Journal of Operation and Automation in Power Engineering

Vol. 10, No. 2, Aug. 2022, Pages: 143-154

http://joape.uma.ac.ir

A Low-Cost Multi-Sized HEVC Core Transform Using Time-Multiplexed DCT Architectures

R. Younesi[1,*], *M. J. R. Fatemi*[1], *M. Rastgarpour*[2]

[1]*Department of Electrical Engineering, Faculty of Engineering, Islamic Azad University, Saveh Branch, Saveh, Iran*
[2]*Department of Computer Engineering, Faculty of Engineering, Islamic Azad University, Saveh Branch, Saveh, Iran*

Abstract- *High Efficiency Video Coding (HEVC) is one of the latest coding standards targeting high-resolution video contents. Due to the high complexity of the existing hardware implementation, this paper presents the low-cost and efficient DCT architectures for HEVC, which are able to perform DCT operation of multiple transform sizes in a single unified architecture. Our objective is to reuse the hardware resources in a DCT architectures using configurable constant multipliers as well as reducing the hardware cost and trading off between hardware complexity and efficiency. We propose three different shift-and-add units with different hardware cost and throughput. The main advantage of the proposed architectures over the existing architectures is a lower hardware and it can also perform DCT transform of different transform units which is available in HEVC standard. The experimental results over 90-nm technology show that the proposed 2D-DCT architecture #1 archives the lowest hardware cost amongst the rest of the architectures with around 57% reduction in gate count, on average. The unfolded 2D-DCT architectures #2 and #3 offer the moderate reduction in gate count around 47%, on average, with a moderate throughput. Apart from architectures #1, #2, and #3, we also develop a reusable architecture by adding an extra* $(N/2)$*-point DCT alongside the main DCT.*

Keyword: DCT; HEVC; Low-Cost; Multi-Sized; Time-multiplexed

1. INTRODUCTION

High Efficiency Video Coding (HEVC) is one of the recent video coding standards, which is widely used in transmission of high definition (HD), and ultra-high definition (UHD) video contents. This emerging video coding was jointly developed by the ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG) and currently has a widespread application in multimedia streaming, broadcast television, multimedia content storage, and mobile communication. The promising feature of the HEVC standard relates to the fact that it offers more than 50% improvement in coding efficiency over its predecessor H.264/MPEG-4 AVC with relatively the same video quality [1]. This higher data compression ratio reduces storage requirements and enables the streaming of higher resolution videos over a limited wireless network. However, HEVC suffers from the higher cost of design complexity by about 40 - 70%, resulting in a higher resource utilization which is not tolerated, especially for resource-constraint IoT devices [2]. This higher complexity mainly pertains to the core transform of HEVC standard in which a Discrete Cosine Transform (DCT) with multiple block sizes is employed. The DCT is a fundamental yet complicated transform that is widely used in different applications of image and video processing [3]–[6], wireless capsule endoscopy [7], and steganography [8]. Unlike its predecessor, the HEVC uses DCT with different transform sizes ranging from 4×4 to 32×32 block sizes. It was found that separate implementation of DCTs of different sizes increases the area utilization remarkably compared to the unified implementation [9]. In order to rectify the higher design complexity of HEVC and enable its simple implementation, two common approaches can be taken into account. The first and the straightforward way is the use of integer approximation of the transform kernel, where the matrix's elements of different sizes are provided by Ref. [10]. The second way is to exploit the commonality in the arithmetic units so as to share the hardware resources as far as possible. In such a case, a unified architecture is developed which is able to perform DCT of different transform sizes. Accordingly, many works till date are being dedicated to develop approximated and integer implementations of multi-sized DCTs so as to alleviate the computational complexity of DCT of different sizes [11]–[15]. The

Received: 19 Jul. 2021
Revised: 28 Sep. 2021
Accepted: 14 Oct. 2021
*Corresponding author:
E-mail: r_younesie@yahoo.com (R. Younesi)
DOI: 10.22098/joape.2022.9656.1675
Research Paper

most common strategy in the recent efforts is the factorization strategy which is based on the partial-butterfly factorization scenario. In this scenario, an N-point DCT is recursively decomposed into an $(N/2)$-point lower point DCT (i.e., even part) and an $(N/2)\times(N/2)$ matrix multiplication (i.e., odd part). The matrix multiplication can be realized by multiple constant multiplication (MCM) problem, where some arithmetic elements are shared between k constant multipliers, producing k multiplication outputs in parallel. Therefore, the integer elements of transform matrix are implemented using simple shift and add units (SAU) rather than fixed-point multipliers.

In this paper, we mainly focus on developing the low-cost and low-complexity SAUs embedded in DCT architecture by exploiting the commonality in the arithmetic unis as well as sharing the hardware resources to trade-off between area consumption and throughput. Accordingly, we propose three different SAUs, presenting different area overhead and throughput. By leveraging time-multiplexing technique, the proposed SAUs can be embedded in the DCT unit to provide an area-efficient HEVC core transform. The exiting architectures based on partial-butterfly factorization only exploit the hardware sharing in even block and the rest of the blocks including odd block cannot be shared any more. However, the proposed architectures can increase the hardware sharing by employing configurable multipliers in the odd blocks of DCT, thereby lowering the hardware cost and area consumption. The main advantage of the proposed architectures over the existing architectures is lower hardware cost in terms of gate count and it can also perform DCT transform of different transform units which is available in HEVC standard. The major contributions of this paper are as follows.

1) We propose three different SAUs by leveraging time-multiplexing technique so as to increase the hardware reusability while trading-off between hardware complexity and throughput.
2) Three low-cost 1D-DCT architectures are presented based on the proposed SAUs, each of which outperformers one another, in terms of area consumption, throughput, and processing frames per second (fps).
3) We develop two 2D-DCT architectures based on the folded and fully-parallel structures. The former one derives an area-efficient architecture, whereas the latter one which benefits from higher throughput, at the cost of additional 1D-DCT block.

The rest of paper is organized as follows: Section 2 provides a literature review. Section 3 presents a brief review on integer-approximated DCT architectures. The proposed 1D-DCT and 2D-DCT architectures are provided in Section 4. The experimental results are provided in Section 5 and finally a comprehensive conclusion is drawn in the last section.

2. LITERATURE REVIEW

In order to reduce hardware complexity, two different approaches toward implementing DCT architecture for HEVC application can be seen from the literature: 1) Fixed-point implementations [2], [16], [17] and 2) Integer approximations of the DCT [11]–[13], [15], [18], [19]. The fixed-point implementations of multi-sized DCT are presented in Refs. [2], [16], [17], [20], [21]. Authors of [16] proposed an architecture that exploits the relationship between Walsh Hadamard transform (WHT) [22] and DCT. However, this architecture requires a large number of rotation units and incurs high design overhead since it uses the precomputed WHT results from the prediction units. To rectify the above limitation, authors of [2] proposed an approximate version that can dynamically skip some rotations based on the content of input signal. In other work [20], [23], authors proposed a multi-sized DCT architecture which used Chen's algorithm [24] as well as WHT based matrix decomposition method to reduce the hardware complexity. To eliminate the rotation units, they replaced all the coefficients using an approximation technique by employing dyadic values. Another fixed point approximation of DCT coefficients is also introduced in Ref. [21] which improves hardware cost with minimal degradation in coding performance. Recently, an efficient variable-sized fixed-point DCT architecture is proposed by Ref. [17] that investigates the existing DCT factorization in order to identify which one minimizes the amount of hardware resources. On the other hand, [11]–[15] deal with the integer DCT approximation defined in the HEVC standard. Meher *et al.* [13] proposed a variable-sized integer architecture using partial-butterfly factorization strategy. Darij *et al.* [14] proposed an efficient architecture of HEVC core transform for computing 4, 8, 16, and 32-point DCT by using the Canonical Signed Digit (CSD) representation and Common Sub-expression Elimination (CSE) technique to reduce the number of adder and shifter blocks.

3. AN INTEGER-APPROXIMATED 1D-DCT FOR HEVC

An N-point 1D-DCT operation can be expressed as (1), where $i = 0, 1, .., N-1$, x_i is an input vector, Y_i is the output coefficients, and c_{ij} is the elements of the DCT transform matrix C which is defined by (2). Here, d is equal to 1 and $\sqrt{2}$ for $i = 0$ and $i > 0$ respectively.

$$Y_i = \sum_{j=0}^{N-1} c_{ij} x_j \tag{1}$$

$$c_{ij} = \frac{d}{\sqrt{N}} \cos\left[\frac{\pi}{N}(j+\frac{1}{2})i\right] \tag{2}$$

According to Eq. (2), c_{ij} represents an $N \times N$ transform matrix with real-valued elements. Realizing infinite precision real-valued transform matrix elements incurs a significant area overhead and power consumption of final implementation. Furthermore, to avoid encoder-decoder mismatch and drift caused by manufacturers realizing DCTs with different floating point representations, the core transform matrices of HEVC decoder are defined as the approximation to the real-valued DCT matrix [9]. Using a matrix representation, one can alternatively define DCT computation of Eq. (1) as $[Y_N]^T = [T_N] * [X_N]^T$, where T_N is an $N \times N$ transform matrix, X_N and Y_N are N-point input samples and output coefficients, respectively. The core transform matrix T_N for different N-point DCTs on the basis of integer approximation is defined by HEVC core transform [9]. According to the core transform specified in [9], the 4-point and 8-point transform kernels can be defined as Eqns. (3) and (4).

$$T_4 = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} \tag{3}$$

$$T_8 = \begin{bmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 \end{bmatrix} \tag{4}$$

Using the partial-butterfly strategy, an N-point DCT kernel matrix can be decomposed into an $(N/2)$-point even and $(N/2)$-point odd parts, where the even part can be further decomposed into the lower point DCTs. In this strategy, the $(N/2)$-point even parts represent the DCT computation of the $(N/2)$-point, while the odd part can be realized by multiplication of an $(N/2)\times(N/2)$ matrix. Accordingly, the 8-point 1D-DCT which is defined as $[Y_8]^T = [T_8] * [X_8]^T$ can be decomposed as expressed in Eq. (5), where T_4 is a transform matrix of 4-point DCT as defined by Eq. (3), O_4 represents the matrix multiplication used in 4-point odd block as Eq. (7), and a_i and b_i are the even and odd outputs of butterfly block, which are defined according to Eq. (6). It is worth pointing out that the $T_{\mathrm{N}/2}$ and $O_{\mathrm{N}/2}$ matrices can be specified by using the even and odd rows of the first half of the basis vectors in matrix T_{N}, respectively, as derived by Eq. (8).

$$\begin{aligned} [y_0, y_2, y_4, y_6] &= T_4 \times [a_0, a_1, a_2, a_3] \\ [y_1, y_3, y_5, y_7] &= O_4 \times [b_0, b_1, b_2, b_3] \end{aligned} \tag{5}$$

$$\begin{aligned} a(i) &= x(i) + x(N-i+1) \\ b(i) &= x(i) - x(N-i+1) \end{aligned} \tag{6}$$

$$O_4 = \begin{bmatrix} 89 & 75 & 50 & 18 \\ 75 & -18 & -89 & -50 \\ 50 & -89 & 18 & 75 \\ 18 & -50 & 75 & -89 \end{bmatrix} \tag{7}$$

$$\begin{aligned} T_{N/2}(i, j) &= T_N(2*i, j) \\ O_{N/2}(i, j) &= T_N(2*i+1, j) \end{aligned} \tag{8}$$

Similarly, the larger N-point DCTs can be computed by incorporating a lower $(N/2)$-point DCT, an $(N/2)$-point odd block, and N-point butterfly structure. Direct realization of algorithm presented in Eqns. (5)-(7) is referred to as reference algorithm in the rest of paper.

4. PROPOSED 1D-DCT ARCHITECTURE USING LOW-COST SHIFT-AND-ADD UNITS

In order to lower the complexity of an N-point 1D-DCT architecture, we mainly focus on developing the low-cost odd blocks of lower points, as the even blocks can be further decomposed into the lower point odd and even blocks. An N-point odd block consists of a matrix multiplication with integer elements as expressed by Eq. (5). A careful analysis of odd matrix (O_{i}) reveals that some integer elements can be reused when calculating that of the other odd outputs. This hardware sharing which targets the reusability of the entire circuit of an element offers the best hardware cost, at the expense of higher latency. In such a case, the multiplexers are employed in such a way that the input samples are time-multiplexed when feeding into the constant multiplication section.

The alternative or complementary approach is to share the constituent elements of arithmetic units in terms of shift and add units to increase hardware sharing as well as reducing the hardware cost. In such a case, multiple constant multipliers are unified by sharing the adder and shift blocks in a time-multiplexed way. The time multiplexing technique is a way of scheduling arithmetic operations in a consecutive manner. This strategy uses the minimum required hardware resources, at the expense of higher latency. In this way, a configurable hardware can be achieved resulting in a more flexible and integrated architecture. In this paper, we have utilized the time-multiplexing technique to design SAU blocks of DCT in order to minimize number of required arithmetic units as well as lowering the hardware cost and increasing the hardware sharing. For doing so, we exploit the redundancy in the adder and shifter blocks required for constant multiplications by using the reconfigurable constant multipliers. In fact, the additions required for different constants are time-multiplexed to reuse the same adders in the final circuit. The configurable multipliers use the multiplexers (Mux) and a control signal to share the same adder and shift blocks. Fig. 1 shows a configurable multiplier, where an input signal x is multiplied by a set of constants $\{C_1, C_1, \dots, C_N\}$ according to the value of control signal sel. By selecting the proper value of sel, the multipliers can be configured based on a specific constant. An example of configurable multiplier is depicted in Fig. 2 (b) in which the input signal is multiplied by 11 and 21 constants. It can be seen from Fig. 1(a) that the configurable constant multiplier consists of two adders, three shift blocks, and one multiplexer. It is worth noting that the configurable constant multiplication is derived by integrating different constant multiplications with an objective of minimizing area consumption. There is another alternative constant multiplier called parallel constant multiplication which in contrast to the configurable constant multiplication, reduces the latency of hardware by increasing number of arithmetic units. Unlike the configurable constant multiplication, the parallel multiplication architecture no longer needs multiplexers, at the expense of higher number of adder and shift blocks. As a result, the area consumption and hardware cost of the configurable constant multiplication is expected to be lower than the parallel one owing to the hardware reusability when realizing DCT architectures.

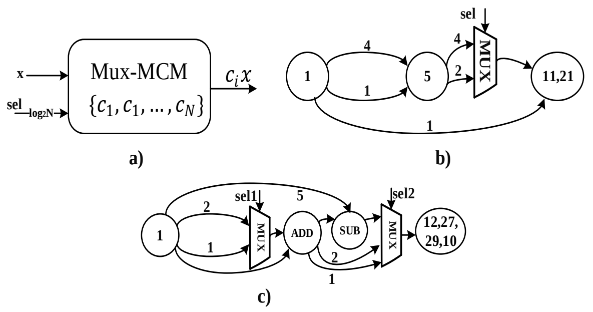


Fig. 1. a) Structure of configurable multiplier; b) example of configurable multiplication with 11 and 21 constants; c) configurable multiplier with 12,27,29,10 constants

4.1. Proposed Shift-and-add Units (SAUs)

Multi-sized processing in DCT means the ability to perform DCT operation on different size of DCT block in a single unified hardware. DCT in HEVC codec uses different size of DCT block ranging from 4*4 to 32*32 square blocks and the separate implementation of DCT targeting a specific block size is not an efficient way from a hardware point of view. A simple but effective approach is to integrate all the hardware required for processing different block of DCT into a single hardware by using the hardware sharing strategy. In such a case, the hardware resources are shared and the hardware cost is reduced accordingly. Due to the prominent feature of the configurable constant multiplication in terms of hardware cost and complexity, we present three different architectures of SAU embedded in our proposed N-point 1D-DCT architectures as shown in Fig. 2.

We have presented three different SAU block, which are embedded in the proposed DCT architecture. The first one is the area-efficient architectures that employs only one configurable multiplier with N/2 constants. In this architecture, we have shared the arithmetic units of all the available constants in SAU to increase the hardware sharing as well as lowering the hardware cost. The second architectures increases the hardware efficiency and throughput by increasing the number of processing elements (i.e., configurable multipliers). This architecture incurs higher hardware cost but it operates in higher throughput and can support higher video resolution. The third design has the same number of multiplier, but with less number of constants in each multiplier, at the expense of adding extra mux unit. The last design further increases the hardware throughput and the hardware cost. These three architectures offer different hardware cost and hardware efficiency for HEVC application. Fig. 2(a) shows the simplest form of SAU (i.e., architecture #1) comprising an $N/2$ to 1 multiplexer unit and one configurable multiplier which integrates $N/2$ constant multipliers in a single low-cost hardware using steering multiplexers. The proposed SAU architecture #1 directs the appropriate odd outputs of butterfly block $B_{N/2} = \{b_0, b_1, \dots, b_{N/2-1}\}$ to the configurable multiplier by using $C1$ signal. Meanwhile, C2 signal is responsible for selecting the suitable constant multiplier in each clock cycle amongst the $N/2$ constant embedded in our configurable multiplier. The $N/2$ constants included in the configurable multiplier are specified according to the first column of odd matrix ($O_{N/2}$) as derived from Eq. (8) which can also be defined by the odd rows of the first half of the transform matrix. In configurable multiplier block, we use a heuristic algorithm presented in Ref. [25] that tries to lower the number of required adder and shift blocks by sharing the redundant elements. Every result coming from the configurable multiplier block is registered and then added by the previous sum which already exists in that of the other register. After $N/2$ clock cycles, the result of $N/2$ constant multiplications have been accumulated in the output register, producing the first output of odd block. As a result, generation of $(N/2)$-point odd outputs requires $N^2/4$ clock cycles.

Fig. 2(b) shows the next SAU architecture (i.e., architectures #2), where the odd outputs are generated every clock cycle. In this architectures, the more hardware resources are utilized relative to the previous architecture by employing a dedicated configurable multiplier for each input sample while increasing the hardware throughput. It consists of $N/2$ configurable multipliers, each of which contains $N/2$ constant multipliers. Finally, an adder tree is deployed to add the intermediates output of configurable multipliers and generate the final output. The adder tree can be realized by the use of conventional full adders or the combination of carry lookahead adder (CLA) and carry save adder (CSA) to boost the performance of architectures. Also, other low-power and high-speed adders can be employed [26]. Signal $C2$ is responsible for selecting the right choice of constant of each configurable multiplier block. The $N/2$ constants included in each configurable multiplier corresponded to $b(i)$ are specified according to i^{th} column of odd matrix ($O_{N/2}$) as derived from (8). With respect to architecture #1, the proposed architecture #2 provides the higher throughput, albeit at the expense of higher area overhead. Fig. 2(c) presents the last SAU architetcure (i.e., architetcure #3) using configurable multipliers. Architetcure #3 consists of a permutation unit, two mutiplexer units, $N/2$ configurable multipliers, and an adder tree. The configurable multipliers can be divided into the right half and the left half, where each half is supplied through a dedicated multiplexer unit. The permutation units is responsible for changing the order of $N/2$ input samples $B_{N/2} = \{b_0, b_1, \dots, b_{N/2-1}\}$ so that it splits the input sample into two parts of $R_{N/4} = \{b_0, b_3, b_4, b_7, \dots, b_{N/2-1}\}$ and $Q_{N/4} = \{b_1, b_2, b_5, b_6, \dots, b_{N/2-2}\}$. The first part of the splitting process ($R_{N/4}$) is supplied to the left half of the multiplier blocks, and that of the other ($Q_{N/4}$) is supplied to the right half of the multipliers. Each configurable multiplier includes two constants in such a way the first and the second constants are specified by the elements of the first and third rows of odd matrix ($O_{N/2}$), respectively. For the left half of the configurable multipliers, the first and the second constants $({m_i}^1, {m_i}^2)$ are $O_{N/2}(0,k)$ and $O_{N/2}(2,k)$, respectively, where k is equal to the index of the ith elements of $R_{N/4}$. The same procedure is also valid for the right half of the configurable multipliers. Similar to architecture #2, the throughput of the proposed architecture #3 is one output per cycle. In comparison to architecture #2, number of constant multiplier is reduced to two constant per multiplier for architecture #3, at the expense of adding additional multiplexer units. Now, we cannot provide any comparison between architecture #2 and #3 in terms of hardware cost. However, such comparison mainly depends upon the amount of hardware sharing that architecture #2 can employ. The more hardware sharing, the less hardware cost it incurs compared to architecture #3. In fact, architecture #2 employs a large number of multiplexers embedded in the configurable multipliers, but architecture #3 has less multiplexers inside the multipliers and more of it in outside of the multipliers.

4.2. Proposed 8-point DCT Architecture

In this section, we present the proposed 8-point DCT architecture by employing a 4-point odd and 4-point even parts. The $(N/2)$-point odd block was introduced in the previous section. The lowest point even part (i.e., 4-point DCT) can be developed in the same way as the odd part. The higher point even parts can be decomposed into the lower odd part and even parts. Therefore, to develop a multi-sized N-point DCT architecture, it is required to develop the $(N/2)$-point odd blocks accompanied by 4-point even block. Accordingly, by employing different SAU units proposed in the previous section, three different DCT architectures can be introduced.

Fig. 3 shows the proposed 8-point DCT architectures by employing 4-point even and 4-point odd blocks. Fig. 3(a) presents the proposed 8-point DCT architecture based on the SAU architecture #1 as introduced in Fig. 2(a). It consists of two configurable multipliers, each of which contains four constants, where the constants related to the odd block are specified according to the first column of O_4 matrix, as expressed by (7). On the other hand, the constants related to the even block (i.e., 4-point DCT) is specified according to the first column of T_4 matrix, as expressed by (4). The butterfly unit consist of eight adders generating four samples of odd (b_i) and four samples of even blocks (a_i). Two 4 to 1 multiplexers are also employed to steer the proper inputs to multipliers. The architecture generates two coefficients every four clock cycles; Thus, for generating eight coefficient of 8-point DCT, sixteen clock cycles are required. Fig. 3(b) shows the next 8-point DCT architecture which is developed based on the proposed SAU architecture #2. It consists of eight configurable multipliers, each of which contains four constants. The four constants related to the multipliers of odd and even blocks attached to b_i and a_i are specified according to the i^{th} column of O_4 and T_4 matrices, respectively. It also includes total of six adders in even and odd blocks together with eight adders of butterfly unit. The throughput of architecture is two coefficients per cycles; thus, for generating eight outputs, four clock cycles are required.

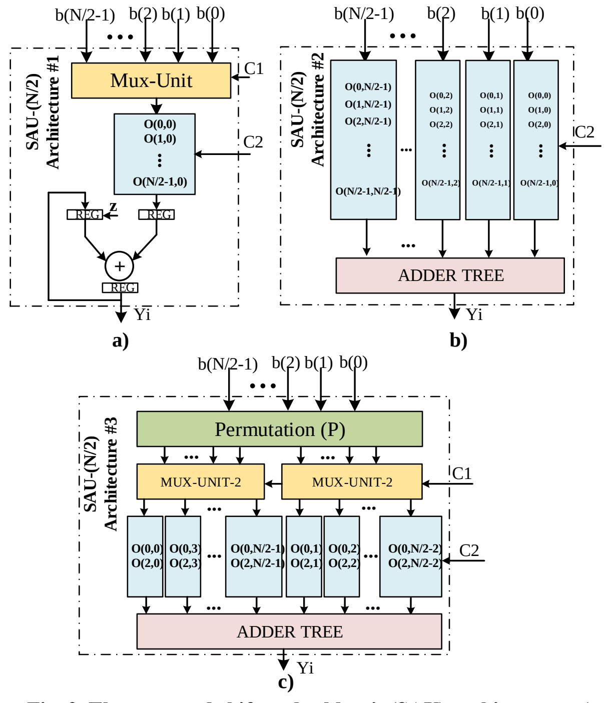
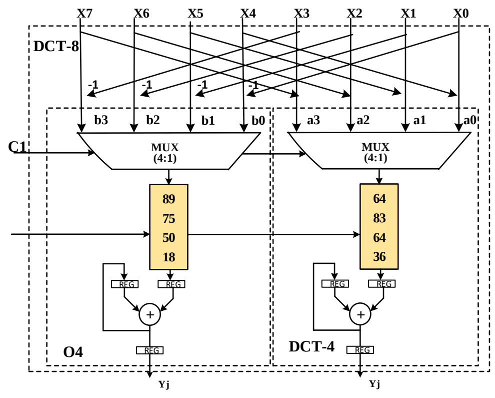


Fig. 2. The proposed shift-and-add unit (SAU) architectures a) architecture #1 with only one multiplier and $N/2$ constants, b) architecture #2 with $N/2$ multipliers and $N/2$ constants, and 3) architecture #3 with $N/2$ multipliers and two constants

Fig. 3(c) shows the last DCT architecture which incorporates the proposed SAU architecture #3. It includes two permutation units ($P1$ and $P2$), one for 4-point odd block (O_4) and that of the other for 4-point DCT. The dashed lines inside the permutation units of $P1$ and $P2$ highlight the splitting process of input samples of b_i and a_i, respectively. The splitting process inside the odd block (i.e., $P1$), divide the input samples $\{b_0, b_1, b_2, b_3\}$ into two partitions of $\{b_0, b_3\}$ and $\{b_1, b_2\}$. However, the splitting process in the even part is quite different so that the $P2$ divides the input samples of $\{a_0, a_1, a_2, a_3\}$ into two partitions of $\{a_0, a_1\}$ and $\{a_2, a_3\}$. It also requires eight 2 to 1 multiplexers, four units for the left half and four units for the right half of the multipliers. Also, eight multipliers are needed, each of them requires two constants. The constants are specified according to the rule which was previously explained in subsection 4.1. Finally, the results of eight multipliers are added through an adder three, producing two output coefficients per cycles. As a result, the throughput of 8-point DCT architectures #2 and #3 are the same.

4.3. Proposed 16-point DCT Architecture

Fig. 4 shows the proposed 16-point DCT architectures by employing the proposed SAUs accompanied by the 8-point DCT architectures of the subsection 4.2. The same procedure is carried out when developing 16-point DCT as 8-point DCT by decomposing it into 8-point even and odd blocks.



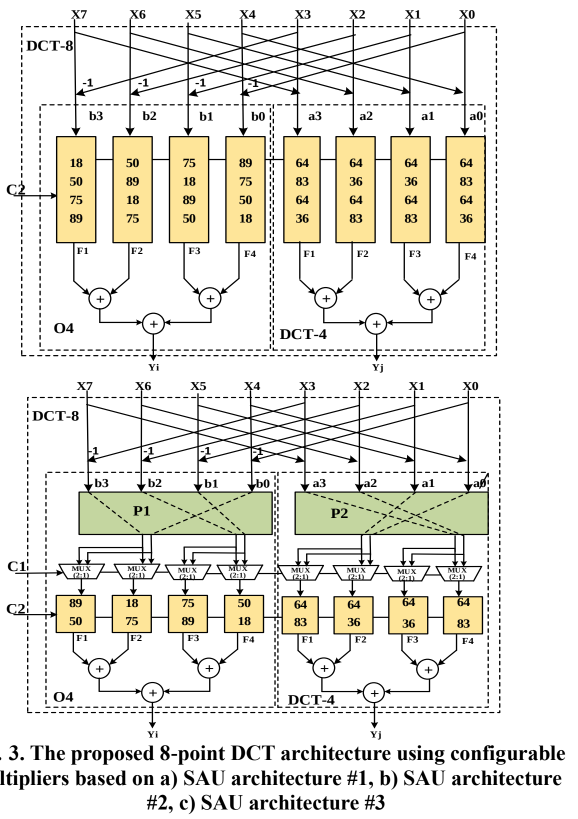
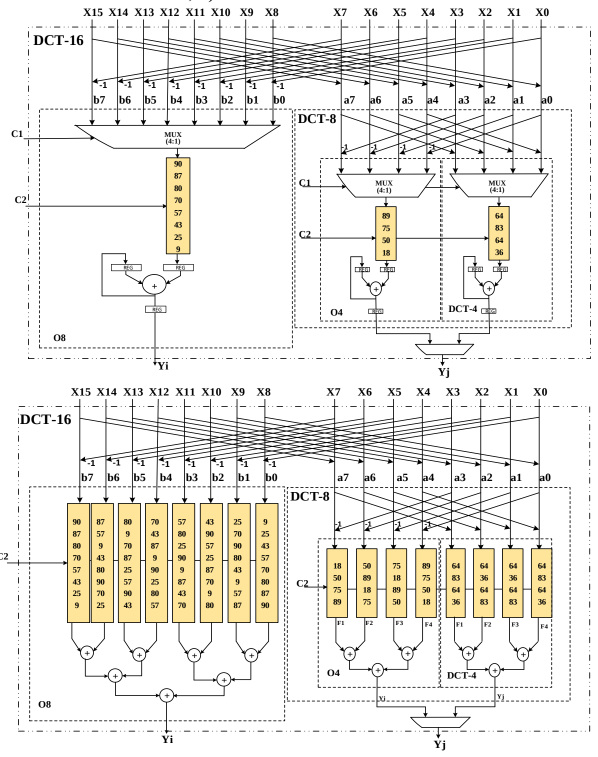


Fig. 3. The proposed 8-point DCT architecture using configurable multipliers based on a) SAU architecture #1, b) SAU architecture #2, c) SAU architecture #3



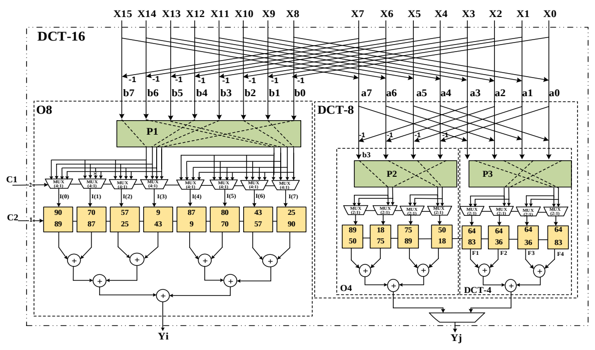


Fig. 4. The proposed 16-point DCT architecture using configurable multipliers based on a) SAU architecture #1, b) SAU architecture #2, c) SAU architecture #3

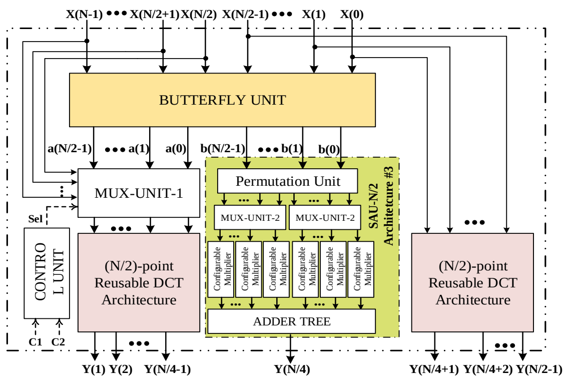


Fig. 5. The proposed reusable architecture by employing an additional $(N/2)$-point DCT and using SAU architecture #3

Fig. 4(a) shows the 16-point DCT architecture based on SAU architecture #1. In order to generate eight output coefficients of odd block (O_8), it requires 64 clock cycles, since each coefficient takes eight clock to be generated in architecture #1. On the other hand, the even part (i.e., DCT-8) requires 16 clock cycles to generate eight output coefficient related to the even block. As a result, 64 clock cycles is required to generate 16 output coefficients of 16-point DCT architecture #1 in a worst case scenario. Fig. 4(b) shows the 16-point DCT architecture based on SAU architecture #2. The throughput of architecture #2 is two output coefficients per cycle; thus, eight clock cycles are required so as to generate 16 coefficients. Lastly, Fig. 4(c) presents 16-point DCT based on SAU architecture #3. The throughput of DCT architecture #3 is the same as architecture #2, but all the multipliers include only two constants. This feature is in contrast to architecture #2, where the constants embedded in multiplier blocks increases by increasing the DCT transform size. Therefore, it is expected that the architecture #3 incurs less hardware cost within multiplier blocks compared to architecture #2, especially for higher transform sizes. However, it should be noted that the number and the size of steering multiplexers of architecture #3 are increased when the DCT transform size increases. In overall, there is not a clear evidence that before implementation indicates which architecture incurs less hardware cost. The same procedure can be carried out for developing the proposed 32-point DCT architecture. The proposed 32-point DCT architecture can compute the DCT computation of the lower points (i.e., one 16-point, one 8-point, and one 4-point) as well, since the lower point DCTs are embedded in higher point DCTs.

To further enhance the throughput of architecture, we can add additional $(N/2)$-point DCT alongside the main $(N/2)$-point DCT and $(N/2)$-point SAU unit. Fig. 5 shows the general architecture of the modified DCT architecture. We called this architecture the reusable architecture for the rest of the paper. This modification doubles the previous throughput, as the 32-point DCT with an additional 16-point DCT can computer one 32-point, two 16-point, four 8-point, and eight 4-point DCTs in parallel. The reusable architecture includes an additional $(N/2)$-point DCT alongside the main $(N/2)$-point DCT. It can process a variable-sized DCT processing meaning that it can perform multiple lower-point DCT in parallel. In contrast to the architectures #1, #2, #3, where they can only perform a N-point DCT one at the time, the reusable architecture can perform multiple lower point DCT in parallel resulting in higher throughput and supporting higher resolution contents in HEVC application. So, the reusable architecture benefits from higher throughput and coding efficiency compared to the other architecture, at the expense of higher hardware cost because of an additional $(N/2)$-point DCT block.

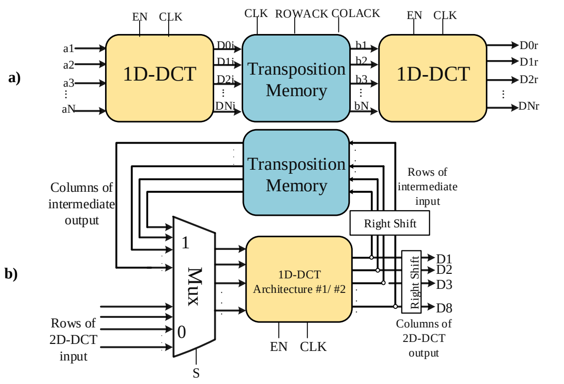


Fig. 6. The proposed 2D-DCT architecture; a) unfolded 2D-DCT, b) folded 2D-DCT structure

4.4. Proposed 2D-DCT Architetcure

The architecture of 2D-DCT can be realized by row-column approach by employing two 1D-DCT blocks and a transposition butterfly [10]. Fig. 6 (a) shows the unfolded 2D-DCT architecture. At first, the first 1D-DCT is fed with the rows of the 32×32 input block, then the intermediate output generated by the first 1D-DCT are stored in transposition memory. After processing all the rows of input block, the column of transposition memory is supplied to the second 1D-DCT generating 32 output coefficients. It should be noted that when processing the second 1D-DCT to generated 2D-DCT outputs, the first 1D-DCT computation of another input block can be performed and its intermediate results can be stored column-wise in transposition memory. This mechanism increases the throughput of the final implementation at the cost of adding additional 1D-DCT architecture.

Another approach that has focused on reducing the hardware cost of 2D-DCT is the folded structure. The folded 2D-DCT employs only one 1D-DCT and a transposition memory. However, the intermediates output samples are time-multiplexed in such a way that the next 1D-DCT operation over the columns of intermediate block is performed on the same 1D-DCT unit using a multiplexer unit. This architecture reduces the hardware cost, at the expense of higher latency since the second 1D-DCT operation must be stall until the first 1D-DCT over the rows of input block is terminated. Fig. 6(b) shows the folded 2D-DCT architecture.

5. EXPERIMENTAL RESULTS

5.1. Details of experiment

We have implemented the proposed 1D-DCT and 2D-DCT architectures using VHDL language and synthesized them using Synopsys Design Compiler® in TSMC 90-nm technology node. The verification of our architecture is carried out by ModelSim software and the experimental results have been performed using a personal computer with Intel Core-i7, 8 gigabyte DRAM. Also, we used the heuristic algorithm which was presented in Ref. [27]. The frequency of the architectures is set to the maximum tolerable frequency of hardware and the library is set to the typical timing condition. Moreover, the wire load is set to the lowest value available by the technology node. All the experimental results are directly extracted from the design compiler tool and compared with the available results in the literature. We develop four different architectures to provide a comprehensive comparison in terms of hardware cost and hardware efficiency. Architectures #1, #2, and #3 refer to the N-point DCT architecture which embeds the proposed SAU architectures #1, #2, and #3, respectively. The proposed reusable architecture refers to the architecture #3 that an additional $(N/2)$-point DCT has been employed alongside the main $N/2$)-point DCT as shown in Fig. 5.

In order to fairly compare the proposed architectures, we need to inspect them from different perspectives. The first item is the hardware cost which is assessed according to the gate count. The gate count is normalized according to the equivalent cell area of the smallest 2-input NAND gate in library. The second item is the throughput of architectures which represents the number of processing pixels per second. The final item the hardware efficiency which provides a trade-off between hardware cost and throughput. The hardware efficiency is defined as the ratio of throughput to the gate count or can be expressed as the normalized area which is defined according to Eq. (9), where T_{ave} and PR_{ave} are the average throughput and processing rate of architecture over different size of transform sizes.

$$Normalized\ Area = \frac{Gate}{T_{ave}} = \frac{Gate}{Freq \times PR_{ave}} \qquad (9)$$

In the first experiment, Table 1 reports the number of arithmetic units in terms of adder, shifter, and multiplexers included in the proposed architectures for different transform sizes. It is worth noting that the shift blocks do not incur any hardware cost in terms of gate count, but it may incur additional overhead due to routing process. It is concluded form Table I that the proposed architecture #1 requires the lowest number of arithmetic units compared with architectures #2 and #3. Another In the first experiment, Table 1 reports the number of arithmetic units in terms of adder, shifter, and multiplexers included in the proposed architectures for different transform sizes. It is worth noting that the shift blocks do not incur any hardware cost in terms of gate count, but it may incur additional overhead due to routing process. It is concluded form Table I that the proposed architecture #1 requires the lowest number of arithmetic units compared with architectures #2 and #3. Another interesting finding from Table 1 is that architecture #3 incurs less number of arithmetic units in comparison to architecture #2 for all the transform sizes excluding 32-point DCT. In fact, the multi-sized 32-point DCT architecture #2 offers the lower hardware cost compared to architecture #3 in terms of adders and multiplexers. This is mainly because of the fact that as the number of constants in the configurable multiplier increases, the likelihood of increasing in the hardware reusability in the arithmetic units increases as well. This increase in hardware reusability eventually reduces the hardware cost.

Table 1. Number of arithmetic circuits required for computing different sizes of 1D-DCT blocks

Design	SAU #1			SAU #2			SAU #3		
	add	shift	mux	add	shift	mux	add	shift	mux
DCT-4	8	5	5	15	20	8	10	8	4
DCT-8	17	12	12	42	40	28	37	35	22
DCT-16	39	22	27	105	120	92	101	93	79
DCT-32	75	34	53	200	312	268	234	205	283

Table 2. Number of arithmetic units of different 32-point 1D-DCTs

Design	Multiplier	Add	Shift	Mux
Partial Butterfly [30]	340	372	2	0
WHT-Based [2], [16]	147	307	8	0
CSD-CSE Based [14]	0	764	306	0
RAG-n Based [2]	0	584	503	0
MCM-Based [13]	0	682	278	0
Proposed #1	0	75	34	53
Proposed #2	0	200	312	268
Proposed #3	0	234	205	283

Table 2 lists the number of arithmetic units in the proposed multi-sized 32-point 1D-DCT architectures #1, #2, and #3 and that of the other architectures. None of the proposed architectures employ multiplier blocks and in turn the fixed point multipliers are replaced with the simple add and shift blocks. It can be seen from Table II that the proposed architecture #1 achieves the lowest hardware cost in terms of number of arithmetic units, amongst the existing DCT architectures, while all the proposed architectures surpasses the existing multi-sized designs in terms of hardware cost. Unlike the existing DCT architectures, the proposed architectures employ multiplexer units. Nevertheless, our analysis on 90-nm technology proved that the cell area consumed by an adder cell is about ≈3.9 larger than the equivalent multiplexer circuit with the same bit width.

5.2. Results of 1D-DCT Architectures

Table 3 provides a detailed comparison between the proposed 1D-DCT architectures and that of the existing 1D-DCT architectures [2], [13], [15], [28] in terms of gate count, processing rate (PR), throughput (T), area-delay-product, and normalized area. According to Table III, the proposed DCT architecture #1 yields only 12 K in gate count under 350 MHz operating frequency. Although this value represents the lowest hardware cost amongst the existing architectures, the throughput of architecture #1 is very low. This implies that the proposed architecture #1 is a suitable candidate only for the resource-constraint devices that do not need very high performance, and suitable in the cases that there is no need to transfer or store high resolution video contents. The second places in terms of gate count goes to architecture #2 with 31 K and right behind that architecture #3 with 37 K. It is worth pointing out that the maximum operating frequency of architectures #1 and #2 is set to 350 MHz, while architecture #3 can operate in higher frequency of 370 MHz. Furthermore, all the proposed architectures #1, #2, and #3 outperforms the existing 1D-DCT architectures [2], [13], [15], [28] in terms of hardware cost.

The proposed architecture #1 requires 16, 16, 64, and 256 clock cycles to process 4-point, 8-point, 16-point, and 32-point DCTs, respectively. Therefore, the PR of architecture #1 in terms of processing pixel per cycle is 0.25, 0.5, 0.25, and 0.125 for 4, 8, 16, and 32-point DCTs, respectively. The average throughput for architecture #1 can be obtained as a weighted sum of throughput of different transform sizes, where the weight associated to each size is the usage statistics of each N-point DCT presented by Ref. [28]. Therefore, the throughput of architecture #1 is obtained as 103-megabyte sample per second (Msps) in 350 MHz.

Table 3. Comparison of different 1D-DCT architectures

Design	Tech	Gate Count	Freq.	PR=Pixels/Cycle	ADP	T (Gsps)	T/Gates (Gsps)	Normalized Area
Goebel [15]	45-nm	97.3 K	50^2 MHz	32^3	1.95	1.6	16.44	60.81
Masera [2]	90-nm	163 K	250 MHz	12.8/12.8/13.4/14.2	0.65	3.212	19.63	50.78
Zhao [28]	45-nm	205 K	333 MHz	1.2/2.8/6.2/13.6≈1.9	0.62	0.634	3.09	324.01
Meher [13]	90-nm	131 K	187 MHz	16	0.70	2.99	22.82	43.78
Proposed #1	90-nm	12 K	350 MHz	0.25/0.5/0.25/0.125≈0.295	0.03	0.103	8.58	116.2
Proposed #2	90-nm	31 K	350 MHz	2	0.88	0.700	22.58	41.89
Proposed #3	90-nm	37 K	370 MHz	2	0.10	0.740	20.00	50.00
Reusable Architecture	90-nm	53 K	370 MHz	4/4/2/1≈ 3.87	0.14	1.430	27.00	37.00

^{1} Only for 32 and 16-point DCTs
^{2} Target frequency is specified for throughput of 1.6 Gsps
^{3} PR was calculated for full-parallel 2-D, for folded structure PR is 16

Table 4. Comparison of different 2D-DCT architectures

Design		Tech	Gate Count	Freq	Pixels/Cycle	ADP	T (Gsps)	Supporting Video format
Masera [2]	Folded	90-nm	157 K	250 MHz	5.2	0.628	1.302	4096×2048@ 60 fps
	Unfolded	90-nm	243 K	250 MHz	12.8/12.8/13.4/14.2	0.97	3.212	7680×4320@ 60 fps
Kalali [29]	LU-Unfolded	90-nm	175 K	140 MHz	4/8/16/32 ≈5.64	1.21	0.79	3840×2160@ 60 fps
	HU-Unfolded	90-nm	197 K	130 MHz	16/16/16/32≈16.11	1.51	2.09	7680×4320@ 30 fps
Meher [13]	Folded	90-nm	208 K	187 MHz	16	1.11	2.99	7680×4320@ 60 fps
	Unfolded	90-nm	347 K	187 MHz	32	1.86	5.98	7680×4320@ 60 fps
Masera [17]	Folded	90-nm	165 K	187 MHz	16	0.88	2.99	7680×4320@ 60 fps
	Unfolded	90-nm	253 K	187 MHz	32	1.35	5.98	7680×4320@ 60 fps
Proposed #1	Unfolded	90-nm	99 K	350 MHz	0.25/0.5/0.25/0.125≈0.295	0.28	0.103	1920×1080@ 30 fps
Proposed #2	Unfolded	90-nm	116 K	350 MHz	2	0.32	0.700	3840×2160@ 56 fps
Proposed #3	Unfolded	90-nm	124 K	370 MHz	2	0.34	0.740	3840×2160@ 60 fps
Proposed Reusable	Folded	90-nm	104 K	370 MHz	4/4/2/1≈3.87	0.28	1.430	4096×2048@ 60 fps
	Unfolded	90-nm	156 K	370 MHz	8/8/4/2≈7.74	0.42	2.860	7680×4320@ 56 fps

The PR of architectures #2 and #3 is two pixels per cycle which is independent of transform sizes. This will result in a throughput of 700 and 740 Msps in 350 MHz and 370 MHz frequency for architecture #2 and #3, respectively. As a result, the throughput of architectures #2 and #3 is about 6.8-fold and 7.2-fold higher than architecture #1, respectively, while architecture #1 archives the remarkable reduction in gate count around 60% and 67% compared to proposed architectures #2 and #3, respectively.

The reusable architecture based on 1D-DCT architecture #3 can process eight 4-point, four 8-point, two 16-point, and one 32-point DCTs in 16, 32, 128, and 512 clock cycles, respectively. Therefore, the PR of the reusable architecture is 4, 4, 2, and 1 for 4, 8, 16, and 32-point DCTs, respectively. The weighted sum of the processing rate for the reusable architecture is approximately 3.87 pixels per cycle, which yields a throughput of 1430 Msps in 370 MHz frequency. The reusable architecture achieves the highest throughput in comparison to the rest of the proposed architectures (i.e., 13.8-fold, 2-fold, and 1.9-fold increase compared to architectures #1, #2, and #3), at the expense of increase in gate count around 30% compared to the base architecture #3 due to an additional $(N/2)$-point 1D-DCT. Moreover, the comparison of hardware efficiency in terms of throughput/gate and normalized area from Table 3 reveals that the reusable architecture and after that architecture #2 offer the best hardware efficiency compared to the rest of the proposed architectures.

In respect to the existing architectures [2], [13], [15], [28], the proposed reusable architecture yields a favorable reduction in gate count and ADP around 61% and 82% respectively, and a significant increase in hardware efficiency about 40%. Therefore, the resurface architecture can be considered as a suitable candidate for resource-constraint and higher performance application of HEVC. Also, the proposed architectures #2 and #3 achieve a remarkable reduction in gate count around 78% and 74%, on average, compared to architectures of [2], [13], respectively, with relatively the same hardware efficiency.

5.3. Results of 2D-DCT Architectures

We implemented the proposed folded and unfolded 2D-DCT architectures. Table 4 reports a detailed comparison between the proposed 2D-DCT architect-ures and that of the other architectures [2], [13], [17], [29]. We select the unfolded structure for the architectures #1, #2, and #3 since they offer less throughput than the reusable architectures and their folded architectures decreases the throughput accordingly. However, the reusable architecture is implemented in two different structure of folded and unfolded 2D-DCTs. In respect to the unfolded architectures, 2D-DCT architectures #1 provides the ultra-low number of gate count, but with a remarkable decrease in throughput. The unfolded 2D-DCT architecture #1 with throughput of 103 Msps can only process 1080p (1920×1080) at 30 fps of high-definition video modes. The unfolded architectures #2 and #3 offer the moderate amount of hardware cost and throughput compared to the rest of proposed architectures. They archive 116 K and 124 K gate count, respectively, and both of them can process 4K (3840×2160) video format at almost 60 fps. The reusable unfolded 2D-DCT architecture, however, provides the highest throughput of 2860 Msps amongst the proposed architectures #1, #2, and #3, at the expense of increase in gate count around 57% 34.5%, and 26%, respectively. It can process UHD 8K (7680×4320) at 56 fps. Moreover, the folded 2D-DCT reusable architecture yields the half of the throughput of unfolded architectures, but with a favorable 30% decrease in gate, and it can process 4K video content at 60 fps. In respect to the existing folded 2D-DCT architectures [2], [13], [17], the proposed folded and reusable 2D-DCT architecture archives a considerable reduction in gate count around 40%, on average, with relatively the same throughput of [2]. This achievement for the proposed unfolded and reusable architecture is around 32% reduction in gate count, on average, compared to the existing unfolded 2D-DCT architectures [2], [13], [17], [29]. Moreover, the unfolded architectures #1, #2, and #3 outperform the existing unfolded architectures [2], [13], [17], [29] by reducing the gate count around 57%, 49% and 46%, on average, respectively.

6. CONCLUSION

In this paper, the new low-cost and multi-sized DCT architectures for HEVC application was proposed. Three different shift-and-add units are developed by using different configurations of configurable constant multiplier to trade-off between hardware cost and hard-ware efficiency. Moreover, the folded and unfolded 2D-DCT architectures based on 1D-DCT architectures #1, #2, #3 were presented. The following conclusion can be drawn from the experimental results on 90-nm technology node: 1) The proposed unfolded 2D-DCT architectures #1 archives the best hardware cost in terms of gate count amongst the proposed architectures and that of the other architectures by a favorable reduction around 57%, but with a lower throughput (Table IV). 2) the proposed unfolded 2D-DCT architectures #2 and #3 offer the moderate reduction in gate count around 47%, on average with moderate throughput compared to the existing architectures (Table IV). 3) The proposed reusable folded and unfolded architectures yield the highest throughput amongst the proposed 2D-DCT architectures and highest efficiency amongst the existing architectures, with 40% reduction in gate count (Table IV). 4) The proposed unfolded 2D-DCT architectures #1, #2, #3, and the reusable architecture can process 2K at 30 fps, 4K at 56 fps, 4K at 60 fps, and 8K 56 fps, respectively. It is worth to note that the proposed architectures do not involve inverse DCT architecture and cannot be directly applied to the other existing coding standards. Accordingly, these issues will be addressed in the future work of this paper

REFERENCES

[1] K. Rao, D. Kim, and J. Hwang, “Video coding standards”, *The Netherlands: Springer*, pp. 51–97, 2014.

[2] M. Masera, M. Martina, and G. Masera, “Adaptive approximated DCT architectures for HEVC”, *IEEE Trans. Circuits Syst. Video Tech.*, vol. 27, no. 12, pp. 2714-25, 2017.

[3] A. Shabani, S. Timarchi, and H. Mahdavi, “Power and area efficient CORDIC-Based DCT using direct realization of decomposed matrix”, *Microelectron. J.*, vol. 91, pp. 11-21, 2019.

[4] A. Seyed et al., “Analysis and synthesis of facial expressions by feature-points tracking and deformable model”, 2007.

[5] A. Abadpour and S. Kasaei, “A novel color image compression method using eigenimages”, *J. Iran. Assoc. Electr. Electron. Eng.*, vol. 5, no. 2, pp. 49-57, 2008.

[6] Z. Mahdavipour, “Image de-noising and micro crack detection of solar cells”, *J. Iran. Assoc. Electr. Electron. Eng.*, vol. 14, no. 4, pp. 55-61, 2018.

[7] A. Shabani and S. Timarchi, “Low-power DCT-based compressor for wireless capsule endoscopy”, *Signal Proc.: Image Commun.*, vol. 59, pp. 83-95, 2017.

[8] M. Saidi et al., “A new adaptive image steganography scheme based on DCT and chaotic map”, *Multimedia Tools Appl.*, vol. 76, no. 11, pp. 13493-510, 2017.

[9] M. Budagavi et al., “Core transform design in the high efficiency video coding (HEVC) Standard”, *IEEE J. Selected Top. Signal Proc.*, vol. 7, pp. 1029-41, 2013.

[10] K. . McCann et al., “High efficiency video coding (HEVC) test model 8 (HM 8) encoder description”, *JCT-VC Documents* , Sweden, 2012.

[11] M. Budagavi and V. Sze, “Unified forward+inverse transform architecture for HEVC”, *Proc. Int. Conf. Image Proc.*, 2012.

[12] S. Park and P. Meher, “Flexible integer DCT architectures for HEVC”, P*roc. IEEE Int. Symp. Circuits Syst.*, 2013.

[13] P. Meher et al., “Efficient integer DCT architectures for HEVC”, *IEEE Trans. Circuits Syst. Video Tech.*, vol. 24, no. 1, pp. 168-78, 2014.

[14] A. Darji and R. Makwana, “High-performance multiplierless DCT architecture for HEVC”, *19th Int. Symp. VLSI Des. Test,* 2015.

[15] J. Goebel et al., “An HEVC multi-size DCT hardware with constant throughput and supporting heterogeneous CUs”, *Proc. IEEE Int. Symp. Circuits Syst.*, 2016.

[16] A. Ahmed, M. Shahid, and A. Rehman, “N point DCT VLSI architecture for emerging HEVC standard”, *VLSI Des.*, 2012.

[17] M. Masera, G. Masera, and M. Martina, “An area-efficient variable-size fixed-point DCT architecture for HEVC encoding”, *IEEE Trans. Circuits Syst. Video Tech.*, vol. 30, no. 1, pp. 232-42, 2020.

[18] S. Shen et al., “A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards”, *Proc. IEEE Int. Conf. Multimedia Expo.*, 2012.

[19] H. Sun et al., “A low-cost VLSI architecture of multiple-Size IDCT for H.265/HEVC”, *IEICE Trans. Fundamentals Electron. Commun. Comput. Sci.*, vol. E97A, no. 12, pp. 2467-76, 2014.

[20] S. Chatterjee and K. Sarawadekar, “WHT and matrix decomposition-based approximated IDCT architecture for HEVC”, *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 66, no. 6, pp. 1043-47, 2019.

[21] S. Chatterjee and K. Sarawadekar, “An optimized architecture of HEVC core transform using real-valued DCT coefficients”, *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 65, no. 12, pp. 2052-6, 2018.

[22] W. Pratt, J. Kane, and H. Andrews, “Hadamard transform image coding”, *Proc. IEEE*, vol. 57, pp. 58-68, 1969.

[23] S. Chatterjee and K. Sarawadekar, “Approximated core transform architectures for HEVC using WHT-based decomposition method”, *IEEE Trans. Circuits Syst. I: Reg. Pap.*, vol. 66, no. 11, pp. 4296-308, 2019.

[24] S. Fralick, “A fast computational algorithm for the discrete cosine transform”, *IEEE Trans. Commun.*, vol. 25, no. 9, pp. 1004-9, 1977.

[25] Y. Voronenko and M. Püschel, “Multiplierless multiple constant multiplication”, *ACM Trans. Algorithms*, vol. 3, no. 2, 2007.

[26] A. Hassanzadeh and A. Shabani, “Low power parallel prefix adder design using two phase adiabatic logic”, *J. Electr. Electron. Eng.*, vol. 3, no. 6, p. 181, 2015.

[27] P. Tummeltshammer, J. Hoe, and M. Puschel, “Time-multiplexed multiple-constant multiplication”, *IEEE Trans. Comp. Aided Des. Integrated Circuits Syst.*, vol. 26, no. 9, pp. 1551-63, 2007.

[28] W. Zhao, T. Onoye, and T. Song, “High-performance multiplierless transform architecture for HEVC”, *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 1668-71, 2013.

[29] E. Kalali, A. Mert, and I. Hamzaoglu, “A computation and energy reduction technique for HEVC Discrete Cosine Transform”, *IEEE Trans. Consumer Electron.*, vol. 62, no. 2, pp. 166-74, 2016.

[30] F. Bossen and K. Sühring, “Joint collaborative team on video coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11”, 2015.