High Performance 2D-DCT Architecture for HEVC Encoder

Maher Abdelrasoul[1], Mohammed Sayed[1], Maha Elsabrouty[1], Victor Goulart[1,2]

[1]ECE department, Egypt-Japan university of Science and Technology (E-JUST), Egypt

[2]Center for Japan-Egypt Cooperation in Science and Technology, Kyushu University, Japan

email: {maher.salem, mohammed.sayed, maha.elsabrouty}@ejust.edu.eg, victor.goulart@acm.org

Abstract— **The revolutionary Ultra-High Definition (UHD) video has found its way to diverse rich multimedia applications. HEVC (H.265) standard is proposed as the gateway to increase the compression rate with no loss in video quality. Large integer DCT, with sizes 16x16 and 32x32, is one of the key new features of the H.265 standard. In this paper, we propose a new optimized architecture for integer DCT in HEVC encoder. The proposed architecture is a fully pipelined architecture with optimized adder-widths. Simulation results confirm the high performance of the optimized adder-width design. For 16-DCT, the proposed architecture increases the maximum clock frequency by 42.7% and decreases area by 3.1%. While for 32-DCT, the proposed design increases the maximum clock frequency by 64.8% with cost of increasing area by only 3%.**

Keywords—Integer DCT, H.265, HEVC, UHD encoding/decoding, Pipelining, Adders, Adder-width.

I. INTRODUCTION

With the increasing competition in consumer electronics market, better quality multimedia is a highly desirable feature in many applications [1]. However, this increased quality/ resolution comes at the expense of increased multimedia size and even bit rate for multimedia transmission systems. One way to provide this highly ultra-high definition media is to improve the compression of the media to target optimizing both the quality of the multimedia content and minimize the bit rate or storage needed to the minimum possible. In addition, real time processing is key for the delay intolerable multimedia transmission and communication. Many video coding standards, for video compression, have been proposed starting by H.261 in 1990 [2], and ending by the High Efficiency Video Coding (HEVC) standard which known as H.265 in 2012. HEVC focuses on two key issues; increased video resolution and increased use of parallel processing architectures [1].

One of the most important blocks in any video encoder/decoder is the transformation block. Discrete Cosine Transform (DCT) is used in most of video/image coding standards because that it can help in separating the image into spectral sub-bands of differing importance. Integer DCT is employed in the latest video coding standards to reduce the computational complexity and to eliminate the error produced by floating point approximations involved in the traditional DCT.

In many video/image coding standards, image or video frame is partitioned into processing blocks (i.e. prediction blocks). The prediction block is partitioned into square transformation blocks. In the new HEVC standard, the prediction block size is variable and can be larger than the prediction block in the previous H.264 standard. Its range changed from 4*4 to 16*16 in H.264 while it changes from 4*4 to 64*64 in HEVC [3],[4]. Changing the range of the prediction block size changes the range of the transform block size form 4*4 to 8*8 in H.264 to 4*4 to 32*32 in HEVC. The new bigger transform block sizes and the new integer DCT transformation matrix make it important to work on building new integer DCT architectures to achieve high performance in terms of throughput and used resources.

The rest of this paper is organized as follows: in Section II, we give an overview on DCT. The related work and the base architecture of DCT are presented in Section III. Then, our proposed modifications are presented in Section IV. In Section V, we evaluate all the architectures and compare between them. Finally, in Section VI, we conclude the paper.

II. OVERVIEW ON INTEGER DCT

DCT is a transformation from spatial domain to the frequency domain. DCT can be represented as shown in Eq. 1.

$$X = Cx$$

$$\begin{bmatrix} X_0 \\ X_1 \\ \dots \\ X_{N-1} \end{bmatrix} = \begin{bmatrix} C_{0,0} & C_{0,1} & \dots & C_{0,N-1} \\ C_{1,0} & C_{1,1} & \dots & C_{1,N-1} \\ \dots & \dots & \dots & \dots \\ C_{N-1,0} & C_{N-1,1} & \dots & C_{N-1,N-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_{N-1} \end{bmatrix} \quad (1)$$

where $\mathbf{x}_{N\times 1}$ is the input array of values in the spatial domain, $X_{N\times 1}$ is an array of the frequency coefficients, and $\mathbf{C}_{N\times N}$ is the transformation matrix. However, floating-point operations are complex and they result in an error due to the approximation done at the transmitter cannot be retrieved at the receiver.

Integer DCT is an integer approximation of the DCT. It is used to simplify the calculations and to avoid mismatching between coders and decoders [12]. The integer DCT basis functions were derived by approximating scaled DCT basis functions, under considerations such as limiting the necessary dynamic range for transform computation and maximizing the precision and closeness to orthogonality. One of the most important features of the transformation matrix is that the columns of the even rows are mirrored (i.e. for *N*-point DCT, column with index i has the same values of the column with index N-1-i) and the columns of the odd rows are negative

mirrored. This feature is very helpful in reducing the calculations by adding or subtracting inputs that have common multiplier. Eq. 2 shows the reduction of Eq. 2.

$$X_{even} = C_{N/2}\,a \quad , \quad X_{odd} = M_{N/2}\,b \tag{2}$$

where the elements values of the matrices $C_{N/2}$, and $M_{N/2}$ are defined as

$$\begin{aligned} C_{N/2}^{i,j} &= C_N^{2i,j} \quad 0 \le i, j \le N/2-1 \\ M_{N/2}^{i,j} &= C_N^{2i+1,j} \quad 0 \le i, j \le N/2-1 \end{aligned} \tag{3}$$

and the element of the arrays a, and b are defined as

$$a(i) = x(i) + x(N-1-i) \quad , \quad b(i) = x(i) - x(N-1-i) \tag{4}$$

It is important to mention that $C_{N/2}$ is the transformation matrix of N/2-point DCT. This enable the reusability of this transformation as it can be used as N/2-point DCT or as a part of *N*-point DCT transformation. Another important feature is the constant values of the transformation matrix elements, which turn the problem from a traditional matrix multiplication into a set of multiple constant multiplications (MCMs). Consequently, a highly regular architecture and a low-complexity implementation can be achieved.

2D-DCT is implemented by two separable 1D-DCTs. The first DCT does the transformation to the rows of the TB. Then a transpose circuit makes a transposition of the result rows to output the columns of the transformed block as new rows for the next step. The second DCT does the transformation to columns of the result block of the first DCT. Fig. 1 shows the steps of 2D-DCT process.

III. Related Work

In the state-of-art, there is a very large and wide research on DCT. For integer DCT, in the last few years, several different transform cores have been presented. However, most of them were targeting H.264/AVC. Furthermore, low work focused on implementing a complete 2D-DCT. In [10], a 2D-DCT transform in H.264 is implemented by using duplicated 1D-DCT transform and parallel register array are used to realize the transpose operation. Chen et al have developed fast DCT algorithm for H.264 using butterfly approach which is used in implementing fast Fourier transform [11]. In [12], Chih proposed a novel 2D-DCT fast algorithm for realization of 4x4 forward integer transform in H.264 based on matrix operations with Kronecker product and direct sum. In addition, many researches targets hardware implementation for the transform block in H.264 as in [13], [14], and others.

Since the HEVC standard was published recently, there are a few published works about hardware implementations of the DCT transforms for this standard. In [15] three flexible architectures are proposed to perform 1D-DCT operation for any size. This work focused on the reusability and flexibility that allow one design do more than one function. The aim of this work was to reduce the complexity of the encoder and to simplify the computation of the transforms. In [16] W. Zhao and T. Song proposed an architecture based on the similarities of the constants that are multiplied by the input values. In [17] P.K. Meher et al. proposed a DCT architecture based on MCM as done in [16]. However, their architecture is modular so that it enables reusability for the N/2-point DCT module in implementing the *N*-point DCT.

IV. The Proposed Optimized Architecture

Most of the previous work focused on implementing the integer DCT algorithm in an efficient way by replacing the multipliers with shifters and adders. In addition, they exploit the similarities in the coefficients of the transformation matrix. However, to the best of our knowledge, no work was done to optimize the DCT implementation itself through pipelining the critical paths and optimizing the adders' lengths. Such optimization is necessary since the computational complexity of the DCT module increased dramatically in the new HEVC standard due to increasing the transformation block size to up to 32*32 pixels instead of 8*8 pixels in the previous H.264 standard. Therefore, we started with the recently published DCT architecture in [17] and optimized it by pipelining the critical paths and optimizing the adders' lengths. We target higher clock frequency to increase the number of blocks that can be processed per second, hence, process higher video resolutions.

As shown in Fig. 1. The base architecture proposed in [17] is divided into three processing stages. The first stage is Input Adder Unit (IAU), which adds and subtracts the input values as presented in Eq. 4. This IAU is useful in reducing the calculation into two parts as shown in Eq. 2. The first one is *N*/2-point DCT, and the second one is N/2 matrix multiplication. The second stage is Shift Add Unit (SAU). The SAU is a MCM circuit, which is used to multiply each of its inputs by a set of constants. *N*-point DCT has N/2 SAU blocks, each SAU multiply its inputs by N/2 constants. The third sage is Output Add Unit (OAU). The OAU is used to do the final additions on the outputs of Different SAUs. In N-point DCT, each OAU makes N/2 additions to produce the final odd coefficients. In addition, the SAU and OAU are running in parallel with the *N*/2-point DCT unit.

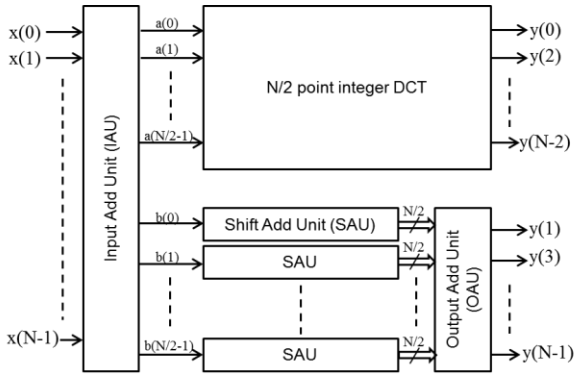

Fig. 1. General Architecture of *N*-point integer DCT

A. Pipelining the architecture

Firstly, we have pipelined the circuit into stages. As shown in Fig. 2, any *N*-point DCT consists of 4 main blocks. The input/add unit (IAU), shift-add unit (SAU), output-add unit (OAU), and *N*/2-point DCT. Therefore, as shown in Fig.3, we have divided the *N*-point DCT into stages equal to the *N*/2-point DCT stages plus one stage for the IAU. The smallest size DCT witch is 4-point DCT is divided into three stages for the

IAU, SAU, and OAU because the calculation of 2-point DCT does not need IAU block.

B. Adder-size optimization.

As mentioned above, the integer DCT implementation, basically, depends on MCM, which took place in the SAU. Several algorithms can be used to generate circuit topologies with shifters and adders to perform MCM process. To the best of our knowledge, the best existing algorithms are BHM [6] which is a modification to BHA [7], and RAG-n [6,8], which is limited to 19 bit constant, and Hcub [9]. Hcub has advantage over the other algorithms since it can optimize the circuit and minimize the critical path by reducing the number of additions in each path although, in some cases, this may add more paths and, hence, increase the total number of adders in the circuit. For example, Fig. 3 shows the circuits generated by the three algorithms to multiply one input by two values (83, and 36).. It can be noted that the circuit generated with the Hcub algorithm has two adders in its critical path while the other circuits have three adders.

After pipelining the DCT architecture, the critical path becomes in the SAU circuits. Using the Hcub algorithm an optimized shift-add structure was generated with the smallest possible depth. In addition, we worked on optimizing the adders' width. Each adder in the SAU circuits was designed with the minimum number of bits that can represent its output. For example, for the adder which results 9x, we can decrease the adder width to be equal the number of the input bits (x size) plus 4 bits only. Adding 4 bits is because the nearest power of two number to 9 and is 16, which mean that the input will be shifted left 4 times. This way of optimization, outcomes two benefits. First, it reduces the delay of each adder and consequently, the delay of the whole circuit. Second, it decreases the area of each adder and hence, the total area of the circuit is decreased.

C. OAU optimization

After optimizing the SAU circuits, the critical path becomes in the OAUs. In *N*-point DCT, OAU adds N/2 inputs. The best way of implementing the OAU in terms of performance is implementing it in tree structure. For any *N*-point DCT, the *N*/2-point DCT takes $\log_2 N$ cycles. Therefore, $\log_2 N-1$ cycles can be used to pipeline the OAU. As the OAU has N/2 inputs, thus it has $\text{Log}_2(N/2)$ stages of adders, which are equal to the number of cycles that can be assign to it. In other words, the OAU is pipelined in such a way that only one adder in each pipeline stage. Fig. 4 shows the pipelining stages of *N*-point DCT. It is clear that pipelining the OAU has no cost in increasing the overall pipeline stages.

V. EVALUATION AND RESULTS

In this section, the performance of the base integer 2D-DCT architecture is evaluated versus the same architecture with optimized SAU and with both optimized SAU and pipelined OAU in terms of frequency and utilized resources. The used prototyping platform is Xilinx Virtex-7 FPGA .

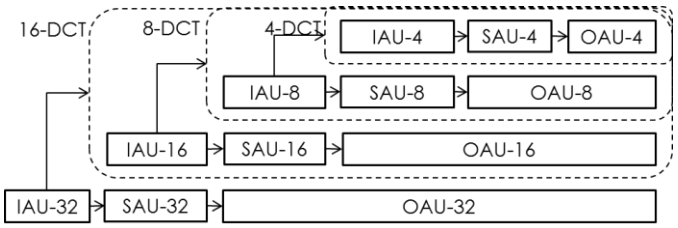

Fig. 2. Pipeline stages of 32-point integer DCT

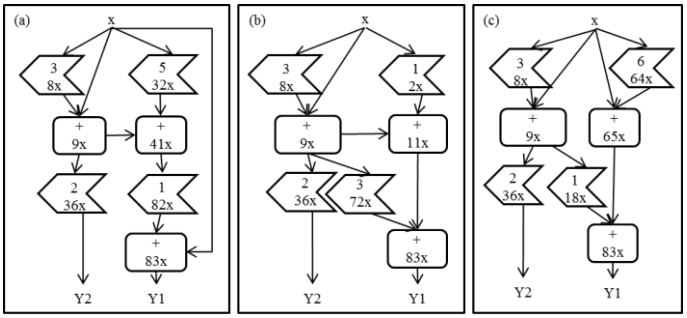

Fig. 3. MCM algorithms (a) BHM (b) RAG-n (c) Hcub

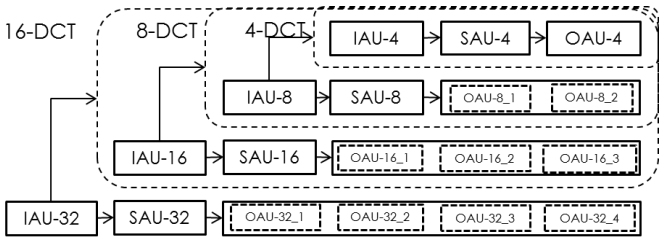

Fig. 4. Pipeline stages of 32-point integer DCT with pipelined OAU

The three architectures were described in Verilog RTL and the correctness of each description is verified by simulating it using ISim tool, which is a tool of Xilinx 14.5 toolset. Then, each design is synthesized and mapped to one of the latest Xilinx FPGA devices (Virtex-7). Virtex-7 family uses 28nm process technology. The slice in Virtex-7 consists of four 6-input LUTs and eight flip-flops [18]. For the synthesis of the designs, Xilinx ISE 14.5 synthesizer was used. Speed was the optimization goal for all designs. The Maximum clock frequency and utilized resources are presented in Fig. 5, and Fig. 6 respectively.

As shown, the SAU optimization effect is presented in terms of maximum clock frequency (Fig.5) for 4,8,16-point DCT. This is because that the SAU is the critical path of the circuit while in 32-point DCT, the OAU is the critical path. Actually, the optimization in the SAU block reduced the SAU delay so that the critical path became in the OAU. Therefore, the next step of optimization by pipelining the OAU clarified the gain achieved by optimizing the SAU which done by reducing the adder sizes in the SAU block to the minimum possible sizes. As mentioned earlier, for 32-point DCT, the critical path is in the OAU. Therefore, the effect of optimizing the SAU block is not shown in the SAU optimization results. However, it will be accumulated to the effect of pipelining at the next step of optimization.

For OAU optimization, as expected the frequency of the circuits increased, mainly for larger DCTs. Where, the OAU logic levels are larger. There was no cost in terms of increasing the number of cycles required to transform one block. For 4-point DCT, the OAU is not pipelined because it is only one level of logic. Therefore, the maximum clock frequency did not change. Indeed, pipelining OAU block shows the benefit of optimizing the SAU, which became the critical path after the pipelining process. The proposed optimizations shows improved maximum clock frequency by approximately 1%, 3.4%, 42.7%, and 64.8% for 4-point, 8-point, 16-point, and 32-point DCT respectively.

For area results (Fig. 6), the area is reduced by optimizing the SAU block. However, due to selecting speed as the optimization goal for the designs, the synthesizer tries to achieve as much speed as possible even if the cost is more area. The proposed optimizations shows a reduction in area resources by approximately 0.5%, 5.4%, and 3.1% for 4-point, 8-point, and 16-point DCT respectively and an increasing by only 3% for 32-point DCT.

The proposed 2D-DCT architecture needs 32.38ns to process one 4*4 block while it needs 57.6ns for 8*8 block, 103.7ns for 16*16 block, and 214.25ns for 32*32 block. In this sense, our architecture can process videos with 8K UHD resolution at 30 fps rate. It may fail to process this resolution only if the whole frame is divided into small 4*4 transform blocks which is a very rare condition that does not happen with such super resolution (i.e. 8K UHD).

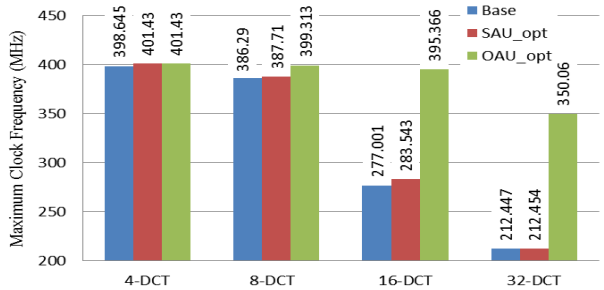

Fig. 5. Maximum clock frequency for different designs of *N*-point integer DCT

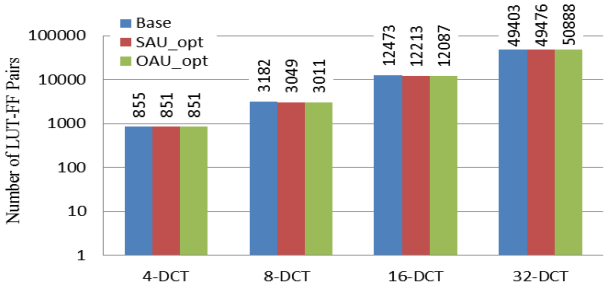

Fig. 6. Number of LUT-FF pairs for different designs of *N*-point integer DCT

VI. CONCLUSIONS

This paper presented an overview on integer DCT for HEVC and the trend of implementing its architecture by MCM algorithms. Further, we have presented a new optimized architecture for 2D-DCT using adder size optimization. The results show a good improvement in both used resources and maximum clock frequency especially for DCT of large block sizes that were added to the new standard to improve video compression for higher resolution videos. For 16-DCT, the proposed architecture increases the maximum clock frequency by 42.7% and decreased number of LUTs by 3.1%. While for 32-DCT, it increases the maximum clock frequency by 64.8% with cost of increasing number of LUTs by only 3%. The proposed architecture can be used in processing 8K UHD video in real time.

REFERENCES

[1] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview ofthe High Efficiency Video Coding (HEVC) Standard," IEEE Transactions on Circuits and Systems for Video Technology,vol.22, NO.12, pp.1649-1668, 2012.

[2] Ohm, J., Sullivan, G.J., Schwarz, H., Thiow Keng Tan, "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:1-15, 2012.

[3] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 9, pp. 1103–1120, 2007.

[4] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Trans. Circuits Syst.Video Technol., vol. 13, no. 7, pp. 560–576, 2003.

[5] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete cosine transform,"IEEE Trans. Comput., vol. C-23, pp. 90-93, 1974.

[6] A. G. Dempster, M.D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal processing, vol.42, no.9, pp.569-577, 1995.

[7] D. R. Bull, D. H. Horrocks, "Primitive operator digital filters," IEE Proceedings of Circuits, Devices and Systems, vol.138, no.3, pp.401-412, 1991.

[8] A. G. Dempster, M.D. Macleod, "Constant integer multiplication using minimum adders," Circuits, Devices and Systems, IEE Proceedings, vol.141, no.5, pp.407-413, 1994.

[9] Y. Voronenko, and M. Püschel, "Multiplierless Multiple Constant Multiplication", ACM Transactions on Algorithms (TALG), vol. 3, Issue 2, May 2007.

[10] L. Ling-Zhi, Q. Lin, R. Meng-Tian, and J. Li, "A 2-D forward/inverse integer transform processor of H.264 based on highly-parallel architecture," in 4th IEEE Int. Workshop System-on-Chip Real-Time Applications, pp. 158–161, 2004.

[11] W. H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," IEEE Trans. Commun.,vol. COM-25, no. 9, pp. 1004–1009, 1977.

[12] C.-P. Fan, "Fast 2-dimensional 4x4 forward integer transform implementation for H.264/AVC," IEEE Trans. Circuits Syst. II, vol. 53, no. 3, pp. 174–177, 2006.

[13] L. Ling-Zhi, Q. Lin, R. Meng-Tian, and J. Li, "A 2-D forward/inverse integer transform processor of H.264 based on highly-parallel architecture," in 4th IEEE Int. Workshop System-on-Chip Real-Time Applications, pp. 158–161, 2004.

[14] C.-P. Fan, "Cost-effective hardware sharing architectures of fast 8x8 and 4x4 integer transforms for H.264/AVC," in 2006 IEEE Asia Pacific Conf. Circuits Syst. (APCCAS'06), pp. 776–779, 2006.

[15] S. Y. Park and P. K. Meher, "Flexible Integer DCT Architectures for HEVC", IEEE International Symposium on Circuits and Systems (ISCAS), 2013.

[16] Wenjun Zhao; Onoye, T. ; Tian Song, "High-performance multiplierless transform architecture for HEVC", IEEE International Symposium on Circuits and Systems (ISCAS), 2013.

[17] 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, 2013.

[18]http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf