A High Throughput Multiple Transform Architecture for H.264/AVC Fidelity Range Extensions

Yao Ma, Yang Song, Takeshi Ikenaga, and Satoshi Goto

Abstract—**In this paper, a high throughput multiple transform architecture for H.264 Fidelity Range Extensions (FRExt) is proposed. New techniques are adopted which (1) regularize the 8×8 integer forward and inverse DCT transform matrices, (2) divide them into four 4×4 sub-matrices so that simple fast butterfly algorithm can be used, (3) because of the similarity of the sub-matrices, mixed butterflies are proposed that all the sub-matrices of 8×8 and matrices of 4×4 forward DCT (FDCT), inverse DCT (IDCT) and Hadamard transform can be merged together. Based on these techniques, a hardware architecture is realized which can achieve throughput of 1.488Gpixel/s when processing either 4×4 or 8×8 transform. With such high throughput, the design can satisfy the critical requirement of the real-time multi-transform processing of High Definition (HD) applications such as High Definition DVD (HD-DVD) (1920x1080@60Hz) in H.264/AVC FRExt. This work has been synthesized using Rohm 0.18um library. The design can work on a frequency of 93MHz and throughput of 1.488Gpixel/s with a cost of 56440 gates.**

Index Terms—**H.264, DCT, fidelity range extensions (FRExt), multiple transform, HD-DVD**

I. Introduction

The newest video coding standard H.264/AVC provides significantly compression performance for video images. This high compression efficiency is obtained by the adopted many new tools, such as variable block size motion estimation, multiple reference frame motion compression, in-loop deblocking filter, integer discrete cosine transform (DCT) and so on. [1]

Transform coding is an efficient technique in video compression by converting the data into the transform domain. Unlike the previous standards, the integer transform is adopted in H.264/AVC. All operations can be carried out using integer arithmetic, without loss of decoding accuracy, and it is possible to ensure zero mismatch between encoder and decoder inverse transforms. Furthermore, the core part of the transform can be implemented using only additions and shifts without multiplications, which can greatly simplify the hardware implementation.

In July, 2004, a new amendment was added to H.264 standard, which is named as Fidelity Range Extensions (FRExt, Amendment I). The H.264 FRExt demonstrates even further coding efficiency against the existed standard.[2] In this amendment, not only the 4×4 transforms are supported as the initial H.264, but also new 8×8 forward and inverse integer transforms are proposed for high quality video coding to satisfy the High Definition (HD) applications such as HD-DVD and DVB (Digital Video Broadcast). In H.264 FRExt, both 4×4 and 8×8 transforms are adopted. For each macroblock, both the transforms should be calculated and the one with the better cost will be chosen, which further increases the throughput requirement. Therefore, how to develop an efficient VLSI architecture to support all types of 4×4 and 8×8 transforms in H.264/AVC with high throughput is an important research topic.

Up to now, lots of works have been done, but mainly

Manuscript received Nov. 2, 2007; revised Nov. 30, 2007.
Graduate school of Information, Production and Systems, Waseda University, Kitakyushu, Japan
E-mail : yaoma@ruri.waseda.jp

concentrate on one specific transform, such as the designs supporting 4 × 4 transform [3], [4] or 8 × 8 transform [5]. One design [6] realized the multiple transform of both 4 × 4 and 8 × 8 by a unified architecture, but did not provide high throughput.

This paper develops a VLSI which not only provides high throughput but also supports all types of the 4×4 and 8×8 transforms in H.264 Fidelity Range Extensions, which fulfils the requirements of transform processing in H.264 FRExt. The following part is organized as follows. In section II, we give an overview of both 4×4 and 8×8 transforms. The proposed architecture is shown in section III. Section IV will discuss the implementation and the comparison with previous work. Finally, we give a conclusion in Section V.

II. Overview of Transforms in H.264

There are three types of 4 × 4 transforms in H.264/AVC, as FDCT, IDCT and Hadamard transform. They are all two-dimensional transforms which can be denoted as $Y = HXH^T$. The matrices of the first dimensional transform of forward and inverse DCT are shown in Equation (1).

$$\begin{bmatrix} Y0 \\ Y1 \\ Y2 \\ Y3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} X0 \\ X1 \\ X2 \\ X3 \end{bmatrix}, \begin{bmatrix} X0 \\ X1 \\ X2 \\ X3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} Y0 \\ Y1 \\ Y2 \\ Y3 \end{bmatrix} \quad (1)$$

The Hadamard transform is used for the DC coefficients in intra macroblocks predicted in 16 × 16 mode. The transform matrix is shown in Equation (2). The forward and inverse Hadamard transform are the same, for it is a symmetric transform matrix.

$$\begin{bmatrix} X0 \\ X1 \\ X2 \\ X3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} Y0 \\ Y1 \\ Y2 \\ Y3 \end{bmatrix} \quad (2)$$

In H.264/AVC FRExt proposal, the integer 8 × 8 FDCT matrix H is as the follow Equation (3), and the 8×8 IDCT matrix is equal to H^T.

$$\begin{bmatrix} Y0 \\ Y1 \\ Y2 \\ Y3 \\ Y4 \\ Y5 \\ Y6 \\ Y7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \frac{3}{2} & \frac{5}{4} & \frac{3}{4} & \frac{3}{8} & -\frac{3}{8} & -\frac{3}{4} & -\frac{5}{4} & -\frac{3}{2} \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 & -1 & -\frac{1}{2} & \frac{1}{2} & 1 \\ \frac{5}{4} & -\frac{3}{8} & -\frac{3}{2} & -\frac{3}{4} & \frac{3}{4} & \frac{3}{2} & \frac{3}{8} & -\frac{5}{4} \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ \frac{3}{4} & -\frac{3}{2} & \frac{3}{8} & \frac{5}{4} & -\frac{5}{4} & -\frac{3}{8} & \frac{3}{2} & -\frac{3}{4} \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} & -\frac{1}{2} & 1 & -1 & \frac{1}{2} \\ \frac{3}{8} & -\frac{3}{4} & \frac{5}{4} & -\frac{3}{2} & \frac{3}{2} & -\frac{5}{4} & \frac{3}{4} & -\frac{3}{8} \end{bmatrix} \begin{bmatrix} X0 \\ X1 \\ X2 \\ X3 \\ X4 \\ X5 \\ X6 \\ X7 \end{bmatrix} \quad (3)$$

III. 8×8 Transform Optimization and Mixed Butterfly Algorithm

To regularize the 8×8 FDCT transform matrix, we use permutation matrix P as shown in Equation (4). This matrix can be easily implemented by wires.

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4)$$

Because the P is unitary matrix satisfying $P \cdot P^T = 1$, we can rewrite the FDCT as Equation (5).

$$Y = HXH^T = HPP^T XPP^T H^T = H'X'H'^T \quad (5)$$

Where

$$H' = HP \, , \; X' = P^T XP \quad (6)$$

Then the regularized FDCT is shown as Equation (7).

$$H' = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \frac{3}{2} & \frac{3}{8} & -\frac{3}{8} & -\frac{3}{2} & \frac{5}{4} & \frac{3}{4} & -\frac{3}{4} & -\frac{5}{4} \\ 1 & -1 & -1 & 1 & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{5}{4} & -\frac{3}{4} & \frac{3}{4} & -\frac{5}{4} & -\frac{3}{8} & -\frac{3}{2} & \frac{3}{2} & \frac{3}{8} \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \frac{3}{4} & \frac{5}{4} & -\frac{5}{4} & -\frac{3}{4} & -\frac{3}{2} & \frac{3}{8} & -\frac{3}{8} & \frac{3}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -1 & 1 & 1 & -1 \\ \frac{3}{8} & -\frac{3}{2} & \frac{3}{2} & -\frac{3}{8} & -\frac{3}{4} & \frac{5}{4} & -\frac{5}{4} & \frac{3}{4} \end{bmatrix} \quad (7)$$

We divide this regularized FDCT matrix into 4 sub-matrices each of which is shown as the following Equation (9) and Equation (10).

$$H' = \begin{bmatrix} H'1 & H'2 \\ H'3 & H'4 \end{bmatrix} \tag{8}$$

$$H'1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \frac{3}{2} & \frac{3}{8} & -\frac{3}{8} & -\frac{3}{2} \\ 1 & -1 & -1 & 1 \\ \frac{5}{4} & -\frac{3}{4} & \frac{3}{4} & -\frac{5}{4} \end{bmatrix}, \quad H'2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \frac{5}{4} & \frac{3}{4} & -\frac{3}{4} & -\frac{5}{4} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ -\frac{3}{8} & -\frac{3}{2} & \frac{3}{2} & -\frac{3}{8} \end{bmatrix} \tag{9}$$

$$H'3 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \frac{3}{4} & \frac{5}{4} & -\frac{5}{4} & -\frac{3}{4} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{3}{8} & -\frac{3}{2} & \frac{3}{2} & -\frac{3}{8} \end{bmatrix}, \quad H'4 = \begin{bmatrix} -1 & -1 & -1 & -1 \\ -\frac{3}{2} & \frac{3}{8} & -\frac{3}{8} & \frac{3}{2} \\ -1 & 1 & 1 & -1 \\ -\frac{3}{4} & \frac{5}{4} & -\frac{5}{4} & \frac{3}{4} \end{bmatrix} \tag{10}$$

We can see that all these 4×4 matrices including the 4×4 FDCT and Hadamard transform have the similar formats which can be expressed as in Equation (11) where $h_0 = 1, -1$, $h_3 = 1, -1, 1/2$. So we proposed a mixed fast butterfly algorithm for FDCT as shown in Fig. 1.

$$H = \begin{bmatrix} h_0 & h_0 & h_0 & h_0 \\ h_1 & h_2 & -h_2 & -h_1 \\ h_3 & -h_3 & -h_3 & h_3 \\ h_4 & h_5 & -h_5 & -h_4 \end{bmatrix} \tag{11}$$

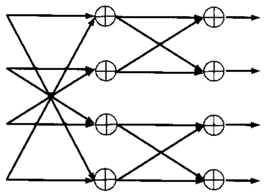


Fig. 1. Mixed butterfly algorithm for FDCT.

The coefficient a, b, c and d in Figure 1 stand for the coefficients in DCT matrix such as (+-)3/2, (+-)3/8, (+-) 5/4, (+-)3/4. All the coefficients can be implemented using only additions and shifters.

Similarly, the mixed butterfly algorithm for IDCT can also be proposed.

Because the 8×8 IDCT matrix is the transpose of 8×8 FDCT matrix, so the permutation matrix for 8×8 IDCT will be P^T as Equation (12).

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{12}$$

We rewrite the IDCT equation as Equation (13).

$$X = H^T Y H = PP^T H^T Y H P P^T = PH'^T YH'P^T \tag{13}$$

So the regularized IDCT matrix H'^T is shown as Equation (14).

$$H'^T = \begin{bmatrix} 1 & \frac{3}{2} & 1 & \frac{5}{4} & 1 & \frac{3}{4} & \frac{1}{2} & \frac{3}{8} \\ 1 & \frac{3}{8} & -1 & -\frac{3}{4} & 1 & \frac{5}{4} & -\frac{1}{2} & -\frac{3}{2} \\ 1 & -\frac{3}{8} & -1 & \frac{3}{4} & 1 & -\frac{5}{4} & -\frac{1}{2} & \frac{3}{2} \\ 1 & -\frac{3}{2} & 1 & -\frac{5}{4} & 1 & -\frac{3}{4} & \frac{1}{2} & -\frac{3}{8} \\ 1 & \frac{5}{4} & \frac{1}{2} & -\frac{3}{8} & -1 & -\frac{3}{2} & -1 & -\frac{3}{4} \\ 1 & \frac{3}{4} & -\frac{1}{2} & -\frac{3}{2} & -1 & \frac{3}{8} & 1 & \frac{5}{4} \\ 1 & -\frac{3}{4} & -\frac{1}{2} & \frac{3}{2} & -1 & -\frac{3}{8} & 1 & -\frac{5}{4} \\ 1 & -\frac{5}{4} & \frac{1}{2} & \frac{3}{8} & -1 & \frac{3}{2} & -1 & \frac{3}{4} \end{bmatrix} \tag{14}$$

The equation of dividing this regularized IDCT matrix into 4 sub-matrices is as Equation (15). And the four sub-matrices is shown as Equation (16) and Equation (17).

$$H'^T = \begin{bmatrix} H'^T_1 & H'^T_3 \\ H'^T_2 & H'^T_4 \end{bmatrix} \tag{15}$$

$$H'^T 1 = \begin{bmatrix} 1 & \frac{3}{2} & 1 & \frac{5}{4} \\ 1 & \frac{3}{8} & -1 & -\frac{3}{4} \\ 1 & -\frac{3}{8} & -1 & \frac{3}{4} \\ 1 & -\frac{3}{2} & 1 & -\frac{5}{4} \end{bmatrix}, \quad H'^T 2 = \begin{bmatrix} 1 & \frac{5}{4} & \frac{1}{2} & -\frac{3}{8} \\ 1 & \frac{3}{4} & -\frac{1}{2} & -\frac{3}{2} \\ 1 & -\frac{3}{4} & -\frac{1}{2} & \frac{3}{2} \\ 1 & -\frac{5}{4} & \frac{1}{2} & \frac{3}{8} \end{bmatrix} \tag{16}$$

$$H^{\prime T}3=\begin{bmatrix}1 & \frac{3}{4} & \frac{1}{2} & \frac{3}{8}\\ 1 & \frac{5}{4} & -\frac{1}{2} & -\frac{3}{2}\\ 1 & -\frac{5}{4} & -\frac{1}{2} & \frac{3}{2}\\ 1 & -\frac{3}{4} & \frac{1}{2} & -\frac{3}{8}\end{bmatrix},\quad H^{\prime T}4=\begin{bmatrix}-1 & -\frac{3}{2} & -1 & -\frac{3}{4}\\ -1 & \frac{3}{8} & 1 & \frac{5}{4}\\ -1 & -\frac{3}{8} & 1 & -\frac{5}{4}\\ -1 & \frac{3}{2} & -1 & \frac{3}{4}\end{bmatrix} \tag{17}$$

The formats of all the sub-matrices including the 4×4 IDCT matrix can all be expressed as Equation (18). So, the mixed fast butterfly algorithm for IDCT is proposed as shown in Fig. 2.

$$H=\begin{bmatrix}h_0 & h_1 & h_3 & h_4\\ h_0 & h_2 & -h_3 & h_5\\ h_0 & -h_2 & -h_3 & -h_5\\ h_0 & -h_1 & h_3 & -h_4\end{bmatrix} \tag{18}$$

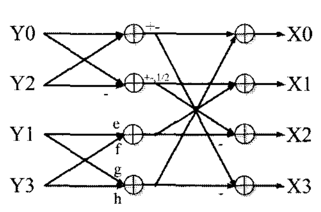


Fig. 2. Mixed butterfly algorithm for IDCT.

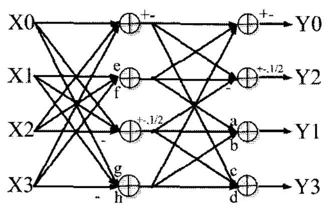


Fig. 3. Mixed butterfly algorithm for multiple transforms.

For the similarity of the mixed butterfly of FDCT and IDCT, we can merge them together to obtain a mixed butterfly algorithm for multiple transforms. The merged transform butterfly algorithm is shown in Fig. 3. The coefficients as a~h in this figure are different due to different matrices and can be selected by multiplexers in the merged architecture. And for the multiple transform processing, we can select the coefficients which are necessary for the current calculation.

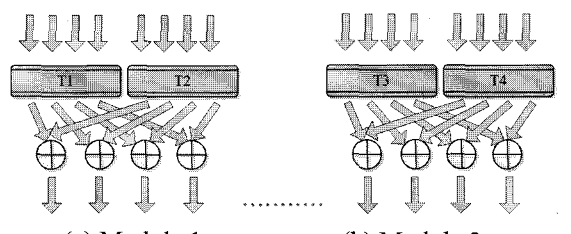
IV. PROPOSED ARCHITECTURE

4.1 Transform Flow Analysis and Module Proposal

With these fast butterfly algorithms, we can calculate the 8×8 FDCT using the 4×4 sub-matrices and similarly the 8×8 IDCT. Therefore, we can design a hardware architecture that provides the processing of all the transforms in H.264/AVC FRExt including the 4×4 DCT, IDCT and Hadamard transform.

Firstly, we discuss the process of 8×8 transforms in detail. The calculation of the first dimensional transform can be written as the following Equation (16).

$$\begin{aligned}HX &= \begin{bmatrix}H_1 & H_2\\ H_3 & H_4\end{bmatrix}\begin{bmatrix}X_1 & X_2\\ X_3 & X_4\end{bmatrix}\\ &=\begin{bmatrix}H_1X_1+H_2X_3 & H_1X_2+H_2X_4\\ H_3X_1+H_4X_3 & H_3X_2+H_4X_4\end{bmatrix}\\ &=\begin{bmatrix}X_{m1} & X_{m2}\\ X_{m3} & X_{m4}\end{bmatrix}=X_m\end{aligned} \tag{19}$$



(a) Module 1 (b) Module 2

Fig. 4. Module for transform.

To accomplish the calculation in Equation (19), we propose two modules of transform as Module1 and Module2 as shown in Fig. 3. In Module1, we merge the butterfly of the first sub-matrix H_1 of both 8×8 FDCT and IDCT to T_1, second sub-matrix H_2 to T_2. In Module2, we merge the butterfly of the third sub-matrix H_1 of both 8×8 FDCT and IDCT to T_3, the fourth sub-matrix H_4 to T_4. If we input the column from X_1 & X_3 or X_2 & X_4 to T_1 & T_2 in Module1, we get the column of X_{m1} or X_{m2}. If we input the column from X_1 & X_3 or X_2 & X_4 to T_3 & T_4 in Module2, we get the column of X_{m3} or X_{m4}. Thus, it can process the first dimensional 8×8 FDCT and IDCT.

Secondly, we merge the butterfly of 4×4 FDCT, IDCT and Hadamard transform to T_1 in Module1 and T_4 in Module2. Then if we input the column of 4×4 block to T_1 or T_4, it can process the first dimensional 4×4 DCT, IDCT and Hadamard transform in either Module1 or Module2, while T_2 and T_3 can be out of power.

The second dimensional 8×8 transform is shown as

Equation (20), X_m denotes the result of the first dimensional transform calculation.

$$X_m H^T = \begin{bmatrix} X_{m1} & X_{m2} \\ X_{m3} & X_{m4} \end{bmatrix} \begin{bmatrix} H^T{}_1 & H^T{}_3 \\ H^T{}_2 & H^T{}_4 \end{bmatrix} = \begin{bmatrix} X_{m1}H^T{}_1 + X_{m2}H^T{}_2 & X_{m1}H^T{}_3 + X_{m2}H^T{}_4 \\ X_{m3}H^T{}_1 + X_{m4}H^T{}_2 & X_{m3}H^T{}_3 + X_{m3}H^T{}_4 \end{bmatrix} = \begin{bmatrix} Y_1 & Y_2 \\ Y_3 & Y_4 \end{bmatrix} = Y \quad (20)$$

From Equation (20), we see that if we input the row from X_{m1} & X_{m2} or X_{m3} & X_{m4} to T_1 & T_2 in Module1, we get the row of Y_1 or Y_3. If we input the row from X_{m1} & X_{m2} or X_{m3} & X_{m4} to T_3 & T_4 in Module2, we get the row of Y_2 or Y_4. In this way, it can process the second dimensional 8 × 8 FDCT and IDCT.

Similarly to the first dimensional processing, T_1 in Module1 and T4 in Module2 take charge of the second dimensional transforms. If we input row of the first dimensional transform result to T_1 & T_4, it can process 4×4 FDCT, IDCT and Hadamard transform. So, all the transforms in H.264/AVC FRExt are supported by Module1 and Module2.

4.2 2-Dimensional Architecture

Based on the mixed butterfly algorithm and the two merged transform module, we propose a 2-Dimensional transform architecture. The proposed 2-D architecture is shown in Fig. 5.

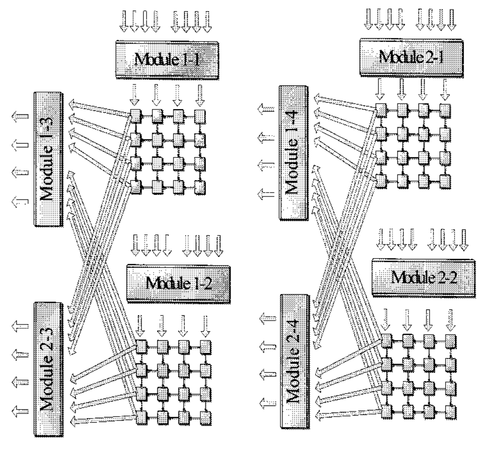


Fig. 5. 2-Dimensional Architecture.

In this architecture, we use four 4×4 register arrays as transpose matrices. Each transpose element contains one register to store the data and one multiplexer to choose data from left, right or itself. Finally, there are four multiplexer to select data from horizontal direction or vertical direction. The structure of each 4×4 register array is similar to the design in [3], and it is shown in Fig. 6.

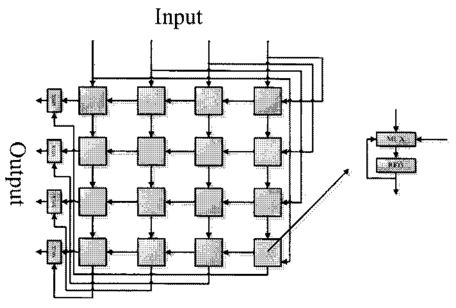


Fig. 6. Structure of each 4×4 register array.

In this 2-Dimensional transform architecture, when processing 8×8 transform, the inputs of Module1-1 and Module2-1 are columns from X_1 and X_3, and the inputs of the Module1-2 and Module2-2 are columns from X_2 and X_4. After four cycles, the results of first dimensional transform are stored in the four transform matrices. Then the transpose elements overlapped from horizontal to vertical direction or reversed to do the second dimensional transform. So the inputs of Module1-3 and Module2-3 are rows from X_{m1} and X_{m2}, and the inputs of the Module1-4 and Module2-4 are rows from X_{m3} and X_{m4}. The outputs of the Module 1-3 and Module2-3 are the rows of Y_1 and Y_2, and the output of the Module1-4 and Module2-4 are the rows of Y_3 and Y_4. Thus totally two column/rows of the 8 × 8 data matrix are inputted/outputted in each cycle. That means we can process 16 pixels in each cycle for 8×8 transform.

When processing 4 × 4 transform, only part of each module is used as T_1 for Module1 and T_4 for Module2. We input one row/column from the different 4×4 block to each of the first dimensional modules as Module1-1, Module2-1, Module1-2 and Module2-2 simultaneity and also get one row/column output from each of the second dimensional modules as Module1-3, Module2-3, Module1-4 and Module2-4 in each cycle. So we can also process 16 pixels in each cycle for 4×4 transform. Overall, this design has a processing capability of 16 pixels per cycle.

Table 1. Performance Comparison.

	Types	Latency (cycle)	Frequency (MHz)	Troughput (Mpixel/s)	Technology (μm)	Area (gates)
Wang [3]	4×4 Multiple	4	80	320	TSMC 0.35	6538
Chen [4]	4×4 Multiple	2	100	800	TSMC 0.18	6482
Amer [5]	8×8 Forward	8	68.5	N/A	N/A	29018(LUT)
Bruguera [6]	4×4 and 8×8 Multiple	8	66	266	AMS 0.35	23800
Proposed	4×4 and 8×8 Multiple	4	93	1488	ROHM 0.18	56440

V. Implementation and Comparison

This work has been coded in Verilog-HDL and synthesized using Rohm 0.18um library. The results are shown in Table 1. After synthesis, the realized design can reach a frequency of 93MHz with a throughput of 1.488Gpixel/s. We use a little more than 2 times of the area to get more than 5 times of throughput comparing to the previous 8×8 multiple one [6]. In that design, 8x8 transpose matrix is used while only subset of the transpose matrix is used when processing 4 × 4 transforms. But in our design, when processing 4x4 transforms, four 4 × 4 transform matrices are used simultaneously. So in our design, the hardware is more efficiently used.

VI. Conclusions

In this paper, a high performance architecture for multiple transforms in H.264/AVC FRExt is proposed with matrix division techniques. By this architecture, we improve the throughput significantly with a acceptable hardware cost. It can satisfy the high performance requirement of High Definition applications such as HD-DVD. This architecture can process all types of 4×4 and 8 × 8 transforms. It can realize the real-time multi-transform processing in H.264/AVC FRExt.

Acknowledgments

This work was supported by fund from the CREST, JST.

References

[1] C. Lauterback, W. Weber, and D. Romer, "Charge-sharing concept and new clocking scheme for power efficiency and electromagnetic emission improvement of boosted charge pumps," *IEEE J. Solid State Circuits,* vol.35, pp.719-723, May. 2000.

[2] T. Tanzawa and S. Atsumi, "Optimization of word-line booster circuits for low-voltage flash memories," *IEEE J. Solid State Circuits,* vol.34, pp.1091-1098, Aug. 1999.

[3] T. Wiegand, G. J. Sullivan, G. Bjtegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Trans. on CSVT,* vol. 13, pp. 560– 576, Jun. 2003.

[4] D. Marpe, T. Wiegand, and S. Gordon, "H.264/MPEG4-AVC Fidelity Range Extensions: Tools, Profiles, Performance, and Application Areas," *IEEE Int. Conf. on Image Processing (ICIP),* pp. 593– 596, Sep. 2005.

[5] T-C. Wang, Y-W. Huang, H-C. Fang, and L-G. Chen, "Parallel 4x4 2D Transform and Inverse Transform Architecture for MPEG-4 AVC/H.264," *IEEE Int. Symp. on Circuits and Systems,* pp. 800– 803, May. 2003

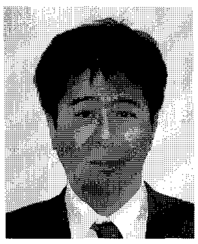
[6] K. Chen and J. Guo, "A High-Performance Direct 2-D Transform Coding IP Design for MPEG-4 AVC/H.264," *IEEE Trans on CSVT,* vol. 16, pp. 472-483, Apr. 2006.

[7] I. Amer, W. Badawy, and G. Jullien, "A High-Performance Hardware Implementation of the H.264 Simplified 8x8 Transformation and Quantization," *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing,* Vol. 2, pp. 1137-1140, Mar. 2005.

[8] J. Bruguera and R. Osorio, "A Unified Architecture for H.264 Multiple Block-size DCT with Fast and Low Cost Quantization," *EUROMICRO Conference on Digital System Design (DSD),* pp. 407-414, 2006.

Yao Ma was born in Beijing, China, 1983. He received the B.E. Degree in Automation from Tsinghua University in 2005 respectively. He is currently working towards the M.E. Degree in LSI System in Graduate School of Information, Production and Systems, Waseda University, Japan. His research interests include H.264/AVC video coding and associated VLSI architectures.

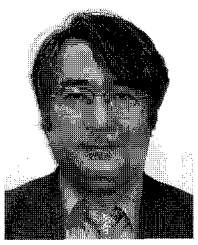
Yang Song received the B.E. degree in Computer Science from Xi'an Jiaotong University, China in 2001, the M.E degree in Computer Science from Tsinghua University, China in 2004 and the Ph.D degree in Waseda University, Japan in 2007. He currently is a visiting research associate in Waseda University. His research interest includes motion estimation, video coding technology and associated VLSI architecture.



Takeshi Ikenaga received his B.E. and M.E. degrees in electrical engineering and Ph. D degree in information & computer science from Waseda University, Tokyo, Japan, in 1988, 1990, and 2002, respectively.

He joined LSI Laboratories, Nippon Telegraph and Telephone Corporation (NTT) in 1990, where he had been undertaking research on the design and test methodologies for high performance ASICs, a real-time MPEG2 encoder chip set, and a highly parallel LSI & system design for image-understanding processing. He is presently an associate professor in the system LSI field of the Graduate School of Information, Production and Systems, Waseda University. His current interests are application SoCs for image, security and network processing. Especially, he engages in the research on H.264 encoder LSI, JPEG2000 codec LSI, LDPC decoder LSI, UWB wireless communication LSI, public key encryption LSI, object recognition LSI, etc.

Dr. Ikenaga is a member of the IEEE, the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan, the Information Processing Society of Japan and the Institute of Image Electoronics Engineers of Japan.



Satoshi Goto was born in Hiroshima, Japan, 1945. He received the B.E. and the M.E. Degrees in Electronics and Communication Engineering from Waseda University in 1968 and 1970 respectively. He also received the Dr. of Engineering from the same University in 1981. He joined NEC Laboratories in 1970 where he worked for LSI design, Multimedia system and Software as GM and Vice President. Since 2003, he has been Professor, at Graduate school of Information, Production and Systems of Waseda University at Kitakyushu. His main interest is now on VLSI design methodologies for multimedia and mobile applications. H has published 7 books, 38 journal papers, 67 international conference papers with reviews. He served as GC of ICCAD and ASPDAC and was a board member of IEEE CAS society. He is IEEE Fellow, IEICE Fellow and Member of Academy Engineering Society of Japan.