Hardware Oriented Re-design and Matrix Approximation Analysis for Transform in High Efficiency Video Coding (HEVC)

Lin SUN* Oscar C.Au* Jiali Li* Ruobing Zou* and Wei Dai*
* Department of Electronic and Computer Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
Email: {lsunece, eeau, jiali, zouruobing, weidai }@ust.hk

***Abstract*—In this paper, we propose an adaptive truncate k-bit re-configurable approximation (aTra) method which can achieve similar video coding efficiency as the original transform but substitute all low efficient multiplication by conformed shifting and addition operations, lifting the utilization of the hardware implementation and making simple hardware implementation and high efficiency pipeline design possible. Also, we may be the first to propose three mathematical constraints for the hardware matrix approximation to make the final performance controllable. The proposed method can achieve regular data flow and massive operation reduction balancing the rate distortion (RD) performance and data throughput. Particularly for the current secondary transform, rotational transform (ROT), we obtain the hardware friendly ROT through our proposed method based on one of the constraints. The simulation results implemented in the HEVC reference software HM1.0 present the validity of our method. Our method achieves similar performance when compared with the original one but it is much better than the simple hardware implementation.**

I. Introduction

The latest state-of-the-art video coding standard is H.264/AVC [1] developed by ITU-T and ISO/IEC/MPEG. H.264 achieves better coding performance than previous video coding standards by applying a number of new tools. Typically, the residual block is formed first by inter and intra prediction. Then the integer discrete cosine transform (ICT) is applied to the residues to exploit the energy compaction. ICT has three major advantages for video coding. Firstly, ICT is generated from the discrete cosine transform (DCT) by replacing the real numbered elements with integers, so multiplication can be substituted by shifting and addition which are beneficial to the hardware implementation. Furthermore, the algorithm can be accelerated by using the butterfly architecture. However, existing research just concentrates on how to change the DCT to ICT and can not be applied to other transforms.

Recently, ITU-T and ISO/IEC/MPEG have formed a Joint Collaborative Team on Video Coding (JCT-VC) on next generation video coding standard called High Efficiency Video Coding (HEVC) [2]. Rotational Transform (ROT) and Mode Dependent Directional Transform (MDDT) [2] are both alternative transforms proposed in the latest video coding standard. Unlike MDDT, ROT is a secondary transform which is applied after the primary DCT without changing the transform core [3] to further exploit the energy compaction. Also, compared to the MDDT, ROT typically shows higher coding gain. The whole transform process with ROT is:

$$m_o = R_v^T D^T m_i D R_h \tag{1}$$

where m_i is input residual matrix, m_o is transformed matrix, D is ICT matrix, R_h and R_v represent the rotation matrices for horizontal and vertical directions with the rotation angles of α_1, α_2, α_3 and α_4, α_5, α_6, respectively:

$$R_h = R_z(\alpha_1)R_x(\alpha_2)R_z(\alpha_3) \tag{2}$$

$$R_v = R_z(\alpha_4)R_x(\alpha_5)R_z(\alpha_6) \tag{3}$$

where R_x and R_z is compound Given's rotation matrices.

The rest of this paper is organized as follows: Section II gives a brief analysis of the drawbacks of simple integer hardware implementation of ROT. Section III proposes our hardware-friendly orientated design, gives mathematical analysis and proposes three constraints. Simulation results of our proposed methods based on the constraints are shown in Section IV. Section V contains our conclusions.

II. Hardware Implementation Analysis

In order to keep the reasonable complexity, ROT totally has four horizontal and four vertical matrices R_h^i and R_v^i corresponding to four sets of rotation angles α_1^i, α_2^i, α_3^i, α_4^i, α_5^i, α_6^i, for i = 1,$\cdots$,4. In implementation, the elements of rotational matrices were scaled and rounded to the nearest integer. For example, one of the horizontal 4×4 ROT matrixes is:

$$R_h^1 = \begin{bmatrix} 3736 & -1603 & 48 & 0 \\ 1597 & 3756 & -397 & 0 \\ 112 & 387 & 4073 & 0 \\ 0 & 0 & 0 & 4096 \end{bmatrix} \tag{4}$$

As we know, the matrix multiplication process consists of two steps. Firstly, each coefficient of the input matrix is multiplied by the transform coefficient at its corresponding position and obtains the temporal result. Such as 10×5 is $(1010)_b \times (0101)_b$ in binary, the temporal result is $(1010)_b$, $(00000)_b$, $(101000)_b$

and $(0000000)_b$. The final result $(0110010)_b$ is then achieved by adding these temporal results together. The multiplication process in computers can be treated as a series of addition and shifting operations, and the operation number is decided by the number of "1" of the multiplicand (coefficient in the transform matrix). Meanwhile the whole system delay is determined by the operation with maximum clock cycles which is also called critical path. Table I shows the number of "1" of $R_h^1(i,j)$, and the number of corresponding addition and shifting operations. For simplicity, we assume that shifting and addition both take one clock cycle. We observe that computing different temporal results need different clock cycles, for example, for the first row of R_h^1, $R_h^1(1,1)$ needs 15 clock cycles and $R_h^1(2,1)$ needs 12 clock cycles but $R_h^1(3,1)$ only need 4. And if we compute one temporal result t_o in parallel:

$$t_o = p(1,1) * R_h^1(1,1) + p(1,2) * R_h^1(2,1) + p(1,3) * R_h^1(3,1) \quad (5)$$

where p is transformed residual matrix. It needs total 15 clock cycles to receive the temporal results. Actually, $R_h^1(2,1)$ and $R_h^1(3,1)$ will be waiting while $R_h^1(1,1)$ is computing. This will significantly decrease the data throughput and increase the power consumption. On the other hand, if we compute the temporal results in a serial mode with re-using identical architecture, the required clock cycles for different coefficients are different and this will result in irregular data flow, complex control signal, and low efficient pipeline design.

TABLE I: R_h^1 Coefficient Operations

Matrix	Original Binary	Shift Num	Adder Num	Clock Cycles
R(1,1)	111010110011	8	7	15
R(1,2)	11001000011	5	4	9
R(1,3)	110000	2	1	3
R(2,1)	11000111101	7	6	13
R(2,2)	111010101100	7	6	13
R(2,3)	110001101	5	4	9
R(3,1)	1110000	3	2	5
R(3,2)	110000011	4	3	7
R(3,3)	111111101001	9	8	17

III. Proposed Hardware Orientated Scheme

In this section, we propose a hardware-friendly re-design method in which all multiplications will be substituted by conformed shifting and addition operations which will result in a large increase of the hardware efficiency.

A. Reconfigurable k-bit Hardware Approximation

All commands, memory storage, input, and output are done based on binary in computer. Particularly with hardware, we do not favor multiplier as it will use significant resources and lower data throughput. Therefore we want to substitute multiplication with shifting and adding. However, if we simply binarize the coefficients, it will not make the hardware implementation efficient as previously discussed. In this section, we propose a hardware-friendly re-design scheme based on our defined mathematical constraints, which will make multiplication efficiently implemented by hardware with negligible degradation.

In [5], Xing proposed a mapping method to use a shifting operation to replace multiplication and with this method, we can use $A^{'}(i,j)$ to approximate the original coefficient *A(i,j)*:

$$A^{'}(i,j) = 2^n; n = floor\lfloor \frac{\ln A(i,j)}{\ln 2} \rfloor \quad (6)$$

Also in [6], the authors give us another mapping method for the filter coefficients,

$$A^{'}(i,j) = 2^{n-10}; n = floor\lfloor \frac{\ln A(i,j) << 2^{10}}{\ln 2} \rfloor \quad (7)$$

However, based on our simulation results, simply mapping as [5] will result in quality drop both in subjective and objective aspects. The simple reason is the difference between approximated coefficients $A^{'}(i,j)$ and the original values *A(i,j)* are relatively huge. Therefore in this paper, a new approximated method, shown in equation (8) which is called adaptive truncation re-configurable approximation (aTra) is proposed.

$$\begin{aligned} &A(i,j)^{'}_{\gamma k} = s_1 2^{n_1} + s_2 2^{n_2} \cdots + s_k 2^{n_k}; \\ &n_1 = floor\lfloor \frac{\ln A(i,j)}{\ln 2} + \gamma_1 \rfloor, \\ &n_2 = floor\lfloor \frac{\ln \left(A(i,j) - 2^{n_1}\right)}{\ln 2} + \gamma_2 \rfloor, \cdots \\ &n_k = floor\lfloor \frac{\ln \left(A(i,j) - 2^{n_1} - \cdots - 2^{n_{k-1}}\right)}{\ln 2} + \gamma_k \rfloor \end{aligned} \quad (8)$$

where s_k is the sign which can be negative or positive and *k* indicates the number of pipeline level. Here *k* can be any integer which makes a tradeoff between the RD performance and the data throughput. A larger *k* will get a good RD performance with low data throughput, while a smaller *k* will degrade the RD performance with high data throughput. Herein γ_i can be negative or positive to make the adaptive truncate, for *i = 1,*$\cdots$*,k* and $\gamma_i \in (-1, 1)$. For simplicity, in the experiment we just set $\gamma_1 = \gamma_2 = \cdots = \gamma_k = \gamma$.The processing diagram is shown in Fig.1. After setting the represented bits for the elements, we can recursively apply the processing procedure to get the final shifting bit and sign bit.

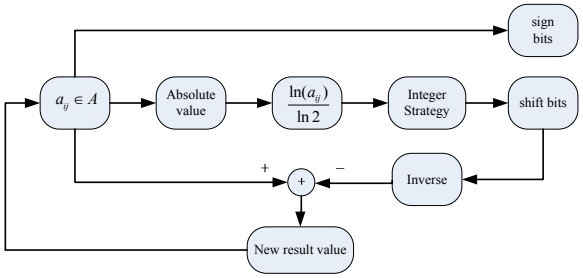

Fig. 1: Proposed minimized difference function with k level pipeline for transform. *A* is the original matrix and a_{ij} is one matrix coefficient at corresponding position (i,j).

The modified method has significant advantages. Firstly, all the temporal results require the same clock cycles *2k-1*, which lead to the regular data flow. That is, the original non-regular

binary matrix becomes the regular k pipeline level for efficient hardware implementation. Secondly, if necessary, the proposed method can largely reduce the number of operations. Furthermore the parameter γ makes the error between the original and the approximated coefficient controllable. Below we will show how to minimize the error based on the constraints.

B. Mathematical Constraints of the Approximation

1) Difference Error(DE): Also the hardware friendly matrix we generate using aTra should largely approximates the original one. Note *A* is the original matrix and $A^{'}$ is the approximated hardware friendly one, therefore the difference error ξ_D after hardware orientated approximation is:

$$\xi_D = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|A(i,j) - A^{'}(i,j)| \tag{9}$$

where N and (i,j) is the dimension and the coefficient position of the matrix *A* and $A^{'}$, respectively.

2) Nonorthogonal Error (NE): If we want to make both the encoder and decoder side matrix hardware friendly, the multiplication of encoder and decoder matrix may not equal to identity matrix. This will bring the reconstruction error for the whole codec. Below we will discuss the reconstruction error without the quantization which we also do not consider in this paper. Consider a vector X with dimension N. X is transformed by matrix T at the encode side. Let Y be the reconstructed matrix, where Y = $T^{'}$TX and $T^{'}$ is the decoder side matrix. $T^{'}T$ may not be equal to the identity matrix *I* for the sake of approximation. The difference between $T^{'}T$ and *I* is denoted as E_r, ie., E_r = $T^{'}T$ - *I*. The average variance of the reconstruction error ξ_R can be expressed as:

$$\begin{aligned}\xi_R &= \frac{1}{N}\sum_{m=0}^{N-1}E[(X_m - Y_m)^2] = \frac{1}{N}E[(X-Y)^T(X-Y)] \\ &= \frac{1}{N}E[(E_rX)^T(E_rX)] = \frac{1}{N}E[X^TE_r^TE_rX]\end{aligned} \tag{10}$$

let $H = E_r^TE_r$, and the deduction continues,

$$\xi_R = \frac{1}{N}E[X^THX] = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}H(i,j)E[x(j)x(i)] \tag{11}$$

In order to give the definition of $E[x(j)x(i)]$, we should analyze the transformed residues first. The inter and intra prediction exploit the image's redundancy to make the residues highly uncorrelated. Also DCT has the ability to de-correlate the type of signals found in image data [7]. Thus where the original one frame video is highly correlated, the transformed residues can be seen as uncorrelated random variables with zero mean. So $\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}E[x(j)x(i)]$ is equal to the variance of the source. Suppose the variance is constant for the signal so the upper bound of the equation (11) is determined by $\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}H(i,j)$. Hence minimize average variance of the reconstruction error is equal to minimize $\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}H(i,j)$. It is, however, slightly complicated to apply the above result to the transform which needs two different matrices at the left and right side, such as ROT. Here we assume T_1 and T_2 is the left and right transform matrix, respectively. After transforming it becomes T_1XT_2 and the reconstructed results is $T_1^{'}T_1XT_2T_2^{'}$, where $T_1^{'}$ and $T_2^{'}$ is the approximated matrix in the decoder side. Of course, they may not be orthogonal to the matrix T_1 and T_2. Equally, let $T_1^{'}T_1 = P$ and $T_2^{'}T_2 = Q$. Then the average variance of reconstruction error ξ_R is:

$$\begin{aligned}\xi_R &= \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}E[(X(i,j) - Y(i,j))^2] \\ &= \frac{1}{N^2}E[(vec(X) - vec(Y))^T(vec(X) - vec(Y))] \\ &= \frac{1}{N^2}E[(vec(X) - vec(PXQ))^T(vec(X) - vec(PXQ))]\end{aligned} \tag{12}$$

Here, vec(X) denotes the vectorization of the matrix X formed by stacking the columns of X into a single column vector. Then, the following relationship can obtain:

$$\begin{aligned}vec(X) - vec(PXQ) &= vec(X) - (Q^T \otimes P)vec(X) \\ &= [I_{n\times n} - (Q^T \otimes P)]vec(X)\end{aligned} \tag{13}$$

where the $'\otimes'$ is the Kronecker product. And then combine (13) and (12), the final ξ_R is:

$$\begin{aligned}\xi_R = \frac{1}{N^2}E[vec(X)^T[I_{n\times n} - (Q^T \otimes P)]^T \\ [I_{n\times n} - (Q^T \otimes P]vec(X))]\end{aligned} \tag{14}$$

Here, similarly let $E_r = I_{n\times n} - (Q^T \otimes P)$ and $H = E_r^TE_r$ and , then the final form of this derivation arrives:

$$\xi_R = \frac{1}{N^2}E[vec(X)^THvec(X))] \tag{15}$$

Similarly, as we derived previously, here, we minimize $\sum_{i=0}^{N\times N-1}\sum_{j=0}^{N\times N-1}H(i,j)$. Let nonorthogonal error be $\xi_R = \sum_{i=0}^{N\times N-1}\sum_{j=0}^{N\times N-1}H(i,j)$.

Below we propose three encoder and decoder side hardware approximation constraints.

(1) Where Difference Error (DE) is used for the encoder and decoder side.

$$\mathscr{C} = \xi_D(\gamma)_{En} + \xi_D(\gamma)_{De}; \tag{16}$$

where $\xi_D(\gamma)_{En}$ is the encoder side difference error and $\xi_D(\gamma)_{De}$ is the decoder side difference error.

(2) Where Non-orthogonal Error (NE) is used for the whole video coding processing.

$$\mathscr{C} = \xi_R(\gamma); \tag{17}$$

(3) Where DE is used for the encoder side constraint and NE for the decoder side constraint.

$$\mathscr{C} = \xi_D^2(\gamma) + \xi_R(\gamma); \tag{18}$$

Note that in eq.(18) $\xi_D(\gamma)$ is squared because the $\xi_R(\gamma)$ is an energy function. The process we adopt in the hardware friendly

approximation is to choose the optimal γ which enforces minimized error function $\mathscr{C}(\gamma)$:

$$\underset{\gamma}{argmin}(\mathscr{C}(\gamma)); \tag{19}$$

Note that the constraints above are parallel, any one of the proposed constraints can be used to obtain the optimal γ. After obtaining the optimal γ, we can use equation (8) to obtain the hardware friendly matrix. In the implementation we should first set the number of pipeline *k* for the encoder and decoder side balancing the complexity and the RD performance, and then perform the approximation based on one of the proposed constraints. Also note that this approximation process will not introduce the drifting error due to the inverse transform mismatch caused by the k-bit approximation of transform coefficients which is already reported in [8].

IV. Software Experimental Results

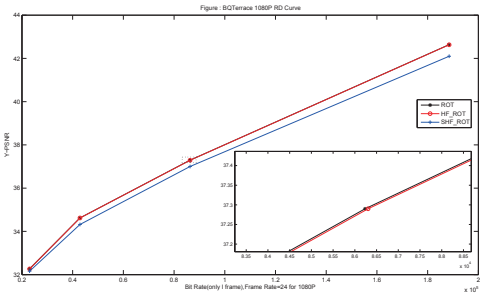

Fig. 2: BQTerrace Intra HE: RD Curve example of Proposed HF_ROT vs. Original ROT and SHF_ROT

Firstly, we apply our proposed method with k = 4 to the matrix R_h^1 illustrated before. The obtained matrix is nearly the same as the original matrix. Furthermore if we set k = 5, the approximated matrix is totally the same as the original. Actually, in real implementation, we set k considering the tradeoff between the RD performance and the complexity. If k is large it will induce more complexity with better RD performance, on the contrary, it will obtain worse RD performance with lower complexity. Also if the approximated matrix is obtained by our proposed method with k pipeline, the shifting number for every coefficient is k and the number of addition is (k-1) which is regular for the whole processing. Now the clock cycles for different coefficients of one pass transform restricts to (2k-1) based on our proposed method, it will largely lift the utilization of hardware resources, reduce the complicated control signal and make the data flow regular.

In the experiment, we set the forward ROT k = 5 and the inverse k = 3 to balance the quantity and quality, obtaining the Hardware Friendly ROT (HF-ROT). As space is limited, here we have only listed the error curve using equation (17) in Fig.3.

After obtaining optimal γ, we can generate the hardware friendly matrix for the encoder and decoder. The test condition satisfies [9] and all the work is based on HM1.0. Note that ROT has been removed from the main part of HEVC but it need further study, so here all the work is based on HM1.0 not the latest version. Here we use BDrate [10] to evaluate the results. Fig.2 illustrates R-D curve of BQTerrace compared original ROT implementation, simple hardware friendly ROT (SHF_ROT) implementation [5] and HF_ROT. From the figure we can see that our proposed method can achieve similar performance as the original one which is much better than the simple hardware implementation and importantly, this design performance can be controlled. Table II shows the "all intra" test case and overall "random access" test case HF-ROT results for different video sequences compared with the original ROT. Note that a negative number in the table means a gain and vice versa. With BD-rate of about -0.2% for intra high efficiency (HE) and -0.3% for intra low complexity (LC) (High Efficiency(HE) and Low Complexity (LC) are both profiles for HEVC) and even the same performance (less than 0.1%) for random access, the proposed method achieves the regular data flow(all the coefficients need the same processing clock), makes hardware implementation easily.

V. Conclusion

In this paper, we propose a hardware-friendly orientated re-design based on minimized error function and the aTra. This re-design can make data flow regular, making the scheme hardware friendly and decrease the time delay, significantly increasing the utilization of the hardware resources. We tested our method on the current secondary rotational transform in HM. Simulation results for the intra and random access based on the test condition show that it can achieve comparable performance as the original one. Of course, this method is not restrict to the ROT, it could be apply to other matrix related hardware oriented design. Importantly, the RD performance and the data through output can be controlled. The hardware design is just the combination of adders and shifters which can be the same for one PE (processing element).

VI. Acknowledgement

This work has been supported in part by the Research Grants Council (RGC) of the Hong Kong Special Administrative Region, China. (GRF Project no. 610210)

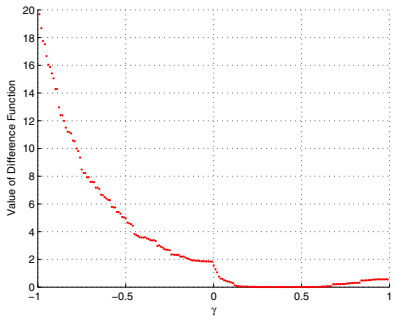

Fig. 3: Total ROT difference function changed with γ

TABLE II: Overall experimental results on the HM1.0 Platform for Intra and Random Access

	Intra HE			Intra Loco			Random HE			Random Loco		
Resolution	Y (BDrate%)	U (BDrate%)	V (BDrate%)	Y	U	V	Y	U	V	Y	U	V
2560×1600	0.3	0.2	0.1	0.4	0.2	0.2	0.1	0.1	0.1	0.1	0.0	0.2
1920×1080	0.2	0.2	0.2	0.4	0.3	0.3	0.1	0.3	0.1	0.1	0.0	0.2
832×480	0.1	0.0	0.0	0.2	0.1	0.1	0.0	0.0	0.1	0.0	0.1	-0.1
416×240	0.1	0.0	0.0	0.2	0.1	0.1	0.0	-0.3	-0.1	0.1	0.0	0.2
1280×720	0.2	0.1	0.0	0.3	0.2	0.1	N/A	N/A	N/A	N/A	N/A	N/A
Average	0.18	0.1	0.06	0.3	0.18	0.16	0.05	0.025	0.05	0.075	0.02	0.1

REFERENCES

[1] T. Wiegand, G.J. Sullivan, G. Bjϕtegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Trans. Circuits and Systems for Video Tech*, 13(7), pp.560-576, August 2003.

[2] K.McCann, W.J. Han, I.K. Kim, "Samsung's Response to the Call for the Proposals on Video Compression Technology," JCTVC-A124, *Joint Collaborative Team on Video Coding*, Dresden, Germany, April 2010.

[3] E.Alshina, A.Alshin, F.C.Fernandes, "Rotational Transform for Image and Video Compression, " *IEEE International Conference on Image Processing*, Brussels, Belgium, Septemper 2011.

[4] T. Wiegand, Han and Ohm, J.-R and Sullivan, G.J., "WD1: Working Draft 1 of High-Efficiency Video Coding," JCTVC-C403, *Joint Collaborative Team on Video Coding Meeting*, Guang Zhou, China, October 2010.

[5] X. Wen, O. C. Au, et al., "Novel RD-optimized VBSME with Matching Highly Data Re-usable Hardware Architecture," *IEEE Trans. on Circults and Systems for Video Technology*, 21(2), pp.206-219, August 2010.

[6] X. Wen, O.C. Au, J. Xu, L. Fang, and R. Cha, "Sub-pixel downsampling of video with matching highly data re-use hardware architecture", *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, 2011, pp.1495-1499.

[7] A.K.Jain, Fundamentals of Digital Image Processing, Englewood Cliffs, NJ: Prentice-Hall, 1989.

[8] X. Wen, O. C. Au, L. Sun, et al., "Hardware Friendly Rotational Transform," *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG114th Meeting*, Daegu, KR, January 2011.

[9] F. Bossen, "Common test conditions and software reference configurations," JCTVC-B300, Geneva, Switzerland, July 2010.

[10] G. Bjϕtegaard, "Calculation of average PSNR differences between RD-Curves," VCEG-M33, *13th VCEG Meeting*, Austin, USA, April 2001.