bitwidth than the original integer transform in the form. The SBT matrices are also sparse in which there are many zero elements. The sparse characteristic of SBT matrices is very helpful for saving the addition operators of SBT. In the proposed architecture, instead of the original integer transform in high bitwidth, the video data can be respectively transformed with the SBT matrices in lower bitwidth. As the result, the delay of transform unit circuit can be significantly reduced with the proposed SBT. Moreover, exploiting the redundant element characteristic of SBT matrices, in which the elements are 0 or ± 1, the adder reuse strategy is proposed for our transform architecture, which can save the circuit area efficiently. The simulation results show that, employing the proposed strategies, the VLSI transform architecture can be synthesized in a proper area with a high working frequency and low latency. The architecture can support all HEVC encoders coding Ultra HD video sequences in real time.**

***Index Terms*—Integer transform, VLSI architecture, HEVC, Bitplane matrix, Ultra HD.**

I. Introduction

DISCRETE cosine transform (DCT) is a key technology for video coding. It is first applied in image coding by Ahmed in 1974 [1]. Subsequently some decades, DCT is widely adopted as a video coding technology. Early video coding standards, such as JPEG [2] MPEG-2/H.262 [3], H.263 [4], employed the real number DCT directly. The real number DCT has to be implemented in float point precision, which is high complexity with up to 64 bitwidth in digital system. It can not be accepted to implement the real number DCT in so high bitwidth. For reducing the complexity of DCT implementation, the integer transforms, the approximated forms of real number DCT, are widely used in the actual encoders replacing the real number DCT. However, the integer transforms were not regulated in standards, which lead to error drifting problems so that greatly deteriorate decoded video. For solving this problem, in the later video standards, such as H.264/AVC [5], AVS [6] and HEVC [7], the explicit integer transforms are defined.

Honggang Qi, Qingming Huang and Wen Gao are with University of Chinese Academy of Sciences, Beijing, China, e-mail: hgqi@ucas.ac.cn. This work was partially supported by the National Science Foundation of China under Grant No.61472388 and 61379100.

Transform is a frequently used module when compressing video, thus the complexity of transform has an important effect on the whole complexity of video encoder. Chen derived the factorization relationship between $N \times N$ and $N/2 \times N/2$ DCT matrices by analyzing the periodic property of cosine function [8]. With the factorization relationship of DCT, the number of arithmetic operations of transform can be reduced. Ahmed et al. [9] decomposed the DCT matrix into sparse submatrices where the multiplications are avoided by using the lifting scheme. Arai et al. [10] proposed AAN fast algorithm based on common factor extraction algorithm in which the complicated common factors were moved from transform kernel to scale part. Only five multipliers are required in the AAN's transform kernel. Multiplier is expensive against adder in integration circuit. Thus, the multiplication operation is usually replaced by adders in the circuit design. Tsui et al. [11] developed an efficient multiplierless FFT-like transform based on a recursive noise model which minimizes the hardware resources of transform maintaining the high performance. In [12], a multiplierless hardware implementation using second-order cone programming technique is presented, and the dynamic ranges of intermediate data are minimized through geometric programming.

HEVC is the latest video coding with higher coding performance than other existing ones. Many novel coding algorithms are introduced in HEVC. Especially in the aspect of transform, up to 32x32 integer transform is applied for improving coding performance. Theoretically, large size transform is usually efficient for coding large size prediction block, vise verse. However, the implementation complexity of transform is increasing with the enlarging transform size. Thus, it is becoming more and more important for reducing the implementation complexity of transform. The 32x32 transform is the most complexity in the transforms of HEVC, thus the improvement of 32x32 transform also can be efficient for the whole transform circuit.

Many researches on transforms implementation optimization for HEVC have been done in the past [13-15]. Meher et al. proposed an efficient constant matrix multiplication scheme to derive parallel architectures of transform for HEVC [13], which can support the real-time ultra HD video codec. In [14], some simplification strategies, such as reuse of transform structure and multiplierless implementation, were adopted for saving the hardware cost. The contribution [15] presented an transform architecture uses the canonical signed digit representation and common sub-expression elimination technique to perform the multiplication with shift-add operation. Based on these optimizations, the transform architecture is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2016.2576061, IEEE Transactions on Circuits and Systems II: Express Briefs

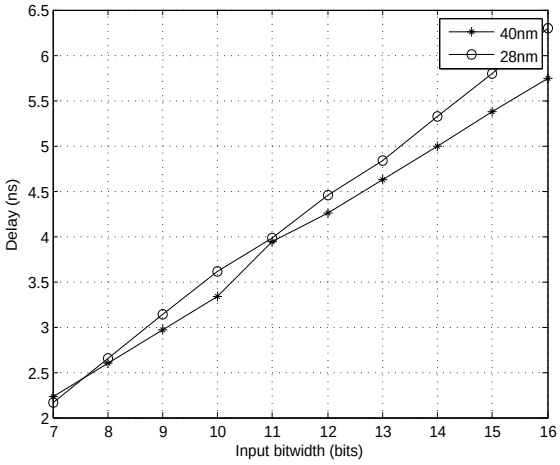

Fig. 1. The delay vs. bitwidth of an adder implemented in 28nm and 40nm CMOS standard cell libraries

greatly simplified for practice application. However, with the increasing applications of HD and ultra HD video coding, the higher processing capacity of codecs are required. Thus, all modules in video codec including the transform need to be further improved for real-time coding with low complexity.

The existing transform architectures consider how to reduce the number of arithmetic operators, such as addition and multiplication, more than the data bitwidth in transform. In fact, the data bitwidth is also an important factor impacting on the circuit speed and area of VLSI architecture. Circuit with large bitwidth needs larger number of fan-in or fan-out of logic gate and more MOS devices are required in the logic gate circuit. Thus, the capacitive load and resistance of logic gate all increase with widening bitwidth. According the first order RC (Resistance and capacitance) circuit model theory, the delay of circuit is related with RC. large RC leads to long circuit delay. The circuit delay varying with the increasing input bitwidth in two typical CMOS processes (SMIC40nm and GF28nm) is shown in Fig.1. As for the adder, the carry chain is the critical path for circuit delay, which is also dependent on the input and output bitwidth. Each extra bit increasing will lead to larger delay. Thus, beside of the number of arithmetic operations, the bitwidth is the other optimization factor for a fast transform architecture. In this paper, we propose a new VLSI architecture for the integer transforms of HEVC standard for reducing the bitwidths of data. The integer transform matrix is decomposed into several signed bitplane transform (SBT) matrices which are used in the proposed architecture. Moreover, a number of adders are reused based on the redundant property of elements of bit matrices. With the bit matrix based transform algorithm, the proposed VLSI transform architecture can process 32 pixels/cycle data throughput maximally with very high working frequency and proper area.

The rest of this paper is organized as follows. In Section II, the proposed transform bitplane decomposition algorithm for optimizing the bitwidth of intermediate data are described and the adder reuse algorithm based on the transform bitplane matrices for reducing the number of adders are introduced. Then, the proposed VLSI transform architecture is presented and the simulation results are shown and discussed in Section III. Finally, we conclude our work in Section IV.

II. Signed Bit Matrix Based Transform Algorithm

A. Bitplane Decomposition of Integer Transform

In order to narrowing the bitwidth of intermediate transformed data, we propose the bit decomposition algorithm which decomposes the integer transform matrix into several SBT matrices.

Let $d_{i,j}$ be the element in ith row and jth column in $N \times N$ integer transform matrix D_N, i.e. $D_N = (d_{i,j})$. If $d_{i,j}$ is positive, the binary expression of $d_{i,j}$ is $(b_{K-1,i,j} \ \cdots \ b_{1,i,j} \ b_{0,i,j})_2,\ b_{k,i,j} \in \{0,1\}$. And then, there is the following relation

$$d_{i,j} = \sum_{k=0}^{K-1} (sgn(d_{i,j}) b_{k,i,j} 2^k),\ b_{k,i,j} \in \{0,1\} \qquad (1)$$

where K is the number of binary significant bits of element $d_{i,j}$, $b_{k,i,j}$ denotes the kth bit of $d_{i,j}$ and $sgn(*)$ is the sign indication function that returns 1 for the positive value and -1 for negative value, i.e. $sgn(x) = \begin{cases} 1, & x > 0 \\ -1, & x < 0 \end{cases}$. Thus, the formula (1) also can be rewritten as

$$d_{i,j} = \sum_{k=0}^{K-1} (b_{k,i,j} 2^k),\ b_{k,i,j} \in \{0, sgn(d_{i,j})\} \qquad (2)$$

The formula (2) is the signed integer binarization. If all elements $d_{i,j}$ in integer transform matrix D_N are binarized and all the kth bits of all binary $d_{i,j}$, $b_{k,i,j}, 0 \le i,j < N$, construct the kth bitplane which is expressed in the form of $N \times N$ SBT matrix $B_{N,k} = (b_{k,i,j})$, there is

$$D_N = \sum_{k=0}^{K-1} (B_{N,k} 2^k) \qquad (3)$$

where K is the binary bitwidth of the maximum element in matrix D_N, i.e. $K = \lceil \log_2^{max(d_{i,j})} \rceil$. The matrix $B_{N,k}$ containing elements of 0 or ± 1 is just the SBT matrix. K SBT matrices are totally generated in the matrix decomposition. Formula (3) specifies the bitplane decomposition from an $N \times N$ integer transform matrix D_N to K $N \times N$ SBT matrices $B_{N,k}$.

Let X_N be the $N \times N$ matrix of input data, according to the bitplane decomposition (3), the integer transform can be equally expressed by the weighted sum of K SBTs as follows

$$D_N X_N = \sum_{k=0}^{K-1} (B_{N,k} 2^k X_N) = \sum_{k=0}^{K-1} (B_{N,k} X_N 2^k) \qquad (4)$$

where $B_{N,k} X_N$ is the kth SBT of input data X_N. Formula (4) specifies the integer transform can be replaced by K SBTs. The final result of integer transform can be equally obtained through weighting (multiplied by 2^k) and accumulating the output data of each SBT as described in formula (4).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2016.2576061, IEEE Transactions on Circuits and Systems II: Express Briefs

Compared with integer transform matrix D_N, the SBT matrices $B_{N,k}$ are simple only including elements of 0 and ± 1. An example for the HEVC 8x8 integer transform D_8 and its the 6th SBT matrix $B_{8,6}$ is shown as follows

$$D_8 = \begin{pmatrix} 90 & 87 & 80 & 70 & 57 & 43 & 25 & 9 \\ 87 & 57 & 9 & -43 & -80 & -90 & -70 & -25 \\ 80 & 9 & -70 & -87 & -25 & 57 & 90 & 43 \\ 70 & -43 & -87 & 9 & 90 & 25 & -80 & -57 \\ 57 & -80 & -25 & 90 & -9 & -87 & 43 & 70 \\ 43 & -90 & 57 & 25 & -87 & 70 & 9 & -80 \\ 25 & -70 & 90 & -80 & 43 & 9 & -57 & 87 \\ 9 & -25 & 43 & -57 & 70 & -80 & 87 & -90 \end{pmatrix} \tag{5}$$

$$B_{8,6} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & -1 & -1 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & 0 & -1 & 1 & 0 & -1 \\ 0 & -1 & 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{pmatrix} \tag{6}$$

As to the matrix multiplication used in transform, if the maximum bitwidth of input data is n bits, the maximum bitwidth of the output data of $N \times N$ 1D transform, $D_N X$, should be $n + \lceil \log_2^{max(\sum_{j=0}^{N-1} |d_{i,j}|)} \rceil$. However, the maximum bitwidth of the output data of kth SBT, $B_{N,k}X_N$, should be $n + \lceil \log_2^{max(\sum_{j=0}^{N-1} |d_{k,i,j})|} \rceil$. Due to $d_{k,i,j} \in \{0, 1, -1\}$, the maximum bitwidth of the output data is no more than $n + \lceil \log_2^N \rceil$. It is easy to be known that $\lceil \log_2^{\sum_{j=0}^{N-1} |d_{i,j}|} \rceil \gg \lceil \log_2^{\sum_{j=0}^{N-1} |d_{k,i,j}|} \rceil$.

Applying the proposed SBT algorithm to the transform architecture, instead of the integer transform matrix circuits, the SBT matrices circuits are implemented and the input data are transformed with each SBT matrix circuits respectively. Due to the simple elements of SBT matrices, the bitwidths of intermediate transformed data and output data are significantly reduced. The bitwidth of output data should be $n + \lceil \log_2^N \rceil$ maximally. Taking 32x32 1D integer transform as an example, the increasing bitwidth of output data is only 5 bits with SBT algorithm, comparing with the 11 bits increasing of straightforward integer transform. The bitwidths of SBT increase slow as the intermediate data processed stage by stage, which shortens circuit delay and constrains the clock cycle smaller. Although, the delay of integer transform circuit is reduced based on the proposed bit transform algorithm, more adders are required due to more SBTs. However, the bitwidths of adders used in SBT are also so low that the addition operation is also very fast. Additional, It can be observed from (6) that many zero elements are in the SBT matrix. The number of actually required addition operations are seldom due to the sparse SBT matrix according to the rule of matrix multiplication. The sparse characteristic of the SBT matrices can benefit for reducing the addition operations in the transform process.

The SBT matrix only contain elements 0 and ± 1. There are many addition operation redundance in SBT. Thus, we propose the adder reuse method based on the element redundance characteristic of SBT matrices for reducing the number of adders in next subsection.

B. Adder Reuse Based on SBT Matrices

If the input data is organized in the form of N dimension column vector, i.e. $\overrightarrow{x} = (x_0, x_1 \cdots, x_{N-1})^T$, the 1D transform of the input data vector can be expressed as

$$D_N \cdot \overrightarrow{x} = \begin{pmatrix} \overrightarrow{d}_0 \\ \overrightarrow{d}_1 \\ \vdots \\ \overrightarrow{d}_{N-1} \end{pmatrix} \cdot \overrightarrow{x} = \begin{pmatrix} \overrightarrow{d}_0 \cdot \overrightarrow{x} \\ \overrightarrow{d}_1 \cdot \overrightarrow{x} \\ \vdots \\ \overrightarrow{d}_{N-1} \cdot \overrightarrow{x} \end{pmatrix} \tag{7}$$

where $\overrightarrow{d}_i = (d_{i,0}, d_{i,1} \cdots, d_{i,N-1})$ is a N dimension row vector which is the ith row elements in transform matrix D_N, and $\overrightarrow{d_i} \cdot \overrightarrow{x} = d_{i,0}x_0 + d_{i,1}x_1 + \cdots + d_{i,N-1}x_{N-1}$. Due to $(d_{i,j})_{10} = (b_{K-1,i,j} \ \cdots \ b_{1,i,j} \ b_{0,i,j})_2$, the kth SBT is

$$B_{N,k} \cdot \overrightarrow{x} = \begin{pmatrix} \overrightarrow{b}_{k,0} \\ \overrightarrow{b}_{k,1} \\ \vdots \\ \overrightarrow{b}_{k,N-1} \end{pmatrix} \cdot \overrightarrow{x} = \begin{pmatrix} \overrightarrow{b}_{k,0} \cdot \overrightarrow{x} \\ \overrightarrow{b}_{k,1} \cdot \overrightarrow{x} \\ \vdots \\ \overrightarrow{b}_{k,N-1} \cdot \overrightarrow{x} \end{pmatrix} \tag{8}$$

where $\overrightarrow{b}_{k,i} = (b_{k,i,0}, b_{k,i,1}, \cdots, b_{k,i,N-1})$ and $\overrightarrow{b}_{k,i} \cdot \overrightarrow{x} = b_{k,i,0}x_0 + b_{k,i,1}x_1 + \cdots + b_{k,i,N-1}x_{N-1}$. If the dimension of vector $\overrightarrow{b}_{k,i}$ is small, it is very probably that $\overrightarrow{b}_{k,i} \cdot \overrightarrow{x} = \overrightarrow{b}_{k,j} \cdot \overrightarrow{x}$. There are many potentially redundant addition operations in $\overrightarrow{b}_{k,i} \cdot \overrightarrow{x}, 0 \le i < N$. Thus, it is reasonable that the adder reuse is efficiently explored in small dimensional vector $\overrightarrow{b}_{k,i}$. However, the dimension of $\overrightarrow{x}$ is usually not very small, such as 16 or 32, which prevents from exploring the element redundance characteristics of SBT. In the proposed adder reuse method, the SBT $\overrightarrow{b}_{k,i} \cdot \overrightarrow{x}$ is split in L parts of sub-SBT $\overrightarrow{b}^l_{k,i} \cdot \overrightarrow{x}^l (0 \le l < L)$, their relationship is

$$\overrightarrow{b}_{k,i} \cdot \overrightarrow{x} = \sum_{l=0}^{L-1} (\overrightarrow{b}^l_{k,i} \cdot \overrightarrow{x}^l) \tag{9}$$

where the number of elements in vector $\overrightarrow{x}^l$ is M, i.e. $dim(\overrightarrow{x}^l) = M$ ($M = N/L$, only considering N divided exactly by L), and $\overrightarrow{b}^l_{k,i} = (b_{k,i,lM}, b_{k,i,lM+1}, \cdots, b_{k,i,(l+1)M-1})$ and $\overrightarrow{x}^l = (x_{lM}, x_{lM+1}, \cdots, x_{(l+1)M-1})^T$. $\overrightarrow{b}^l_{k,i} \cdot \overrightarrow{x}^l$ is named as M dimension sub-SBT (M-SSBT) and the vector $\overrightarrow{b}^l_{k,i}$ is named as M-SSBT vector. Through the split of the SBT matrix, the SSBT with small dimension is obtained. The situations of element combinations in $\overrightarrow{b}^l_{k,i}$ decide the number of adders. The number of all the possible element combinations of M-SSBT vector is α^M, where α denotes the number of all the possible values of an element in vector. For vector $\overrightarrow{b}^l_{k,i}$ from SBT matrix, $b_{k,i,j} \in \Phi = \{0, 1, -1\}$, $\alpha = |\Phi| = 3$, where $|\Phi|$ is the cardinality of set Φ. For reusing adders more efficiently, the M should be smaller. In this architecture, let $M = 2$, i.e. $dim(\overrightarrow{b}^l_{k,i}) = 2$. The SBT

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2016.2576061, IEEE Transactions on Circuits and Systems II: Express Briefs

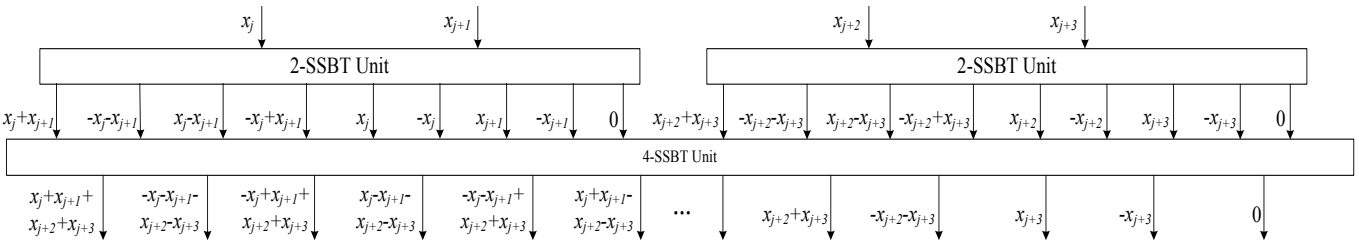

Fig. 2. The hierarchical structure of SBT

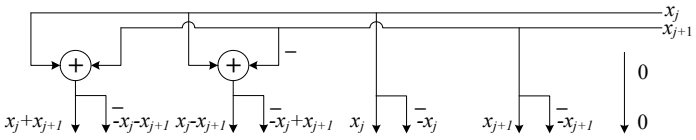

Fig. 3. The adder reuse circuit of 2-SSBT

vector is divided into multiple 2-SSBT vectors. The number of all the possible element combination situations of 2-SSBT is $3^2 = 9$. The 9 element combinations are (1, 1), (1, -1), (-1, 1), (-1, -1), (1, 0), (-1, 0), (0, 1), (0, -1), (0, 0). As for 2-SSBT vector, all possible addition operations for $\overrightarrow{b}^{l}_{k,i} \cdot \overrightarrow{x}^{l}$ can be expressed as

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_j \\ x_{j+1} \end{pmatrix} = \begin{pmatrix} x_j + x_{j+1} \\ x_j - x_{j+1} \\ -x_j + x_{j+1} \\ -x_j - x_{j+1} \\ x_j \\ -x_j \\ x_{j+1} \\ -x_{j+1} \\ 0 \end{pmatrix} \tag{10}$$

It can be seen from (10) that four adders are required in the sub-transform of 2-SSBT. In fact, considering the relationship of positive and negative signs, the all additions of a 2-SSBT are done with only two adders. The adders used in the additions $i_x + i_{x+1}$ and $i_x - i_{x+1}$ can be reused by the additions $-(i_x + i_{x+1})$ and $-(i_x - i_{x+1})$ respectively with two extra negative operators. The negative operator, which is implemented through reversing each bit and then adding 1, is very simple compared with the addition operator with negligible circuit implementation cost. Exploring the addition redundancy of 2-SSBT vector, 2 adders are really required for computing $\overrightarrow{b}^{l}_{k,i} \cdot \overrightarrow{x}^{l}$. The inner circuit design of 2-SSBT unit is described in Fig.2.

According to the relationship between M-SSBT and $M/2$-SSBT, the SBT can be implemented in hierarchical way. A M-SSBT can be implemented through jointing two $M/2$-SSBTs. The hierarchical structure of 4-SSBT as an example is illustrated in Fig.3 where the output of two 2-SSBT unit are input to a 4-SSBT unit for 4-SSBT computation. The inner circuit design of 4-SSBT circuit is also shown in Fig.4.

It can be summarized that the number of adders, #ADDER, used in the proposed adder reuse scheme for SBT can be

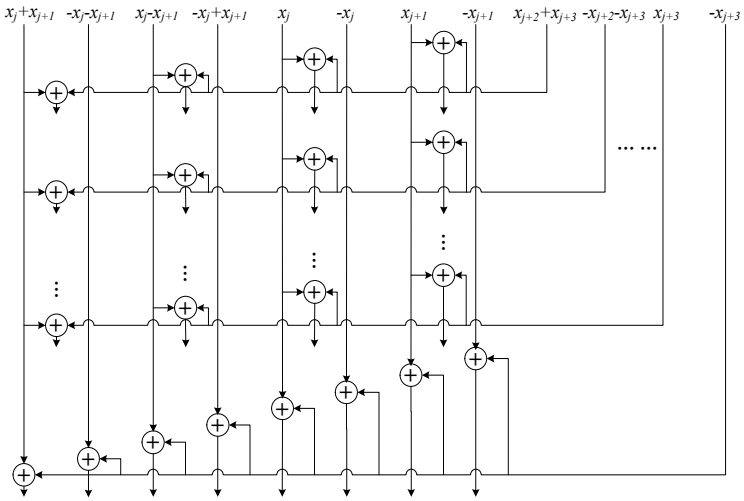

Fig. 4. The part of the adder reuse circuit of 4-SSBT

calculated according to the following expression

$$\#ADDER = \frac{\alpha^M - 1}{2} - (\alpha^{M/2} - 1) \tag{11}$$

Taking 2-SSBT as an example, $M = 2$ and $\alpha = 3$, thus the number of adders of a 2-SSBT is 2.

III. VLSI Architecture and Simulation Results

HEVC adopts several integer transforms in different sizes from 4x4 to 32x32, which are integrated in the proposed transform VLSI architecture. The transform architecture is implemented based on Chen's transform framework [10]. Chen proposed a fast DCT algorithm based on the factorization framework that N-point II-type DCT T_N can be recursively factorized into a $N/2$-point II-type DCT $T_{N/2}$ and a $N/2$-point IV-type DCT $V_{N/2}$. The transform in size N is decomposed into transforms in size $N/2$ recursively. The 32x32 1D transform architecture based on Chen's fast transform algorithm is illustrated in Fig.5, which is the top-level architecture of the proposed 1D transform architecture. The SBT algorithm and corresponding adder reuse algorithm are incorporatively implemented in each $N \times N$ transform unit in Fig.5. For implementing 2D transform, two 1D transforms are respectively implemented and connected by a transpose buffer. The Meher's transpose buffer solution [13] is employed in our 2D transform architecture. The transpose buffer is designed to be capable pipeline the data with only several initial latency cycles, which can guarantee the 2D transform circuit is the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2016.2576061, IEEE Transactions on Circuits and Systems II: Express Briefs

TABLE I
COMPARISONS BETWEEN EXISTING AND PROPOSED 1D DCT ARCHITECTURES

Architecture	Technology (nm)	Gate counts (10^3)	MAX Working frequency (MHz)	Max Throughput (ppc)	Decoding capability
Meher's[13]	90	131	187	16	7680x4320@60fps
Zhao's[14]	45	206	333	32	4096x2048@30fps
Darji's[15]	90	149	100	16	-
Proposed	40	255	450	32	7680x4320@60fps
Proposed	28	223	700	32	7680x4320@60fps

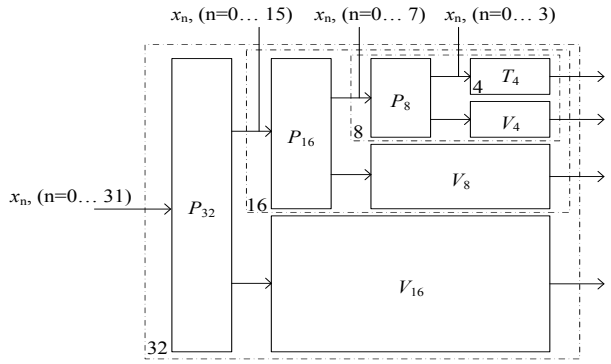

Fig. 5. The 1D 32 × 32 transform top-level architecture

same throughput with the 1D transform circuit. The proposed 1D transform architecture is synthesized with SMIC40nm and GF28nm CMOS standard cell Libraries, which is compared with existing solutions of transform circuits [13-15]. The compared results are shown in Table I. In this table, the circuit area is estimated from the number of transistors by normalizing with respect to a basic 2-input NAND gate. The synthesized results show that, with the 16-bit input data, the proposed architecture costs about 255K logic gates in the maximum working frequency 450 MHz in 40nm CMOS process and 223K logic gates in the maximum working frequency 700 MHz in 28nm CMOS process. The high frequency is achieved from the optimized intermediate data with narrower bitwidth. The working frequency can be further improved through optimizing the critical paths at the cost of increasing circuit area. It can be easily compared from the simulation results that the proposed 1D DCT architecture has almost twice gate count as many as and significant higher working frequency than Meher's [13] and Darji's [15], and has compared circuit area and stronger coding capability than Zhao's [14]. High working frequency is meaningful that the proposed transform circuit can synchronize to other high-frequency modules in video encoder. The working frequency of the encoder modules have to be very high for real-time coding ultra HD video.

IV. Conclusion

A fast integer transform VLSI architecture based sparse SBT is proposed for real-time Ultra HD video coding conforming HEVC standard. Considering the bitwidth effect on circuit delay, the bitwidth of integer transform matrix is optimized in the proposed VLSI architecture. The integer transform matrix with high-bitwidth elements are decomposed into several SBT matrices with low-bitwidth elements based on the proposed matrix Bitplane decomposition method. The transform architecture with the SBT algorithm can work more efficiently for the low-bitwidth computations. The circuit reuse strategy is especially proposed for the SBT to reduce the number of adders of the VLSI architecture. A large number of adders for the SBT is saved using the proposed circuit reuse strategy. The proposed transform hardware architecture can process video data with higher speed and proper area compared with previous work.

References

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "On image processing and a discrete cosine transform", IEEE Transactions on Computer, vol. C-23, (1), pp. 90-93, Jan. 1974.
[2] G. K. William, "JPEG still image data compression standard ", Communications of the ACM, vol. 34, no. 4, pp. 30-44, 1991.
[3] ITU-T Recommendation H.262, ISO/IEC 13818-2 (MPEG-2), "Generic coding of moving pictures and associated audio information Part 2: Video ITU-T and ISO/IEC JTC1", Nov. 1994.
[4] ITU-T Recommendation H.263, "Video coding for low bit-rate communication Version 1", Nov. 1995.
[5] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, Jul. 2003.
[6] Lu Yu, Sijia Chen, and Jianpeng Wang, "Overview of AVS-video coding standards", Signal Processing: Image Communication, vol. 24, no. 4, pp. 247-262, 2009.
[7] G. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard", IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.
[8] W. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform", IEEE Transaction on Communication, vol. com-25, (9), pp. 1004-1009, Sep. 1977.
[9] A. Ahmed, M. U. Shahid, and A. Rehman, "N-point DCT VLSI architecture for emerging HEVC standard", VLSI Design, vol. 2012, article ID 752024, pp. 1C13, 2012.
[10] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images", Transaction IEICE, vol. E71, no. 11, pp. 1095-1097, Nov. 1988.
[11] K. M. Tsui, and S. C. Chan, "Error analysis and efficient realization of the multiplierless FFT-like transformation (ML-FFT) and related sinusoidal transformations", Journal of VLSI Signal Process, vol. 44, no. 1-2, pp. 97-115, 2006.
[12] S. H. Zhao, and S. C. Chan, "Design and multiplierless realization of digital synthesis filters for hybrid-filter-bank A/D converters", IEEE Transactions on Circuits and Systems I, vol. 56, no.10, pp. 2221-2233, 2009.
[13] P. Meher, P. Yoon, B. Mohanty, L. Seong, and Y. Hao, "Efficient integer DCT architectures for HEVC", IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, no. 1, pp. 168-178, Jan. 2014.
[14] W. Zhao, T. Onoye, and T. Song, "High-performance multiplierless transform architecture for HEVC", IEEE International Symposium on Circuits and Systems, pp. 1668-1671, May 2013.
[15] A. D. Darji, and P. M. Raviraj, "High-performance multiplierless DCT architecture for HEVC", IEEE International Symposium on VLSI Design and Test, pp. 1-5, Jun. 2015.