An Area-Efficient Variable-Size Fixed-Point DCT Architecture for HEVC Encoding

Maurizio Masera, Guido Masera, *Senior Member, IEEE,* and Maurizio Martina, *Senior Member, IEEE*

***Abstract*—This paper proposes an area-efficient fixed-point architecture for the computation of the Discrete Cosine Transform (DCT) of multiple sizes in HEVC. This result is obtained by comparing different DCT factorizations in order to find the most suitable one for implementation in the HEVC encoder. The recursive structure of fast algorithms, which decompose the N-point DCT by means of two $N/2$-point DCTs, is exploited to execute computations of small-size DCTs in parallel, thus maximizing the hardware re-usability while maintaining a constant throughput. Simulation results prove that the proposed solution features reduced rate-distortion losses, with relevant complexity saving compared with state-of-the-art implementations. Finally, the proposed architecture is exploited to design two families of architectures for the 2D-DCT, namely *Folded* and *Full-parallel*.**

***Index Terms*—Discrete Cosine Transform, DCT, VLSI Architecture, High Efficiency Video Coding.**

I. Introduction

THE latest High Efficiency Video Coding (HEVC) standard [1] aims to double the coding efficiency with respect to the previous Advanced Video Coding standard (AVC) [2]. One of the features introduced by the new standard to achieve such a result, is the ability to compute transforms on squared blocks of multiple sizes. Discrete Cosine Transforms (DCTs) of size from 4×4 to 32×32 and the 4×4 Discrete Sine Transform (DST) are specified as core transforms in the HEVC standard [3]. The use of multiple DCT sizes improves the video coding performance at the expense of increasing the computational complexity. For this reason, some hardware architectures have been proposed in the literature to accelerate the DCT computation for HEVC. However, despite being important, it is not easy to estimate the throughput requirements of the different subblocks inside the HEVC encoder. Some works in the literature [4], [5] addressed the problem, showing that HEVC encoding complexity is strictly related to the use of certain coding tools and, as a consequence, to the application. In particular, [5] shows that the number of DCTs performed by the encoder, and so the required throughput, changes dramatically when enabling or disabling certain tools. For this reason most of the works proposed in the literature, describing architectures for the DCT, rely on an application-independent approach.

It is worth noting that the HEVC standard defines the DCT used at the decoder side, which is an integer approximation of the DCT and is obtained through the optimization process discussed in [3]. As a consequence, different solutions, trading rate-distortion performance for complexity, can be explored at the encoder side, leading to two possible scenarios: i) fixed-point implementations of DCT factorizations, ii) integer approximations of the DCT. In the first scenario, where fixed-point implementations of DCT factorizations are considered, e.g. [6], [7], rate-distortion performance degradation due to quantization needs to be carefully studied. On the other hand, in the second scenario there are two possible cases, which depend on the selected approximation. If the integer DCT approximation at the encoder side is the same one imposed at the decoder side by the standard, as in [8]–[11], then there is no need to investigate the rate-distortion performance, as it is already known [1]. On the contrary, other integer DCT approximations, such as [12]–[16], can lead to relevant complexity reduction at the expense of significant coding loss, especially at high bit-rates [7], [15].

In the following, the one-dimensional N-point DCT and the two-dimensional DCT over an $N \times N$ block are referred to as 1D-DCT and 2D-DCT respectively. The solutions implemented in [6], [7] for the 1D-DCT are fixed-point architectures, which rely on the work of Chen *et al.* [17], namely factorizing the 1D-DCT as the cascade of the Walsh-Hadamard Transform (WHT) and a set of Givens rotations. On the other hand, [8]–[11], [18] deal with the integer DCT approximation defined in the HEVC standard. Budagavi *et al.* [8] propose a unified forward and inverse transform unit by exploiting the symmetry within the transform defined in the HEVC standard. Meher *et al.* [9] present an efficient integer DCT architecture, where the 1D-DCT core implements the partial-butterfly factorization suggested in [3] by means of one $N/2$-point DCT and one $N/2 \times N/2$ matrix multiplication resorting to the Multiple Constant Multiplication (MCM) technique. Zhao *et al.* [10] reduced the hardware cost of the design by exploiting add-and-shift operations to represent the integer DCT coefficients. In [11] Dias *et al.* exploit a systolic-array-based design to propose a multi-standard architecture, which supports both H.264/AVC and HEVC integer DCTs. Goebel *et al.* [18] have recently proposed an HEVC multi-size DCT architecture, which is able to support heterogeneous partitioning of CUs, thanks to its ability to compute DCTs of different length in parallel. Unlike the aforementioned works, several low-complexity approximate DCTs are shown and compared in [12]. A similar approach is developed in [14]–[16], where a generalized algorithm and efficient architectures are derived by recursive decomposition of sparse DCT approximations.

As observed in most of previous works, including [9], [11], the separability property permits to implement the 2D-DCT by applying 1D-DCTs first on the rows, then on the columns of the input block of samples. As a consequence, a transposition structure is required to feed the 1D-DCT architecture either

The authors are with the Electronics and Telecommunications Department - Politecnico di Torino, 10129 Torino, Italy (e-mail: maurizio.masera@polito.it; maurizio.martina@polito.it; guido.masera@polito.it).

row-wise or column-wise. In [11] this operation is implemented by the means of a transposition switch, which is made of multiplexers with no need for additional memory resources. On the contrary, in [9] transposition relies on a buffer, which contains $N \times N$ registers and N multiplexers to select the data either row-wise or column-wise. Then, two architectures are derived, where the first one, referred to as *Folded*, relies on one 1D-DCT and a transposition buffer; the second one, referred to as *Full-parallel*, exploits two 1D-DCTs and a transposition buffer to achieve high throughput. Since the buffer requires to be filled with the intermediate results, it introduces a latency of N clock cycles. Moreover, as the number of pixels in each block varies from 4 to 32, the solutions presented in [9], [18] suggest to transform 32 pixels per clock cycles, which means to process up to 32/N blocks of N pixels concurrently. Such solutions lead to architectures which are able to sustain a constant throughput, independently of N.

From the analysis of previous works proposed in the literature, it is clear that designing an architecture able to support all the DCT sizes specified in the HEVC standard, leads to some area overhead. In particular, [9] shows that thanks to the even-odd decomposition about 50% the architecture of the N-point 1D-DCT, can be exploited to implement one $N/2$-point 1D-DCT, with low area overhead. As a consequence, being able to increase to more than 50% the reuse of the N-point 1D-DCT for small size ones is important, especially in fixed throughput architectures. This aspect motivated us to analyze alternative approaches to implement the DCT, such as resorting to DCT factorizations. Stemming from [9], this current work describes two families of 2D-DCT architectures (both pipelined and not-pipelined), each of which includes a transposition buffer and one or two 1D-DCT modules. It also aims to analyze and evaluate the complexity and the rate-distortion performance degradation of fixed-point implementations of DCT factorizations. Indeed, several factorizations proposed in the literature, such as [17], [19]–[27] can be considered. However, despite these factorizations have been partially compared for fixed N values in previous works, such as [28], to the best of our knowledge there is no results in the open literature showing: i) the amount of resource reusability featured by different factorizations when variable-size DCT is required, as in HEVC; ii) if some factorizations are competitive with state-of-the-art solutions from the hardware complexity point of view. This motivated us to investigate these directions discovering that, if variable-size is taken into account when choosing the factorization, then the obtained architecture features a large complexity reduction with respect to state-of-the-art solutions. Based on these observations, this work aims to provide the following novel contributions: i) to analyze different 1D-DCT factorizations from the literature and to identify for each one both the amount of required resources and the degree of re-usability to support variable-size transforms, as required for HEVC; ii) to choose the factorization which minimizes the amount of hardware resources while maximizing the re-usability, which has not been previously shown in the open literature; iii) to show the rate-distortion performance degradation in HEVC of the selected factorization when fixed-point implementation is used. In particular, the selected factorization with fixed point implementation features in the worst case an overall Bjøntegaard Delta rate loss lower than the 2%; iv) to present a novel fixed-point 1D-DCT architecture of the selected factorization to support variable-size transform and to study the effect of pipelining on area and maximum sustained throughput. The proposed architecture is significantly smaller than other architectures described in the literature and it achieves lower power and energy consumption.

The paper is organized as follows. Different alternatives to design a variable-size architecture (including DCT factorizations and DCT integer approximations) are presented and compared in Section II, where the characteristics of the selected factorization (Lee's factorization [21]) are briefly summarized. Section III illustrates the proposed architecture, which is able to support all the DCT sizes specified in the HEVC standard with high resource sharing. Finally, implementation results are presented in Section IV, while Section V concludes the paper.

II. DCT Algorithms for HEVC

Since the proposed architecture has to support multiple transform sizes (i.e. N = 4, 8, 16, 32) and aims to minimize the hardware cost by exploiting resource sharing, a comparison of both exact factorizations and integer approximations proposed in literature for the 1D-DCT is briefly reviewed.

A. Exact DCT Factorizations

According to [29], the 1D-DCT can be computed as:

$$\mathbf{X} = \mathbf{C}_N^{II} \cdot \mathbf{x}, \tag{1}$$

where $\mathbf{x} = \{x_0, \ldots, x_{N-1}\}$ and $\mathbf{X} = \{X_0, \ldots, X_{N-1}\}$ are the column vectors of input samples and DCT coefficients respectively, and $\mathbf{C}_N^{II}$ is the N-order type-II DCT matrix, which coefficients are:

$$(\mathbf{C}_N^{II})_{l,k} = \sigma_k \cos\left[\left(l + \frac{1}{2}\right)\frac{k\pi}{N}\right] \quad k, l = 0, \ldots, N-1, \tag{2}$$

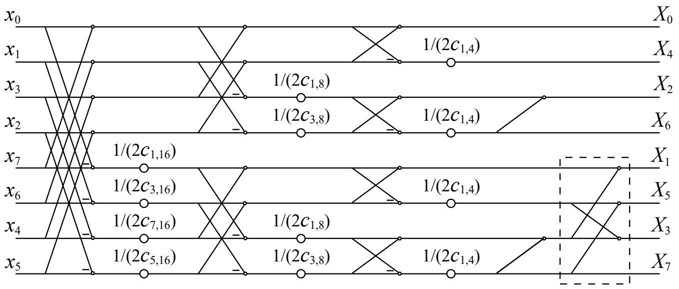
where $\sigma_k = 1/\sqrt{2}$ only for $k = 0$, $\sigma_k = 1$ otherwise.

In the past, several fast algorithms have been proposed for computing the DCT [17], [19]–[27].

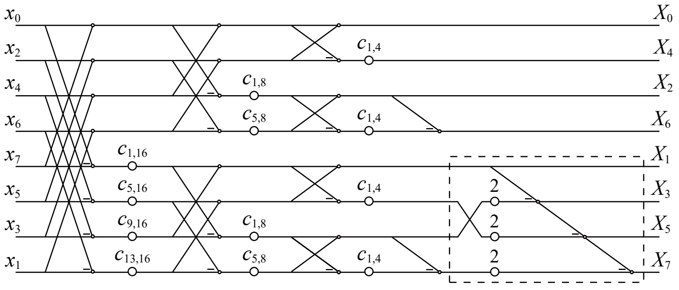
The WHT-based factorization was initially proposed in [17] for the case N = 8. Then, it has been extended and exploited in [6], [7] to reduce the 1D-DCT computation to F_N simple addition/subtraction butterflies and R_N Givens rotations (see the first row in Table I).

Loeffler [27] decomposed the 1D-DCT matrix by means of one $N/2$-point DCT and a non-regular $N/2 \times N/2$ rotation matrix, which cannot be shared with other DCT sizes. It is worth noting that Loeffler's factorization achieves the theoretical lower bound on the number of required multiplications for N = 8. The extension of Loeffler's factorization to larger N values is described in [30].

On the other hand, the $\mathbf{C}_N^{II}$ matrix can be recursively factorized by splitting the computation into one type-II DCT matrix of size $N/2$ ($\mathbf{C}_{\frac{N}{2}}^{II}$) and one type-IV DCT matrix of size $N/2$ ($\mathbf{C}_{\frac{N}{2}}^{IV}$) [29]. Some works in the literature implement $\mathbf{C}_{\frac{N}{2}}^{IV}$ by the means of sparse matrices [19], [20]. Other works

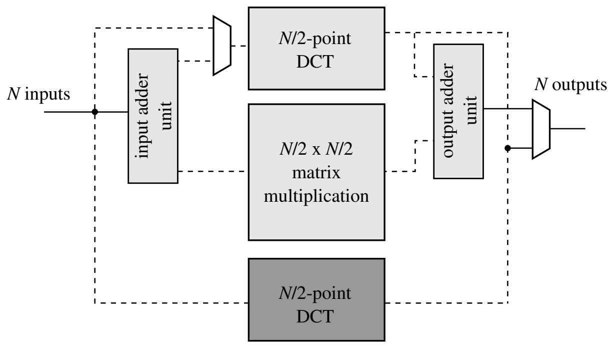


(a) Lee 1D-DCT [21].

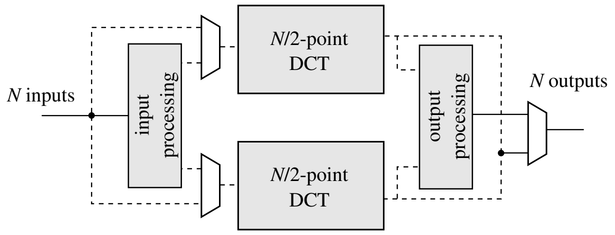


(b) Lee and Huang 1D-DCT [26].

Fig. 1. Flowgraphs of 1D-DCT for $N = 8$ of factorizations in [21] and [26].



(a) Recursive scheme in [9].



(b) Proposed recursive scheme.

Fig. 2. 1D-DCT block schemes. Continuous lines represent N-samples data-flow and dashed lines $N/2$-samples data-flow.

rewrite $\mathbf{C}^{IV}_{\frac{N}{2}}$ as a function of $\mathbf{C}^{II}_{\frac{N}{2}}$ [21]–[26] to obtain solutions based on $\mathbf{C}^{II}_{\frac{N}{2}}$ only. The formulas to compute the number of additions and multiplications as a function of N, referred to as A_N and M_N, for [6], [7], [19]–[26] are given in Table I, whereas for [27] no closed-form is available [30].

Although factorizations in [21]–[26] feature equal computational complexity, they adopt different flowgraphs and coefficient values. As an example, Fig. 1 shows the data flow, for the case $N = 8$, of two representative factorizations, namely [21] and [26]. As it can be observed, the additions and the multiplications in the initial stage are performed concurrently in both factorizations. The coefficients in the multiplications are inverse cosine $1/(2 \cdot c_{i,j})$ values in [21] and cosine $c_{i,j} = \cos\frac{i\pi}{j}$ values in [26]. Moreover, as highlighted by the dashed boxes in the right-most part of Fig. 1 (a) and (b), the final stage is different. The data flow in [21] performs concurrent computation of the last $N/2-1$ additions, whereas cascaded additions are used in [26]. As a consequence, the two data flows have the same critical path except for the final stage. Indeed, the delay of the last stage is one adder for [21] and $N/2-1$ adders for [26]. For this reason, the factorization in [21] provides the shortest critical path among [21]–[26], being the most suitable one (in the group of $\mathbf{C}^{II}_{\frac{N}{2}}$ only factorizations) for high throughput hardware implementation.

B. Integer DCT Approximations

These approximations aim to reduce the complexity by placing integer coefficients into the DCT matrix. Depending on the choice of the coefficients, several different rate-distortion/complexity trade-offs can be achieved.

The DCT used in HEVC is an integer DCT approximation [3], which has been designed to achieve several properties, including almost orthogonal basis vectors with almost equal norm, symmetry and the possibility to embed small transform matrices in large ones.

Other approaches can be used to design low-complexity approximations of the DCT for HEVC as well. The works in [14], [15] present a general recursive algorithm to derive an approximate N-point DCT from a pair of $N/2$-point DCTs with an additional cost of $N/2$ butterflies. As shown in Table II, such solutions feature very low complexity compared to the other ones. However, as argued in [15], these works give coding loss between 2% and 30% in terms of bit-rate, when used for video compression. Another integer cosine transform has been proposed in [16], to provide the same coding performance of HEVC while reducing the complexity. Formulas to compute A_N and M_N for [3], [14]–[16] are given in Table I.

C. Variable-size Analysis

As highlighted in Section I, the complexity of a multiple-size DCT architecture depends on both the complexity of the employed factorization and the possibility to share resources for different values of N. Even if some factorizations, such as [27], exhibit low complexity when each N value is taken as a stand-alone point in the design space (see Table II), they suffer from some complexity overhead when resource sharing is required to support variable-size transforms. As stated in Section I, this current work exploits a fixed throughput solution, where the 1D-DCT architecture processes 32/N data blocks concurrently, i.e. one 1D-DCT32, two 1D-DCT16, four 1D-DCT8 or eight 1D-DCT4. As a consequence, resources allocated for large N values must be reused for small N values. However, all the factorizations which employ one $N/2$-point DCT and a custom matrix multiplication, such as [3], [16], [19], [20], [27], are not suited for variable-size

TABLE I
FORMULAS TO COMPUTE THE NUMBER OF ADDITIONS (A_N) AND MULTIPLICATIONS (M_N) OF DCTS ALGORITHMS

[6], [7]	$A_N = 2 \cdot F_N + 3 \cdot R_N$ $M_N = 3 \cdot R_N$	$F_N = 2 \cdot F_{\frac{N}{2}} + \frac{N}{2}$ $R_N = 2 \cdot R_{\frac{N}{2}} + N \cdot \sum_{i=2}^{\log_2 N} \frac{1}{2^i}$			$F_4 = 2$ $R_4 = 1$
[19]	$A_N = A_{\frac{N}{2}} + \frac{3N}{4} \log_2 N$ $M_N = M_{\frac{N}{2}} + \frac{N}{2} \log_2 N - \frac{N}{4}$	$A_4 = 8$ $M_4 = 6$	[3]	$A_N = A_{\frac{N}{2}} + \frac{N}{2} + \frac{N^2}{4}$ $M_N = M_{\frac{N}{2}} + \frac{N^2}{4}$	$A_4 = 8$ $M_4 = 4$
[20]	$A_N = A_{\frac{N}{2}} + \frac{N}{8} \cdot (7 \cdot \log_2 \frac{N}{2} - 2) + N$ $M_N = M_{\frac{N}{2}} + \frac{N}{8} \cdot (3 \cdot \log_2 \frac{N}{2} + 2)$	$A_4 = 9$ $M_4 = 5$	[14] [15]	$A_N = 2 \cdot A_{\frac{N}{2}} + N,\ M_N = 0$	$A_8 = 22$ $A_4 = 8$
[21]–[26]	$A_N = 2 \cdot A_{\frac{N}{2}} + N + \frac{N}{2} - 1$ $M_N = 2 \cdot M_{\frac{N}{2}} + \frac{N}{2}$	$A_4 = 9$ $M_4 = 4$	[16]	$A_N = A_{\frac{N}{2}} + \frac{3N}{2} + N \cdot \log_2 \frac{N}{4}$ $M_N = M_{\frac{N}{2}} + N \cdot \log_2 \frac{N}{2}$	$A_4 = 8$ $M_4 = 6$

TABLE II
COMPUTATIONAL COMPLEXITY COMPARISON AMONG DIFFERENT DCT ALGORITHMS

N	Factorizations										Integer Approximations									
	WHT based		$\mathbf{C}^{IV}_{\frac{N}{2}}$ as sparse matrices				$\mathbf{C}^{II}_{\frac{N}{2}}$ only		Loeffler		HEVC		sparse matrices based					
	[6], [7]		[19]		[20]		[21]–[26]		[27]		[3]		[14]		[15]		[16]	
	M_N	A_N	M_N	A_N	M_N	A_N	M_N	A_N	M_N	A_N	M_N	A_N	M_N	A_N	M_N	A_N	M_N	A_N
4	3	11	6	8	5	9	4	9	3	9	4	8	-	-	0	8	6	8
8	15	39	16	26	13	29	12	29	11	29	20	28	0	22	0	24	22	28
16	51	115	44	74	35	83	32	81	31	81	84	100	0	60	0	64	70	84
32	147	307	116	194	91	219	80	209	79	209	340	372	0	152	0	160	198	228
All	147	307	216	352	172	396	80	209	144	384	480	560	0	152	0	160	336	400

support. Indeed, in [9], variable-size support is achieved by adding a further $N/2$-point DCT, shown in dark-shaded gray in Fig. 2 (a). Therefore, the total amount of additions and multiplications needed by these algorithms is:

$$A_{All} = A_{32} + A_{16} + 2 \cdot A_8 + 4 \cdot A_4 \tag{3}$$

$$M_{All} = M_{32} + M_{16} + 2 \cdot M_8 + 4 \cdot M_4 \tag{4}$$

where the terms A_N and M_N are computed as in Table I.

On the other hand, all the factorizations which rely on two $N/2$-point DCTs (see Fig. 2 (b)), such as [6], [7], [14], [15], [21]–[26], achieve a higher degree of re-usability as they do not pay the overhead introduced by the custom matrix multiplication. In this case the resources needed for N = 32 are reused for the concurrent computation of small-size transforms. As a consequence, adders and multipliers achieve a 100% utilization, $A_{All} = A_{32}$ and $M_{All} = M_{32}$, where A_{32} and M_{32} for [6], [7], [14], [15], [21]–[26] are computed as in Table I.

The amount of additions and multiplications required by each algorithm to implement a variable-size DCT architecture for the HEVC encoder are shown in the last row of Table II, where they are labeled as *All*. As it can be observed, factorizations in [21]–[26] achieve the lowest complexity, being all good candidates for fixed-point implementations. However, as highlighted in Section II-A, [21] features the shortest critical path among [21]–[26]. Thus, in the following sections an architecture for Lee's factorization [21] is proposed and analyzed.

D. Lee's DCT Factorization

According to Lee's factorization [21] and neglecting σ_k in (2), the DCT matrix can be recursively decomposed as:

$$\mathbf{C}^{II}_N = \mathbf{P}_N \cdot \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & \\ & \mathbf{R}_{\frac{N}{2}} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{C}^{II}_{\frac{N}{2}} & \\ & \mathbf{C}^{II}_{\frac{N}{2}} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & \\ & \mathbf{D}_{\frac{N}{2}} \end{bmatrix} \cdot \mathbf{B}_N, \tag{5}$$

where the N-order butterfly matrix $\mathbf{B}_N$ is:

$$\mathbf{B}_N = \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & \bar{\mathbf{I}}_{\frac{N}{2}} \\ \bar{\mathbf{I}}_{\frac{N}{2}} & -\mathbf{I}_{\frac{N}{2}} \end{bmatrix} \tag{6}$$

and $\mathbf{I}$ and $\bar{\mathbf{I}}$ are the identity and the reverse identity matrices respectively. On the other hand, $\mathbf{D}_{\frac{N}{2}}$ is an anti-diagonal matrix which contains $N/2$ inverse cosine coefficients:

$$d_{k,N} = \frac{1}{2 \cdot \cos \frac{(2k+1)\pi}{2N}} \quad k = 0, \ldots, \frac{N}{2} - 1. \tag{7}$$

Besides, matrix $\mathbf{R}_{\frac{N}{2}}$ is an upper bidiagonal matrix, which performs data recombination and it can be written as:

$$\mathbf{R}_{\frac{N}{2}} = \begin{bmatrix} 1 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 1 \end{bmatrix}. \tag{8}$$

Finally, $\mathbf{P}_N$ is the alternating permutation matrix defined by Φ_N:

$$\Phi_N = \begin{pmatrix} k \\ \phi_N(k) \end{pmatrix} \quad k = 0, \ldots, N-1, \tag{9}$$

with

$$\phi_N(k) = \begin{cases} \lfloor \frac{k}{2} \rfloor & k \text{ even} \\ \lfloor \frac{k}{2} \rfloor + \frac{N}{2} & k \text{ odd} \end{cases}. \tag{10}$$

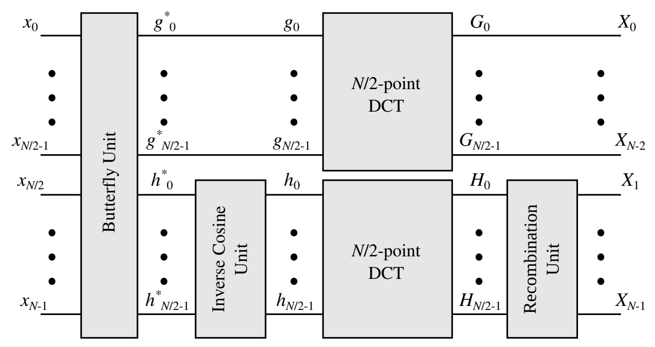


Fig. 3. Proposed recursive architecture of 1D-DCT for generalized N.

III. Hardware Architecture

This Section describes the proposed baseline 1D-DCT hardware architecture, derived from the Lee's factorization [21]. Then, architecture modifications to support variable-size DCT are explained and two schemes to implement the 2D-DCTs are presented as well. Finally, it is shown how pipelining can improve the maximum sustained throughput of the architectures.

A. 1D-DCT Architecture

The proposed recursive 1D-DCT architecture, shown in Fig. 3, corresponds to the implementation of Lee's factorization as in (5), where the N-point 1D-DCT computation is performed by the means of two $N/2$-point DCTs plus the Butterfly Unit, the Inverse Cosine Unit and the Recombination Unit, implementing $\mathbf{B}_N$, $\mathbf{D}_{\frac{N}{2}}$ and $\mathbf{R}_{\frac{N}{2}}$ respectively.

The Butterfly Unit implements the $N/2$ butterfly operators defined by $\mathbf{I}_{\frac{N}{2}}$ and $\bar{\mathbf{I}}_{\frac{N}{2}}$ in (6), which combine $\mathbf{x}$ values into $\mathbf{g}^*$ and $\mathbf{h}^*$ as:

$$\begin{array}{l} g^*_k = x_k + x_{N-k-1} \\ h^*_k = x_k - x_{N-k-1} \end{array} \qquad k = 0, \ldots, \frac{N}{2} - 1, \tag{11}$$

where x_k, g^*_k and h^*_k are the k-th elements of $\mathbf{x}$, $\mathbf{g}^*$ and $\mathbf{h}^*$, respectively. As it can be observed, $\mathbf{g}^*$ is directly connected to the input of the upper $N/2$-point DCT (i.e. $\mathbf{g} = \{g_0, \ldots, g_{N/2-1}\}$) and $\mathbf{h}^*$ is the input of the Inverse Cosine Unit. This block multiplies each h^*_k value with the proper inverse cosine coefficient to produce:

$$h_k = (\delta_{k,N} \cdot h^*_k) >> N_q \qquad k = 0, \ldots, \frac{N}{2} - 1, \tag{12}$$

where $\mathbf{h} = \{h_0, \ldots, h_{N/2-1}\}$ is the input of the lower $N/2$-point DCT,

$$\delta_{k,N} = \lfloor d_{k,N} \cdot 2^{N_q} \rceil \tag{13}$$

are the coefficients in (7) scaled up by N_q bits and rounded to the nearest integer and $>>$ represents the right-shift operation. As an example, $d_{0,2} = 1/[2 \cdot \cos(\pi/4)] = 1/\sqrt{2}$ with N_q = 7 gives $\delta_{0,2}$ = 91. As a consequence, depending on N_q, different trade-offs between DCT accuracy and hardware cost have been defined. The results of this trade-off exploration are reported in Section IV. Since $\delta_{k,N}$ are constants, multiplications are simplified into add-and-shift blocks by exploiting the Reduced Adder Graph (RAG-n) technique [31]. As an example, the

TABLE III
Scaled Values of Inverse Cosine Coefficients $\delta_{k,N}$ and Number of Add $A_{k,N}$ and Shift $S_{k,N}$ Operations for RAG-n Implementation with N_q = 7

(k,N)	$\delta_{k,N}$	$A_{k,N}$	$S_{k,N}$	(k,N)	$\delta_{k,N}$	$A_{k,N}$	$S_{k,N}$
0,2	91	3	3	0,32	64	0	1
0,4	69	2	2	1,32	65	1	1
1,4	167	3	3	2,32	66	1	2
0,8	65	1	1	3,32	68	1	2
1,8	77	3	3	4,32	71	2	2
2,8	115	3	3	5,32	75	2	2
3,8	328	2	3	6,32	80	1	2
0,16	64	0	1	7,32	86	3	4
1,16	67	2	2	8,32	95	2	2
2,16	73	2	2	9,32	107	3	3
3,16	83	3	3	10,32	124	1	2
4,16	101	3	3	11,32	150	2	3
5,16	136	1	2	12,32	190	2	3
6,16	220	2	3	13,32	263	2	2
7,16	653	3	3	14,32	436	3	4
-	-	-	-	15,32	1304	3	4

product $\delta_{k,N} \cdot h^*_k$ in (12) when $k = 0$, $N = 2$ and $N_q = 7$ becomes:

$$\delta_{0,2} \cdot h^*_0 = 91 \cdot h^*_0 = (h^*_0 << 6) + (3 \cdot h^*_0 << 3) + 3 \cdot h^*_0, \tag{14}$$

where $3 \cdot h^*_0 = (h^*_0 << 1) + h^*_0$ and $<<$ represents the left-shift operation. Thus, the multiplication is implemented with the RAG-n technique as three additions ($A_{0,2} = 3$) and three shift operations ($S_{0,2} = 3$). The $\delta_{k,N}$ values, as well as the number of add ($A_{k,N}$) and shift ($S_{k,N}$) operations required to implement each coefficient through the RAG-n representation for N_q = 7 are reported in Table III. As it can be observed, when k=15 and N=32, Eq. (7) gives $d_{15,32} \approx 10.19$, which means that in the worst case 4 bits are required to correctly represent the integer part of $d_{k,N}$.

The last stage in Fig. 3 connects the result of the upper $N/2$-point DCT (i.e. $\mathbf{G} = \{G_0, \ldots, G_{N/2-1}\}$) to even-position 1D-DCT results, X_{2k} with $k = 0, \ldots, N/2 - 1$. On the other hand, the output of the lower $N/2$-point DCT (i.e. $\mathbf{H} = \{H_0, \ldots, H_{N/2-1}\}$) is fed to the Recombination Unit, which calculates odd-position 1D-DCT results (X_{2k+1} with $k = 0, \ldots, N/2 - 1$) as described in (8), namely:

$$X_{2k+1} = \begin{cases} H_k + H_{k+1} & k = 0, \ldots, \frac{N}{2} - 2 \\ H_k & k = \frac{N}{2} - 1 \end{cases}. \tag{15}$$

Architectures for N = 8, 16, 32 are obtained by applying the recursion shown in Fig. 3 until the 4-point DCT is found. The inner 4-point DCT architecture, which is the smallest non-recursive unit, is depicted in Fig. 4. Fig. 5 (a)-(c) illustrates the internal structures of Butterfly, Inverse Cosine and Recombination units for N = 8.

B. Variable-size 1D-DCT Architecture

As stated in Section II-C, the proposed architecture has to concurrently compute one 32-point DCT or two 16-point DCTs or four 8-point DCTs or eight 4-point DCTs, so it sustains a processing rate of 32 coefficients per cycle independently of the size N. Thus, the 1D-DCT architecture shown in Fig. 3 has been modified as depicted in Fig. 6 to support variable-size. As it can be observed, the reconfigurability of

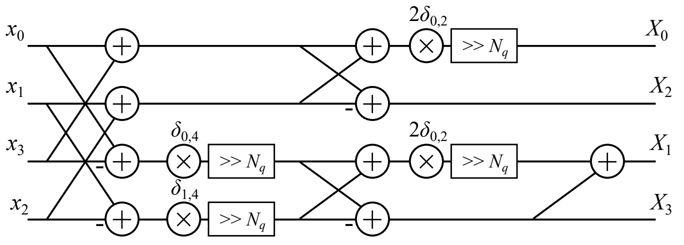


Fig. 4. Proposed architecture of 1D-DCT for N = 4.

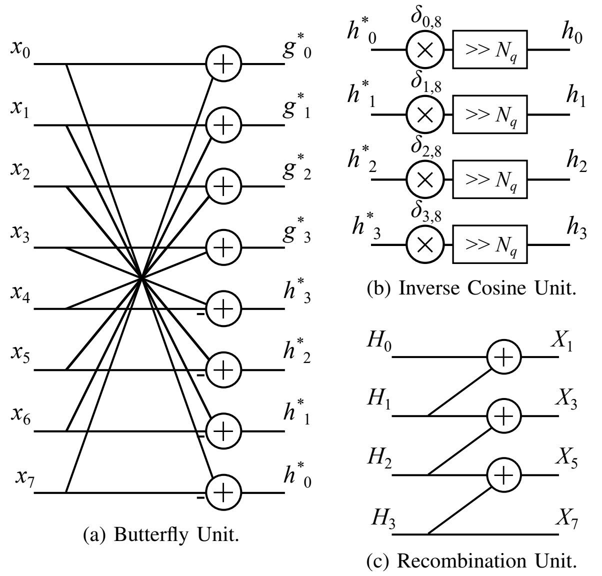
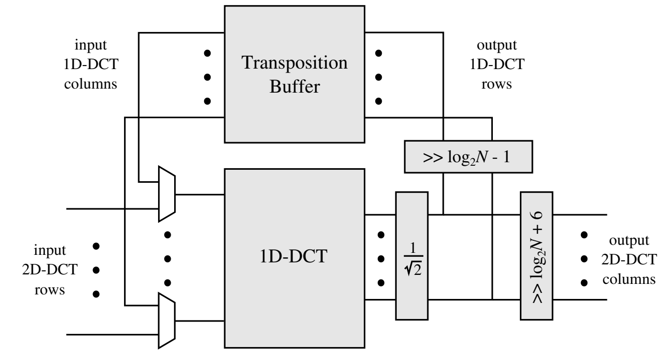


Fig. 5. Proposed architectures of 1D-DCT internal units for N = 8.

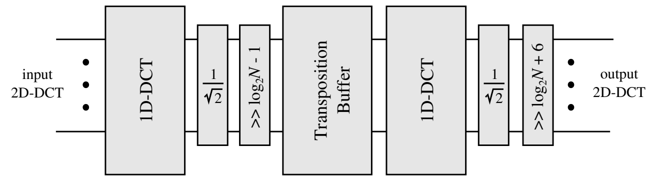
the architecture for variable-size computation is achieved by the means of two banks of 2:1 multiplexers. The selection signal of each multiplexer, referred to as sel, is driven by simple logic depending on current N value. In particular, when the architecture is used for N-point DCT computation, each $N/2$-point DCT block is fed with the output of the Butterfly and Inverse Cosine units, i.e. $\mathbf{g}$ and $\mathbf{h}$. On the contrary, when the architecture is used to compute small-size DCTs, each $N/2$-point DCT block receives $N/2$ input samples, i.e. $(x_0, \ldots, x_{N/2-1})$ and $(x_{N/2}, \ldots, x_{N-1})$. As it can be observed, the result produced by the upper $N/2$-point DCT block is directly connected to the output, independently of the selected transform size. The second bank of multiplexers is used to output the Recombination Unit results when N-point DCT computation is selected. On the other hand, it connects the results of the lower $N/2$-point DCT block to $\mathbf{X}$, when small-size DCT computation is required. The complete variable-size computation required by HEVC is achieved by applying this scheme recursively from the maximum size N = 32 to the smallest non-recursive 4-point DCT.

C. 2D-DCT Architecture

As described in Section I, the separability property is exploited to design two variable-size 2D-DCT architectures, based on the solution proposed in [9], which are composed of one (*Folded*) or two (*Full-parallel*) variable-size 1D-DCT blocks with maximum size N = 32 and a transposition buffer to store the intermediate results, as shown in Fig. 7.



(a) Folded.



(b) Full-parallel.

Fig. 7. 2D-DCT architectures as in [9].

The *Folded* architecture relies on one 1D-DCT block only, which is reused when computing the 2D-DCT. In the first 32 cycles, the 1D-DCT block is fed with 32 samples taken row-wise from the $32^2/N^2$ input data blocks of size $N \times N$, as in [9]. Its outputs are then scaled, to be consistent with the HEVC encoder [32], and stored row-wise in the transposition buffer. During the following 32 cycles, successive columns of the transposition buffer are read and processed by the 1D-DCT block, which outputs the final DCT results. The whole computation takes 64 clock cycles to compute 32 $\times$ 32 results independently of the DCT size N. Therefore, the achieved processing rate is equal to 16 results per cycle.

On the other hand, the *Full-parallel* architecture in Fig. 7 (b) achieves double processing rate by exploiting two 1D-DCT modules to calculate the 2D-DCT. In the first 32 cycles, 32 input samples are fed row-wise to the first 1D-DCT block, which writes the scaled partial results into the transposition buffer. Then, during the following 32 cycles, the DCT is computed column-wise by the second 1D-DCT module. At the same time, a new computation can start in the first module, thus achieving a processing rate of 32 results per cycle. Since the two 1D-DCT blocks have to concurrently write/read to/from the transposition buffer, the first 1D-DCT block may overwrite part of the data which the second 1D-DCT block is going to read in the next clock cycles. Even if this problem can be avoided by doubling the transposition buffer, in this work we exploit the simple and effective solution proposed in [9], which requires only one transposition buffer. Namely, the first 1D-DCT block starts writing row-wise in the transposition buffer. When the buffer is full, the second 1D-DCT block reads the data column-wise. As a consequence, now the first 1D-DCT block can write column-wise in the transposition buffer. These new data will be read row-wise by the second 1D-

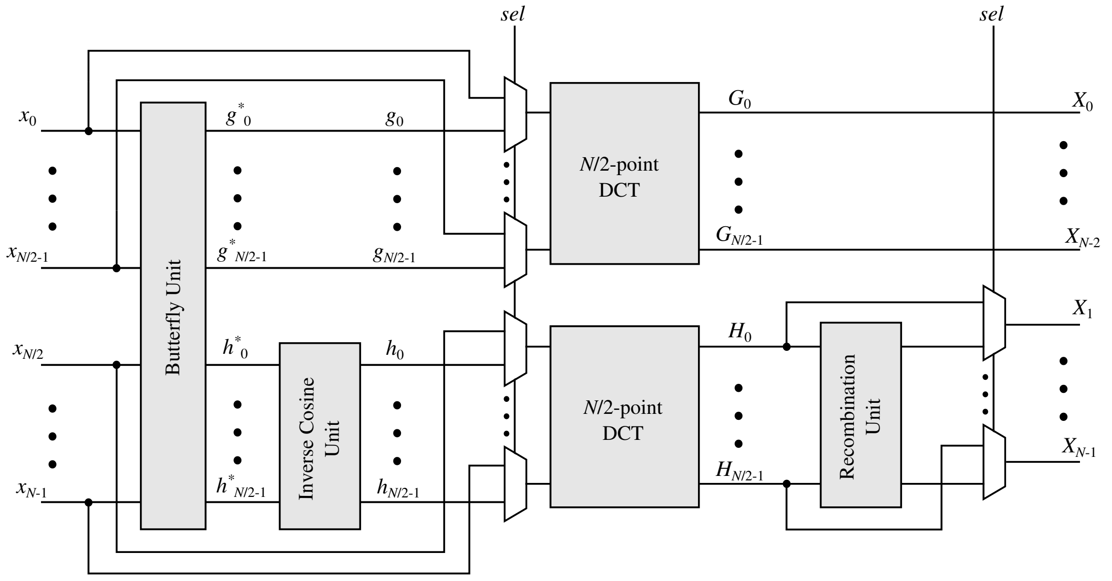


Fig. 6. Proposed variable-size 1D-DCT architecture for N = 8, 16 and 32.

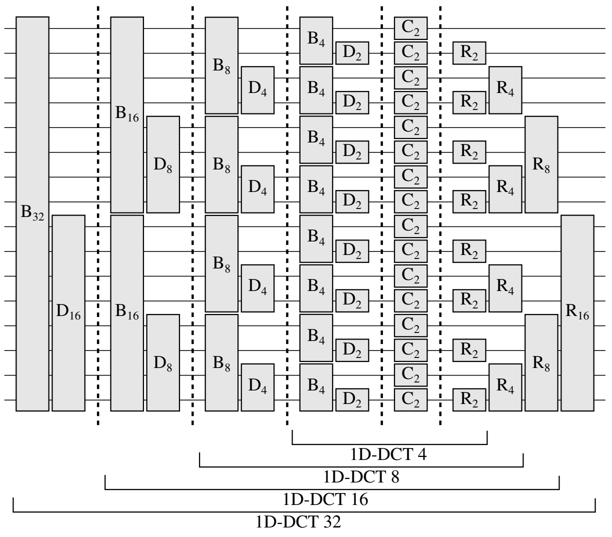


Fig. 8. Pipeline stages of variable-size 1D-DCT.

DCT block. Thus, by alternating row-wise and column-wise writing/reading, which is implemented with some multiplexers and few logic, only one transposition buffer is needed.

D. Pipelined Architectures

The performance of the proposed variable-size DCT architectures can be improved by applying pipelining to the inner 1D-DCT blocks. It is worth noting that the adoption of pipelining inside the 1D-DCT block increases its latency. Since the 2D-DCT requires that all the partial results are calculated before starting the computation on the transposed samples, the maximum sustained throughput of the 2D-DCT architectures is affected by the latency of the core 1D-DCT module. However, to understand the effectiveness of pipelining, maximum sustained throughput analysis is required. In general, the maximum sustained throughput of the complete 2D-DCT architecture processing $32^2/N^2$ blocks of $N \times N$ pixels and employing pipelining can be expressed as:

$$T = \frac{32^2}{N^2} \cdot \frac{N^2}{\alpha \cdot (32 + L)} \cdot f_{CK}, \qquad (16)$$

where α is 2 and 1 for the *Folded* and the *Full-parallel* architectures respectively, L is the number of pipeline stages and f_{CK} is the operating clock frequency. It is worth noting that if L is independent of N, then the maximum sustained throughput is independent of N as well. In particular, the case when L = 0 corresponds to the constant-throughput architectures described in Section III-C. As it can be inferred from (16), deep pipelining (large L), which is useful to reduce the critical path of large-size DCTs, increases the latency (i.e. $32 + L$), thus leading to a severe T reduction. As an example, considering the *Folded* architecture and L = 5, the processing rate of the architecture decreases from 16 to 13.8 samples/cycle. However, Fig. 6 highlights that multiplexers, required to support variable-size DCTs, act as bypass elements. As a consequence, the number of pipeline stages required for small-size DCTs (e.g. N = 4) is lower than the one required for large-size DCTs (e.g. N = 32). Thus, the maximum sustained throughput of the architecture T improves if L becomes a function of N (L_N - adaptive pipelining). In this case, where the throughput T_N is a function of N, the maximum sustained throughput T can be obtained as a weighted sum of T_N, where the weight associated to each size is the usage statistic of the corresponding DCT [10]; such values well approximate the experimental ones measured in [7]. To implement adaptive pipelining, we added pipeline stages to limit the critical path to one adder and one add-and-shift multiplier. Therefore, L_N = 2, 3, 4, 5 have been used for N = 4, 8, 16, 32, respectively, as shown in Fig. 8. Thus, processing rates of 15.0, 14.6, 14.2 and 13.8 samples/cycle are achieved for N = 4, 8, 16, 32 respectively and 14.9 samples/cycle on average, in the *Folded* architecture. On the other hand, the *Full-parallel* structure achieves 30.1, 29.3, 28.4, 27.7 samples/cycle and

29.8 samples/cycle on average, where the average processing rate has been computed as a weighted sum, as suggested in [7], [10].

IV. Implementation Results

A. Data Representation

The internal parallelism and the scaling operations have been sized as specified in the HEVC standard [3]. Indeed, input samples are represented with 9 bits, as they are residuals of the intra and inter prediction modules. Besides, intermediate as well as final results are represented with 16 bits. Moreover, 1D-DCT results are scaled prior to being stored in the transposition buffer ($>> \log_2 N - 1$) and before being output ($>> \log_2 N + 6$), as shown in Fig. 7, where the $1/\sqrt{2}$ scaling required in (2) for σ_k is shown as well.

B. HEVC Performance/Resources Analysis

As introduced in Section III, it is possible to define trade-offs between rate-distortion loss and hardware cost by changing N_q, the number of bits to represent the fractional part of each inverse cosine coefficient.

In order to select the proper N_q value in the context of HEVC encoding, we have integrated Lee's DCT algorithm [21] into the HEVC reference software HM-16.3 [32][1]. In this work two setups have been considered. In the first one, referred to as *setup1*, only the forward transform has been modified, whereas the decoder implements the original HEVC transform. In the second one, referred to as *setup2*, both the forward and the inverse transforms have been modified as well as the decoder.

Simulations have been performed on all the video sequences taken from classes A, B, C, D, E and F, as specified in the common test conditions reported in [34]. All-Intra (AI), Low-Delay (LD) and Random-Access (RA) main configurations and quantization parameters 22, 27, 32 and 37 have been used. The rate-distortion loss has been studied via Bjøntegaard Delta (BD) metrics [35], namely by computing the BD-rate between curves obtained by encoding the sequences with the proposed solution, both for *setup1* and *setup2*, and with the original partial-butterfly approach used in HEVC, which is available in the HM-16.3 software model and has been used as anchor in the comparison. Table IV shows the average BD-rate variations both for *setup1* and *setup2* per each class of sequences in the three encoding configurations as a function of N_q in the range from 4 to 7. As it can be observed, in the worst case the BD-rate loss decreases significantly when increasing N_q. Besides, by choosing N_q = 7 bits, the overall BD-rate loss becomes negligible for *setup1* and less than 2% for *setup2*. It is worth noting that the rate-distortion performance of *setup2* are generally worse. This fact is due to the adopted numeric approximation of the coefficients $\delta_{k,N}$ in the inverse DCT. Indeed, for the aim of simplicity, in this work the same calculated coefficients $\delta_{k,N}$ have been used in both the forward and inverse DCT. More accurate results for *setup2* can be obtained by choosing the set of coefficients $\delta_{k,N}$ which minimize the mismatch between the forward and the inverse DCT matrices. From these experimental results, one can infer that the proposed 1D-DCT, based on Lee's factorization, achieves nearly the same rate-distortion performance as the partial-butterfly approach implemented in the reference software, therefore it is suitable for HEVC applications.

[1]Modified reference software HM-16.3 is available at [33].

TABLE IV
BD-rate [%] Comparison of the Lee's DCT in HEVC for *setup1* and *setup2*:
(a) All Intra, (b) Low Delay, (c) Random Access.

N_q	*setup1*				*setup2*			
	4	5	6	7	4	5	6	7
Class A	2.15	0.93	0.27	0.12	9.32	6.14	4.53	3.46
Class B	0.94	0.39	0.14	0.04	5.40	3.76	2.97	2.21
Class C	0.53	0.28	0.07	0.03	2.35	1.56	1.26	0.99
Class D	0.44	0.20	0.05	0.03	2.03	1.17	0.96	0.72
Class E	0.72	0.38	0.09	0.05	4.82	3.64	3.03	2.55
Class F	0.32	0.20	0.06	0.08	1.71	1.19	1.04	0.83
Overall	0.86	0.40	0.12	0.06	4.30	2.91	2.30	1.78

(a)

N_q	*setup1*				*setup2*			
	4	5	6	7	4	5	6	7
Class A	-	-	-	-	-	-	-	-
Class B	0.49	0.07	0.10	0.00	3.41	2.25	2.18	1.83
Class C	0.25	0.13	0.09	0.00	2.15	1.45	1.38	1.16
Class D	0.31	0.17	0.12	0.05	1.76	1.11	1.00	0.92
Class E	0.48	0.20	0.04	0.12	3.74	2.68	2.72	2.43
Class F	0.22	0.09	-0.02	-0.06	1.84	1.32	1.05	0.99
Overall	0.35	0.13	0.07	0.02	2.57	1.74	1.64	1.44

(b)

N_q	*setup1*				*setup2*			
	4	5	6	7	4	5	6	7
Class A	1.73	0.15	0.35	0.00	7.81	4.29	3.94	3.06
Class B	0.63	0.20	0.09	0.02	4.30	2.85	2.40	1.86
Class C	0.32	0.17	0.00	0.01	1.71	1.09	1.00	0.83
Class D	0.25	0.15	0.05	0.03	1.67	1.10	0.90	0.76
Class E	-	-	-	-	-	-	-	-
Class F	0.28	0.15	0.05	0.08	1.61	1.01	0.90	0.76
Overall	0.64	0.16	0.11	0.02	3.80	2.31	2.03	1.62

(c)

As explained in Section III-A, multiplications have been simplified to add-and-shift blocks, by exploiting the RAG-n technique [31]. Similar approaches have been followed in [7], [9], where multiplications were simplified through the RAG-n and Multiple Constant Multiplication algorithms, respectively. Thus, Table V shows the detailed number of adders required by these multiplierless algorithms both for individual values of N and for the variable-size 1D-DCT architectures (see the last row labeled as *All*). In [7] the use of RAG-n leads to an architecture made of A_{All} = A_{32} = 584 adders. According to (3), the work in [9] requires 1024 adders. In the proposed architecture the number of adders is A_{All} = A_{32} and it grows as the BD-rate loss decreases (i.e. increasing N_q), as expected. Nevertheless, the number of adders used in the proposed variable-size 1D-DCT architecture ranges from 314 (N_q = 4) to 394 (N_q = 7), so it is noticeably lower than the one required by [7] and [9], for all tested N_q values.

TABLE V
NUMBER OF ADDERS OF THE STATE-OF-THE-ART AND THE PROPOSED MULTIPLIERLESS ALGORITHMS AS FUNCTION OF THE NUMBER OF FRACTIONAL BITS N_q

N	[7]	[9]	N_q			
			4	5	6	7
4	17	14	16	16	17	20
8	69	50	47	49	52	60
16	213	186	125	133	139	159
32	584	682	314	335	350	394
All	584	1024	314	335	350	394

TABLE VI
COMPARISON OF VARIABLE-SIZE 1D-DCT ARCHITECTURE AS FUNCTION OF THE NUMBER OF FRACTIONAL BITS N_q

N_q	Maximum Frequency		Minimum Area	
	f_{CK} [MHz]	Gates	f_{CK} [MHz]	Gates
4	380	86 K	185	63 K
5	343	96 K	174	70 K
6	326	106 K	161	75 K
7	296	111 K	147	83 K

C. Synthesis of Variable-size 1D-DCT

The proposed architecture has been described in VHDL, verified and synthesized with a 90-nm CMOS standard-cell library. The performance and the gate count when synthesizing the variable-size 1D-DCT architecture for maximum frequency and for minimum area are reported in Table VI. As expected, the equivalent gate count of the proposed architecture increases with the number of fractional bits N_q, whereas the operating frequency is reduced because of longer critical paths. Moreover, when searching for the maximum performance, the frequency doubles while the gate count increases only from the 33% to the 40% with respect to the minimum area implementations.

Table VII reports the technology, the operating frequency (f_{CK}), the number of processed samples per cycle for each DCT size, the throughput (T) and the equivalent gate count of different variable-size 1D-DCT architectures for HEVC. The processing rate of the 1D-DCT architectures is calculated considering their usage in the *Folded* 2D structure. Thus, they have been synthesized at an operating frequency of 187 MHz and 401 MHz, for the non-pipelined and the pipelined architectures, respectively. The proposed non-pipelined and pipelined architectures respectively show smaller gate count and higher throughput when compared to the other state-of-the-art DCT implementations. In particular, only the solution proposed in [14] provides lower gate count at the expense of very high rate-distortion loss. Area saving of about 33% has been achieved by the proposed non-pipelined architecture with respect to the best implementation in [9] for equal throughput. Since the proposed pipelined variable-size 1D-DCT allows to speed-up the computation with respect to the non-pipelined architecture, it provides double throughput at the cost of 28% more area, due to the implementation of pipe registers.

D. Synthesis of 2D-DCT

Table VIII compares the proposed 2D-DCT with other existing architectures for HEVC in terms of technology, operating frequency (f_{CK}), processing rate, throughput (T), gate count, throughput-area ratio, power consumption (P), energy-per-sample (EPS) and BD-rate for *setup1* and All Intra configuration. Both pipelined and non-pipelined variable-size 1D-DCTs have been used to implement the corresponding *Folded* and *Full-parallel* 2D-DCT structures, where the two transposition buffers presented in [9] have been used as well. The values related to the proposed architectures refer to the synthesis with N_q = 7, whereas Architecture 1 MODE0 has been chosen for fair comparison with the work in [7], since it uses a totally unfolded 1D-DCT and provides negligible rate-distortion losses. As highlighted by the ratio of throughput over area and by the energy-per-sample metric, the proposed architectures provide very high area and power efficiency. The architecture proposed in [14] achieves the best area efficiency at the cost of a very high rate-distortion performance degradation. With respect to the best state-of-the-art implementations in [9], the proposed 2D architectures based on non-pipelined variable-size 1D-DCT achieve 20% and 27% area reduction for equal throughput in the *Folded* and *Full-parallel* schemes respectively. On the other hand, the architectures based on the pipelined 1D-DCT achieve the best area efficiency for negligible losses by nearly doubling the achievable throughput and showing lower gate count with respect to the architectures in [9].

V. CONCLUSION

In this paper, an efficient variable-size 1D-DCT architecture for HEVC has been proposed. A comparison among the existing DCT factorizations has been presented and exploited to identify the one which minimizes the amount of hardware resources required to support variable-sizes. For the selected factorization, which is Lee's factorization, fixed-point analysis has been performed by evaluating the rate-distortion loss in the HEVC context and an area-efficient architecture has been derived as well. Then, the architectures of both 1D-DCT and 2D-DCT have been implemented and characterized, achieving significant area reduction with respect to other DCT architectures for HEVC, available in the literature. Furthermore, an adaptive pipeline scheme has been applied to the proposed variable-size 1D-DCT, which has been exploited to implement area and power efficient 2D-DCT architectures.

ACKNOWLEDGMENT

The authors would like to thank the HPC@POLITO, a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it), which has provided the computational resources.

REFERENCES

[1] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec 2012.

[2] J. Ohm, G. 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 Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1669–1684, Dec 2012.

TABLE VII
COMPARISON OF VARIABLE-SIZE 1D-DCT ARCHITECTURES

Design		Technology [nm]	f_{CK} [MHz]	Samples/Cycle	T [Gsps]	Gates
Zhao *et al.* [10]		45	333	1.2/2.8/6.2/13.6	0.634	205 K
Meher *et al.* [9]		90	187	16	2.992	131 K
Masera *et al.* [7]		90	250	12.8/12.8/13.4/14.2	3.212	163 K
Jridi *et al.* [14]		90	322	16	5.152	31 K
Goebel *et al.* [18]		45	50	32	1.600	97 K
Proposed	N_q=7	90	187	16	2.992	88 K
Proposed Pipelined	N_q=7	90	401	15.0/14.6/14.2/13.8	5.985	113 K

TABLE VIII
COMPARISON OF 2D-DCT ARCHITECTURES

Design		Technology [nm]	f_{CK} [MHz]	Samples/Cycle	T [Gsps]	gates	T/gates [Gsps]	P [mW]	EPS [pJ]	BD-rate [%]
Zhao *et al.* [10]		45	333	1.2/2.8/6.2/13.6	0.634	345 K	1837	-	-	0.00
Ahmed *et al.* [6]		90	150	1.0/2.6/6.4/7.5	0.246	149 K	1651	-	-	-
Meher *et al.* [9]	Folded	90	187	16	2.992	208 K	14384	40.04	13.38	0.00
	Full-parallel	90-nm	187	32	5.984	347 K	17244	67.57	11.29	0.00
Masera *et al.* [7]	Architecture 1	90	250	12.8/12.8/13.4/14.2	3.212	243 K	13218	51.72	16.10	0.01
Jridi *et al.* [14]	Folded	90	322	16	5.152	108 K	47703	-	-	7.07
Proposed	Folded	90	187	16	2.992	165 K	18133	31.95	10.67	0.06
	Full-parallel	90	187	32	5.984	253 K	23652	49.08	8.20	
Proposed Pipelined	Folded	90	401	15.0/14.6/14.2/13.8	5.985	190 K	31500	64.18	10.72	0.06
	Full-parallel	90	401	30.1/29.3/28.4/27.7	11.969	303 K	39501	113.56	9.49	

[3] M. Budagavi, A. Fuldseth, G. Bjontegaard, V. Sze, and M. Sadafale, "Core Transform Design in the High Efficiency Video Coding (HEVC) Standard," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 1029–1041, Dec 2013.

[4] M. Naccari, A. Gabriellini, M. Mrak, S. G. Blasi, I. Zupancic, and E. Izquierdo, "HEVC Coding Optimisation for Ultra High Definition Television Services," in *Picture Coding Symposium*, May 2015, pp. 20–24.

[5] M. Masera, L. R. Fiorentin, E. Masala, G. Masera, and M. Martina, "Analysis of HEVC Transform Throughput Requirements for Hardware Implementations," *Elsevier Signal Processing: Image Communication*, vol. 57, pp. 173–182, 2017.

[6] A. Ahmed, M. U. Shahid, and A. Rehman, "N Point DCT VLSI Architecture for Emerging HEVC Standard," *VLSI Design*, vol. 2012, Article 752024, pp. 1–13, 2012.

[7] M. Masera, M. Martina, and G. Masera, "Adaptive Approximated DCT Architectures for HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.

[8] M. Budagavi and V. Sze, "Unified Forward+Inverse Transform Architecture for HEVC," in *Proc. 19th IEEE Int. Conf. Image Processing*, Sept 2012, pp. 209–212.

[9] P. Meher, S. Y. Park, B. Mohanty, K. S. Lim, and C. Yeo, "Efficient Integer DCT Architectures for HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 1, pp. 168–178, Jan 2014.

[10] W. Zhao, T. Onoye, and T. Song, "High-Performance Multiplierless Transform Architecture for HEVC," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, May 2013, pp. 1668–1671.

[11] T. Dias, N. Roma, and L. Sousa, "High performance multi-standard architecture for DCT computation in H.264/AVC High Profile and HEVC codecs," in *Proc. Conf. Design and Architectures for Signal and Image Processing*, Oct 2013, pp. 14–21.

[12] A. Madanayake, R. Cintra, V. Dimitrov, F. Bayer, K. Wahid, S. Kulasekera, A. Edirisuriya, U. Potluri, S. Madishetty, and N. Rajapaksha, "Low-Power VLSI Architectures for DCT/DWT: Precision vs Approximation for HD Video, Biomedical, and Smart Antenna Applications," *IEEE Circuits Syst. Mag.*, vol. 15, no. 1, pp. 25–47, Firstquarter 2015.

[13] U. Sadhvi Potluri, A. Madanayake, R. Cintra, F. Bayer, S. Kulasekera, and A. Edirisuriya, "Improved 8-Point Approximate DCT for Image and Video Compression Requiring Only 14 Additions," *IEEE Trans. Circuits Syst. I*, vol. 61, no. 6, pp. 1727–1740, Jun 2014.

[14] M. Jridi, A. Alfalou, and P. Meher, "A Generalized Algorithm and Reconfigurable Architecture for Efficient and Scalable Orthogonal Approximation of DCT," *IEEE Trans. Circuits Syst. I*, vol. 62, no. 2, pp. 449–457, Feb 2015.

[15] M. Jridi and P. Meher, "A Scalable Approximate DCT Architecture for Efficient HEVC Compliant Video Coding," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.

[16] C. K. Fong, Q. Han, and W. K. Cham, "Recursive Integer Cosine Transform for HEVC and Future Video Coding Standards," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.

[17] Y. J. Chen, S. Oraintara, and T. Nguyen, "Video Compression Using Integer DCT," in *Proc. Int. Conf. Image Processing*, 2000, pp. 844–845.

[18] J. Goebel, G. Paim, L. Agostini, B. Zatt, and M. Porto, "An HEVC Multi-Size DCT Hardware with Constant Throughput and Supporting Heterogeneous CUs," in *IEEE International Symposium on Circuits and Systems*, 2016, pp. 2202–2205.

[19] W.-H. Chen, C. Smith, and S. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE Trans. Commun.*, vol. 25, no. 9, pp. 1004–1009, Sep 1977.

[20] Z. Wang, "Fast Algorithms for the Discrete W Transform and for the Discrete Fourier Transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 4, pp. 803–816, Aug 1984.

[21] B. Lee, "A New Algorithm to Compute the Discrete Cosine Transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 6, pp. 1243–1245, Dec 1984.

[22] Z. Cvetkovic and M. V. Popovic, "New Fast Recursive Algorithms for the Computation of Discrete Cosine and Sine Transforms," *IEEE Trans. Signal Process.*, vol. 40, no. 8, pp. 2083–2086, Aug 1992.

[23] H. S. Hou, "A Fast Recursive Algorithm for Computing the Discrete Cosine Transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 35, no. 10, pp. 1455–1461, Oct 1987.

[24] S. C. Chan and K. L. Ho, "Direct Methods for Computing Discrete Sinusoidal Transforms," *IEE Proceedings F - Radar and Signal Processing*, vol. 137, no. 6, pp. 433–442, Dec 1990.

[25] C. W. Kok, "Fast Algorithm for Computing Discrete Cosine Transform," *IEEE Trans. Signal Process.*, vol. 45, no. 3, pp. 757–760, Mar 1997.

[26] P. Lee and F.-Y. Huang, "Restructured Recursive DCT and DST Algorithms," *IEEE Trans. Signal Process.*, vol. 42, no. 7, pp. 1600–1609, Jul 1994.

[27] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Practical Fast 1-D DCT Algorithms with 11 Multiplications," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, May 1989, pp. 988–991.

[28] W. Yuan, P. Hao, and C. Xu, "Matrix Factorization for Fast DCT Algorithms," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 3, May 2006, pp. 948–951.

[29] V. Britanak, P. C. Yip, and K. R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. Elsevier, Sep. 2006.

[30] Y. M. Hong, I.-K. Kim, T. Lee, M.-S. Cheon, E. Alshina, W.-J.

Han, and J.-H. Park, “New Fast DCT Algorithms Based on Loeffler’s Factorization,” in *Proc. SPIE*, vol. 8499, 2012, pp. 84 990U–84 990U–8.

[31] A. Dempster and M. MacLeod, “Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters,” *IEEE Trans. Circuits Syst. II*, vol. 42, no. 9, pp. 569–577, Sep 1995.

[32] Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, *HM 16.3 Reference Software*. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.3/

[33] M. Masera, *An Area-Efficient Variable-Size Fixed-Point DCT Architecture for HEVC Encoding: Modified HM 16.3 Reference Software*. [Online]. Available: http://personal.det.polito.it/maurizio.masera/material/HM-16.3_DCTLee.zip

[34] F. Bossen, *Common Test Conditions and Software Reference Configurations*, document JCTVC-L1100, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Geneva, CH, Jan 2013.

[35] G. Bjontegaard, *Calculation of Average PSNR Differences Between RD Curves*, document VCEG-M33, ITU-T SG16/Q6, Austin, TX, Apr 2001.

PLACE PHOTO HERE

Maurizio Masera received the M.Sc. and the Ph.D. degrees in electronic engineering from Politecnico di Torino, in 2014 and 2018 respectively. His research activities include design of digital VLSI circuits and implementation of algorithms and architectures for multimedia applications.

PLACE PHOTO HERE

Guido Masera (SM’07) received the Dr. Ing. Degree (summa cum laude) in 1986 and the Ph.D. degree in electronic engineering from the Politecnico di Torino, Torino, Italy, in 1992. Since 1992, he has been an Assistant Professor and then Associate Professor with the Electronic Department, where he is member of the VLSI-Lab group. His research interests include several aspects in the design of digital integrated circuits and systems, with special emphasis on high-performance architecture development and on-chip interconnect modeling and optimization. He is an associate editor of IEEE Transactions on Circuits and Systems II.

PLACE PHOTO HERE

Maurizio Martina (S’98-M’94-SM’15) received the M.S. and Ph.D. in electrical engineering from Politecnico di Torino, Italy, in 2000 and 2004, respectively. He is currently an Associate Professor of the VLSI-Lab group, Politecnico di Torino. His research interests include VLSI design and implementation of architectures for digital signal processing, video coding, communications, artificial intelligence, machine learning and event-based processing. He published 3 book chapters on VLSI architectures and digital circuits for wireless communications and error correcting codes. He has more than 100 scientific publications and is author of 2 patents. He is now an Associate Editor of IEEE Transactions on Circuits and Systems - I, Hindawi Journal of Electrical and Computer Engineering and Journal of Circuits, Systems and Computers. He has been Tutorials, Special Sessions, and Awards co-Chair of NGCAS 2017 and part of the organizing committee, demo co-Chair and treasurer of IEEE BioCAS 2017. Recently, he has been guest editor of two special issues of selected papers from NGCAS 2017 (JOLPE and Integration, the VLSI Journal). In 2012 he had been guest editor of a special issue on VLSI Circuits, Systems, and Architectures for Advanced Image and Video Compression Standards, (Hindawi VLSI Design). He is a member of the Circuits and Systems Society and reviewer for several journals and conferences of the CAS society. Currently, he is the counselor of the IEEE Student Branch at Politecnico di Torino and a professional member of IEEE HKN.