www.ietdl.org

Published in IET Image Processing
Received on 3rd April 2014
Revised on 16th October 2014
Accepted on 16th October 2014
doi: 10.1049/iet-ipr.2014.0277

ISSN 1751-9659

Hardware architectures for the H.265/HEVC discrete cosine transform

Grzegorz Pastuszak

Institute of Radioelectronics, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland
E-mail: G.Pastuszak@ire.pw.edu.pl

Abstract: This study presents a design methodology for the two-dimensional (2D) discrete cosine transform dedicated for H.265/HEVC hardware encoders. The methodology decomposes matrix multiplications for different transform sizes into some steps based on the division of transform units into fixed-size blocks. The modified order of processed blocks allows a significant reduction of the size of the transposition buffer. As a consequence, the resource consumption of the whole 2D-transform architecture is decreased. Separate transform cores assigned to two transform stages increase the throughput more than twice. The decomposition enables different hardware configurations of the architectures. Particularly, the architectures applying the proposed methodology are parametrically specified in VHDL, and configuration parameters enable the tradeoff between resources and the throughput. Furthermore, the interface adaptation to desired horizontal and vertical sizes is possible. The use of regular multipliers allows the support for transforms specified in other video standards. Computational elements embedded in architectures are well-suited to FPGA devices, which improves the area-speed efficiency. Synthesis results show that they can operate at 200 and 400 MHz when implemented in FPGA Arria II and TSMC 90 nm, respectively.

1 Introduction

The latest research and standardisation efforts within Joint Collaborative Team – Video Coding led to the specification of the H.265/HEVC (high-efficiency video coding) standard in 2013 [1–4]. It significantly improves the rate-distortion efficiency as compared to its predecessor H.264/AVC (advanced video coding) [5]. However, the improvement involves the increased computational complexity. In the case of the transform stage, H.265/HEVC specifies larger block sizes. In particular, 16×16 and 32×32 transforms can be computed apart from 4×4 and 8×8 transforms. The larger transform units require the increased size of the transposition buffer used to store one-dimensional (1D)-transformed coefficients. In the hardware framework, the computation of such a transform requires a significant amount of resources. H.265/HEVC applies integer basis functions (specified in transform matrices) which better approximate original discrete cosine transform (DCT). The extended range of values included in the transform matrix (from −90 to 90) involves the use of multipliers. For 4×4 intra predictions, H.265/HEVC exploits the integer approximation of discrete sine transform.

Highly-efficient video coding involves a great amount of computations. Hardware encoders apply a number of parallelisation techniques to satisfy real-time requirements. In the literature, there are some architectures developed to accelerate the DCT/IDCT (inverse DCT) computation for H.265/HEVC [6–20]. Some of them do not support all transform sizes [10–12, 14, 18, 20]. All the designs embed (or assume [10, 17]) the full-size transposition buffer (32×32 samples) implemented either as a register matrix [6–9, 11, 16, 18] or memory modules [8, 13, 15, 19, 20]. In the either case, the buffer contributes a significant amount of hardware resources. For example, the reported logic area of the register implementation is equivalent to 139 k [6], 126 k [8], 183 k [14] or 125 k [15] basic gates. As the designs are dedicated to specific throughputs, they cannot trade resources for the throughput. Implementations embedding regular multipliers and the full-size transposition buffer involve a large amount of hardware resources. To save them, some designs implement multiplications with constant multipliers [6, 10, 11, 15, 16]. This approach is area-efficient when the output contains the number of samples/coefficients equal to the 1D transform size (i.e. 32) [13]. On the other hand, this output format is inconvenient and requires additional registers for the adaptation to smaller sizes. In the referenced architecture, the support for transform specified in other video standards requires additional design efforts and significantly increases the resource consumption [13]. All the referenced designs are not efficient in terms of FPGA devices because of a great number of cascaded additions/subtractions and the large transposition buffer. These features lead to the high occupancy of general-purpose logic and long critical paths [18, 19].

To allow fast and computationally-configurable architectures with the reduced transposition buffer, the novel design methodology for DCT is proposed. The methodology uses regular multiplication units and applies the decomposition of the matrix multiplication into steps performed in successive clock cycles on fixed-size blocks. The number of arithmetic operations performed in each clock cycle is configurable at the VHDL level. Hence, different architectures can be generated with the tradeoff between logic resources and the throughput. In addition, input/output interfaces can be adapted to various block sizes corresponding to the throughput. Other configuration options enable the selection of transform sizes/types specified by the H.265/HEVC standard and the extension to other video standards. The assumed order of blocks computed in the pipeline at both transform stages (vertical and horizontal) allows the decreased size of the transposition buffer. In particular, the gate-equivalent area of the buffer is reduced by 54–88% depending on the selected configuration. As a consequence, the architecture outperforms other designs in terms of the area–speed efficiency. This advantage is especially evident in the case of FPGA devices. Compared to other designs, the achieved latencies are the lowest.

The rest of the paper is organised as follows: Section 2 reviews basic algorithms used for DCT. Sections 3 and 4 describe the proposed methodology and architecture design, respectively. Implementation results are given in Section 5. The comparison with other architectures is provided in Section 6. Finally, the paper is concluded in Section 7.

2 DCT basics

The 1D horizontal forward transform of the $N \times N$ block $\boldsymbol{X}$ can be written as follows

$$Y(i, j) = \sum_{k=0}^{N-1} C(j, k) \times X(i, k) \tag{1}$$

where $\boldsymbol{C}$ and $\boldsymbol{Y}$ are the transform matrix and the result, respectively. This operation involves N^3 multiplications, where N is the transform size. For H.265/HEVC, N can be equal to either 4, 8, 16 or 32. The algorithm used to compute the 2D transform according to (1) is presented as Algorithm 1 (see Fig. 1) below. In DCT, even and odd rows in the transform matrix $\boldsymbol{C}$ are symmetric and anti-symmetric, respectively. This property is usually utilised to decompose the transform formula into two cases corresponding to even and odd positions

$$Y(i, 2j) = \sum_{k=0}^{N/2-1} C(2j, k) \times [X(i, k) + X(i, N-k-1)] \tag{2}$$

$$Y(i, 2j+1) = \sum_{k=0}^{N/2-1} C(2j+1, k) \times [X(i, k) - X(i, N-k-1)] \tag{3}$$

where $j = 0, 1, \ldots, N/2-1$.

This type of the decomposition (using butterfly structures at the input stage) reduces the number of multiplications by half. The algorithm used to compute the 2D transform with the one butterfly stage is presented as Algorithm 2 (see Fig. 2). In the case of the original DCT/IDCT, the transform can be recursively decomposed according to Chen's [21] scheme to reduce the number of multiplications. This scheme cannot be directly applied in H.265/HEVC. In particular, the scheme takes advantage of the orthogonality of cosine functions whereas the standard specifies their integer (not orthogonal) approximation. The standard specifies the 32×32 matrix which is directly applied to compute 32-point transform. Smaller transforms are computed with subsampled versions of the matrix. The 2D-DCT is obtained in two steps. In the first step, 1D transform is applied on each input row. In the second step, the 1D transform is applied to each column of the result of the first step. Owing to the increased size of the transforms, the dynamic range after both stages is limited by downshifting.

Algorithm 1

```
// Horizontal transform
For (i=0; i < N; i++) // for each row
  For (j=0; j < N; j++)
    Y(i, j) = 0;
    For (k=0; k < N; k++)
      Y(i, j) += C(j, k) × X(i, k) ;
// Vertical transform
For (j=0; j < N; j++) // for each column
  For (i=0; i < N; i++)
    Z(i, j) = 0;
    For (k=0; k < N; k++)
      Z(i, j) += C(i, k) × Y(k, j) ;
```

Fig. 1 *General 2D transform*

Algorithm 2

```
// Horizontal transform
For (i=0; i < N; i++) // for each row
  For (j=0; j < N/2; j++)
    Y(i, 2j) = 0;
    Y(i, 2j+1) = 0;
    For (k=0; k < N/2; k++)
      Y(i, 2j) += C(2j, k) × [X(i, k) + X(i, N-k-1)];
      Y(i, 2j+1) += C(2+1, k) × [X(i, k) - X(i, N-k-1)];
// Vertical transform
For (j=0; j < N; j++) // for each column
  For (i=0; i < N/2; i++)
    Z(2i, j) = 0;
    Z(2i+1, j) = 0;
    For (k=0; k < N/2; k++)
      Z(2i, j) += C(2i, k) × [Y(k, j) + Y(N-k-1, j)];
      Z(2i+1, j) += C(2i+1, k) × [Y(k, j) - Y(N-k-1, j)];
```

Fig. 2 *2D transform algorithm with the butterfly*

3 Methodology

The main novelty of the proposed methodology is the combination of three elementary techniques to reduce the size of the transposition buffer. These techniques include the decomposition, the modified order of processed blocks, and the parallel processing at two transform stages. They are described in the following subsections.

3.1 Decomposition

The 1D transform is the particular case of the matrix–matrix multiplication, which can be described by the three-level nested loops. The proposed decomposition scheme decomposes each loop into smaller parts, and the common loop body performs the computations of one partitioned data cube (each cube dimension corresponds to one loop). This approach is basically similar to the partitioning scheme developed for the systolic array design, where different mappings (projection and partitioning) can be applied to obtain different architectures. The computation of the 1D

Algorithm 3

```
// Horizontal transform
For (ii=0; ii < N; ii+=H) // for each block row
  For (jj=0; jj < N; jj+=Wo)
    For (kk=0; kk < N/2; kk+=Wi)
      For (i=0; i < N/H; i++) // for each row in block
        For (j=0; j < N/(2Wo); j++)
          Y(ii+i, jj+2j) = 0;
          Y(ii+i, jj+2j+1) = 0;
          For (k=0; k < N/(2Wi); k++)
            Y(ii+i, jj+2j) += C(jj+2j, kk+k) × [X(ii+i, kk+k) + X(ii+i, N-kk-k-1)];
            Y(ii+i, jj+2j+1) += C(jj+2j+1, kk+k) × [X(ii+i, kk+k) - X(ii+i, N-kk-k-1)];
// Vertical transform
For (jj=0; jj < N; jj+=W) // for each block column
  For (ii=0; ii < N; ii+=Ho)
    For (kk=0; kk < N/2; kk+=Hi)
      For (j=0; j < N/W; j++) // for each column in block
        For (i=0; i < N/(2Ho); i++)
          Z(ii+2i, jj+j) = 0;
          Z(ii+2i+1, jj+j) = 0;
          For (k=0; k < N/(2Hi); k++)
            Z(ii+2i, jj+j) += C(ii+2i, kk+k) × [Y(kk+k, jj+j) + Y(N-kk-k-1, jj+j)];
            Z(ii+2i+1, jj+j) += C(ii+2i+1, kk+k) × [Y(kk+k, jj+j) - Y(N-kk-k-1, jj+j)];
```

Fig. 3 *2D transform algorithm after the decomposition*

transform is susceptible to parallelise in three ways. Firstly, a part of input coefficient/sample contributions to each output can be accumulated in one step. Secondly, a number of accumulators can be incorporated, each of which is assigned to one coefficient output. Thirdly, some rows/columns can be processed simultaneously because of their independence. By combining the three parallelisms, the 1D transform can be decomposed into some steps/phases as shown in Algorithm 3 (see Fig. 3). The decomposition is specified by block sizes W (width) and H (height). Subscripts 'i' and 'o' indicate that the width/height corresponds to either input or output interface of a given 1D transform. If there is no index, the width/height corresponds to both interfaces. Apart from the parallelism, the width/height parameters also determine the size of the interfaces. Therefore, the modification of the parameters enables the interface adaptation. For a given 1D transform, three parameters are set. One of them determines the size of both input and output interfaces, whereas the remaining two correspond to one interface. As a consequence, the block height of the horizontal transform and the block width of the vertical transform are selected jointly for the input and output interface.

The decomposition divides input and output domains into blocks as shown in Fig. 4. Owing to the butterfly operation performed for both 1D transforms, the input block consists of two subblocks located symmetrically around the horizontal/vertical middle axis. In each step, one input block is accessed to compute the partial sum for one output block. Partial sums are accumulated in successive steps. In the horizontal transform, each output block is computed from input blocks located in the corresponding row. For example, the output block indexed by (Y,1,2) in Fig. 4*b* is computed from eight input subblocks with indices (0,1,X), (1,1,X), (2,1,X) and (3,1,X) in Fig. 4*a*. If the width of input blocks is small, more steps are required to obtain the output block. The decomposition allows a fixed number of arithmetic operations performed in one step/phase. Particularly, the total number of multiplications is equal to $H \times W_i \times W_o$ and $W \times H_i \times H_o$ for the horizontal and vertical transform, respectively. If the larger transform is computed, more steps are required.

3.2 Order control

The decomposition described in the previous subsection allows the computation of successive blocks at each transform stage. However, the traditional order of computed blocks (the raster order for the horizontal transform) requires the full-size $N \times N$ transposition buffer to save 1D-transformed coefficients $Y(i, j)$. Furthermore, the processing for the vertical transform cannot be started if the processing for the horizontal one is not finished. To overcome this limitation, the processing order is modified, as presented in Algorithm 4 (see Fig. 5). Firstly, both transform stages (horizontal and vertical) are controlled by the same external loop determining the block column. Secondly, the width of the output block at both stages is determined by the same parameter W_o (the vertical

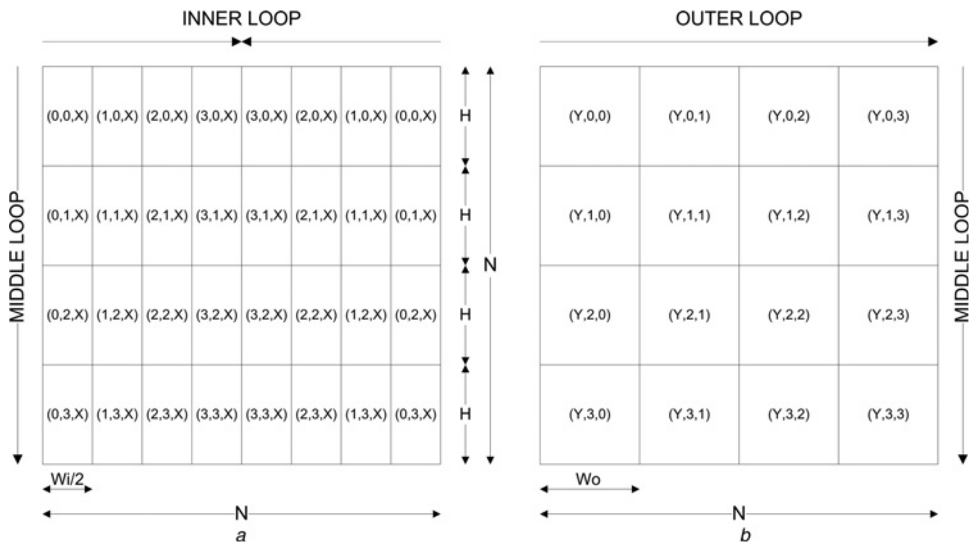

Fig. 4 *Example of the partitioning of the input and output domains into blocks*

a Input domain partitioning
b Output domain partitioning
Numbers indicate the loop iteration in which a block is accessed/computed
The first, the second and the third index correspond to the iteration of the inner, the middle and the outer loop, respectively

Algorithm 4

```
// Horizontal transform
For (jj=0; jj < N; jj+=Wo) // external loop
  For (ii=0; ii < N; ii+=H) // middle loop
    For (kk=0; kk < N/2; kk+=Wi) // inner loop
      For (i=0; i < N/H; i++)
        For (j=0; j < N/(2Wo); j++)
          Y(ii+i, jj+2j) = 0;
          Y(ii+i, jj+2j+1) = 0;
          For (k=0; k < N/(2Wi); k++)
            Y(ii+i, jj+2j) += C(jj+2j, kk+k) × [X(ii+i, kk+k) + X(ii+i, N-kk-k-1)];
            Y(ii+i, jj+2j+1) += C(jj+2j+1, kk+k) × [X(ii+i, kk+k) - X(ii+i, N-kk-k-1)];
// Vertical transform
  For (ii=0; ii < N; ii+=Ho) // middle loop
    For (kk=0; kk < N/2; kk+=Hi) // inner loop
      For (j=0; j < N/Wo; j++)
        For (i=0; i < N/(2Ho); i++)
          Z(ii+2i, jj+j) = 0;
          Z(ii+2i+1, jj+j) = 0;
          For (k=0; k < N/(2Hi); k++)
            Z(ii+2i, jj+j) += C(ii+2i, kk+k) × [Y(kk+k, jj+j) + Y(N-kk-k-1, jj+j)];
            Z(ii+2i+1, jj+j) += C(ii+2i+1, kk+k) × [Y(kk+k, jj+j) - Y(N-kk-k-1, jj+j)];
```

Fig. 5 *2D transform algorithm with the modified order*

transform has $W_\text{i} = W_\text{o}$). The modifications allow the reduction of the transposition buffer since only one block column ($W_\text{o} \times N$ registers) is required to store 1D-transformed coefficients during each iteration of the external loop. In the case of the horizontal transform, the inner loop determines horizontal coordinates of accessed input subblocks (see Fig. 4*a*). Vertical and horizontal coordinates of output blocks are determined in the middle and outer loop, respectively (see Fig. 4*b*). In the horizontal processing, the middle loop also controls the vertical position of the input subblocks. The order determined by the loops enables the processing of successive columns of output blocks.

The throughput of both transform stages should be balanced to achieve the best hardware utilisation. Since one parameter (W_o) determining the throughput is common for the two stages, multiplications of the remaining two parameters featuring each stage ($H \times W_\text{i}$ for the horizontal transform and $H_\text{i} \times H_\text{o}$ for the vertical transform) should give the same result. The selection of parameters greater than the transform size N would lead to the hardware inefficiency. This stems from the fact that the only N inputs are required to compute each of N outputs in the 1D transform. On the other hand, small parameter values decreases the throughput. It is convenient to set the input width (W_i) for the horizontal transform and the input height (H_i) for the vertical transform to 8. This setting limits the hardware efficiency/utilisation only for the 4×4 transform, whereas the high throughput can be achieved. In the case of FPGA devices, the fixing of the parameters to 8 facilitates the fitting to DSP units, which improves the utilisation of hardware resources.

The number of iterations of each loop is proportional to the transform size N and inversely proportional to the width or height of input/output blocks. For the horizontal and vertical transform, the total number of iterations can be expressed as $N^3/(W_\text{o} \times H \times W_\text{i})$ and $N^3/(W \times H_\text{o} \times H_\text{i})$, respectively. In reality, the number of iterations must not be less than one, that is, at least one iteration is indispensable to perform the transformation. However, the expression can give the result smaller than one if the transform size is small and block sizes are large. It means that resources are not utilised fully for computations. To achieve a better utilisation, one larger (e.g. 8×8) block composed of four smaller (e.g. 4×4) transform blocks should be processed. The parallel processing of smaller blocks involves the reorganisation of subblocks directed to the butterfly stage since subblocks determined for larger transforms do not correspond to those for smaller transforms. Generally, transforms with different sizes can be computed on fixed-size blocks by changing multiplication coefficients. Moreover, the implementation of other transform matrices enables the support for other video standards such as H.264/AVC, AVS, VC-1 and so on.

3.3 *Parallel processing at two transform stages*

The sequential processing of two 1D transforms in the same engine decreases the throughput because of its delay. In particular, the vertical processing must wait until the horizontal one writes all results to the transposition buffer. Furthermore, the average throughput of the folded 2D processing is reduced to a half of that achieved for the 1D-transform engine. The delay of the engine additionally decreases the throughput. These properties are important if the throughput of the engine is limited. The folded 2D processing has a negative impact on the hardware utilisation of the quantisation which does not receive coefficients during the horizontal processing. Therefore the proposed methodology assumes the use of separate pipelined engines for both transform stages.

The diagram of 2D processing is depicted in Fig. 6. Each column of blocks is forwarded to vertical processing immediately after the horizontal engine finishes the computation of the last block in the column. In order to avoid read/write conflicts, the horizontal and vertical engines must operate on separate parts of the transposition buffer in the parallel. Whenever a column of blocks is written to the buffer and the vertical engine is ready, assignments of the two parts are exchanged according to the ping-pong scheme. As a consequence, the buffer should keep two columns of blocks, and its width is increased to $2 \times W_\text{o}$. Provided that the block sizes are powers of two, the proposed methodology has the advantage for W_o not greater than $N/4$. For W_o equal to $N/2$, the transposition buffer has the size the same as in the traditional approach.

4 Architecture

The general DCT architecture applying the proposed methodology and its timing diagram is depicted in Figs. 7 and 8, respectively. The module reads residuals from two memory buffers which can keep a variable number of different-size transform/prediction blocks. The architecture consists of three main submodules: the horizontal transformation, the transposition buffer and the vertical transformation. Transformation engines are pipelined to maximise the clock rate. The first stage performs the butterfly operation. The second stage embeds multiplications, whereas the third includes the adder tree with the accumulator. The fourth stage consists of the normalisation downshifter and the rounding adder. The adder is not present in the ASIC version since the accumulator is initialised with non-zero value corresponding to the rounding fraction (1/2). Apart from the arithmetic operations, the transformation submodules embed the control logic. It provides coordinates for input/output blocks according to the order determined by three nested loops, as described in the previous section. The horizontal transform engine reads residuals as $W_\text{i} \times H$ blocks from two external

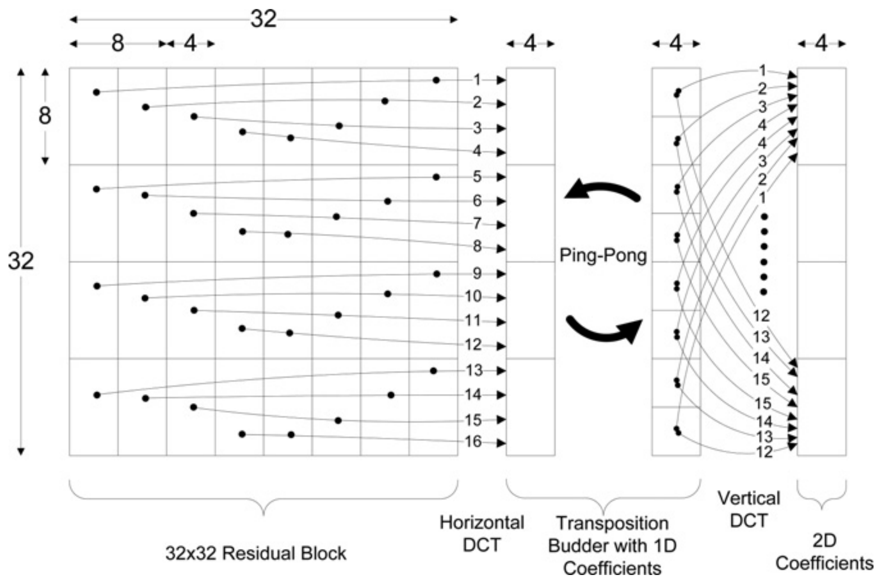

Fig. 6 *Column-oriented computation of the 2D 32 × 32 DCT*
Input domain is divided into 4 × 8 subblocks read subsequently
Pair of subblocks is read in one step
Arrows with numbers indicate the order of subblock contributions
Output subblock size at both stages is set to 4 × 8

memory buffers. Two buffers are indispensible to provide two generally-not-adjacent subblocks ($W_i/2 \times H$) required by the input butterfly operation. Horizontally-transformed $W_o \times H$ blocks are written to the first register queue in the transposition buffer. The number of writes for one row is equal N/H. Fig. 7 shows the case for four writes. Both register queues have the width of W_o. When processing of one column of blocks ($W_o \times N$) is finished and the vertical transform engine is ready, the content of the first queue is written to the second one. Next, the horizontal transformation starts to write data from the following column of blocks to the same queue, whereas the vertical transformation reads the second queue. The second queue is composed of two parts providing pairs of subblocks ($W \times H_i/2$) to the butterfly operation. Since the same data have to be accessed for vertically-consecutive output blocks, the second queue is designed as two ring buffers including four register sections. As a consequence, four read accesses are needed to compute one 2D-transformed block ($W_o \times H$) for 32 × 32 transform. For smaller transform sizes, only a part of registers is used, and the number of reads is decreased. 2D-transform coefficients produced by the vertical processing are released from the transform module and forwarded to the quantisation module. Provided that the same clock drives both modules, the quantisation module should embed $W_o \times H_o$ parallel quantisers.

The architecture of the 1D transformation dedicated for ASIC designs is depicted in Fig. 9. Although $W_i/2$ is fixed to 4, it can be changed if needed. The input stage includes two layers of multiplexers. The first layer exchanges two input subblocks when the transform size is larger ($N = 16$ or 32) and the block has the odd horizontal index (assumed

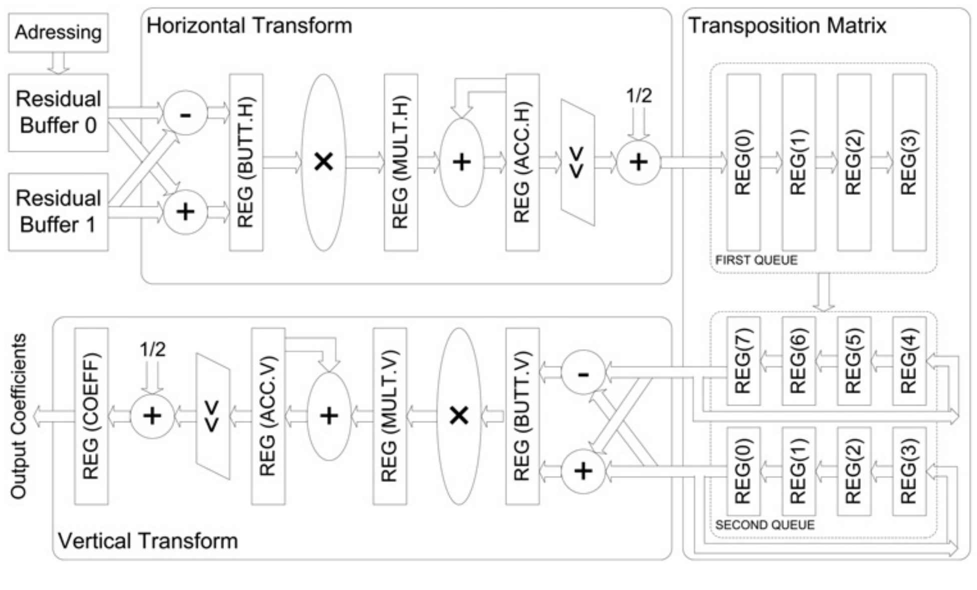

Fig. 7 *General DCT architecture*

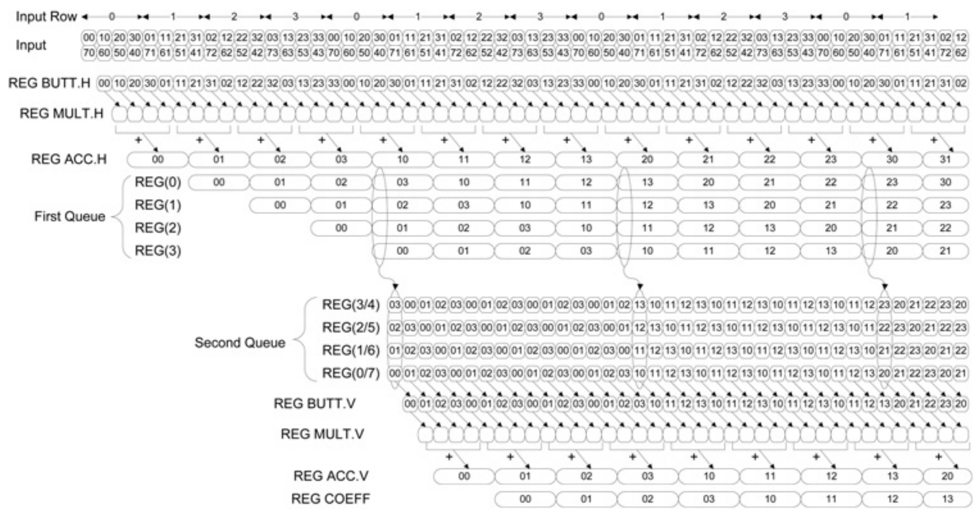

Fig. 8 *Timing diagram of the 2D 32 × 32 transform for the ASIC implementation*

Blocks are indexed by horizontal and vertical coordinates
Input, queue, and residual indices corresponds to input, 1D-transformed, and residual blocks, respectively
For clarity, indices identify only upper blocks in the second queue

horizontal transform). The exchange allows a fixed assignment of particular subblocks to two memory buffers for any transform size. The second layer of multiplexers reconfigures the circuit to compute two 4-point transforms (T4) instead of one larger. Note that subblocks for the 4-point transform have the width equal to 2 rather than 4. The 1D engine incorporates $W_o \times H_o$ DSP-4 units, each of which embeds four multiplications, three adders and one accumulator. The operations are performed at two pipeline stages. However, the stages can be merged at the cost of the decreased maximal frequency. The rounding is implemented at the accumulation stage by providing an appropriate offset to the adder in the first cycle (phase). Totally, the 2D-transform module embeds $8 \times W_o \times \mathrm{H}$ multipliers and $8 \times W_o \times H + 16 \times H$ adders/subtractors.

In FPGA technology, there are modules dedicated for typical DSP operations such as multiplications and accumulations. They allow designs to achieve higher performance in terms of speed and resources. Particularly, the modules can be pipelined to obtain higher frequencies. Since the transformation involves a great amount of multiplications and accumulations, it is beneficial to utilise DSP modules while implementing the design in FPGA devices. However, the architecture optimised for ASIC technology in not always suitable to obtain the best performance in FPGA, and vice versa. Moreover, the same rule can apply to FPGA families from different vendors.

The architecture of the 1D transformation using Altera DSP modules is depicted in Fig. 10. $W_i/2$ is fixed to 4 to fit FPGA resources. There are three main differences compared to the ASIC-optimised design: additional registers before multipliers, the moving of some (first-level) adders to the first stage (just after multiplications), and the use of separate adders for the final rounding. These modifications enable fitting to the structure of DSP modules available in Altera devices. Input registers embedded in DSP eliminate the impact of signal delay inferred from routing. Totally, the 2D transform implemented in FPGA embeds $2 \times W_o \times H$ DSP units and $2 \times W_o \times H + 16 \times H$ adders/subtractors.

The proposed architecture can be reconfigured to support different transform sizes and transform matrices by the selection of appropriate coefficients used in multiplications.

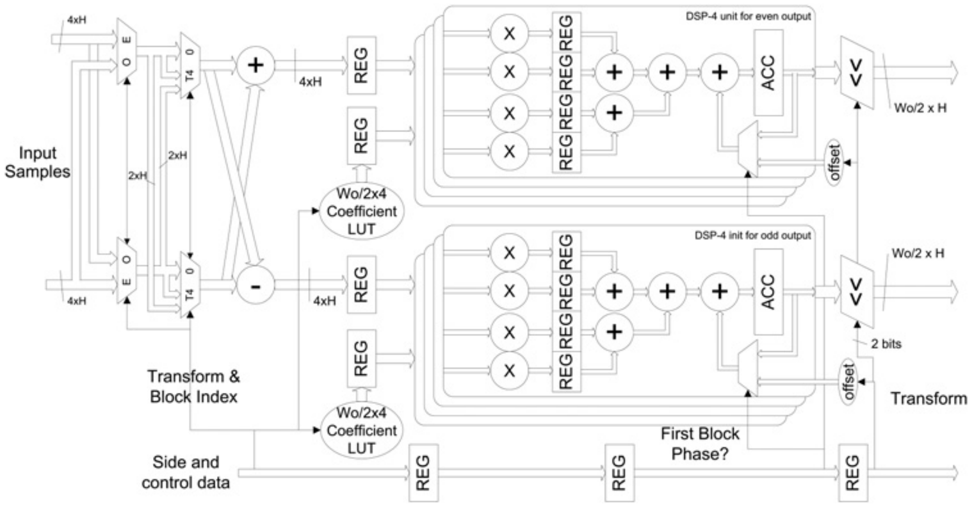

Fig. 9 *ASIC architecture of the 1D (horizontal) transformation*

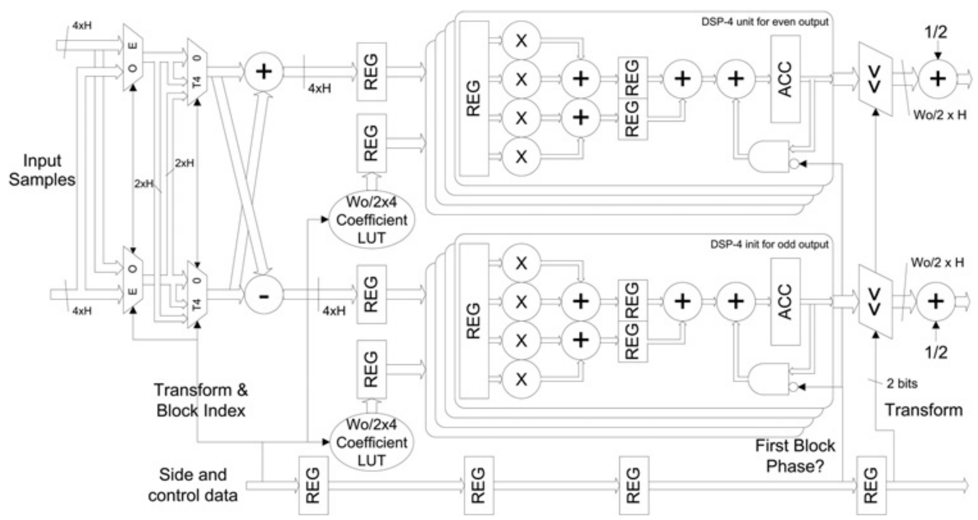

Fig. 10 *FPGA architecture of the 1D (horizontal) transformation*

The coefficients correspond to particular matrix entries, and their selection is made with LUTs shown in Figs. 9 and 10. The architecture can support DCT-like transforms if corresponding coefficients can be read from LUTs. This allows the multi-standard processing.

5 Implementation results

Two versions of the a DCT architecture are specified in VHDL and verified with the HM 13 reference model [2]. The first version is optimised for ASIC designs, whereas the second version is dedicated for FPGA devices. Differences between the two versions are described in the previous section. The synthesis is performed for FPGA and ASIC technologies using the Altera Quartus II (ver. 12.0) software and Synopsys Design Compiler (ver. 2012.06-SP5), respectively. Particularly, FPGA synthesis is performed for Arria II GX FPGA devices, whereas TSMC 90 nm is selected as the ASIC technology. In the second case, the design takes advantage of clock gating to reduce power consumption. Achieved frequencies are 200 and 400 MHz for the FPGA and ASIC technologies, respectively. If two pipeline stages in DSP-4 units are merged, the frequencies drop. In particular, 170 and 333 MHz are obtained for the FPGA and ASIC technologies, respectively.

Table 1 shows the resource consumption for different sizes of output blocks and corresponding throughputs for three transform sizes ($8 \times 8/16 \times 16/32 \times 32$). The synthesis results show that the dependence between throughput and resource consumption is almost proportional. The proportionality stems from the fact that the number of DSP-4 units (main contribution) is equal to the doubled size of the output block. At the same throughput, the design requires less resource when decreasing the block width. This stems from the fact that the size of the transposition buffer depends on the output-block width rather than the height. The impact of the buffer size on the resource consumption is strong for FPGA, as each flip-flop must be mapped into a separate logic element (ALUT). Thus, the full-size transposition buffer would consume 16 k logic elements, which is much more than the whole DCT architecture with the 8×8 output. If the output block width W_o is set to two, the buffer needs about 2 k logic elements, that is, the reduction of 87% is achieved. In the case of AISC technologies, such a buffer consists of 16 k flip-flops and corresponding two-input multiplexers. For the TSMC 90 nm general-purpose nominal-threshold-voltage library, the smallest area of one flip-flop/multiplexer pair is equivalent to eight basic gates. Therefore the buffer consumes resources equivalent to 128 k gates. Buffer sizes reported in Table 1 are significantly smaller, that is, the buffer is decreased by about 54–88%.

The support of the transform module for particular video resolutions depends on the encoder dataflow. If the transformation is executed only once, the required throughput is relatively low. On the other hand, the requirement increases with the number of performed transformations and the transform size. Provided that the 32-point transform for luma and 16-point transform for chroma are executed only once, the module with the 4×8 output interface can support 7680 × 4320p@60 fps videos at the frequency of 400 MHz. Particularly, 128 and 32 clock cycles are utilised to feed input with luma and chroma samples, respectively. The delay of 24 cycles must also be taken into account. Totally, 184 cycles are required, whereas 204 are available. In practice, the reserve must be much higher than 20 cycles (204–184) since the inverse

Table 1 Resource consumption for different configurations of the output interface

Block size	8 × 8	8 × 4	4 × 8	4 × 4	8 × 2	2 × 8	4 × 2	2 × 4	2 × 2
throughput [sample/cycle]	64/32/16	32/16/8	32/16/8	16/8/4	16/8/4	16/8/4	8/4/2	8/4/2	4/2/1
TSMC 90 nm [gate] – total	672 903	375 287	328 755	184 731	230 783	170 215	115 808	96 300	60 020
TSMC 90 nm [gate] – buffer	59 332	59 643	29 673	29 680	59 604	14 829	29 683	14 845	14 827
Arria II GX [ALUT] – total	12 704	11 520	7269	6402	10 879	4539	5871	3792	3331
Arria II GX [ALUT] – buffer	8198	8195	4099	4099	8195	2051	4100	2051	2051
Arria II GX [DSP]	256	128	128	64	64	64	32	32	16

Table 2 Resource consumption for different configurations of supported transform sizes

Configurations							
4 × 4	8 × 8	16 × 16	32 × 32	Multi-standard	Stages merged	Arria II GX	TSMC 90 nm
×	×	×	×			7269	328 235
	×	×	×			6928	326 181
		×	×			6821	326 176
			×			6792	325 659
×	×	×				5014	294 399
×	×					3436	196 918
	×	×				4921	292 453
×	×	×	×	×		7429	328 754
	×	×	×	×		7078	326 302
×	×	×	×		×	7269	295 580

4 × 8 output interface is selected

transform following the forward one must be completed. If more transforms are computed, lower video resolutions can be supported. The continuous processing for different video standards improves the computational efficiency. It can be achieved by the interleaving of candidate blocks based on their modes (intra/inter, chroma/luma) and sizes. The proposed architecture can interrupt and resume the transformation since each column of blocks can be computed independently. Particularly, iterations of the external loop can be divided into separate groups. This feature facilitates the interleaving for parts of large blocks (e.g. 32 × 32).

The use of a smaller-size transform increases the throughput and reduces the delay. Particularly, the throughput of the 2D transform module is inversely proportional to N (except $N = 4$). If only 8-point and 4-point transforms are allowed, the module can compute 4 32 × 32-predictions or 64 8 × 8 candidates in 137 (4 × 32 + 9 or 64 × 2 + 9) cycles. The limit of 204 cycles per macroblock exits also when processing 3840 × 2160p@60 fps videos with the H.264/AVC encoder. Provided the continuous processing, 102 candidate blocks (4 × 4 or 8 × 8) can be transformed using the module with the 4 × 8 output interface. If the frame rate or the frame area is decreased by half, the number of allowable candidate blocks is doubled. A similar scaling can be applied for H.265/HEVC. The greater number of candidates allows the encoder to improve its compression efficiency.

Table 2 shows the resource consumption for different configurations of supported transform sizes. Moreover, the support for transforms specified in other video compression standards is studied. In particular, the multi-standard extension allows the processing for H.264/AVC, AVS and VC-1. The results are provided for the output block size of 4 × 8 coefficients. Note that the number of DSP units in the FPGA implementation is constant and equal to 128 (256 multiplications). If the 32-point transform is supported, the resource consumption is not changed significantly regardless of the configuration for the remaining ones and the multi-standard support. This observation stems from the fact that look-up tables for transform coefficients take relatively a small amount of resources. Therefore the extended support for other symmetric transform matrices should slightly affect the hardware complexity provided that the coefficient accuracy and transform sizes are kept constant. On the other hand, removing the support for large transforms significantly reduces the amount of resources. The main reason is the decrease of the transposition buffer. If the 2D-transform module has the extended support for other video standards, the increase of the resource consumption is slight. The last row in Table 2 shows resource savings when two pipeline stages in DSP-4 units are merged. The modification reduces the logic area by 10% for the ASIC design, whereas there is no change in the case of the FPGA implementation.

6 Design comparison

Table 3 provides the comparison with different DCT architectures described in the literature. If applicable, throughputs are provided for different transform sizes (4 × 4/8 × 8/16 × 16/32 × 32). Two versions of the proposed architecture having throughputs close to those of referenced designs are provided (output block size equal to 2 × 8 and 4 × 8). The proposed designs can operate at the highest frequencies. The estimated power consumption of the proposed architecture is similar to those of other designs despite of the highest frequency. If two pipeline stages in DSP-4 units are merged, the power consumption is reduced to 35.1 and 54.6 mW for the 2 × 8 and 4 × 8 configurations, respectively. Even without the merging, the proposed architectures have the lowest energy consumption per output sample. The proposed methodology allows the lowest latencies. Owing to the decreased transposition buffer, the proposed architectures consume less resource at similar or higher throughputs. To show the design efficiency, Table 3 includes the ratio of average throughput (samples/clock) to gate count. The average assumes the same weights for each transform size. Proposed architectures achieve the highest ratios.

Although the architecture [8] incorporating memories need fewer gates, its total resource cost should be increased significantly. In particular, the area of the 16 kbit on-chip memory divided into 32 modules is equivalent to 108.1 k gates [8]. The design proposed by Meher *et al.* [9] allows the throughput of 32 samples/clock regardless the transform size. Taking into account the frequency, it outperforms the proposed architecture with the 4 × 8 interface only for the 32-point transform. However, the Meher's architecture has other disadvantages. Firstly, it consumes more resource even though the output registers and the interface adaptation are neglected. Secondly, the pruning involves losses in the compression efficiency. Thirdly, the latency is higher. Fourthly, the implementation in FPGA devices is not efficient because of the mapping to general-purpose resources (lower frequencies and a great number of logic elements utilised). Fifthly, the support for transforms specified in other video standards requires more resources

Table 3 Comparison of different DCT architectures

Design	Zhao *et al.* [6]	Ahmed *et al.* [7]	Zhu *et al.* [8]	Meher *et al.* [9]		This work	
technology, nm	45	90	TSMC 90	TSMC 90	TSMC 90	TSMC 90	TSMC 90
gate count, k	345.0	~173	412.3/320.0	208.0	347.0	170.3	328.2
memory, bit	–	–	0/16384	–	–	–	–
clock frequency, MHz	333	150	311	187	187	400	400
throughput 2D [samples/clock]	1.2/2.8/6.2/ 13.6	4/8/16/8	4/8/16/32	16	32	16/16/8/4	32/32/16/8
throughput 2D, Gsamples/s	0.4/0.9/2.0/4.5	0.6/1.2/2.4/1.2	1.25/2.5/5/10	2.99	5.98	6.4/6.4/3.2/ 1.6	12.8/12.8/ 6.4/3.2
average throughput, Gsamples/s	1.95	1.35	4.69	2.99	5.98	4.4	8.8
aver. throughput/area	5652	7803	11 369	14 375	17 233	25 837	26 813
latency [clock]	9/15/25/43	12/16/24/72	10/14/22/38	32	32	9/9/12/24	9/9/12/24
transposition buffer	32 × 32 registers	32 × 32 registers	32 × 32 registers/ memory	32 × 32 registers	32 × 32 registers	4 × 32 registers	8 × 32 registers
power, mW	–	–	61.5/89.4	40.04	67.57	43.4	76.9
power/throughput, pJ/ sample	–	–	13.12/19.07	13.51	11.30	9.86	8.74
features	2D-DCT	2D-DCT, 13-bit registers	2D-DCT, IDCT and Hadamard	2D-DCT, losses in compression-efficiency		2D-DCT	

and additional design efforts. Sixthly, input/output interfaces cannot be adapted, that is, the input and the output are fixed to the 32 sample row and the 32 coefficient column, respectively. The throughput and the clock frequency can be increased by the pipelining. However, this approach would increase the resource/power consumption and the latency.

The implementation of the referenced architectures in FPGA devices would lead to lower frequencies and a great number of logic elements utilised. In this technology, each bit adder/subtractor is mapped to a separate logic element. As referenced DCT/IDCT architectures embed many such operations, their resource consumption is huge. For example, Meher's implementation requires 682 additions for each 1D-DCT. Provided that arguments have 16 bits, the 2D-DCT needs about 22 k logic elements. In addition, 16 k logic elements must be occupied to implement the transposition buffer. The DCT implementation proposed by Zhao *et al.* [6] consumes 40541 logic elements. The IDCT implementation developed by Conceição *et al.* [18] requires 28311 ALUTs and 15367 registers (implemented in separate ALUTs). Although carry-chains are highly optimised, the wiring between successive layers of adders/ subtractors introduces significant latencies. Hence, the clock frequency is decreased dramatically (43.6 MHz) [18]. The pipelining can increase the frequency (150 MHz) [19]. The high frequency does not assure the high throughput. Although the design proposed by Belghith *et al.* [20] can operate at high frequency (251 MHz), its 2D throughput is relatively low (0.31/0.57/1 samples per clock cycle) due to the inefficient dataflow. Since the proposed architecture exploits DSP units, its frequency is significantly higher despite the less-advanced FPGA family and the moderate speed grade (5). Moreover, the utilisation of DSP units decreases the consumption of general-purpose logic several times. This provides a better balance between different-type resources available in FPGA devices.

As DCT usually process data in different order than intra/ inter prediction modules, input buffers are indispensible to form the processing path. Architectures based on the 32-point core require the adaptation of interfaces to the block processing order. In the proposed methodology, the adaptation is natural since the block width and height are configurable. On the other hand, the input buffer needs to be continuously accessed.

7 Conclusion

This paper proposes the architecture design for the DCT used in H.265/HEVC. Depending on the required throughput a number of architectures can be generated by the specification of the size of the input/output interfaces. The design methodology can also be applied in the development of the inverse transform. The methodology enables the reduction of the transposition matrix, the resource-throughput tradeoff, and the interface adaptation to desired block sizes. Although regular multiplication units are used, the methodology allows the design to consume less resource compared to other approaches. The advantage of the proposed solution is particularly evident in the FPGA technology. Furthermore, architectures can support transforms specified in other video standards at the slight increase of hardware resources.

8 References

1 ITU-T Recommendation H.265 and ISO/IEC 23008-2 MPEG-H Part 2: 'High efficiency video coding (HEVC)'. 2013

2 HEVC software repository – HM-13 reference model

3 Sullivan, G.J., Ohm, J., Woo-Jin, H., Wiegand, T.: 'Overview of the high efficiency video coding (HEVC) standard', *IEEE Trans. Circuits Syst. Video Technol.*, 2012, **22**, (12), pp. 1649–1668

4 Grois, D., Marpe, D., Mulayoff, A., Itzhaky, B., Hadar, O.: 'Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/ MPEG-AVC encoders'. Proc. Picture Coding Symp. (PCS), December 2013, pp. 394–397

5 ITU-T Recommendation H.264 and ISO/IEC 14496-10 MPEG-4 Part 10: 'Advanced video coding (AVC)'. 2003

6 Zhao, W., Onoye, T., Song, T.: 'High-performance multiplierless transform architecture for HEVC'. Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS), 2013, pp. 1668–1671

7 Ahmed, A., Shahid, M.U., ur Rehman, A.: 'N point DCT VLSI architecture for emerging HEVC standard'. VLSI Design, 2012

8 Zhu, J., Liu, Z., Wang, D.: 'Fully pipelined DCT/IDCT/Hadamard unified transform architecture for HEVC codec'. Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 2013), May 2013, pp. 677–681

9 Meher, P.K., Park, S.Y., Mohanty, B.K., Lim, K.S., Yeo, C.: 'Efficient integer DCT architectures for HEVC', *IEEE Trans. Circuits Syst. Video Technol.*, 2014, **24**, (1), pp. 168–178

10 Jeske, R., de Souza, J.C., Wrege, G., *et al.*: 'Low cost and high throughput multiplierless design of a 16 point 1-D DCT of the new HEVC video coding standard'. Proc. VIII Southern Conf. on Programmable Logic, March 2012
11 Edirisuriya, A., Madanayake, A., Cintra, R.J., Bayer, F.M.: 'A multiplication-free digital architecture for 16 × 16 2-D DCT/DST transform for HEVC'. Proc. IEEE 27th Conf. of Electrical & Electronics Engineers in Israel, November 2012, pp. 14–17
12 Martuza, M., Wahid, K.: 'A cost effective implementation of 8 × 8 transform of HEVC from H.264/AVC'. Proc. IEEE Canadian Conf. on Electrical & Computer Engineering, April 2012
13 Shen, S., Shen, W., Fan, Y., Zeng, X.: 'A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards'. Proc. IEEE Int. Conf. on Multimedia and Expo, July 2012, pp. 788–793
14 Park, J.-S., Nam, W.-J., Han, S.-M., Lee, S.: '2-D large inverse transform (16 × 16, 32 × 32) for HEVC (High Efficiency Video Coding)', *J. Semicond. Technol. Sci.*, 2012, **12**, (2), pp. 203–211
15 Tikekar, M., Huang, C.-T., Juvekar, C., Sze, V., Chandrakasan, A.P.: 'A 249-Mpixel/s HEVC video-decoder chip for 4 K ultra-HD applications', *IEEE J. Solid-State Circuits*, 2014, **49**, (1), pp. 61–72
16 Chiang, P.-T., Chang, T.S.: 'A reconfigurable inverse transform architecture design for HEVC decoder'. Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS 2013), May 2013, pp. 1006–1009
17 Budagavi, M., Fuldseth, A., Bjøntegaard, G., Sze, V., Sadafale, M.: 'Core transform design in the high efficiency video coding (HEVC) standard', *IEEE J. Sel. Top. Signal Process.*, 2013, **7**, (6), pp. 1029–1041
18 Conceicao, R., Souza, J.C., Jeske, R., Porto, M., Mattos, J., Agostini, L.: 'Hardware design for the 32 × 32 IDCT of the HEVC video coding standard'. 26th Symp. on Integrated Circuits and Systems Design (SBCCI), September 2013
19 Kalali, E., Ozcan, E., Yalcinkaya, O.M., Hamzaoglu, I.: 'A low energy HEVC inverse DCT hardware'. IEEE Third Int. Conf. on Consumer Electronics (ICCE-Berlin), September 2013, pp. 123–124
20 Belghith, F., Loukil, H., Masmoudi, N.: 'Efficient hardware architecture of a modified 2-D transform for the HEVC standard', *Int. J. Comput. Sci. Appl. (IJCSA)*, 2013, **2**, (4), pp. 59–69
21 Chen, W.H., Smith, C.H., Fralick, S.C.: 'A fast computational algorithm for the discrete cosine transform', *IEEE Trans. Commun.*, 1977, **COM-25**, (9), pp. 1004–1009