An Efficient ASIC Design of Variable-Length Discrete Cosine Transform for HEVC

Niras C. Vayalil, Joshua Haddrill and Yinan Kong
Department of Engineering
Macquarie University
Sydney, NSW, 2109 Australia
niras.cheeckottu-vayalil@mq.edu.au, joshua.haddrill@students.mq.edu.au, yinan.kong@mq.edu.au

***Abstract*—The latest video coding standard introduced by the joint collaborative team on video coding (JCT-VC) is known as high-efficiency video coding (HEVC) or H.265. HEVC/H.265 is mainly targeted for high-definition videos, and offer more compression than its predecessor. The discrete cosine transform (DCT) is widely used for image and video compression including HEVC. This paper proposes a variable-length DCT architecture for encoding video according to the HEVC/H.265 specifications. The architecture is optimized for most likely block sizes in ultra-high definition (UHD) video, and eliminates unnecessary complexities found in many architectures proposed. The synthesized results with Synopsys design tools show that the proposed method can encode 8K UHD videos @ 60 fps in real-time and accomplishes more than 60% in hardware savings.**

Keywords–Discrete cosine transform (DCT); H.265; high efficiency coding (HEVC); high definition;

I. Introduction

As the demand for high-definition (HD) video content increases, so does the need for efficient compression techniques. The high efficiency video coding (HEVC/H.265) standard [1] is a relatively new codec that is poised to replace advanced video coding (AVC/H.264) [2] as the standard for high-definition video encoding. HEVC/H.265 offers more compression, approximately a 50% bit-rate reduction, than its predecessor AVC/H.264 for an equivalent subjective reproduction quality [3]. The use of the discrete cosine transform (DCT) is a common method in several previous codecs and could be a key factor in the development of compression techniques for HEVC due to its near-optimal efficiency for performing this task. To be compatible for proper use with HEVC/H.265 the DCT needs to be computed for a matrix of varying length.

To accommodate the varying size of the architecture it would be ideal to develop components that can be utilized by other lengths such that the architecture is more area efficient, but the common method of multiplying by a constant matrix would not be effective in this case, due to its architecture not being able to be reused for other lengths.

Mehr et al proposed a reusable integer DCT architecture [4] providing same throughput in all supported transform lengths, but resulting in a higher area or gate count. An approximated architecture of DCT through the Walsh-Hadamard Transform (WHT) followed by a set of Givens rotations in [5] reduces gate count. Another approximate DCT architecture is proposed in [6] and offers better peak signal-to-noise ratio (PSNR). High-resolution video such as ultra-high-definition (UHD) video is more likely to have large smoother regions [7], thus transforms of larger size are mostly used. Hence the design is targeted mainly for the most likely block sizes instead of all possible sizes, and this assumption can reduce the hardware complexity significantly. This paper proposes a 2D-DCT architecture which has substantial throughput, with block sizes 16×16 and 32×32 for real-time encoding of 8K UHD. This architecture leads to a simple memory and DCT hardware structure and thus is smaller (lower gate count) and faster than many proposals in the literature.

II. Hardware Architecture for DCT computation

The DCT is a Fourier-related transform that only uses real numbers to represent a set number of discrete data points within a signal; unlike the discrete Fourier transform (DFT) the DCT only uses cosine functions to represent the data points [8]. There are multiple versions of the DCT that range from DCT-I to DCT-IV, the most common of which is DCT-II and referred to as 'the DCT' and defined as

$$X_k = \sum_{n=0}^{N-1} \cos\left[\frac{\pi}{N}(n+\frac{1}{2})k\right] x_n \quad 0 \le k < N \qquad (1)$$

For processing two-dimensional signals such as images, a two-dimensional version of the DCT (2D-DCT) is used; it is a trivial expansion of the standard DCT, given as

$$X_{k_1,k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \cos\left[\frac{\pi}{N_1}(n_1+\frac{1}{2})k_1\right] \cos\left[\frac{\pi}{N_2}(n_2+\frac{1}{2})k_2\right] x_{n_1,n_2} \qquad (2)$$

where $0 \le k_1 < N_1, 0 \le k_2 < N_2$.

One property of the 2D-DCT is separability [9], i.e. the 2-D DCT can be computed in two steps, a column-wise 1-D DCT followed by a row-wise 1-D DCT, or vice versa. This procedure of calculating a multidimensional

2473-3539/16 $31.00 © 2016 IEEE
DOI 10.1109/EMS.2016.43

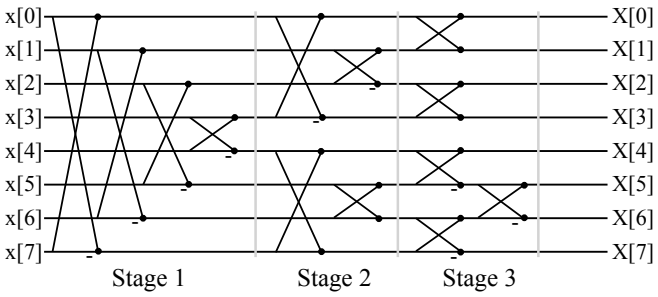

Figure 1. Stick diagram of the butterfly technique applied to the DCT.

separable transform is called row-column decomposition, which reduces the number of computations.

The DCT has become a staple in image compression, specifically in the JPEG format, due to the resulting lossy compression that occurs as a result of the transform, allowing larger image data to be compressed. This is done by applying the DCT to a quantization of an image's pixels to obtain an approximation that requires less data to be stored. DCT possesses a strong energy compaction property [10]; most of the signal information tends to be concentrated in few low-frequency components, making DCTs useful for image compression.

The related Fourier properties of the DCT make it possible to use the butterfly multiplication approach described by Budagavi et al, in such a way that the overall transformation completes in sections that effectively 'fold' into the next section [11]. This method is ideal as it is more efficient than the brute-force matrix multiplication method which is very costly in terms of computing time [11]. A stripped-down representation of this process can be seen in Fig. 1, where the horizontal lines represent the input and manipulated data after operations while a $(-)$ beneath the dot represents subtraction or addition.

A. Four-point DCT architecture

The four-point DCT module is based on the algorithm by Meher et al [4]. The algorithm used to implement the DCT for a 4×4 matrix is outlined in stages in Table I. For efficient implementation the algorithm is divided into an input adder unit (IAU), a glssau and an output adder unit (OAU), such that each stage can be examined and implemented individually to ensure that necessary values are available as each stage completes. This algorithm is used to replicate the kernel matrix represented by equation (3) to perform the transform without directly performing matrix multiplication. This is done to improve the computational speed and efficiency of the architecture.

$$C_4 = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} \tag{3}$$

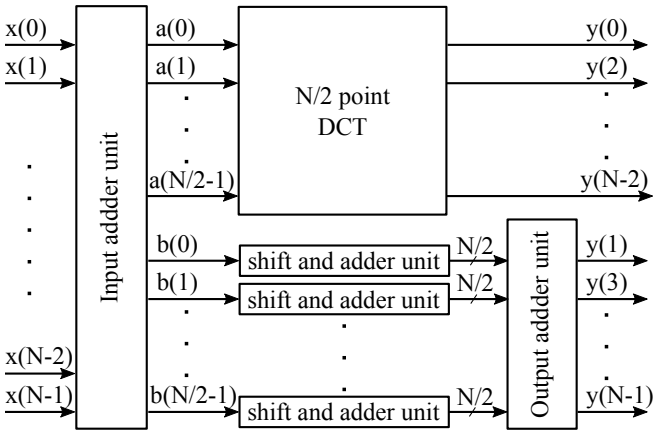

Figure 2. A generalized structure of higher radix DCTs, where N = 8, 16, 32. [4]. The N point DCT is build upon $N/2$ DCT with adder units and shift units.

B. Architecture for higher length 1-D DCTs

Higher-length DCTs with N = 8, 16 and 32 are built upon 4 point DCTs recursively. A generalized structure of $N = 8, 16, 32$ point integer DCTs are shown in Fig. 2. At each stage of this process the input data is first manipulated by an IAU to create intermediate data that is then used as the input for further operations. The even-numbered rows/columns, including zero, are processed as an $N/2$ point DCT to obtain the corresponding output values. The odd-numbered rows and columns are passed through a shift adder unit (SAU), which is specific to the point length of the DCT being performed, to produce the corresponding values in the output matrix.

Once the lower level transforms and operations have been completed the resulting data is then further manipulated by an OAU to complete the transformation. The complete architecture implementation operates recursively by gradually calling $N/2$-point DCTs until it reaches the four-point DCT.

III. Proposed hardware architecture for variable-length two-dimensional DCT

The separability property is used to design the 2-D DCT because the row-column decomposition results in computational savings but this introduces another problem of data storage or memory for saving first-step results. It is clear that the second step (row/column 1D-DCT) can only begin after completing the first step (column/row 1D-DCT), thus it is necessary to save all data and retrieve it in a transposed order. In this proposed architecture we use a 2 dimensional register array to save and transpose first 1-D DCT results, which results in an efficient 2D-DCT architecture.

The architecture of the proposed method is shown in Fig. 3, where the transpose module has a size of 32×32 words (1-D DCT input and output have 16 bit word length) which can hold all column transforms of a 32×32 blocks pixels. Initially, the 2D shift registers are set into 'shift-in mode' and

TABLE I
FOUR-POINT DCT ALGORITHM BY STAGE

Stage	Computation	Binary expression	Notes
Stage 1 (IAU)	$a(i) = x(i) + x(3-i)$ $b(i) = x(i) - x(3-i)$		for $i = 0$ to 3
Stage 2 (SAU)	$m_{i,9} = 9b(i)$ $m_{i,64} = 64a(i)$ $t_{i,83} = 83b(i)$ $t_{i,36} = 36b(i)$	$(b(i) << 3) + b(i)$ $a(i) << 6$ $(b(i) << 6) + (m_{1,9} << 1) + b(i)$ $m_{1,9} << 1$	for $i = 0$ to 3
Stage 3 (OAU)	$y(0) = t_{0,64} + t_{1,64}$ $y(1) = t_{0,83} + t_{1,36}$ $y(2) = t_{0,64} + t_{1,64}$ $y(3) = t_{0,36} + t_{1,83}$		

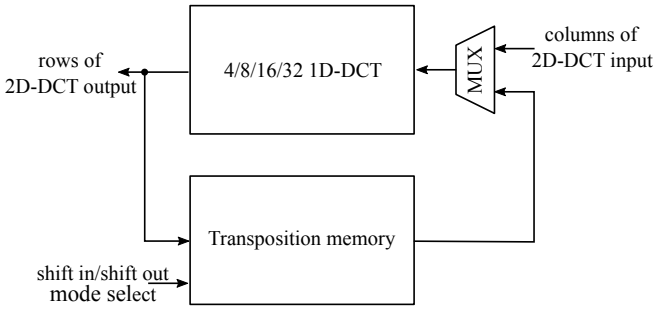

Figure 3. The proposed 2D-DCT architecture, transposition memory implemented using a 2-D register array.

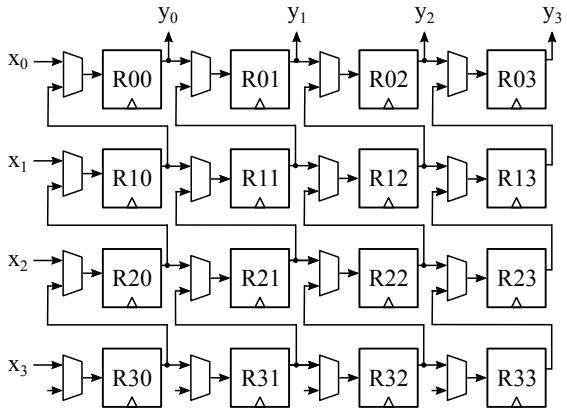

Figure 4. The proposed 2D shift register architecture, showing 4 inputs and 4 outputs. Data is shifted in the horizontal direction from left to right, and shifted out in the up direction; all MUX selection changes accordingly.

input data is given into the 1D-DCT module column-wise. During the 'shift-in mode', the results of the first 1D-DCTs are stored into the leftmost column of the 2D register array, and each column of data in the register array is shifted rightward at every clock cycle. A detailed digram of the 2D shift register arrangement is shown in Fig. 4.

After completion of all column transformations, the 2D shift register changes into 'shift-out' mode and the DCT module input connects to the 2D shift register outputs with the help of a multiplexer (MUX). In 'shift-out' mode the shift register's data shifts in the upward direction, and the output is taken from the top row. This write and read arrangement facilitates the transpose operation. The shift registers do not load data in this mode of operation. Since the DCT module completes a row transform in each cycle, the proposed architecture requires $2N$ clock cycles to complete an N point 2D-DCT transform. An an example, for a 32 point 2D-DCT, the first 32 clock cycles are required to complete all column transforms which are stored into the shift registers, and another 32 clock cycles are required to shift-out these data row-wise and complete all row transformations.

In HEVC/H.265, the transform is performed after intra and inter prediction, on the residues obtained by the differences between the original pixels and predicted pixels. For the residual coding, HEVC/H.265 employs recursive quad tree-structured partitioning of coding blocks [1]. The HEVC/H.265 specification supports four transform sizes: 4×4, 8×8, 16×16 and 32×32. The different block sizes in the specification are introduced for accommodating varying space-frequency characteristics of the residuals. The rate distortion (RD) cost computation is to be done for all coding unit (CU) sizes to select the best among the various block sizes. However this 'trial and error' method has a very high computational cost. Several algorithms are proposed for early transform unit (TU) decision reducing this complexity. Chio et al. propose a method for early TU decision by determining the number of nonzero DCT coefficients as a threshold to stop further RD cost evaluation in the quad tree structure [12]. But this method still has enough complexity, especially for sequences with active motion or rich textures, thus further optimizations are proposed in [13]. Quad-tree TU encoding process termination based on the residual coefficients is proposed in [14]–[17]

One of the options in the HEVC test model (HM) [18] to reduce the computational complexity is to use the largest

TABLE II
COMPARISON OF 2D-DCT ARCHITECTURES

Design	Technology	Gate Count	Max. Freq.	Throughput	Supported Video format
TCSVT'14 [4] Architecture-1	90 nm	347 k	187 MHz	5.984 G	8K UHD @ 60 fps
TCSVT'14 [4] Architecture-2	90 nm	208 k	187 MHz	2.992 G	8K UHD @ 60 fps
TCSVT'16 [5] Architecture-1	90 nm	243 k	250 MHz	3.212 G	8K UHD @ 64 fps
TCSVT'16 [5] Architecture-2	90 nm	157 k	250 MHz	1.302 G	8K UHD @ 26 fps
Proposed	32 nm	96 k	450 MHz	3.600 G	8K UHD @ 60 fps

available transform size. The homogeneity of the transform block residuals has a strong relation to the homogeneity of input block; when the TU covers multiple prediction units (PUs) these transform residues may not be consistent and also there is a chance for introducing blocks artifacts which in turn increases the high-frequency energy in the residuals. To cope with computational complexity and the aforementioned problems, this architecture decided to use the maximum TU size that fits in the PU as the TU size. This decreases the computational complexity but the Bjøntegaard delta (BD)-rate [19] increases by 3.02% in the low-delay P configuration [20].

IV. RESULTS AND COMPARISON

The proposed architecture is written in the VHDL hardware description language, and is verified by simulating the design in ModelSim. The design is synthesized using Synopsys Design Compiler version K-2015.06 with Synopsys Armenia Educational Department (SAED) design kit $32\,\mathrm{nm}$ standard logic cell libraries, for operating conditions of $1.16\,\mathrm{V}$ and a worst-case temperature $125\,^{\circ}\mathrm{C}$. The highest throughput of the architecture is 16 pixels per clock cycle while processing a 32×32 block within 64 clock cycles, and varies to a worst-case 2 pixels per clock cycle when processing blocks are in 4×4 size. In high-resolution video, especially for 8K UHD video, lower block sizes are rarely expected, hence as an average, throughput of 16×16 blocks are taken for calculation purposes. Thus to encode 8K UHD @ $60\,\mathrm{Hz}$ in 4:2:0 YUV format requires $7680 \times 4320 \times 60 \times 1.5/8$ clock cycles per second or $374\,\mathrm{MHz}$. The design can operate up to $450\,\mathrm{MHz}$, a much higher clock frequency than is required, and the synthesized results are in Table II.

The synthesized design has an area of $0.2443\,\mathrm{mm}^2$ or a $96\ \mathrm{k}$ standard 2-input NAND equivalent gate count. There are two architectures proposed in each of [4] and [5] for the 2D-DCT, based on unfolded and folded 1D-DCT modules, and are referred to as Architecture-1 and Architecture-2 respectively. The unfolded or full-parallel structures have higher throughput at the expense of larger area or gate count. A comparison is given in the table with the proposed architecture, and is clear that the proposed method has approximately 61% gate count of [5] Architecture-2 and is twice as fast. For an equivalent throughput, the proposed method saves more than 60% gate count of the designs in the table.

V. CONCLUSION

In this paper we propose a 2-D DCT architecture for encoding UHD video in the HEVC/H.265 standard. The hardware has substantial throughput for block sizes that are more likely to be found in HD or UHD video. This assumption removes several unnecessary complexities established in many other architectures. The proposed method has an efficient and fast DCT structures as well as transposition memory. Thus the synthesized results show a lower gate count or a smaller area than the architectures in the literature.

REFERENCES

[1] G. Sullivan, J. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[2] ITU-T and ISO/IEC JTC, *Advanced video coding for generic audio-visual services*, ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC), 2003.

[3] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards – including high efficiency video coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669–1684, Dec. 2012.

[4] P. K. Meher, S. Y. Park, B. K. Mohanty, K. S. Lim, and C. Yeo, "Efficient integer DCT architectures for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 1, pp. 168–178, Jan. 2014.

[5] M. Masera, M. Martina, and G. Masera, "Adaptive approximated DCT architectures for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, no. 99, pp. 1–1, 2016.

[6] M. Jridi and P. Meher, "A scalable approximate DCT architectures for efficient HEVC compliant video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, no. 99, pp. 1–1, 2016.

[7] M. T. Pourazad, C. Doutre, M. Azimi, and P. Nasiopoulos, "HEVC: The new gold standard for video compression: How does HEVC compare with H.264/AVC?" *IEEE Consumer Electronics Magazine*, vol. 1, no. 3, pp. 36–46, July 2012.

[8] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, Jan 1974.

[9] N. C. Vayalil, A. Safari, and Y. Kong, "Overlapped block-processing VLSI architecture for separable 2D filters," in *Electronics, Communications and Networks IV*, Jun 2015, pp. 1355–1358.

[10] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press, Boston, 1990.

[11] M. Budagavi, A. Fuldseth, G. Bjntegaard, V. Sze, and M. Sadafale, "Core transform design in the high efficiency video coding (HEVC) standard," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 1029–1041, Dec 2013.

[12] K. Choi and E. S. Jang, "Early TU decision method for fast video encoding in high efficiency video coding," *Electronics Letters*, vol. 48, no. 12, pp. 689–691, June 2012.

[13] C. C. Wang, Y. C. Liao, J. W. Wang, and C. W. Tung, "An effective TU size decision method for fast HEVC encoders," in *Computer, Consumer and Control (IS3C), 2014 International Symposium on*, June 2014, pp. 1195–1198.

[14] J. Su, K. Nitta, M. Ikeda, and A. Shimizu, "Residue role assignment based transform partition predetermination on HEVC," in *2013 IEEE International Conference on Image Processing*, Sept 2013, pp. 2019–2023.

[15] J. Kang, H. Choi, and J. G. Kim, "Fast transform unit decision for HEVC," in *Image and Signal Processing (CISP), 2013 6th International Congress on*, vol. 01, Dec 2013, pp. 26–30.

[16] Z. Pan, J. Lei, Y. Zhang, W. Yan, and S. Kwong, "Fast transform unit depth decision based on quantized coefficients for hevc," in *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, Oct 2015, pp. 1127–1132.

[17] J. T. Fang, Y. C. Tsai, J. X. Lee, and P. S. Yu, "Computation reduction in transform unit of high efficiency video coding based on zero-coefficients," in *2016 International Symposium on Computer, Consumer and Control (IS3C)*, July 2016, pp. 797–800.

[18] HEVC reference software 16.3. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/

[19] G. Bjøntegaard, *Calculation of average PSNR differences between RD-curves*, ITU-T SG16 Document VCEG-M33, Joint Collaborative Team on Video Coding (JCTVC), Apr. 2001.

[20] V. Sze, M. Budagavi, and G. J. Sullivan, Eds., *High Efficiency Video Coding (HEVC) Algorithms and Architectures*. Springer, 2014.