Core Transform Design in the High Efficiency Video Coding (HEVC) Standard

Madhukar Budagavi, *Senior Member, IEEE*, Arild Fuldseth, Gisle Bjøntegaard, Vivienne Sze, *Member, IEEE*, and Mangesh Sadafale

***Abstract*—This paper describes the core transforms specified for the high efficiency video coding (HEVC) standard. Core transform matrices of various sizes from 4×4 to 32×32 were designed as finite precision approximations to the discrete cosine transform (DCT). Also, special care was taken to allow implementation friendliness, including limited bit depth, preservation of symmetry properties, embedded structure and basis vectors having almost equal norm. The transform design has the following properties: 16 bit data representation before and after each transform stage (independent of the internal bit depth), 16 bit multipliers for all internal multiplications, no need for correction of different norms of basis vectors during quantization/de-quantization, all transform sizes above 4×4 can reuse arithmetic operations for smaller transform sizes, and implementations using either pure matrix multiplication or a combination of matrix multiplication and butterfly structures are possible. The transform design is friendly to parallel processing and can be efficiently implemented in software on SIMD processors and in hardware for high throughput processing.**

***Index Terms*—Discrete cosine transform, high efficiency video coding, transform design.**

I. Introduction

THE HEVC standard [1] specifies core transform matrices of size 4×4, 8×8, 16×16, and 32×32 to be used for two-dimensional transforms in the context of block-based motion-compensated video compression. Similar to previous video coding standards, HEVC specifies two-dimensional transforms resembling the inverse discrete cosine transform (IDCT) for all transform sizes. Multiple transform sizes improve compression performance, but also increase the implementation complexity. Hence a careful design of the core transforms is needed. HEVC also specifies an alternate 4×4 integer transform based on the Discrete Sine Transform (DST) for coding 4×4 Intra blocks [2]. The focus of this paper will be on the IDCT-based HEVC core transform.

In the H.261, MPEG-1, H.262/MPEG-2, and H.263 video coding standards, an 8-point IDCT was specified with infinite precision. To ensure interoperability and to minimize drift between encoder and decoder implementations using finite precision, two features were included in the standards. First, block-level periodic intra refresh was mandatory. Second, a conformance test for the accuracy of the IDCT using a pseudo-random test pattern was specified.

In the H.264/MPEG-4 Advanced Video Coding (AVC) standard [3], the problem of encoder-decoder drift was solved by specifying integer valued 4×4 and 8×8 transform matrices. The transforms were designed as approximations to the IDCT with emphasis on minimizing the number of arithmetic operations. These transforms had large variations of the norm of the basis vectors. As a consequence of this, non-flat default de-quantization matrices were specified to compensate for the different norms of the basis vectors [4].

During the development of HEVC, several different approximations of the IDCT were studied for the core transform. The first version of the HEVC Test Model HM1 used the H.264/AVC transforms for 4×4 and 8×8 blocks and integer approximation of Chen's fast IDCT [5] for 16×16 and 32×32 blocks. The HM1 inverse transforms had the following characteristics [6], [7]:

- Non-flat de-quantization matrices for all transform sizes: While acceptable for small transform sizes, the implementation cost of using de-quantization matrices for larger transforms is high because of larger block sizes,
- Different architectures for different transform sizes: This leads to increased area since hardware sharing across different transform sizes is difficult,
- A 20-bit transpose buffer used for storing intermediate results after the first transform stage in 2D transform: An increased transpose buffer size leads to larger memory and memory bandwidth. In hardware, the transpose buffer area can be significant and comparable to transform logic area [8],
- Full factorization architecture requiring cascaded multipliers and intermediate rounding for 16- and 32-point transforms: This increases data path dependencies and impacts parallel processing performance. It also leads to increased bit width for multipliers and accumulators (32 bits and 64 bits respectively in software). In hardware, in addition to area increase, it also leads to increased circuit delay thereby limiting the maximum frequency at which the inverse transform block can operate.

To address the complexity concerns of the HM1 transforms, a matrix multiplication based core transform was proposed in [9] and eventually adopted as the HEVC core transform. The

Manuscript received January 16, 2013; revised May 10, 2013; accepted June 13, 2013. Date of publication June 20, 2013; date of current version November 18, 2013. The guest editor coordinating the review of this manuscript and approving it for publication was Prof. Yun He.

A. Fuldseth and G. Bjøntegaard are with Cisco Systems Norway, 1366 Lysaker, Norway (e-mail: arild.fuldseth@cisco.com; gbjonteg@cisco.com).

M. Budagavi and V. Sze are with Texas Instruments, Inc., Dallas, TX 75243 USA (e-mail: madhukar@ti.com, sze@ti.com).

M. Sadafale is with Signalchip Innovations, Bangalore 560 052, India, (e-mail: mangesh@signalchip.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTSP.2013.2270429

design goal was to develop a transform that was efficient to implement in both software on SIMD machines and in hardware. Alternative proposals to the HEVC core transform design can be found in [10]–[12].

The HEVC core transform matrices were designed to have the following properties [9]:

- Closeness to the IDCT
- Almost orthogonal basis vectors
- Almost equal norm of all basis vectors
- Same symmetry properties as the IDCT basis vectors
- Smaller transform matrices are embedded in larger transform matrices
- 8-bit representation of transform matrix elements
- 16-bit transpose buffer
- Multipliers can be represented using 16 bits or less with no cascaded multiplications or intermediate rounding
- Accumulators can be implemented using less than 32 bits

This paper is organized as follows. Section II describes the HEVC core transform design in detail, including matrix elements selection, intermediate scaling and the associated quantization and de-quantization. In Section III, complexity analysis including arithmetic operation counts and hardware analysis is provided. Section IV presents coding performance of the HEVC transforms and also coding performance comparison to the H.264/AVC transforms. Finally, conclusions are given in Section V.

II. HEVC Core Transform Design

A. Use of Transforms in Block-Based Video Coding

In the block-based hybrid video coding approach, transforms are applied to the residual signal resulting from inter- or intra-frame prediction as shown in Fig. 1. At the encoder, the residual signal of a frame is divided into square blocks of size $N \times N$ where $N = 2^M$ and M is an integer. Each residual block ($\mathbf{U}$) is then input to a two-dimensional $N \times N$ forward transform. The two-dimensional transform can be implemented as a separable transform by applying an N-point one-dimensional transform to each row and each column separately. The resulting $N \times N$ transform coefficients ($coeff$) are then subject to quantization (which is equivalent to division by quantization step size $Qstep$) to obtain quantized transform coefficients ($level$). At the decoder, the quantized transform coefficients are then de-quantized (which is equivalent to multiplication by $Qstep$). Finally, a two-dimensional $N \times N$ separable inverse transform is applied to the de-quantized transform coefficients ($coeff'_Q$) resulting in a residual block of quantized samples which is then added to the intra- or inter-prediction samples to obtain the reconstructed block.

Typically, the forward- and inverse transform matrices are transposes of each other and are designed to achieve near lossless reconstruction of the input residual block when concatenated without the intermediate quantization and de-quantization steps.

In video coding standards such as HEVC, the de-quantization process and inverse transforms are specified, while the forward transforms and quantization process are chosen by the implementer (subject to constraints on the bit-stream). In the following, however, and unless otherwise specified, we will discuss the design and the properties of the HEVC core transforms in terms of the forward transform matrix. It should be understood that the inverse transforms are specified in the HEVC standard as the corresponding transpose matrices.

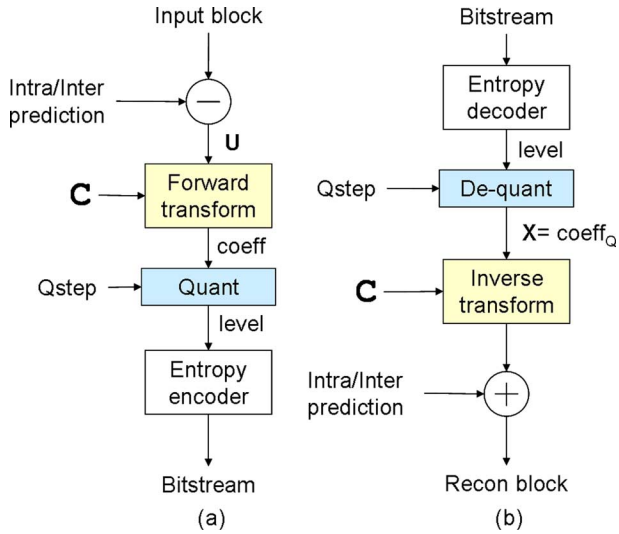

Fig. 1. Block-based hybrid video coding. (a) Encoder, (b) Decoder. $\mathbf{C}$ is the transform matrix and $Qstep$ is the quantization step size.

B. Discrete Cosine Transform

The N transform coefficients w_i of an N-point 1D DCT applied to the input samples u_i can be expressed as

$$w_i = \sum_{j=0}^{N-1} u_j c_{ij} \tag{1}$$

where $i = 0, \ldots, N-1$. Elements c_{ij} of the DCT transform matrix $\mathbf{C}$ are defined as

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

where $i, j = 0, \ldots, N-1$ and where A is equal to 1 and $2^{1/2}$ for $i = 0$ and $i > 0$ respectively. Furthermore, the basis vectors $\mathbf{c}_i$ of the DCT are defined as $\mathbf{c}_i = [c_{i0}, \ldots, c_{i(N-1)}]^T$ where $i = 0, \ldots, N-1$.

The DCT has several properties that are considered useful both for compression efficiency and for efficient implementation.

1) The basis vectors are orthogonal, i.e., $\mathbf{c}_i^T \mathbf{c}_j = 0$ for $i \neq j$. This property is desirable for compression efficiency by achieving transform coefficients that are uncorrelated.
2) The basis vectors of the DCT have been shown to provide good energy compaction which is also desirable for compression efficiency.
3) The basis vectors of the DCT have equal norm, i.e., $\mathbf{c}_i^T \mathbf{c}_i = 1$ for $i = 0, \ldots, N-1$. This property is desirable for simplifying the quantization/de-quantization process. Assuming that equal frequency-weighting of the quantization error is desired, equal norm of the basis vectors eliminates the need for quantization/de-quantization matrices.
4) Let $N = 2^M$. The elements of a DCT matrix of size $2^M \times 2^M$ is a subset of the elements of a DCT matrix of

size $2^{(M+1)} \times 2^{(M+1)}$. More specifically, the basis vectors of the smaller matrix is equal to the first half of the even basis vectors of the larger matrix. This property is useful to reduce implementation costs as the same multipliers can be reused for various transform sizes.

5) The DCT matrix can be specified by using a small number of unique elements. By examining the elements c_{ij} of (2) it can be shown that the number of unique elements in a DCT matrix of size $2^M \times 2^M$ is equal to $2^M - 1$. As shown in Section III, this is particularly advantageous in hardware implementations.
6) The even basis vectors of the DCT are symmetric, while the odd basis vectors are anti-symmetric. This property is useful to reduce the number of arithmetic operations.
7) The coefficients of a DCT matrix have certain trigonometric relationships that allows for a reduction of the number of arithmetic operations beyond what is possible by exploiting the (anti-)symmetry properties. These properties can be utilized to implement fast algorithms such as the Chen's fast factorization [5].

C. Finite Precision DCT Approximations

The core transforms matrices of HEVC are finite precision approximations of the DCT matrix. The benefit of using finite precision in a video coding standard is that the approximation to the real-valued DCT matrix is specified in the standard rather than being implementation dependent. This avoids encoder-decoder mismatch and drift caused by manufacturers implementing the IDCT with slightly different floating point representations. On the other hand, a disadvantage of using approximate matrix elements is that some of the properties of the DCT discussed in Section II-B may not be satisfied anymore. More specifically, there is a trade-off between the computational cost associated with using high bit-depth for the matrix elements and the degree to which some of the conditions of Section II-B are satisfied.

A straightforward way of determining integer approximations to the DCT matrix elements is to scale each matrix element with some large number (typically between 2^5 and 2^{16}) and then round to the closest integer. However, this approach does not necessarily result in the best compression performance. As shown in Section II-D, for a given bit-depth of the matrix elements, a different strategy for approximating the DCT matrix elements results in a different trade-off between some of the properties of Section II-B.

D. HEVC Core Transform Design Principles

The DCT approximations used for the core transforms of HEVC were chosen according to the following principles. First, properties 4, 5 and 6 of Section II-B were satisfied without any compromise. This choice ensures that several implementation friendly aspects of the DCT are preserved. Second, for properties 1, 2, 3 and 7, there were trade-offs between the number of bits used to represent each matrix element and the degree by which each of the properties were satisfied.

To measure the degree of approximation for properties 1, 2, and 3, the following measures are defined for an integer N-point DCT approximation with scaled matrix elements equal to d_{ij} and basis vectors equal to $\mathbf{d}_i = [d_{i0}, \ldots, d_{i(N-1)}]^T$ where $i = 0, \ldots, N-1$.

1) Orthogonality measure: $o_{ij} = \mathbf{d}_i^T \mathbf{d}_j / \mathbf{d}_0^T \mathbf{d}_0$, $i \neq j$
2) Closeness to DCT measure: $m_{ij} = |\alpha c_{ij} - d_{ij}| / d_{00}$
3) Norm measure: $n_i = |1 - \mathbf{d}_i^T \mathbf{d}_i / \mathbf{d}_0^T \mathbf{d}_0|$

where $i, j = 0, \ldots, N-1$, c_{ij} are the DCT matrix elements of (2), and the scale factor α is defined as $d_{00} N^{1/2}$.

TABLE I
COMPARISON OF TRANSFORM DESIGN METHODS.

	HEVC core transforms	Scaling and rounding
Orthogonality	$o_{ij} < 0.0029$	$o_{ij} < 0.0037$
Closeness to DCT	$m_{ij} < 0.0213$	$m_{ij} < 0.0077$
Norm	$n_i < 0.0014$	$n_i < 0.0109$

As a result of careful investigation, it was decided to represent each matrix coefficient with 8 bit (including sign bit), and choosing the elements of the first basis vector to be equal to 64 (i.e., $d_{0j} = 64$, $j = 0, \ldots, N-1$). Note that this results in a scale factor of $2^{(6+M/2)}$ for the HEVC transform matrix when compared to the orthonormal DCT. The remaining matrix elements were hand-tuned (within the constraints of properties 4, 5, and 6) to achieve a good balance between properties 1, 2 and 3. The hand-tuning was performed as follows. First, the real-valued scaled DCT matrix elements, αc_{ij}, were derived. Next, for each unique number in the resulting matrices, each integer value in the interval $[-1.5, 1.5]$ around αc_{ij} was examined and the resulting values of o_{ij}, m_{ij}, and n_i were calculated. Since there are only 31 unique numbers in the transform matrices (see Section II-E), various permutations can be examined systematically (although not exhaustively). The final integer matrix elements were chosen to give a good compromise between all measures o_{ij}, m_{ij}, and n_i. The resulting worst case values of o_{ij}, m_{ij}, and n_i are shown in the second column of Table I. The norm was considered to be sufficiently close to 1 (i.e., the norm measure n_i is sufficiently close to 0) to justify not using a non-flat default de-quantization matrix in HEVC (i.e., all transform coefficients scaled equally).

For comparison purposes, the resulting measures when multiplying the real-valued DCT matrix elements with $2^{(6+M/2)}$ and rounding to the closest integer are listed in the third column of Table I. As can be seen from the table, although the matrix elements of the HEVC transforms are farther from the scaled DCT matrix elements, they have better orthogonality and norm properties.

Finally, by using only 8 bit representation, property 7 of Section II-B (trigonometric relationship between matrix elements) was not easily preserved. The authors are not aware of any trigonometric property of the HEVC core transforms that can be utilized to reduce the number of arithmetic operations below those required when using the (anti-) symmetry properties.

E. Basis Vectors of the HEVC Core Transforms

The left half of the 32×32 matrix specifying the 32-point forward transform is shown in Fig. 2. The right half can be derived by using the (anti-)symmetry properties of the basis vectors (property 6 of Section II-B). The inverse transform matrix of HEVC is defined as the transpose of the matrix resulting from

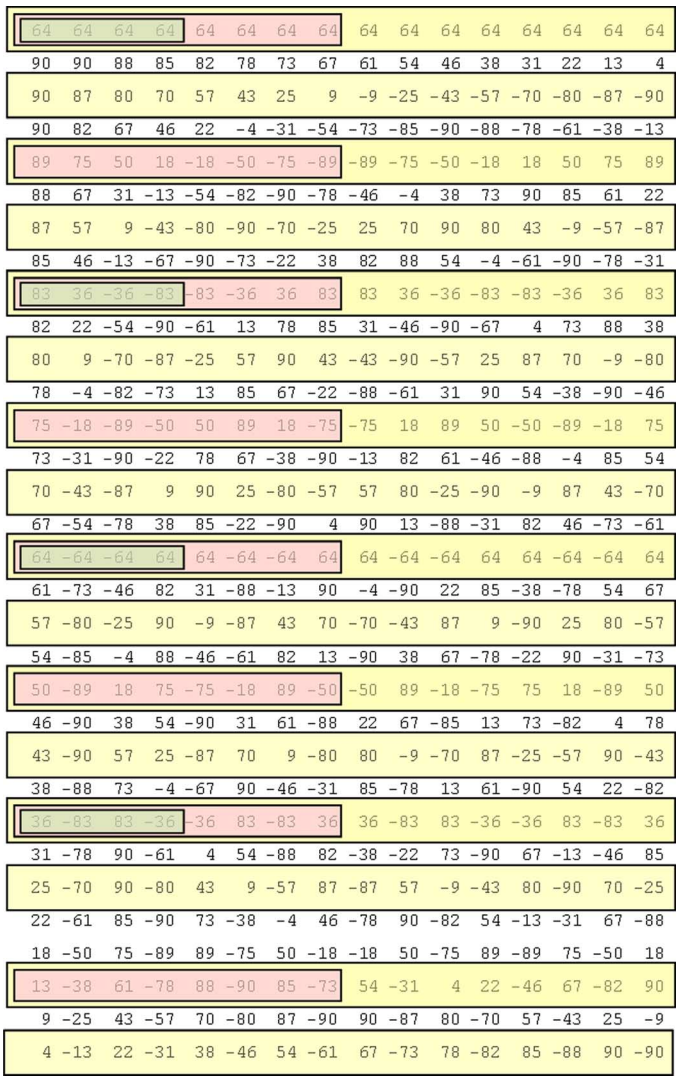

Fig. 2. Left half of the 32 × 32 matrix specifying the 32-point forward transform. Embedded 4-point (green shading), 8-point (pink shading) and 16-point (yellow shading) forward transform matrices are also shown in the figure.

the figure. The 32 × 32 matrix contains up to 31 unique numbers as follows.

$$d^{32}_{i,0},\ i = 1, \ldots, 31 = (90, 90, 90, 89, 88, 87, 85, 83, 82, 80, 78, 75, 73, 70, 67, 64, 61, 57, 54, 50, 46, 43, 38, 36, 31, 25, 22, 18, 13, 9, 4) \quad (3)$$

These unique numbers are elements 1 to 31 of the first column of the forward transform matrix. Note that although the number 90 occurs three times, this is by accident and not generally true. The unique numbers property was used in [13] to enable 25% area reduction for hardware designs with practical throughput.

Furthermore, the coefficients d^N_{ij} of the smaller transform matrices ($N = 4$, 8, 16) can be derived from the coefficients d^{32}_{ij} of the 32 × 32 transform matrix as:

$$d^N_{ij} = d^{32}_{i(32/N),j},\ i, j = 0, \ldots, N-1. \quad (4)$$

Let $\mathbf{D}_4$ denote the 4 × 4 transform matrix. By using (4) and Fig. 2, $\mathbf{D}_4$ can be obtained as:

$$\mathbf{D}_4 = \begin{bmatrix} d^{32}_{0,0} & d^{32}_{0,1} & d^{32}_{0,2} & d^{32}_{0,3} \\ d^{32}_{8,0} & d^{32}_{8,1} & d^{32}_{8,2} & d^{32}_{8,3} \\ d^{32}_{16,0} & d^{32}_{16,1} & d^{32}_{16,2} & d^{32}_{16,3} \\ d^{32}_{24,0} & d^{32}_{24,1} & d^{32}_{24,2} & d^{32}_{24,3} \end{bmatrix} = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} \quad (5)$$

The 8 × 8 transform matrix $\mathbf{D}_8$ and the 16 × 16 transform matrix $\mathbf{D}_{16}$ can be similarly obtained from the 32 × 32 transform matrix as shown in Fig. 2 where different colors are used to highlight the embedded 16 × 16, 8 × 8 and 4 × 4 forward transform matrices. This property allows for different transform sizes to be implemented using the same architecture thereby facilitating hardware sharing between different transform sizes as shown in Section III.

Note that from the unique numbers property of (3) and the (anti-)symmetry properties, $\mathbf{D}_4$ is also equal to:

$$\mathbf{D}_4 = \begin{bmatrix} d^{32}_{16,0} & d^{32}_{16,0} & d^{32}_{16,0} & d^{32}_{16,0} \\ d^{32}_{8,0} & d^{32}_{24,0} & -d^{32}_{24,0} & -d^{32}_{8,0} \\ d^{32}_{16,0} & -d^{32}_{16,0} & -d^{32}_{16,0} & d^{32}_{16,0} \\ d^{32}_{24,0} & -d^{32}_{8,0} & d^{32}_{8,0} & -d^{32}_{24,0} \end{bmatrix} \quad (6)$$

F. Intermediate Scaling

Since the HEVC matrices are scaled by $2^{(6+M/2)}$ compared to an orthonormal DCT transform, and in order to preserve the norm of the residual block through the forward and inverse two-dimensional transforms, additional scale factors – S_{T1}, S_{T2}, S_{IT1}, S_{IT2} – need to be applied as shown in Fig. 3. Note that Fig. 3 is basically a fixed point implementation of the transform and quantization in Fig. 1. While the HEVC standard specifies the scale factors of the inverse transform (i.e., S_{IT1}, S_{IT2}), the HEVC reference software also specifies corresponding scale factors for the forward transform (i.e., S_{T1}, S_{T2}). The scale factors were chosen with the following constraints:

1) All scale factors shall be a power of two to allow the scaling to be implemented as a right shift.
2) Assuming full range of the input residual block (e.g., a DC block with all samples having maximum amplitude), the bit depth after each transform stage shall be equal to 16 bits (including the sign bit). This was considered a reasonable trade-off between accuracy and implementation costs.
3) Since the HEVC matrices are scaled by $2^{(6+M/2)}$, cascading of the two-dimensional forward and inverse transform will results in a scaling of $2^{(6+M/2)}$ for each of the 1D row forward transform, the 1D column forward transform, the 1D column inverse transform, and the 1D row inverse transform. Consequently to preserve the norm through the two-dimensional forward and inverse transforms, the product of all scale factors shall be equal to $(1/2^{(6+M/2)})^4 = 2^{-24}2^{-2M}$.

The process of selecting the forward transform scale factors is illustrated using the 4 × 4 forward transform as an example in Fig. 4. When video has a bit depth of B bits, the residual will

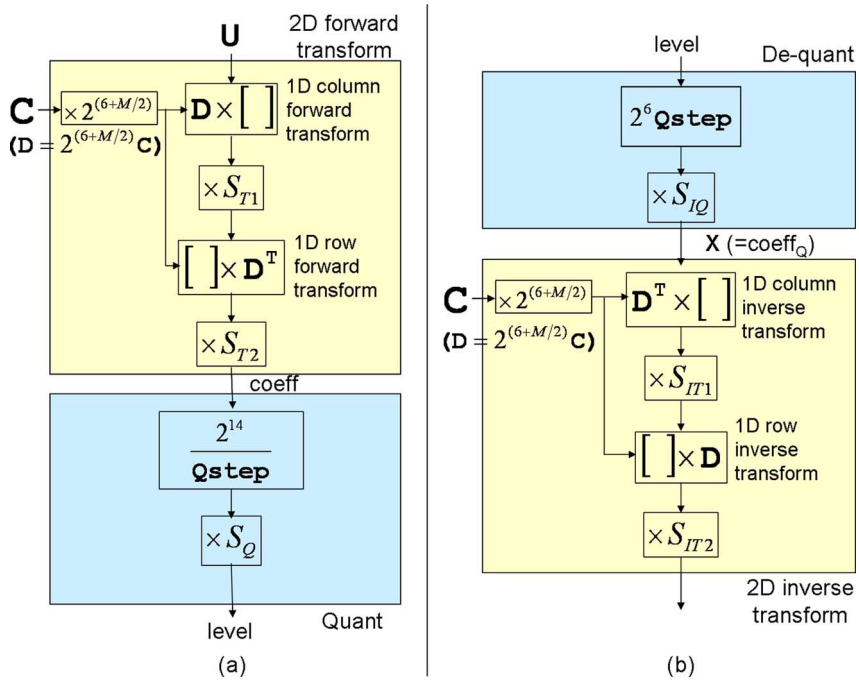

Fig. 3. Additional scale factors ($S_{T1}, S_{T2}, S_{IT1}, S_{IT2}, S_Q, S_{IQ}$) required to implement HEVC integer transform and quantization. (a) Forward transform and quantization, (b) Inverse transform and quantization. 2D forward and inverse transform is implemented as separable 1D column and row transforms. $\mathbf{C}$ is the orthonormal DCT matrix. $\mathbf{D}$ is the scaled approximation of the DCT matrix. $M = \log_2(N)$ where N is the transform size.

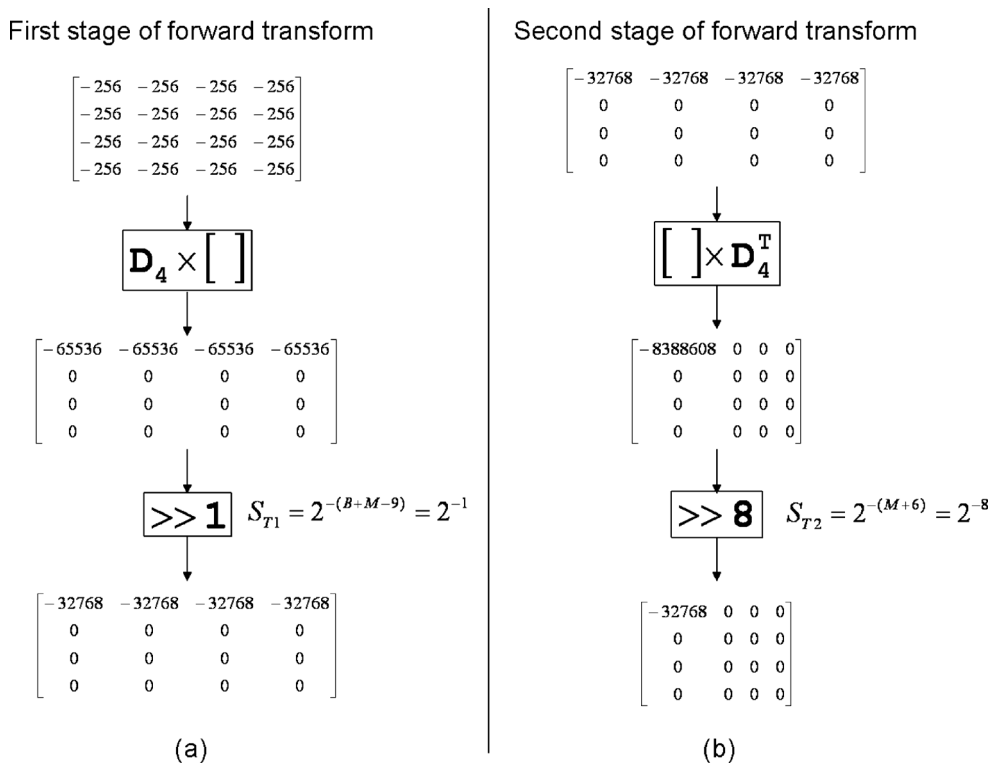

Fig. 4. Intermediate scaling factor determination for forward transform so that intermediate and output values fit within 16-bits. B is video bit depth and $M = \log_2(N)$ where N is the transform size. Worst case bit-depth analysis is done assuming a residual block with all samples having maximum amplitude equal to -2^B (where $B = 8$ is the video bit depth), as input to the first stage of the forward transform.

be in the range of $[-2^B + 1, 2^B - 1]$ requiring $(B + 1)$ bits to represent it. In the following worst case bit-depth analysis we will assume a residual block with all samples having maximum amplitude equal to -2^B as input to the first stage of the forward transform. We believe this is a reasonable assumption since all basis vectors have almost the same norm. Note also that we are using -2^B instead of $-2^B + 1$ or $2^B - 1$ in the worst case analysis since it is a power of 2. The scale factor derivation becomes simpler assuming input to be -2^B (which still fits within $(B+1)$ bits) since all the scale factors are a power of 2. For this

worst case input block, the maximum value of an output sample will be $-2^B \times N \times 64$. This corresponds to the dot product of the first basis vector (of length N with all values equal to 64) with input vector consisting of values equal to -2^B. Therefore, with $N = 2^M$, for the output to fit within 16 bits (i.e., maximum value of -2^{15}) a scaling of $1/(2^B \times 2^M \times 2^6 \times 2^{-15})$ is required. Consequently, the scale factor after the first transform stage is chosen as $S_{T1} = 2^{-(B+M-9)}$.

The second stage of the forward transform consists of multiplication of the result of the first transform stage with $\mathbf{D}_4^T$. The input into the second stage of the forward transform is the output from the first stage which is a matrix with all elements in the first row having a value of -2^{15}. All other elements will be zero as shown in Fig. 4(b). The output of multiplication with $\mathbf{D}_4^T$ will be a matrix with only a DC value equal to $-2^{15} \times 2^M \times 2^6$ and all remaining values equal to 0. This implies that the scaling required after the second stage of transform is $S_{T2} = 2^{-(M+6)}$ in order for the output to fit within 16 bits.

The first stage of the inverse transform consists of multiplication of the result of the forward transform with $\mathbf{D}_4^T$. The input into the first stage of the inverse transform is the output matrix from the forward transform which is a matrix with only the DC element equal to -2^{15}. The output of multiplication with $\mathbf{D}_4^T$ will be a matrix with first column elements equal to $-2^{15} \times 2^6$. Consequently, the scaling required after the first stage of the inverse transform for the output to fit within 16 bits is $S_{IT1} = 2^{-6}$.

The second stage of the inverse transform consists of multiplication of the result of the first stage of the inverse transform with $\mathbf{D}_4$. The input into the second stage of the inverse transform is the output matrix from the first stage of inverse transform which is a matrix with first column elements equal to -2^{15}. The output of multiplication with $\mathbf{D}_4$ will be a matrix with all elements equal to $-2^{15} \times 2^6$. So the scaling required after the second stage of inverse transform to get the output values into the original range of $[-2^B, 2^B - 1]$ is $S_{IT2} = 2^{-(21-B)}$.

In summary the constraints imposed in this section result in the following scale factors after different transform stages:

- After the first forward transform stage: $S_{T1} = 2^{-(B+M-9)}$
- After the second forward transform stage: $S_{T2} = 2^{-(M+6)}$
- After the first inverse transform stage: $S_{IT1} = 2^{-6}$
- After the second inverse transform stage: $S_{IT2} = 2^{-(21-B)}$

where B is the bit depth of the input/output signal (e.g., 8 bit) and $M = \log_2(N)$.

Without quantization/de-quantization, this choice of scale factors ensures a bit depth of 16 bit after all transform stages. However, quantization errors introduced by the quantization/de-quantization process might increase the dynamic range before each inverse transform stage to more than 16 bit. For example, consider the situation where $B = 8$ and all input samples to the forward transform are equal to 255. In this case, the output of the forward transform will be a DC coefficient with value equal to $255 \ll 7 = 32640$. For high QP values and with a quantizer rounding upwards, the input to each inverse transform stage can easily exceed the allowed 16 bit dynamic range of $[-32768, 32767]$. While clipping to 16 bit range was considered trivial after the de-quantizer, it was considered undesirable after the first inverse transform stage. In order to allow for quantization error of some reasonable magnitude and at the same time limit the dynamic range between the two inverse transform stages to 16 bit, the choice of scale factors for the inverse transform was finally modified as follows[1]:

- After the first inverse transform stage: $S_{IT1} = 2^{-7}$
- After the second inverse transform stage: $S_{IT2} = 2^{-(20-B)}$

The use of the inverse transform scale factors is illustrated in Fig. 5 using the 4×4 inverse transform as an example assuming the input to be the final output of Fig. 4.

Tables II and III summarize the different scaling factors of the forward and inverse transform respectively when compared to the orthonormal DCT.

The HEVC specification specifies an offset value to be added before scaling to carry out rounding. This offset value is equal to the scale factor divided by 2. The offset is not explicitly shown in Figs. 3–5.

Finally, two useful consequences of using 8-bit coefficients and limiting the bit-depth of the intermediate data to 16 bit is that all multiplications can be represented with multipliers having 16 bits or less and that the accumulators before right shift can be implemented with less than 32 bits for all transform stages

G. Quantization and De-Quantization

Quantization consists of division by a quantization step size ($Qstep$) and inverse quantization consists of multiplication by the quantization step size. Similar to H.264/AVC [14], a quantization parameter (QP) is used to determine the quantization step size in HEVC. QP can take 52 values from 0 to 51. An increase of 1 in QP means an increase of the quantization step size by approximately 12% (i.e., $2^{1/6}$). An increase of 6 leads to an increase in the quantization step size by a factor of 2. In addition to specifying the *relative* difference between the step-sizes of two consecutive QP values, there is also a need to define the *absolute* step-size associated with the range of QP values. This was done by selecting $Qstep = 1$ for $QP = 4$.

The resulting relationship between QP and the equivalent quantization step size for an orthonormal transform is now given by:

$$Qstep(QP) = (2^{1/6})^{QP-4} \tag{7}$$

Equation (7) can be also be expressed as:

$$Qstep(QP) = G_{QP\%6} \ll \frac{QP}{6} \tag{8}$$

where

$$\mathbf{G} = [G_0, \ldots, G_5]^T = [2^{-4/6}, 2^{-3/6}, 2^{-2/6}, 2^{-1/6}, 2^0, 2^{1/6}]^T \tag{9}$$

HEVC quantization and dequantization are basically fixed point approximations of (8). Additional scale factors S_Q and S_{IQ} as shown in Fig. 3 are introduced to restore the norm of the residual block which gets modified because of the scaling used in fixed point implementation of (8).

[1]Note that in the final HEVC specification [1], a clipping operation is introduced after the first inverse transform stage, mainly to allow for random quantization that could be used to create "evil" bitstreams used for stress testing video decoders. With the clipping introduced, the modification to the inverse transform scale factors is not necessary but has been retained in the HEVC specification and Test Model software for maturity reasons.

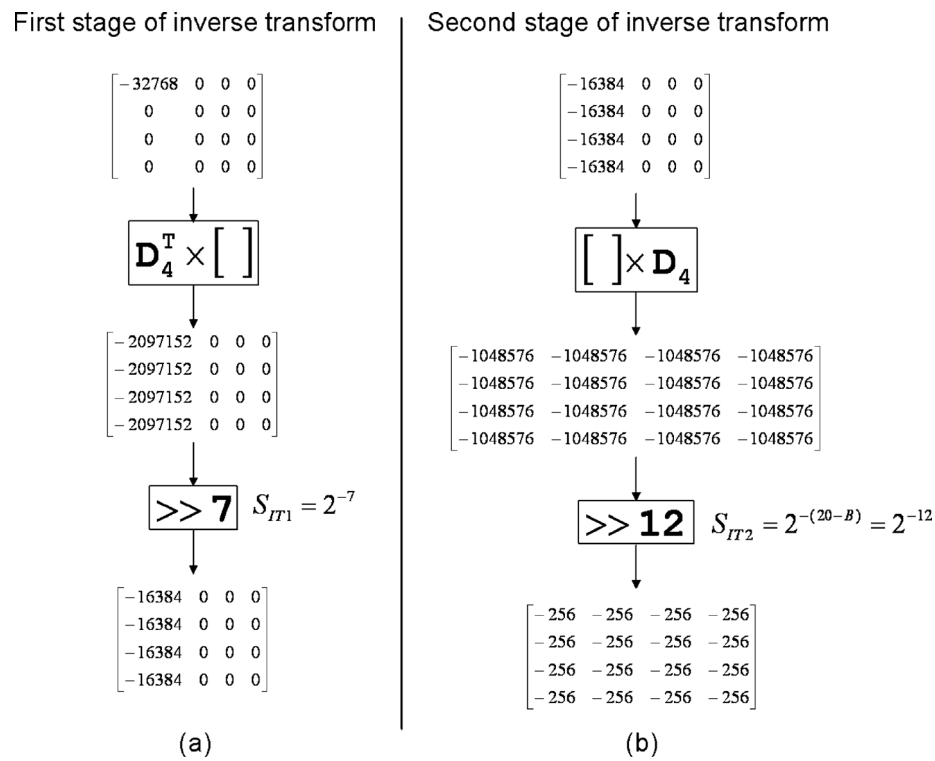

Fig. 5. Use of the inverse transform scale factors assuming the input to be the final output of Fig. 4. Video bit depth $B = 8$.

TABLE II
SCALING IN DIFFERENT STAGES FOR THE 2D FORWARD TRANSFORM.

	Scale factor
First forward transform stage	$2^{(6+M/2)}$
After the first forward transform stage (S_{T1})	$2^{-(B+M-9)}$
Second forward transform stage	$2^{(6+M/2)}$
After the second forward transform stage (S_{T2})	$2^{-(M+6)}$
Total scaling for the forward transform	$\mathbf{2^{(15-B-M)}}$

TABLE III
SCALING IN DIFFERENT STAGES FOR THE 2D INVERSE TRANSFORM.

	Scale factor
First inverse transform stage	$2^{(6+M/2)}$
After the first inverse transform stage (S_{IT1})	2^{-7}
Second inverse transform stage	$2^{(6+M/2)}$
After the second inverse transform stage (S_{IT2})	$2^{-(20-B)}$
Total scaling for the inverse transform	$\mathbf{2^{-(15-B-M)}}$

The fixed point approximation of (8) in HEVC is given by

$$g_{QP\%6} = round(2^6 \times G_{QP\%6}). \tag{10}$$

This results in

$$\mathbf{g} = [g_0, \ldots, g_5]^T = [40, 45, 51, 57, 64, 72]^T. \tag{11}$$

For a quantizer output, $level$, the de-quantizer is specified in the HEVC standard as

$$coeff_Q = \left(\left(level \times \left(g_{QP\%6} \ll \frac{QP}{6}\right)\right) + offset_{IQ}\right) \gg shift1 \tag{12}$$

where $shift1 = (M - 9 + B)$ and $offset_{IQ} = 1 \ll (M - 10 + B)$.

The scale factor S_{IQ} of Fig. 3 is equal to $2^{-shift1}$ and is obtained as follows: When $QP = 4$ (i.e., $Qstep = 1$) the combined scaling of the inverse transform and de-quantization in Fig. 3 when multiplied together should result in a product of 1 to maintain the norm of the residual block through inverse transform and inverse quantization i.e.,

$$S_{IQ} \times g_4 \times 2^{-(15-B-M)} = 1. \tag{13}$$

This results in $S_{IQ} = 2^{-(M-9+B)}$ leading to $shift1$ being equal to right shift by $(M - 9 + B)$. The scale factor $2^{-(15-B-M)}$ in (13) is obtained from Table III.

For the output sample of the forward transform, $coeff$, a straightforward quantization scheme can be implemented as follows:

$$level = \left((coeff \times f_{QP\%6} + offset_Q) \gg \frac{QP}{6}\right) \gg shift2 \tag{14}$$

where $shift2 = 29 - M - B$, and

$$\mathbf{f} = [f_0, \ldots, f_5]^T = [26214, 23302, 20560, 18396, 16384, 14564]^T \tag{15}$$

Note that $f_{QP\%6} \approx 2^{14}/G_{QP\%6}$. The value of $shift2$ is obtained by imposing similar constraints on the combined scaling in the forward transform and the quantizater as in (13), i.e., $S_Q \times f_4 \times 2^{(15-B-M)} = 1$, where $S_Q = 2^{-shift2}$.

Finally, $offset_Q$ is chosen to achieve the desired rounding.

To summarize, the quantizer multipliers, f_i, and dequantizer multipliers, g_i, were chosen to satisfy the following conditions

1) Ensure that g_i can be represented with signed 8 bit data type (i.e., $g_i < 2^7, i = 0, \ldots, 5$)
2) Ensure an almost equal increase in step size from one QP value to the next (approximately 12%) (i.e., $g_{i+1}/g_i \approx 2^{1/6}, i = 0, \ldots, 4$ and $2g_0/g_5 \approx 2^{1/6}$)
3) Ensure approximately unity gain through the quantization and de-quantization processes (i.e., $f_i \times g_i \approx 1 \ll (shift1 + shift2) = 2^6 \times 2^{14}, i = 0, \ldots, 5$)
4) Provide the desired absolute value of the quantization step size for $QP = 4$ (i.e., $level = coeff \times 2^{-(15-B-M)}$ for $QP = 4$).

III. Complexity Analysis

A. Arithmetic Operations

With straightforward matrix multiplication, the number of operations for the 1D inverse transform is N^2 multiplications and $N(N-1)$ additions. For the 2D transform, the number of multiplications required is $2N^3$ and the number of additions required is $2N^2(N-1)$. However, by utilizing the (anti-) symmetry properties of each basis vector inherited from DCT, the number of arithmetic operations can be significantly reduced. We refer to the algorithm that does this as the Even-Odd decomposition in this paper (it was also referred to as partial butterfly during HEVC development) [15]. Even-Odd decomposition is illustrated below using the 4- and 8-point inverse transform.

Consider the 4-point forward transform matrix defined in (6). For notational simplicity the constants $d^{32}_{i,0}$ of (6) will be denoted by d_i. Using the new notation (6) becomes

$$\mathbf{D}_4 = \begin{bmatrix} d_{16} & d_{16} & d_{16} & d_{16} \\ d_8 & d_{24} & -d_{24} & -d_8 \\ d_{16} & -d_{16} & -d_{16} & d_{16} \\ d_{24} & -d_8 & d_8 & -d_{24} \end{bmatrix} \tag{16}$$

The inverse transform matrix is given by $\mathbf{D}_4^T$. Let $\mathbf{x} = [x_0, x_1, x_2, x_3]^T$ be the input vector and $\mathbf{y} = [y_0, y_1, y_2, y_3]^T$ denote the output. The 1D 4-point inverse transform is given by the following equation:

$$\mathbf{y} = \mathbf{D}_4^T \mathbf{x} \tag{17}$$

The Even-Odd decomposition of the inverse transform of an N-point input consists of the following three steps:

1) Calculate the even part using a $N/2 \times N/2$ subset matrix obtained from the even columns of the inverse transform matrix. ((18) shows an example).
2) Calculate the odd part using a $N/2 \times N/2$ subset matrix obtained from the odd columns of the inverse transform matrix. ((20) shows an example).
3) Add/subtract the odd and even parts to generate N-point output. ((21) shows an example).

Even-odd decomposition of the inverse 4-point transform is given by (19) to (21):

Even part:

$$\begin{bmatrix} z_0 \\ z_1 \end{bmatrix} = \begin{bmatrix} d_{16} & d_{16} \\ d_{16} & -d_{16} \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \end{bmatrix} \tag{18}$$

The even part can be further simplified as:

$$\begin{aligned} t_0 &= d_{16} x_0 \\ t_1 &= d_{16} x_2 \\ \begin{bmatrix} z_0 \\ z_1 \end{bmatrix} &= \begin{bmatrix} t_0 + t_1 \\ t_0 - t_1 \end{bmatrix} \end{aligned} \tag{19}$$

Odd part:

$$\begin{bmatrix} z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} -d_{24} & d_8 \\ -d_8 & -d_{24} \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} \tag{20}$$

Add/sub:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} z_0 - z_3 \\ z_1 - z_2 \\ z_1 + z_2 \\ z_0 + z_3 \end{bmatrix} \tag{21}$$

The direct 1D 4-point transform using (17) would require 16 multiplications and 12 additions. The 2D transform will require 128 multiplications and 96 additions. Even-Odd decomposition on the other hand requires a total of 6 multiplications and 8 additions for 1D transform using (19)–(21). The 2D transform using Even-Odd decomposition will require a total of 48 multiplications and 64 additions which is 62.5% savings in number of multiplications and 33.3% savings in number of additions when compared to direct matrix multiplication

The 8-point 1D inverse transform is defined by the following equation:

$$\mathbf{y} = \mathbf{D}_8^T \mathbf{x}, \tag{22}$$

where $\mathbf{x} = [x_0, x_1, \ldots, x_7]^T$ is input and $\mathbf{y} = [y_0, y_1, \ldots, y_7]^T$ is output, and $\mathbf{D}_8$ is given by:

$$\mathbf{D}_8 = \begin{bmatrix} d_{16} & d_{16} & d_{16} & d_{16} & d_{16} & d_{16} & d_{16} & d_{16} \\ d_4 & d_{12} & d_{20} & d_{28} & -d_{28} & -d_{20} & -d_{12} & -d_4 \\ d_8 & d_{24} & -d_{24} & -d_8 & -d_8 & -d_{24} & d_{24} & d_8 \\ d_{12} & -d_{28} & -d_4 & -d_{20} & d_{20} & d_4 & d_{28} & -d_{12} \\ d_{16} & -d_{16} & -d_{16} & d_{16} & d_{16} & -d_{16} & -d_{16} & d_{16} \\ d_{20} & -d_4 & d_{28} & d_{12} & -d_{12} & -d_{28} & d_4 & -d_{20} \\ d_{24} & -d_8 & d_8 & -d_{24} & -d_{24} & d_8 & -d_8 & -d_{24} \\ d_{28} & -d_{20} & d_{12} & -d_4 & d_4 & -d_{12} & d_{20} & -d_{28} \end{bmatrix} \tag{23}$$

Even-odd decomposition for the 8-point inverse transform is given by (24) to (27)

Even part:

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} d_{16} & d_8 & d_{16} & d_{24} \\ d_{16} & d_{24} & -d_{16} & -d_8 \\ d_{16} & -d_{24} & -d_{16} & d_8 \\ d_{16} & -d_8 & d_{16} & -d_{24} \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} \tag{24}$$

Odd part:

$$\begin{bmatrix} z_4 \\ z_5 \\ z_6 \\ z_7 \end{bmatrix} = \begin{bmatrix} -d_{28} & d_{20} & -d_{12} & d_4 \\ -d_{20} & d_4 & -d_{28} & -d_{12} \\ -d_{12} & d_{28} & d_4 & d_{20} \\ -d_4 & -d_{12} & -d_{20} & -d_{28} \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix} \quad (25)$$

Add/sub:

$$\mathbf{y} = [z_0 - z_7, z_1 - z_6, z_2 - z_5, z_3 - z_4, z_3 + z_4, z_2 + z_5, z_1 + z_6, z_0 + z_7]^T \quad (26)$$

Note that the even part of the 8-point inverse transform is actually a 4-point inverse transform (by comparing (24) with transpose of $\mathbf{D}_4$ in (16)) i.e.,

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \mathbf{D}_4^T \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} \quad (27)$$

So the Even-Odd decomposition of the 4-point inverse transform (19)–(21) can be used to further reduce computational complexity of the even part of the 8-point transform in (24).

The direct 1D 8-point transform using (22) would require 64 multiplications and 56 additions. The 2D transform will require 1024 multiplications and 896 additions. An even-odd decomposition on the other hand requires 6 multiplications for (27) and 16 multiplications for (25) resulting in a total of 22 multiplications. It requires 8 additions for (27), 12 additions for (25) and 8 additions for (27) resulting in a total of 28 additions. The 2D transform using Even-Odd decomposition will require a total of 352 multiplications and 448 additions.

The computational complexity calculation for the 4-point and 8-point inverse transform can be extended to inverse transforms of larger size. In general, the resulting number of multiplications and additions (excluding the rounding operations associated with the shift operations) for the two-dimensional N-point inverse transform can be shown to be

$$O_{mult} = 2N\left(1 + \sum_{k=1}^{\log_2 N} 2^{2k-2}\right) \quad (28)$$

$$O_{add} = 2N\left(\sum_{k=1}^{\log_2 N} 2^{k-1}(2^{k-1}+1)\right) \quad (29)$$

The number of arithmetic operations for the inverse transform can be further reduced if knowledge about zero-valued input transform coefficients is assumed. In an HEVC decoder, this information can be obtained from the entropy decoding or de-quantization process. Furthermore, for typical video content many blocks of size $N \times N$ will have non-zero coefficients only in a $K \times K$ low frequency sub-block. For example in [16] it was found that on average around 75% of the transform blocks had non-zero coefficients only in $K \times K$ low frequency sub-blocks. Computations can be saved in two ways for such transform blocks. Fig. 6 shows the first way. Columns that are completely zero need not be inverse transformed. So only K 1D IDCTs along columns needs to be carried out. However, all N rows will need to be transformed subsequently. The second way

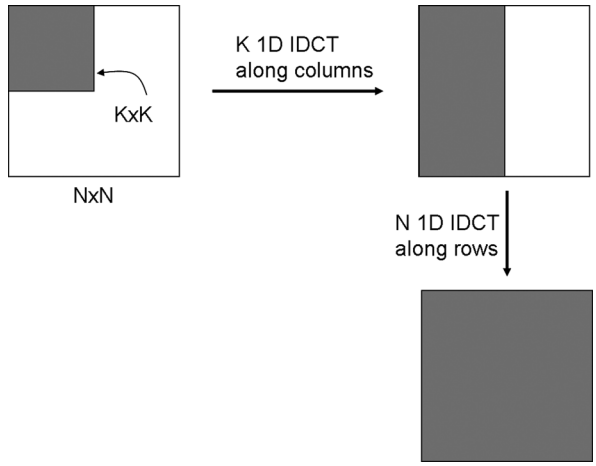

Fig. 6. Efficient implementation of inverse transform of a block with non-zero coefficients in only the $K \times K$ low frequency sub-block. Shaded regions denote the regions that can contain non-zero coefficients. Only K 1D IDCTs are required along columns.

TABLE IV
ARITHMETIC OPERATION COUNTS FOR HEVC TWO-DIMENSIONAL INVERSE TRANSFORMS.

N	K	Omult	Oadd
4	4	48	64
8	8	352	448
8	4	132	228
16	16	2752	3200
16	8	1032	1512
16	4	420	820
32	32	21888	23808
32	16	8208	10320
32	8	3400	5320
32	4	1476	3060

to reduce computations is by exploiting the fact that each of the column and row IDCT is on a vector that has non-zero values only in the first K locations. For example with $K = N/2$, $x_4 = x_5 = x_6 = x_7 = 0$, roughly half the computations for the inverse transformation can be eliminated by simplifying (24)–(25) to

Even part:

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} d_{16} & d_8 \\ d_{16} & d_{24} \\ d_{16} & -d_{24} \\ d_{16} & -d_8 \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \end{bmatrix} \quad (30)$$

Odd part:

$$\begin{bmatrix} z_4 \\ z_5 \\ z_6 \\ z_7 \end{bmatrix} = \begin{bmatrix} -d_{28} & d_{20} \\ -d_{20} & d_4 \\ -d_{12} & d_{28} \\ -d_4 & -d_{12} \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} \quad (31)$$

In general, the number of multiplications can be reduced approximately by a factor of $(N/K)^2$ for the first stage and a factor of (N/K) for the second stage. Table IV shows the number of arithmetic operations for various values of N and K.

Note that the majority of the arithmetic operations listed in Table IV can be efficiently implemented using SIMD instructions since the operations are matrix multiply operations. For example, for an 8×8 inverse transform implementation, (25)

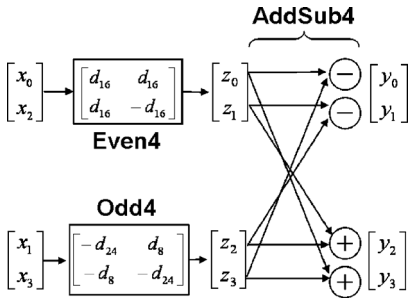

Fig. 7. 4-point inverse transform architecture.

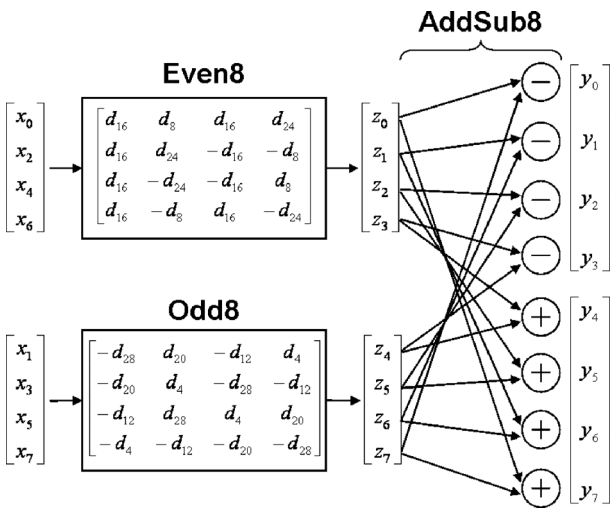

Fig. 8. 8-point inverse transform architecture.

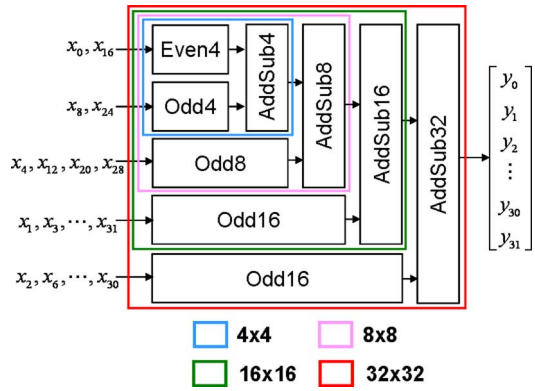

Fig. 9. 32-point inverse transform architecture.

TABLE V
AREA BREAKDOWN OF DIFFERENT COMPONENTS IN 32-POINT INVERSE TRANSFORM.

	Percent Area
Even4	0.5%
Odd4	1.2%
AddSub4	0.6%
Odd8	4.8%
AddSub8	1.4%
Odd16	17.3%
AddSub16	2.7%
Odd32	59.8%
AddSub32	4.6%
Mux, Demux, Rounding	7.1%
Total	100%

can be efficiently implemented on a 4-way SIMD processor in 4 cycles v/s 16 cycles on a processor without SIMD acceleration. Software performance using SIMD acceleration on various Intel processor architectures for the 8×8, 16×16, and 32×32 transform sizes are provided in [17] and [18].

B. Hardware Analysis

1) Inverse Transform Implementation: This sub-section describes hardware implementation of the 32-point inverse transform using the Even-Odd decomposition. It also shows how the smaller sized transforms are embedded within the larger size transforms so that hardware sharing is maximized.

Fig. 7 shows the architecture of the 4-point inverse transform which is the direct implementation of (19)–(21). The even matrix multiplication is denoted as Even4 and is implemented using (19). The odd matrix multiplication is denoted as Odd4 and implemented in (20). The outputs of the Even4 and Odd4 blocks are added and subtracted as in (21) to get the 4-point inverse transform output. The addition/subtraction network is denoted as AddSub4.

Fig. 8 shows the architecture of the 8-point inverse transform which is direct implementation of (24)–(27). The even part of the transform is exactly the 4-point inverse transform and is implemented using the architecture shown in Fig. 7. The odd part of the transform is denoted by Odd8 and implements (25). Similar to the 4-point inverse transform, the output of the even and odd parts of the transform are added and subtracted to get the 8-point inverse transform output. The addition/subtraction network is denoted as AddSub8.

The Even-Odd architecture can be extended to the 16-point and 32-point inverse transform in a similar fashion. The data flow is shown in Fig. 9. Note that the 4-point transforms are embedded within the 8-point transform which in turn is embedded in the 16-point transform and so on. Also note that the multipliers are not cascaded thereby reducing circuit delay.

The 32-point inverse transform was implemented in RTL for a throughput of one 32-point 1D transform per cycle. A 32×32 2D transform requires 64 cycles. The implementation was synthesized in 45-nm library. The 32-point inverse transform requires around 130 kGates at 250 MHz. Table V lists the area breakdown of the different components in the 32-point inverse transform implementation.

2) Unified Forward-Inverse Transform Implementation [19]: With the proliferation of products such as camera phones, tablets, video-conferencing, set-top boxes with digital video recording feature, etc., it is necessary to support video capture in addition to video playback on the same device. As a result both the forward and inverse transforms need to be implemented in the same device and techniques that can reduce the overall area of the hardware block that implements both the forward and the inverse transform are desirable. This sub-section describes one such technique that makes use of symmetry between the forward and inverse transform to share hardware between the forward and inverse HEVC transform. Examples of unified implementations for the H.264/AVC transforms can be found in [20], [21].

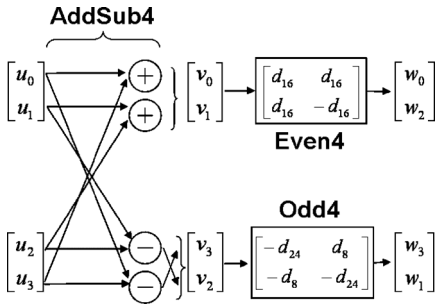

Fig. 10. 4-point forward transform architecture.

First we show the Even-Odd decomposition of the forward transform and then show the commonality in the Even-Odd decomposition of the forward and the inverse transform. This commonality is exploited to reduce the hardware area.

Let $\mathbf{u} = [u_0, u_1, u_2, u_3]^T$ be the input and $\mathbf{w} = [w_0, w_1, w_2, w_3]^T$ be the output of the 4-point 1D forward transform. The 4-point 1D forward transform is given by the following equation:

$$\mathbf{w} = \mathbf{D}_4\mathbf{u} \tag{32}$$

Even-odd decomposition of the 4-point forward transform is given by (33) to (35):

Add/sub:

$$\begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} u_0 + u_3 \\ u_1 + u_2 \\ u_2 - u_1 \\ u_3 - u_0 \end{bmatrix} \tag{33}$$

Even part:

$$\begin{bmatrix} w_0 \\ w_2 \end{bmatrix} = \begin{bmatrix} d_{16} & d_{16} \\ d_{16} & -d_{16} \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \end{bmatrix} \tag{34}$$

Odd part:

$$\begin{bmatrix} w_1 \\ w_3 \end{bmatrix} = \begin{bmatrix} -d_{24} & -d_8 \\ d_8 & -d_{24} \end{bmatrix} \begin{bmatrix} v_2 \\ v_3 \end{bmatrix} \tag{35}$$

Or equivalent odd part:

$$\begin{bmatrix} w_3 \\ w_1 \end{bmatrix} = \begin{bmatrix} -d_{24} & d_8 \\ -d_8 & -d_{24} \end{bmatrix} \begin{bmatrix} v_3 \\ v_2 \end{bmatrix} \tag{36}$$

Fig. 10 shows the architecture of the 4-point 1D forward transform implementation.

For a unified forward-inverse transform implementation, additional symmetry between the forward and the inverse transform matrices can be exploited to further reduce area. Comparing (18) and (34), it can be observed that the even matrices of the forward and the inverse transform are identical. Comparing (20) and (36), it can be observed that the odd matrices of the forward and the inverse transform are also identical. So a unified 4-point transform circuit can share hardware for even

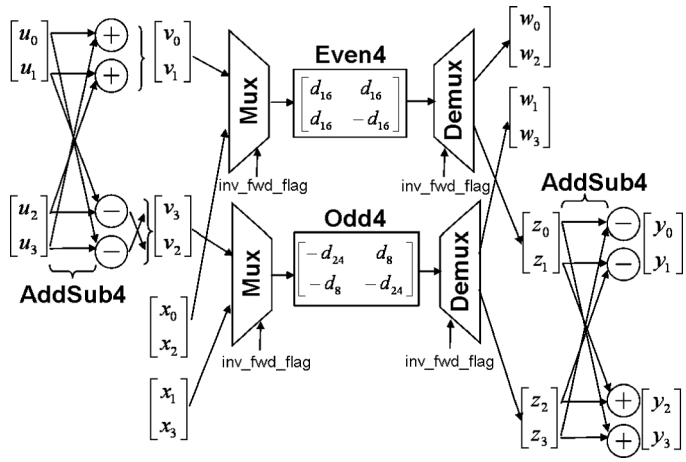

Fig. 11. Unified forward-inverse 4-point transform architecture.

TABLE VI
HEVC 32-POINT 1D TRANSFORM AREA ESTIMATE.

Forward	148 kGates
Inverse	130 kGates
Separate Fwd+Inv	278 kGates
Unified Fwd+Inv	156 kGates
%Area savings	44%

matrix and the odd matrix multiplication. The commonality between the 4-point forward and inverse transforms can also be observed by comparing Figs. 7 and 10. Fig. 11 shows the architecture of a unified forward-inverse 4-point transform with even and odd matrix sharing (labeled as Even4 and Odd4 respectively in the figure). The add-sub network on the left is used for the forward transform whereas the add-sub network on the right is used for the inverse transform. The add-sub networks are labeled as AddSub4 in Fig. 11. A control signal (inv_fwd_flag) selects whether the circuit behaves as a forward or as an inverse transform. When the inverse transform is desired, the mux/demuxes are switched down and for the forward transform, the mux/demuxes are switched up.

The 8-point, 16-point and 32-point unified transform architectures are similar to Fig. 11. More details can be found in [19].

Note that the final results of both the forward and the inverse transform are rounded before being stored. This is not explicitly shown in the figures but is included in the hardware results. The rounding circuit is also shared between forward and inverse transform.

The unified forward-inverse transform was implemented in RTL for a throughput of one 32-point 1D transform per cycle. Separate forward (and inverse) transforms were also implemented. The implementations were synthesized in 45-nm library. Table VI lists the area estimates (in kGates) at 250 MHz for separate and unified implementations. The unified implementation requires around 44% less area than a separate implementation. Hardware area savings at other frequencies are provided in [22] and are in range of 43–45%.

IV. CODING PERFORMANCE OF HEVC TRANSFORMS

The different transform sizes used in a coding block in HEVC are signaled in a quadtree structure [23]. The maximum trans-

TABLE VII
BD-RATE SAVINGS OF USING LARGER TRANSFORM SIZES (16 × 16 AND 32 × 32) ON TOP OF THE SMALLER TRANSFORM SIZES (4 × 4 AND 8 × 8).

	All Intra	Random access	Low delay B
4K	-9.1%	-10.1%	
1080p	-6.7%	-8.0%	-9.1%
WVGA	-2.5%	-4.3%	-6.0%
WQVA	-2.2%	-2.8%	-3.7%
720p	-7.7%		-8.4%
Overall	-5.6%	-6.4%	-6.8%

TABLE VIII
BD-RATE SAVINGS OF THE HEVC 4 × 4 AND 8 × 8 TRANSFORMS VERSUS THE H.264/AVC 4 × 4 AND 8 × 8 TRANSFORMS.

	All Intra	Random access	Low delay B
4K	-1.2%	-0.7%	
1080p	-0.6%	-0.4%	-0.3%
WVGA	-0.2%	-0.2%	-0.1%
WQVA	-0.1%	0.0%	-0.1%
720p	-0.5%		-0.2%
Overall	-0.5%	-0.3%	-0.2%

form size to use in a coding block is signaled in the sequence parameter set. Table VII compares the coding performance of HEVC when all transform sizes (up to 32 × 32) are used to the coding performance when only 4 × 4 and 8 × 8 transforms are used as in H.264/AVC. The standard Bjøntegaard Delta-Rate (BD-Rate) metric [24] is used for comparison. Table VII shows that there is a bit rate savings in the range of 5.6% to 6.8% on average because of the introduction of larger transform sizes (16 × 16 and 32 × 32) in HEVC. The bit rate savings are higher at larger resolution video such as 4 K (2560 × 1600) and 1080 p (1920 × 1080). The HEVC test model, HM-9.0.1 [25] was used for the simulations and the video sequences and coding conditions used were as described in [26].

Table VIII compares the coding performance of the HEVC 4 × 4 and 8 × 8 transforms to that of the corresponding H.264/AVC transforms. The H.264/AVC 4 × 4 and 8 × 8 transforms were converted to 8-bit precision and implemented in the HM-9.0.1 test model. Only the 4 × 4 and 8 × 8 transform sizes were enabled in the simulations. It can be seen from Table VIII that the HEVC 4 × 4 and 8 × 8 transforms perform better than the corresponding H.264/AVC transforms in terms of coding performance.

V. CONCLUSION

The core transform design for HEVC has been described in detail. Closeness to the DCT for finite precision approximation has been discussed as a trade-off between various measures. Implementation friendliness has been achieved by preserving symmetry properties, an embedded structure and careful limitations of the required bit depths. Finally, the implementation complexity has been discussed in terms of arithmetic operation count and hardware implementation analysis. Coding performance results show that larger transform sizes in HEVC provide significant bit rate savings especially at higher video resolutions and that the smaller transforms (4 × 4 and 8 × 8) are better than the corresponding H.264/AVC transforms in terms of coding performance.

REFERENCES

[1] "ITU-T Rec. H.265 and ISO/IEC 23008-2: High efficiency video coding," ITU-T and ISO/IEC, 2013.

[2] A. Saxena and F. Fernandes, "JCTVC-E125: CE7: Mode-dependent DCT/DST without 4 ∗ 4 full matrix multiplication for intra prediction," Joint Collaborative Team on Video Coding (JCT-VC), 2011.

[3] "ITU-T Rec. H.264 and ISO/IEC 14496-10:2009: Advanced video coding," ITU-T and ISO/IEC, 2010.

[4] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 598–603, Jul. 2003.

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

[6] T. Wiegand, W.-J. Han, J.-R. Ohm, and G. J. Sullivan, "JCTVC-C403: High efficiency video coding (HEVC) text specification working draft 1," Joint Collaborative Team on Video Coding (JCT-VC), 2010.

[7] M. Sadafale and M. Budagavi, "JCTVC-C226: Low-complexity configurable transform architecture for HEVC," Joint Collaborative Team on Video Coding (JCT-VC), 2010.

[8] M. Zhou and V. Sze, "JCTVC-C056: TE 12: Evaluation of transform unit (TU) size," Joint Collaborative Team on Video Coding (JCT-VC), 2010.

[9] A. Fuldseth, G. Bjøntegaard, M. Sadafale, and M. Budagavi, "JCTVC-G495: Core transform design for HEVC," Joint Collaborative Team on Video Coding (JCT-VC), 2011.

[10] W. Dai, M. Krishnan, J. Topiwala, P. Topiwala, and E. Alshina, "JCTVC-G266: Lossless core transforms for HEVC," Joint Collaborative Team on Video Coding (JCT-VC), 2011.

[11] R. Joshi, J. Sole, and M. Karczewicz, "JCTVC-G579: Scaled integer transform supporting recursive factorization structure," Joint Collaborative Team on Video Coding (JCT-VC), 2011.

[12] E. Alshina, A. Alshin, W. Lee, and J. Park, "JCTVC-G737: Full factorization core transforms for HEVC," Joint Collaborative Team on Video Coding (JCT-VC), 2011.

[13] M. Tikekar, C.-T. Huang, C. Juvekar, and A. Chandrakasan, "JCTVC-G265: Core transform property for practical throughput hardware design," Joint Collaborative Team on Video Coding (JCT-VC), 2011.

[14] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–570, Jul. 2003.

[15] C.-Y. Hung and P. Landman, "Compact inverse discrete cosine transform circuit for MPEG video decoding," in *Proc. IEEE SIPS*, Nov. 1997, pp. 364–373.

[16] M. Budagavi, "JCTVC-E386: IDCT pruning," Joint Collaborative Team on Video Coding (JCT-VC), 2011.

[17] F. Bossen, "JCTVC-G757: On software complexity," Joint Collaborative Team on Video Coding (JCT-VC), 2011.

[18] A. Fuldseth, L. P. Endresen, S. Selnes, V. Arbatov, F. Franchetti, and M. Puschel, "JCTVC-G497: SIMD Optimization of proposed HEVC transforms," Joint Collaborative Team on Video Coding (JCT-VC), 2011.

[19] M. Budagavi and V. Sze, "Unified forward + inverse architecture for HEVC," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2012, pp. 209–212.

[20] Y. Li, Y. He, and S. Mei, "A highly parallel joint VLSI architecture for transforms in H.264/AVC," *J. Signal Process. Syst.*, vol. 50, no. 1, pp. 19–32, Jan. 2008.

[21] W. Hwangbo and C.-M. Kyung, "A multitransform architecture for H.264/AVC high-profile coders," *IEEE Trans. Multimedia*, vol. 12, no. 3, pp. 157–167, Apr. 2010.

[22] M. Budagavi, V. Sze, and M. Sadafale, "JCTVC-G132: hardware analysis of transform and quantization," Joint Collaborative Team on Video Coding (JCT-VC), 2011.

[23] M. Winken, P. Helle, D. Marpe, H. Schwarz, and T. Wiegand, "Transform coding in the HEVC test model," in *Proc. IEEE Int. Conf. Image Process.*, 2011, pp. 3693–3696.

[24] G. Bjøntegaard, "VCEG-M33: Calculation of average PSNR differences between RD curves," ITU-T Video Coding Experts Group, 2001.

[25] HEVC Test Model HM-9.0.1 Nov. 2012 [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/ svn_HEVCSoftware/tags/HM-9.0.1/

[26] F. Bossen, "JCTVC-K1100: Common HM test conditions and software reference configurations," Joint Collaborative Team on Video Coding (JCT-VC), 2012.

Madhukar Budagavi received the B.E. degree (first class with distinction) in electronics and communications engineering from the National Institute of Technology, Trichy, India, in 1991, the M.Sc.(Eng.) degree in electrical engineering from the Indian Institute of Science, Bangalore, in 1994, and the Ph.D. degree in electrical engineering from Texas A & M University, College Station, in 1998.

From 1993 to 1995, he was with Motorola India Electronics, Ltd., developing DSP software and algorithms for Motorola DSP chips. Since 1998, he has been with the Texas Instruments (TI) Embedded Processing Systems R & D Center doing video coding, 3D graphics, and image processing research, design and implementation. From 2003 to 2007, he was also an Adjunct Assistant Professor in Southern Methodist University teaching courses in DSP and image processing to undergraduate and graduate students.

He has published over 35 journal and conference papers (including 7 book chapters) in the field of video coding, multimedia communications, DSP programming, speech coding and biomedical data compression. He has been representing TI in ITU and ISO international video coding standardization activity. His most recent participation has been in the next generation video coding standard HEVC being standardized by JCTVC committee of ITU and ISO. Within the committee he has helped coordinate core experiments and AhG activity on spatial transforms, quantization, entropy coding, in-loop filter, intra prediction, screen content coding and scalable HEVC (SHVC). He is a Senior Member of IEEE.

Arild Fuldseth received his B.Sc. degree from the Norwegian Institute of Technology in 1988, his M.Sc. degree from North Carolina State University in 1989, and his Ph.D. degree from Norwegian University of Science and Technology in 1997, all degrees in Signal Processing.

From 1989 to 1994, he was a Research Scientist in SINTEF, Trondheim, Norway. From 1997 to 2002 he was a Manager of the signal processing group of Fast Search and Transfer, Oslo, Norway. Since 2002 he has been with Tandberg Telecom, Oslo, Norway (now part of Cisco Systems) where he is currently a Principal Engineer working with video compression technology.

Gisle Bjøntegaard received the Dr. Philos degree in physics from University of Oslo, Norway in 1974.

From 1974 to 1996 he worked as senior scientist with Telenor Research and Development in Oslo, Norway. The areas of work included: radio link network design, reflector antenna design and construction, and digital signal procession. The main area of work from 1980 was development of digital video compression methods. He contributed actively in development of the ITU video standards H.261, H.262, H,263, H.264 as well as ISO/IEC MPEG2 and MPEG4. From 1996 to 2002 he worked as a group manager at Telenor Broadband Services in Oslo, Norway. Work areas included design of point to point satellite communication and development of digital satellite TV platform. He produced numerous contributions towards the development of the ITU-T standards H.263 and H.264. From 2002 he worked as principal scientist at Tandberg Telecom in Lysaker, Norway working with video coding development and implementation. Since 2006 he has worked on further improvement of digital video coding and is at present taking part in the definition of HEVC developed jointly between ITU and ISO. In 2010 Tandberg Telecom was acquired by Cisco Systems and he was promoted to Cisco Fellow and is presently working with Cisco Systems Norway.

Vivienne Sze received the B.A.Sc. (Hons) degree in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 2004, and the S.M. and Ph.D. degree in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 2006 and 2010 respectively. She received the Jin-Au Kong Outstanding Doctoral Thesis Prize, awarded for the best PhD thesis in electrical engineering in 2011.

Since September 2010, she has been a Member of Technical Staff in the Embedded Processing Systems R & D Center at Texas Instruments (TI), Dallas, TX, where she designs low-power algorithms and architectures for video coding. She also represents TI at the international JCT-VC standardization body developing HEVC, the next generation video coding standard. Within the committee, she is the primary coordinator of the core experiment on coefficient scanning and coding.

She was a recipient of the 2007 DAC/ISSCC Student Design Contest Award and a co-recipient of the 2008 A-SSCC Outstanding Design Award. She received the Natural Sciences and Engineering Research Council of Canada (NSERC) Julie Payette fellowship in 2004, the NSERC Postgraduate Scholarships in 2005 and 2007, and the Texas Instruments Graduate Woman's Fellowship for Leadership in Microelectronics in 2008.

Mangesh Sadafale received BE degree in electronics engineering from Nagpur University, Nagpur, India in 1998 and ME degree in microelectronics from BITS, Pilani, India in 2000.

From 2000 to 2011, he worked at different groups in Texas Instruments, India where he designed and developed synthesizable (soft) c64x+ datapath and different signal processing hardware accelerators for DSL and video. He also represented TI at the international JCT-VC standardization body developing HEVC. In 2011 he was co-founder at Signalchip Innovations, India.