Journal of Southeast University (English Edition) Vol. 29, No. 3, pp. 276 - 281 Sept. 2013 ISSN 1003—7985

High-performance multi-transform architecture for H. 264/AVC

Wang Gang[1, 2] Wang Qing[1] Li Bing[2] Chen Rui[3]

([1] School of Instrument Science and Engineering, Southeast University, Nanjing 210096, China)
([2] Wuxi Branch, Southeast University, Wuxi 214135, China)
([3] Institute of Electronics, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: In order to increase the hardware utilization and minimize the chip area, a multi-transform coding architecture which includes 4×4 forward integer transform, 4×4 inverse integer transform, 4×4 Hadamard transform and 2×2 Hadamard transform is proposed. By simplifying these transforms and exploring their similarities, the proposed design merges the architectures, processing individual transforms into a high-performance multi-transform coding architecture. Using a semiconductor manufacturing international corporation (SMIC) 0.18 μm complementary metal oxide semiconductor (CMOS) technology, the proposed architecture achieves the maximum operating clock frequency of 200 MHz and the throughput rate of 800×10^6 pixel/s with the hardware cost of 3 704 gates. The results demonstrate that the data throughput rate per unit area (DTUA) of this design is at least 40.28% higher than that of the reference design. This design can meet the requirements of real-time decoding digital cinema video (4 096 × 2 048@30 Hz) at 62.9 MHz, which helps to reduce the power consumption.

Key words: H. 264/AVC; multi-transform architecture; Hadamard transform; integer transform

doi: 10. 3969/j. issn. 1003 - 7985. 2013. 03. 009

To obtain better compression performance, the H. 264/AVC video coding standard has defined various transforms for different prediction modes, such as an 8×8 integer transform, a 4×4 integer transform, a 4×4 Hadamard transform, a 2×2 Hadamard transform and their inverses. The integer transforms, which are similar to a discrete cosine transform (DCT), can be used to avoid mismatch and reduce computation complexity. Among them, the 8×8 integer transform is only adopted in the high profile and the Hadamard transform is utilized for 4×4 array luma DC coefficients of intra 16×16 prediction modes and 2×2 array chroma DC coefficients. This paper focuses on the 4×4 integer transform and the 2×2 Hadamard transform.

Received 2013-01-25.

Biographies: Wang Gang (1977—), male, graduate; Wang Qing (corresponding author), male, doctor, professor, w3398@263. net.

Foundation item: The National Key Technology R&D Program of China during the 12th Five Year Plan Period (No. 2013BAJ05B03).

Citation: Wang Gang, Wang Qing, Li Bing, et al. High-performance multi-transform architecture for H. 264/AVC[J]. Journal of Southeast University (English Edition), 2013, 29(3): 276 - 281. [doi: 10. 3969/j. issn. 1003 - 7985. 2013. 03. 009]

In accordance with the architecture, the existing designs can be divided into three kinds. The first kinds of architectures are the parallel architectures[1-4]. Hong et al.[1] proposed a parallel 4×4 transform architecture based on bit-extended arithmetic. Rubin[2] proposed a parallel architecture based on bit-serial shared memory. Both bit-extended arithmetic and bit serial shared memory are used to improve the processing rate. To realize forward transforms, Porto et al.[3] proposed a parallel architecture which is called T module, and this kind of architecture achieves a very high throughput. And the design of Wang et al.[4] is another parallel architecture to realize multiple transforms. These parallel architectures achieve high throughput at the expense of high area cost. The second kinds of architectures are reconfigurable architectures[5-6]. Cao et al.[5] proposed a reconfigurable 2-D architecture of two novel signal flow graphs (SFG) of 4×4 forward and inverse transforms which achieves a data processing rate up to 16 pixels/cycle, with greatly increased throughput. By the revised design, Cao et al.[6] optimized the architecture to reduce the data processing rate and throughput. Reconfigurable architectures improve area efficiency and flexibility. However, these architectures are very complex. The third kinds of architectures are direct 2-D architectures[7-9]. Chen et al.[7] proposed a direct 2-D transform algorithm and a correspondent direct 2-D transform architecture. Peng et al.[8] combined direct 2-D transform with quantization. Hwangbo et al.[9] realized inverse transform by dividing the 4×4 matrix multiplication into four 2×2 components, utilizing block multiplication instead of 4×4 matrix multiplication which still belongs to direct 2-D architecture. Direct 2-D transform architectures can improve data throughput and efficiency. However, these architectures need more hardware resources. Some architectures to support multi-standard video applications with the adaptive block-size transform are proposed[10-11]. The throughput values of these architectures are not sufficient to satisfy the real-time requirement of transform in real-time decoding digital cinema video.

To obtain better performance at low-cost, a new multi-transform architecture is proposed in this paper. A new algorithm is proposed to integrate a 4×4 forward integer transform, a 4×4 inverse integer transform, a 4×4 Hadamard transform and a 2×2 Hadamard transform into a single block. A low-cost and high-performance architecture is proposed to realize the multi-transform algorithm. Experimental results demonstrate that the data throughput rate per unit area (DTUA) of the proposed architecture is at least 40. 28% higher than the reference design under the area cost of 3 704 gates. In addition, this architecture satisfies the requirements of real-time decoding digital cinema video (4 096 × 2 048@30 Hz).

1 Transform Coding in H. 264

This paper mainly focuses on implementing 4 × 4 transforms and the 2 × 2 Hadamard transform of H. 264/AVC because all these 4 × 4 and 2 × 2 transforms can be used in every H. 264/AVC profile/level combination.

1.1 4 × 4 transforms of H. 264/AVC

The 4 × 4 forward integer transform (FIT), 4 × 4 inverse integer transform (IIT) and the 4 × 4 Hadamard transform (HT) are defined as

$$\boldsymbol{Y}_{\rm f} = \boldsymbol{C}_{\rm f}\boldsymbol{X}\boldsymbol{C}_{\rm f}^{\rm T}, \quad \boldsymbol{Y}_{\rm i} = \boldsymbol{C}_{\rm i}\boldsymbol{X}\mathrm{C}_{\rm i}^{\rm T}, \quad \boldsymbol{Y} = \boldsymbol{H}\boldsymbol{X}\boldsymbol{H}^{\rm T} \tag{1}$$

As can be seen in Eq. (1), the forms of 4 × 4 transforms employed in H. 264/AVC are similar to each other. The transform matrices for the 4 × 4 transforms are given as

$$\boldsymbol{C}_{\rm f} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

$$\boldsymbol{C}_{\rm i} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0.5 & -0.5 & -1 \\ 1 & -1 & -1 & 1 \\ 0.5 & -1 & 1 & -0.5 \end{bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{2}$$

where $\boldsymbol{C}_{\rm f}$, $\boldsymbol{C}_{\rm i}$ and $\boldsymbol{H}$ are transform matrices for 4 × 4 FIT, 4 × 4 IIT and 4 × 4 HT, respectively. The H. 264/AVC standard has defined fast algorithms for implementing 1-D matrix multiplication, and 2-D matrix multiplication can be realized by cascading two 1-D matrix multiplications. And the 1-D matrix multiplication only needs shift, addition and subtraction operations, as shown in Fig. 1.

1.2 2 × 2 Hadamard transform

The 2 × 2 Hadamard transform, which is always applied to a 2 × 2 array of DC coefficients of each chroma component, is defined as

$$\boldsymbol{Y} = \boldsymbol{H}_{\rm t}\boldsymbol{X}\boldsymbol{H}_{\rm t}^{\rm T} \tag{3}$$

where $\boldsymbol{X}$ is a 2 × 2 residual block input to the Hadamard transform. The transform matrix is given as

$$\boldsymbol{H}_{\rm t} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4}$$

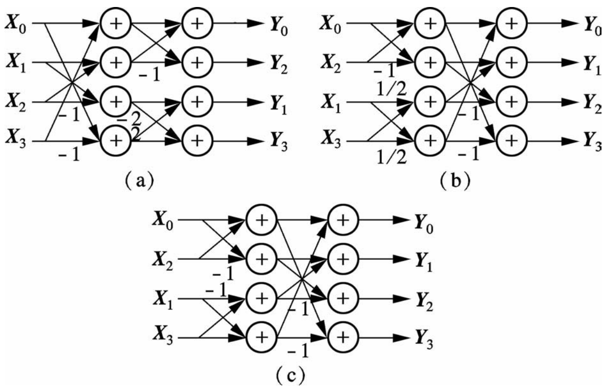

Fig. 1 Signal flow of 4 × 4 inverse integer transform. (a) 4 × 4 forward integer transform; (b) 4 × 4 inverse integer transform; (c) 4 × 4 Hadamard transform

2 Proposed Algorithm for Multi-Transform

The 4 × 4 transforms and the 2 × 2 Hadamard transform defined in the H. 264/AVC standards are characterized by their high regularity and low complexity. A generic 2-D transform is illustrated as

$$W_{ij} = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} X_{kl}C_{ik}C_{jl} \qquad i,j = 0,2,\dots,N-1 \tag{5}$$

where X, W and C denote the input data, the output data and the transform coefficient, respectively.

To improve the hardware utilization rate and to achieve viable hardware implementations, algorithm decomposition is used. The row-column decomposition method for realizing N-point 2-D transforms is as follows:

$$M_{il} = \sum_{k=0}^{N-1} X_{kl}C_{ik} \qquad i,l = 0,2,\dots,N-1 \tag{6}$$

$$W_{ij} = \sum_{l=0}^{N-1} C_{jl}\boldsymbol{M}_{li}^{\rm T} \qquad i,j = 0,2,\dots,N-1 \tag{7}$$

where $\boldsymbol{M}$ denotes the intermediate data between the first-dimensional and the second-dimensional transforms, and $\boldsymbol{M}^{\rm T}$ denotes the transpose matrix of $\boldsymbol{M}$. Eq. (6) represents the column-wise transform and Eq. (7) represents the row-wise transform.

The row-column decomposition reduces the hardware cost of the circuit when implementing the transform algorithm, which only consists of a simpler 1-D matrix multiplication, and it significantly increases the utilization rate of its processor elements (PEs). PEs, which are designed to compute a restricted and very well defined set of operations, are proposed in this paper, and the detailed formulae are presented.

From Eq. (1), Eq. (2) and Fig. 1, it is easy to find that there are similarities among these fast algorithms and matrices. For the matrices, the differences are the coefficients in corresponding position. For example, the first coefficient in the second row of $\boldsymbol{C}_{\mathrm{f}}$ is 2, while those of $\boldsymbol{C}_{\mathrm{i}}$ and $\boldsymbol{H}$ are 1. The signal flow of the 4×4 inverse integer transform and the 4×4 Hadamard transform are the same. Therefore, these matrices may be integrated into one matrix. And these fast algorithms are possible to be realized by one fast algorithm.

Eq. (1) can be expanded as

$$\begin{bmatrix}\boldsymbol{Y}_0\\ \boldsymbol{Y}_1\\ \boldsymbol{Y}_2\\ \boldsymbol{Y}_3\end{bmatrix}=\begin{bmatrix}\boldsymbol{C}_{\mathrm{f}} & \boldsymbol{C}_{\mathrm{f}} & \boldsymbol{C}_{\mathrm{f}} & \boldsymbol{C}_{\mathrm{f}}\\ 2\boldsymbol{C}_{\mathrm{f}} & \boldsymbol{C}_{\mathrm{f}} & -\boldsymbol{C}_{\mathrm{f}} & -2\boldsymbol{C}_{\mathrm{f}}\\ \boldsymbol{C}_{\mathrm{f}} & -\boldsymbol{C}_{\mathrm{f}} & -\boldsymbol{C}_{\mathrm{f}} & \boldsymbol{C}_{\mathrm{i}}\\ \boldsymbol{C}_{\mathrm{f}} & -2\boldsymbol{C}_{\mathrm{f}} & 2\boldsymbol{C}_{\mathrm{f}} & -\boldsymbol{C}_{\mathrm{f}}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_0\\ \boldsymbol{X}_1\\ \boldsymbol{X}_2\\ \boldsymbol{X}_3\end{bmatrix} \tag{8}$$

Then $\boldsymbol{Y}$ can be decomposed as

$$\left.\begin{aligned}\boldsymbol{Y}_0&=(\boldsymbol{X}_1+\boldsymbol{X}_2)+(\boldsymbol{X}_0+\boldsymbol{X}_3)\\ \boldsymbol{Y}_1&=(\boldsymbol{X}_1-\boldsymbol{X}_2)+2(\boldsymbol{X}_0-\boldsymbol{X}_3)\\ \boldsymbol{Y}_3&=-[2(\boldsymbol{X}_1-\boldsymbol{X}_2)-(\boldsymbol{X}_0-\boldsymbol{X}_3)]\\ \boldsymbol{Y}_2&=-[(\boldsymbol{X}_1+\boldsymbol{X}_2)-(\boldsymbol{X}_0+\boldsymbol{X}_3)]\end{aligned}\right\} \tag{9}$$

where

$$\begin{aligned}\boldsymbol{Y}_0&=(Y_{00},Y_{01},Y_{02},Y_{03})^{\mathrm{T}}, & \boldsymbol{X}_0&=(X_{00},X_{01},X_{02},X_{03})^{\mathrm{T}}\\ \boldsymbol{Y}_1&=(Y_{10},Y_{11},Y_{12},Y_{13})^{\mathrm{T}}, & \boldsymbol{X}_1&=(X_{10},X_{11},X_{12},X_{13})^{\mathrm{T}}\\ \boldsymbol{Y}_2&=(Y_{20},Y_{21},Y_{22},Y_{23})^{\mathrm{T}}, & \boldsymbol{X}_2&=(X_{20},X_{21},X_{22},X_{23})^{\mathrm{T}}\\ \boldsymbol{Y}_3&=(Y_{30},Y_{31},Y_{32},Y_{33})^{\mathrm{T}}, & \boldsymbol{X}_3&=(X_{30},X_{31},X_{32},X_{33})^{\mathrm{T}}\end{aligned}$$

Applying the same process as Eq. (1), Eq. (2) and Eq. (3) can be expanded as

$$\begin{bmatrix}\boldsymbol{Y}_0\\ \boldsymbol{Y}_1\\ \boldsymbol{Y}_2\\ \boldsymbol{Y}_3\end{bmatrix}=\begin{bmatrix}\boldsymbol{C}_{\mathrm{i}} & \boldsymbol{C}_{\mathrm{i}} & \boldsymbol{C}_{\mathrm{i}} & \dfrac{\boldsymbol{C}_{\mathrm{i}}}{2}\\ \boldsymbol{C}_{\mathrm{i}} & \dfrac{\boldsymbol{C}_{\mathrm{i}}}{2} & -\boldsymbol{C}_{\mathrm{i}} & -\boldsymbol{C}_{\mathrm{i}}\\ \boldsymbol{C}_{\mathrm{i}} & -\dfrac{\boldsymbol{C}_{\mathrm{i}}}{2} & -\boldsymbol{C}_{\mathrm{i}} & \boldsymbol{C}_{\mathrm{i}}\\ \boldsymbol{C}_{\mathrm{i}} & -\boldsymbol{C}_{\mathrm{i}} & \boldsymbol{C}_{\mathrm{i}} & -\dfrac{\boldsymbol{C}_{\mathrm{i}}}{2}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_0\\ \boldsymbol{X}_1\\ \boldsymbol{X}_2\\ \boldsymbol{X}_3\end{bmatrix} \tag{10}$$

$$\left.\begin{aligned}\boldsymbol{Y}_0&=(\boldsymbol{X}_0+\boldsymbol{X}_2)+\left(\boldsymbol{X}_1+\frac{\boldsymbol{X}_3}{2}\right)\\ \boldsymbol{Y}_1&=(\boldsymbol{X}_0-\boldsymbol{X}_2)+\left(\frac{\boldsymbol{X}_1}{2}-\boldsymbol{X}_3\right)\\ \boldsymbol{Y}_2&=(\boldsymbol{X}_0-\boldsymbol{X}_2)-\left(\frac{\boldsymbol{X}_1}{2}-\boldsymbol{X}_3\right)\\ \boldsymbol{Y}_3&=(\boldsymbol{X}_0+\boldsymbol{X}_2)-\left(\boldsymbol{X}_1+\frac{\boldsymbol{X}_3}{2}\right)\end{aligned}\right\} \tag{11}$$

$$\begin{bmatrix}\boldsymbol{Y}_0\\ \boldsymbol{Y}_1\\ \boldsymbol{Y}_2\\ \boldsymbol{Y}_3\end{bmatrix}=\begin{bmatrix}\boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H} & \boldsymbol{H}\\ \boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H}\\ \boldsymbol{H} & -\boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H}\\ \boldsymbol{H} & -\boldsymbol{H} & \boldsymbol{H} & -\boldsymbol{H}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_0\\ \boldsymbol{X}_1\\ \boldsymbol{X}_2\\ \boldsymbol{X}_3\end{bmatrix} \tag{12}$$

$$\left.\begin{aligned}\boldsymbol{Y}_0&=(\boldsymbol{X}_0+\boldsymbol{X}_2)+(\boldsymbol{X}_1+\boldsymbol{X}_3)\\ \boldsymbol{Y}_1&=(\boldsymbol{X}_0-\boldsymbol{X}_2)+(\boldsymbol{X}_1-\boldsymbol{X}_3)\\ \boldsymbol{Y}_2&=(\boldsymbol{X}_0+\boldsymbol{X}_2)-(\boldsymbol{X}_1+\boldsymbol{X}_3)\\ \boldsymbol{Y}_3&=(\boldsymbol{X}_0-\boldsymbol{X}_2)-(\boldsymbol{X}_1-\boldsymbol{X}_3)\end{aligned}\right\} \tag{13}$$

Comparing the three equations with each other, it is easy to find that Eqs. (9), (11) and (13) are similar. Based on this similarity, a new fast algorithm to implement 1-D matrix multiplication for all the three 4×4 transforms is proposed. For the 2×2 Hadamard transform, the expansion equation of Eq. (3) is

$$\begin{bmatrix}Y_{00} & Y_{01}\\ Y_{10} & Y_{11}\end{bmatrix}=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}X_{00} & X_{01}\\ X_{10} & X_{11}\end{bmatrix}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \tag{14}$$

And Eq. (14) can be further expanded as

$$\begin{bmatrix}Y_{00} & Y_{01}\\ Y_{10} & Y_{11}\end{bmatrix}=\begin{bmatrix}X_{00}+X_{01}+X_{10}+X_{11} & X_{00}-X_{01}+X_{10}-X_{11}\\ X_{00}+X_{01}-X_{10}-X_{11} & X_{00}-X_{01}-X_{10}+X_{11}\end{bmatrix} \tag{15}$$

Finally, the following equation is obtained, which is similar to the 4×4 Hadamard transform.

$$\begin{bmatrix}Y_{00}\\ Y_{10}\\ Y_{11}\\ Y_{01}\end{bmatrix}=\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix}\begin{bmatrix}X_{00}\\ X_{01}\\ X_{10}\\ X_{11}\end{bmatrix}=\boldsymbol{H}_{\mathrm{f}}\begin{bmatrix}X_{00}\\ X_{01}\\ X_{10}\\ X_{11}\end{bmatrix} \tag{16}$$

Based on Eq. (16), the 2×2 Hadamard transform is integrated into the 4×4 Hadamard transform, which can be treated as a 1-D 4×4 Hadamard transform.

According to the previous analysis, a fast algorithm which integrates four kinds of 1-D matrix multiplication is proposed. As demonstrated in Fig. 2, $\boldsymbol{X}_0$ to $\boldsymbol{X}_3$ represent the data input, and $\boldsymbol{Y}_0$ to $\boldsymbol{Y}_3$ are the 1-D matrix multiplication results. And the fast algorithm contains three modes corresponding to the three transforms. If the algorithm mode is not the 4×4 forward integer transform, then the data input and output orders are $\boldsymbol{X}_0$, $\boldsymbol{X}_2$, $\boldsymbol{X}_1$, $\boldsymbol{X}_3$ and $\boldsymbol{Y}_0$, $\boldsymbol{Y}_1$, $\boldsymbol{Y}_2$, $\boldsymbol{Y}_3$, respectively; else, the data input and output order will change to $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, $\boldsymbol{X}_0$, $\boldsymbol{X}_3$ and $\boldsymbol{Y}_0$, $\boldsymbol{Y}_1$, $\boldsymbol{Y}_3$, $\boldsymbol{Y}_2$, respectively (notice that, as shown in Eq. (9), the output signals $\boldsymbol{Y}_3$ and $\boldsymbol{Y}_2$ exchange position). Coefficients on the dataflow arrow (for example, -1, $1/2$, 1) in Fig. 2 correspond to the 4×4 Hadamard transform, the 4×4 inverse integer transform and the 4×4 forward integer transform, respectively. If there is no coefficient on the dataflow arrow, the coefficient will be 1 for all transforms. If the coefficient on the dataflow arrow is -1, the coefficient will be -1 for all transforms.

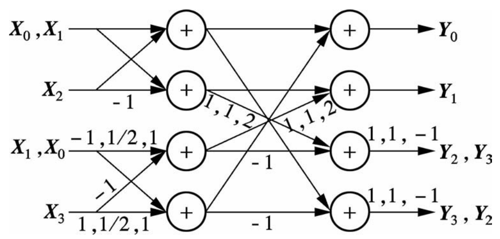

Fig. 2 Proposed fast algorithm

3 Proposed Architecture for Multi-Transform

The proposed fast algorithm yields a high-performance architecture for multiple transforms. It adopts the method defined in the H. 264/AVC standard, which implements 2-D matrix multiplication by cascading two 1-D matrix multiplications. Although these two 1-D matrix multiplications are all based on the proposed algorithm, there are some differences in their structures. The first 1-D one is controlled by a finite state machine (FSM), while the second one is adopted to realize Eq. (8), Eq. (10) and Eq. (12), which multiplies the transform matrix with the results of the first 1-D matrix multiplication. The proposed architecture for multi-transform is illustrated in Fig. 3, which contains three parts, PE1 array, MODE generator and PE0.

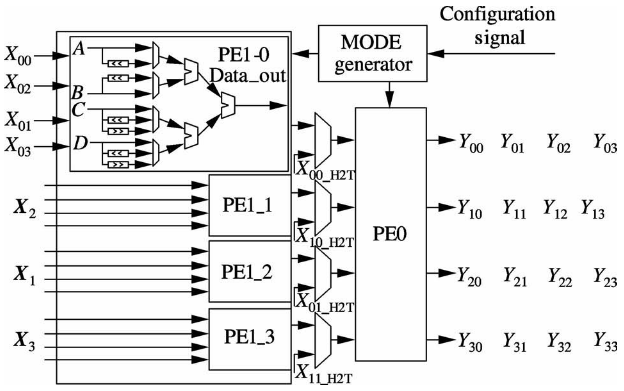

Fig. 3 Block diagram of proposed architecture

3.1 PE0

The block diagram of PE0 is shown in Fig. 4. As the second 1-D matrix multiplication, it aims at obtaining the product of the transform matrix and multiplying the results of the first 1-D matrix multiplication. PE0, which is composed of registers, left shifters, right shifters, MUXs, subtractors and adders, and controlled by a two bit mode selected signal generated by the mode generator, can realize 1-D 4 × 4 FIT, 1-D 4 × 4 IIT and 1-D 4 × 4 HT. The left and right shifters are used to substitute for "× 2" and "/2" operations, respectively. MUXs are controlled by input mode selected signals, and the registers are used to ensure the validity of timing. The values of the mode selected signal (named as PE0_SEL) for selecting operation mode are illustrated in Tab. 1.

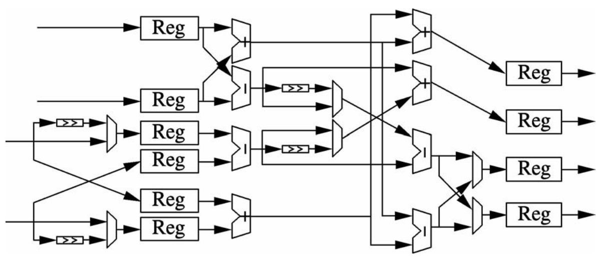

Fig. 4 Block diagram of PE0

Tab. 1 PE0_SEL and PE0 functions

PE0_SEL	Function
00	4 × 4 FIT
01	4 × 4 IIT
10	4 × 4 HT
11	2 × 2 HT

3.2 PE1 array

2-D matrix multiplication can be implemented by cascading two 1-D matrix multiplications. In the proposed architecture, PE0 represents a 1-D matrix multiplication and PE1 array represents another. PE1 is also composed of left shifters, right shifters, adders/subtractors and MUXs, which is controlled by PE1_SEL and PE0_SEL. And these two signals are generated by the mode generator. PE0_SEL is used to select the transform mode and PE1_SEL is used to control the operation of PE1s. Each transform mode has four PE1 operation modes, which corresponds to the equations embraced in the brace in Eqs. (8), (10) and (12). PE1 operation modes can be controlled by an FSM, which has four statements corresponding to the four equations embraced in the brace of the third column in Eqs. (9), (11) and (13).

3.3 MODE generator

There are four functions of the MODE generator: PE0_SEL generation, PE1_SEL generation, FSM for PE1 array and PE0 data input selection. PE0_SEL is equal to the configuration signal, which is a 5-bit width signal used to control MUXs and adders/subtractors. If the transform mode is a 2 × 2 Hadamard transform, the data input of PE0 is X_{00_H2T}, X_{01_H2T}, X_{10_H2T} and X_{11_H2T}. Else, the data inputs are the outputs of PE1 array.

4 Synthesis Results and Validation

The proposed architecture for multi-transform is implemented by Verilog HDL, functionally simulated with ModelSim SE 6. 6 and synthesized by Synopsys Design Compiler under a SMIC 0. 18 μm CMOS technology. Tab. 2 shows the hardware cost (in terms of gate count), optimum operating frequency, critical path delay, throughput and DTUA. DTUA is defined as the ratio of data throughput rate over gate count. The higher the DTUA is, the more efficient the architecture is.

The reference designs listed in Tab. 2 are all synthesized by CMOS technology and the multiple transform is realized by utilizing reconfigurable architecture[6], parallel structure[4], direct 2-D structure[7,12] and multi-transform architecture[10]. The results shown in Tab. 2 indicate that the DTUA of the proposed architecture is at least 40.28% higher than the reference design. And the hardware cost (in terms of gate count) is smaller than all of the reference designs. Additionally, it does not need transpose memory.

Tab. 2 Performance comparison for the proposed architecture

Design	Technology/μm	Area/gate	Frequency/MHz	Throughput/ (10^6 pixel · s^{-1})	DTUA/ (pixel · s^{-1} · $gate^{-1}$)
This work	0. 18	3 704	200	800	215 980
Ref. [4]	0. 35	6 538	80	320	48 900
Ref. [7]	0. 18	6 482	100	800	123 420
Ref. [6]	0. 18	5 140	200	800	155 600
Ref. [10]	0. 18	39 800	50	400	10 100
Ref. [11]	0. 18	63 618	200	3 200	50 300

The data processing rate of the proposed multi-transform architecture is 4 pixels/cycle. Therefore, decoding a 4 × 4 block needs 4 cycles and utilizing the proposed architecture decoding a macroblock needs about 64 cycles.

As for digital cinema video (4 096 × 2 048 @ 30 Hz), the real-time decoding requirement is 983 040 macroblocks per second and the proposed architecture utilized for decoding one frame of digital cinema video needs 64 × 983 040 cycles, which is 62 914 560. Thus, by using the proposed architecture, the minimum operating frequency to decode this kind of video format is about 62. 9 MHz, which is much smaller than the optimum operating frequency (200 MHz). So, it is concluded that the proposed architecture satisfies the real-time decoding requirement of digital cinema video.

5 Conclusion

This paper proposes a high-performance multiple transform architecture for H. 264/AVC video coding standard and a novel fast algorithm for 1-D matrix multiplication of multi-transforms with the data processing rate of 4 pixels/cycle. By using the SMIC 0. 18 μm CMOS technology, the maximum operating clock frequency of the proposed multi-transform architecture is 200 MHz and the data throughput rate can achieve as many as 800 × 10^6 pixels/s with the hardware cost of 3 704 gates. The synthesize results indicate that the proposed architecture is able to increase at least 40. 28% of the DTUA and efficiently reduce the hardware cost to satisfy the requirement of real-time decoding digital cinema video (4 096 × 2 048 @ 30 Hz).

References

[1] Hong E P, Jung E G, Fraz H, et al. Parallel 4 × 4 transform architecture based on bit extended arithmetic for H. 264/AVC [C]//*Proc of International Symposium on Signals, Circuits and Systems*. New York, USA, 2005, **1**: 95 – 98.

[2] Rubin G. Parallel 4 × 4 transform on bit serial shared memory architecture for H. 264/AVC [C]//*Proc of the 16th International Conference on Mixed Design of Integrated Circuits & Systems*. Lodz, Poland, 2009: 675 – 680.

[3] Porto R, Bampi M, Agostini S, et al. High throughput architecture for forward transforms of H. 264/AVC video coding standard [C]//*Proc of the 14th IEEE International Conference on Electronics, Circuits and Systems*. Marrakech, Morocco, 2007: 150 – 153.

[4] Wang T C, Huang Y W, Fang H C, et al. Parallel 4 × 4 2D transform and inverse transform architecture for MPEG-4 AVC/H. 264 [C]//*Proc of the 2003 International Symposium on Circuits and Systems*. Bangkok, Thailand, 2003: Ⅱ-800 – Ⅱ-803.

[5] Cao W, Hou H, Lai J M, et al. A high-performance reconfigurable 2-D transform architecture for H. 264 [C]//*Proc of the 15th IEEE International Conference on Electronics*, Circuits and Systems. St. Julien's, 2008: 606 – 609.

[6] Cao W, Hou H, Lai J M, et al. A novel dynamic reconfigurable VLSI architecture for H. 264 transforms [C]//*Proc of IEEE Asia Pacific Conference on Circuits and Systems*. Macao, China, 2008: 1810 – 1813.

[7] Chen K H, Guo J I, Wang J S. A high-performance direct 2-D transform coding IP design for MPEG-4 AVC/H. 264 [J]. *IEEE Transactions on Circuits and Systems for Video Technology*, 2006, **16**(4): 472 – 483.

[8] Peng C, Yu D, Cao X, et al. A new high throughput VLSI architecture for H. 264 transform and quantization [C]//*Proc of the 7th International Conference on ASIC*. Guilin, China, 2007: 950 – 953.

[9] Hwangbo W, Kim J, Kyung C M. A high-performance 2-D inverse transform architecture for the H. 264/AVC decoder[C]//*Proc of IEEE International Symposium on Circuits and Systems*. New Orleans, LA, USA, 2007: 1613 – 1616.

[10] Hwangbo W, Kyung C M. A multi-transform architecture for H. 264/AVC high-profile coders [J]. *IEEE Transactions on Multimedia*, 2010, **12**(3): 157 – 167.

[11] Wang K W, Chen J L, Cao W, et al. A reconfigurable multi-transform VLSI architecture supporting video codec design[J]. *IEEE Transactions on Circuits and Systems* Ⅱ, 2011, **58**(7): 432 – 436.

[12] Huang C Y, Chen L F, Lai Y K. A high-speed 2-D transform architecture with unique kernel for multi-standard video applications [C]//*Proc of IEEE International Symposium on Circuits and Systems*. Seattle, WA, USA, 2008: 21 – 24.

一种用于 H. 264/AVC 的高性能多变换结构

王　刚[1 2]　王　庆[1]　李　冰[2]　陈　锐[3]

([1] 东南大学仪器科学与工程学院,南京 210096)
([2] 东南大学无锡分校,无锡 214135)
([3] 中国科学院电子学研究所, 北京 100190)

摘要:为了提高硬件使用率、减少芯片面积,提出了一种包括 4 ×4 正反变换、4 ×4 哈达码变换及 2 ×2 哈达码变换在内的多变换编码结构. 通过将这几种变换算法化简,并且研究它们之间的相似性,将处理单个变换的结构合并成一个高性能多变换编码结构. 采用 SMIC 0. 18 μm CMOS 工艺,该结构最高工作频率可达 200 MHz,并在消耗 3704 门电路的情况下,达到 800×10^6 pixel/s 的吞吐率. 结果表明,该设计的吞吐率和电路规模的比值 DTUA 比参考设计至少提高了 40. 28% ,并且在 62. 9 MHz 的频率下支持分辨率为 4 096 × 2 048、刷新频率为 30 Hz 的视频实时解码需求,从而有助于降低功耗.

关键词:H. 264/AVC;多变换结构;哈达码变换;整数变换

中图分类号:TP302