A Low-Cost VLSI Architecture of Multiple-Size IDCT for H.265/HEVC

Heming SUN††a), Nonmember, Dajiang ZHOU†, Member, Peilin LIU††, Nonmember, and Satoshi GOTO†, Fellow

SUMMARY In this paper, we present an area-efficient 4/8/16/32-point inverse discrete cosine transform (IDCT) architecture for a HEVC decoder. Compared with previous work, this work reduces the hardware cost from two aspects. First, we reduce the logical costs of 1D IDCT by proposing a reordered parallel-in serial-out (RPISO) scheme. By using the RPISO scheme, we can reduce the required calculations for butterfly inputs in each cycle. Secondly, we reduce the area of transpose architecture by proposing a cyclic data mapping scheme that can achieve 100% I/O utilization of each SRAM. To design a fully pipelined 2D IDCT architecture, we propose a pipelining schedule for row and column transform. The results show that the normalized area by maximum throughput for the logical IDCT part can be reduced by 25%, and the memory area can be reduced by 62%. The maximum throughput reaches 1248 Mpixels/s, which can support real-time decoding of a 4K × 2K 60 fps video sequence.

key words: HEVC, IDCT, SRAM, area-efficient, video coding

1. Introduction

High Efficiency Video Coding (HEVC) [1] is a new video compression standard that has been approved as an ITU-T standard. HEVC aims to double the compression ratio compared with H.264. To achieve this target, many new coding tools have been used [2], [3]. Since the development of H.264, the discrete cosine transform (DCT) has been widely employed to perform the transform operation since it can provide energy compaction. To reduce the computational complexity and solve the mismatch problem between forward and inverse transforms, integer DCT has been adopted instead of floating DCT. The largest DCT size provided in H.264 is 8 × 8, while larger DCTs including 16 × 16 and 32 × 32 are provided in HEVC since a larger transform size may also enable higher compression ratios.

In H.264, in order to reduce the complexity of the transform, some area-efficient inverse DCT (IDCT) architecture designs have been presented. In [4], the authors proposed an architecture for a multi-standard inverse transform; two circuit share strategies, a factor share, and adder share strategies were applied to reduce the required circuit resource. A low-cost hardware sharing architecture was proposed in [5] by adding the offset computations and pipelined design. In [6], Wang et al. designed a reconfigurable architecture that provided the fusing strategies that generate constant multipliers in the matrix calculation blocks. In addition, an adder-sharing strategy was adopted to save circuit area. A high-performance inverse transform circuit was also developed in [7] based on the butterfly architecture, and this work shares the resources efficiently by exploiting the similarities between transforms. In [8], extensive mathematical analysis and decomposition were performed for the H.264 transform and quantization so that all multiplication and division operations were avoided effectively. The above low-cost architectures were all designed for the 4/8-point IDCT in H.264, and they cannot be directly used in HEVC due to the differences in the transform size.

The large transform in HEVC will cause significant complexity problems from two perspectives. One is the logical computation part of the transform; a larger transform leads to a greater number of multipliers and adders causing the required amount of hardware resources to increase. The other problem is that the transpose memory required to store more intermediate results, so the transpose architecture consumes more area. In order to solve the above two problems, several recently developed low-cost architectures have been reported for the HEVC transform.

To reduce the hardware cost of the logical IDCT computation, [9] utilize the Chen’s algorithm [19] to reuse the small transform in the larger transform architecture, the author used the multiple constant multiplication (MCM) for 4/8-point IDCT and share the regular multipliers for 16/32-point IDCT. In [10], the multiplications of 16/32-point IDCT are implemented with input-muxed constant multipliers. Still based on Chen’s algorithm, a fully pipelined transform architecture was proposed for an HEVC codec in [11]. The authors in [12] proposed a 16/32-point IDCT architecture that had no multiplications in the design. The similarity property of transform matrices was utilized in [13], so a calculation unit can be shared for different transform mode. In [14], the authors presented a unified forward/inverse transform architecture for HEVC; the unified architecture makes use of symmetrical properties that exist in HEVC transform matrices to achieve hardware sharing. In [15], the authors turned the multiplications by constant into shift and sum operations to achieve a multiplierless HEVC transform architecture. In [16], the N-point 1D DCT was performed using an (N/2)-point 1D DCT unit and a constant matrix multiplication recursively. [17] developed two efficient DCT architectures for HEVC, one is based on Chen’s algorithm and implemented without multipliers. Moreover, the other one uses pruning scheme to truncate some transformed bits in

††The author is with Shanghai Jiao Tong Univ., 200240, China. a)E-mail: terrysun1989@gmail.com DOI: 10.1587/transfun.E97.A.2467
order to save the complexity. The pruned architecture cannot be used in the decoder since the transformed bits are truncated.

To reduce the area of the transpose memory, SRAM instead of register is utilized in some previous works. [9] used 4 single-port 8 × 512-bit SRAM. [11] used 32 dual-port 32 × 16-bit SRAM. The total bits of SRAM are 16384 bits in both designs. For the transpose memory design in [9] and [11], the data width of SRAM is bigger than the data parallelism of logical IDCT, so the I/O utilization of each SRAM cannot reach 100%.

The main contributions and new features of this paper are listed as follows:

1) To reduce the hardware resources of IDCT logical computation, we utilize the symmetric property in the butterfly structure. We propose a reordered parallel-in serial-out (RPISO) scheme to reduce the redundant inputs of butterfly and the required calculations for butterfly inputs in each cycle.

2) To reduce the hardware area of transpose memory, we present a cyclic data mapping that achieves 100% I/O utilization for each SRAM. In addition, to make a fully pipelined design for 2D IDCT, two-port SRAM is used. A pipelining schedule for row and column 1D IDCT is proposed to avoid the address conflict of writing and reading.

The results show that our design can significantly reduce the circuit area in the logical computation part and transpose memory compared with existing works. The maximum throughput is able to support real-time decoding of a 4 K × 2 K 60 fps video sequence.

The rest of this paper is arranged as follows. Section 2 explains the inverse transform in HEVC. Section 3 demonstrates the proposed 1D IDCT architecture design. Section 4 presents the SRAM-based transpose memory. Section 5 shows the experimental results, followed by the conclusion in the last section.

2. HEVC Inverse Transform

HEVC IDCT uses the integer transform and the largest transform size is 32 × 32. A complete 2D IDCT can be decomposed to two 1D IDCTs, and the decomposition method is shown in the following equation.

\[ T_{2N}^T \cdot [Y] \cdot T_{2N} = ([Y] \cdot [T_{2N}])^T \cdot [T_{2N}]^T \]  

where \( T_{2N} \) is a 2N × 2N transform matrix defined by HEVC, \( Y \) is the 2N × 2N matrix before the IDCT, and \( X \) is the result after the 2D IDCT operation. The first 1D IDCT (IT1) is called a row IDCT, and the second one (IT2) is called a column IDCT. For a 2N-point input \([Y_0, Y_1, \ldots, Y_{2N-1}]\), which is defined as \( Y_{2N} \), a 1D IDCT operation \([Y_{2N}]^* \cdot [T_{2N}] \) requires \( 2N^2 \) multiplications and \( 2N*(2N-1) \) additions.

In fact, the transform matrix \( T_{2N} \) has the property of symmetry so that it can be decomposed. For example, \( T_8 \) is given by Eq. (2). We can see that the matrix has both symmetric and anti-symmetric properties, as shown in Eq. (3).

\[ T_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 75 & -18 & -89 & 50 & 50 & 89 & 18 & -75 \\ 64 & -64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 50 & -89 & 18 & 75 & -75 & 18 & 89 & -50 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & 75 & 50 & -18 \end{bmatrix} \]  

(2)

\[ T_8(i, j) = T_8(i, 7 - j) \quad (i = 0, 2, 4, 6) \]

\[ T_8(i, j) = -T_8(i, 7 - j) \quad (i = 1, 3, 5, 7) \]  

(3)

The matrices for each transform size can be found in [18]. Using the symmetry, a fast algorithm was developed to decompose the transform matrix [19]. Using Chen’s algorithm, the decomposition method for the HEVC transform matrix is shown in the following equation.

\[ [T_{2N}] = [P_{2N}] \cdot [T_N] \cdot [O_N] \cdot [B_{2N}] \]  

(4)

where \( T_{2N} \) is a 2N × 2N transform matrix, \( T_N \) is an N × N transform matrix, \( O_N \) is an odd part matrix, \( P_{2N} \) is a permutation matrix, and \( B_{2N} \) is a 2N-point butterfly structure. Detailed definitions for \( P_{2N}, B_{2N}, T_N, \) and \( O_N \) are given as follows.

\[ P_{2N}(i, j) = \begin{cases} 1, & i = 2 \cdot j \text{ or } i = (j-N) \cdot 2 + 1 \\ 0, & \text{otherwise} \end{cases} \]

\[ B_{2N}(i, j) = \begin{cases} -1, & i = j \text{ and } i < N \text{ or } (i+j = 2N-1) \\ 0, & \text{otherwise} \end{cases} \]

\[ T_N(i, j) = T_{2N}(2 \cdot i, j) \]

\[ O_N(i, j) = T_{2N}(2 \cdot i + 1, N-1-j) \]

By using the decomposition, \([Y_{2N}]^* \cdot [T_{2N}] \) can be expressed as the following equation.

\[ [Y_{2N}] = [Y_{2N}] \cdot [P_{2N}] \cdot [T_N] \cdot [O_N] \cdot [B_{2N}] \]

(5)

where \( [Y_{2N}]^* \cdot [T_N] \) and \( [Y_{2N}]^* \cdot [O_N] \) are defined as the results of the even and odd parts, respectively. The implementation of \([Y_{2N}]^* \cdot [T_N] \) and \( [Y_{2N}]^* \cdot [O_N] \) cost \( N^2 \) multiplications and \( N^2*(N-1) \) additions, respectively, while \( B_{2N} \) requires \( 2N \) additions. Therefore, a total of \( 2N^2 \) multiplications and \( 2N^2 \) additions are required. Compared with the original method before decomposition, the required number of computations is significantly reduced.

In fact, \( T_N \) can be decomposed into the \( T_{N/2} \) and \( O_{N/2} \) in the same way to further reduce the number of computations for the even part. In the HEVC Test Model (HM) [20], Chen’s algorithm is adopted in the implementation of IDCT.

3. Proposed 1D IDCT Architecture

First, we design an 8-point 1D IDCT architecture. Figure 1
In our design, the parallelism is set at 4 pixels/cycle. Therefore, the 4-point IDCT can be finished within one cycle, and the 8/16/32-point IDCT can be finished within 2/4/8 cycles. Using the 8-point 1D IDCT as an example, two cycles are required to generate the eight results Xₙ (n = 0, 1, 2, ..., 7). Conventionally, {X₀, X₁, X₂, X₃} are generated in the first cycle, and {X₄, X₅, X₆, X₇} are generated in the second cycle. In this way, according to Eq. (8), all of the Eₙ and Oₙ results are required in both cycles. To generate all of the Eₙ and Oₙ results in each cycle, the architecture for the odd and EE engines is the same as shown in Fig. 2.

3.1 Conventional Parallel-In Serial-Out Scheme

In order to reduce the redundant inputs of butterfly and the required calculations for butterfly inputs in each cycle, we reorder the outputs. After the reordering, in the first cycle, {X₀, X₁, X₂, X₇} are generated. In the second cycle, {X₃, X₄, X₅, X₆} are generated. To calculate X₀₁₄₇ in the first cycle, O₁₃ and E₀₃ are required, while to obtain E₀₃, E₀₀ is required. In the second cycle, to calculate X₁₂₅₆, O₁₂ and E₁₂ are required, while to obtain E₁₂, EE₁ is required. Within a single cycle, only one result is generated from the EE part and two results are generated from Odd part. It should be noted that the EO engine is similar to the EE engine; within each cycle, only one result is generated from the EO engine.

Using the RPISO scheme, the architecture for the EE engine is shown in Fig. 3. In the first cycle, EE₀ is selected for the calculation, and the multiplexers select T[0, 0] and T[4, 0]. In the second cycle, EE₁ is required, thus the multiplexers select T[0, 1] and T[4, 1]. Compared with the original EE engine in Fig. 2(a), the number of EE sample generators is reduced by one half with some extra multiplexers. The architecture for the odd engine is shown in Fig. 3. The multiplexers are used to select the transform coefficients. In the first cycle, the transform coefficients for O₁₃ are selected, while in the second cycle, the transform coefficients for O₁₂ are selected. Compared with the original architecture shown in Fig. 2(b), one half of the multiplexers and adders can be reduced for the odd engine. The proposed architecture for an 8-point 1D IDCT is shown in Fig. 4(A) where Yₙ represents 8-point inputs, the circuits for EE₈ and EO₈ are shown in Fig. 3. The values of sel₈₈ and sel₈₀ and the reordered outputs in each cycle are shown in Fig. 4(B).
It has to be noted that the pixel parallelism is not necessary to be 4. When the parallelism is 8 or 16, the RPISO scheme can still reduce the redundant inputs of butterfly and the required calculations for butterfly inputs in each cycle.

### 3.3 Unified 8/16/32-Point IDCT Architecture Based on RPISO

A 16-point 1D IDCT is similar with 8-point IDCT. The overall flowchart is shown in Fig. 5. It is composed of permutation part, even part, odd part and butterfly structure. By using a 16-point butterfly structure, $X_n$ and $X_{15-n}$ are calculated by the same $E_n$ and $O_n$ as shown in the following equation.

$$X_n = E_n + O_n, \quad X_{15-n} = E_n - O_n$$

By using the proposed RPISO scheme, $X_n$ and $X_{15-n}$ can share $\{E_n, O_n\}$ in the same cycle. To generate four outputs in one cycle, two even results and two odd results are calculated in one cycle. The reordered outputs in each cycle are stated in Fig. 6(A). Four cycles are required to generate 16-point results.

For the even part of 16-point IDCT, $E_n$ is equal to the result of 8-point IDCT according to [19]. So the architecture in Fig. 4(A) can be reused as the even engine for 16-point IDCT. Within each cycle, four results are generated from 8-point IDCT. A multiplexer is required to select two as the results for even part. The selection signal is $sel_{e16}$. The selection signal $sel_{e8}$ and $sel_{e8}$ can select four results generated by 8-point IDCT. The values for the selection signals in each cycle are given in Fig. 6(A). For example, in the first cycle, $sel_{e16}$ is 0, the corresponding constants for $O_0$ and $O_7$ are selected so that $\{O_0, O_7\}$ are generated as the odd results for 16-point IDCT.

For the odd part of 16-point IDCT, $O_n$ is generated by multiplication and addition. An odd engine for 16-point IDCT (O16) can generate one $O_n$ result. As shown in Fig. 7, O16 is composed of 8 multiple constant multiplications (MCM) and 7 adders. For each MCM in O16, one input is the odd-index input $Y_n$, and the other is constant coefficient. The constants are selected by a Look Up Table (LUT16) and the selection signal is $sel_{o16}$. For example, in the first cycle, $sel_{o16}$ is 0, the corresponding constants for $O_0$ and $O_7$ are selected so that $\{O_0, O_7\}$ are generated as the odd results for 16-point IDCT.

Similarly, the 32-point IDCT can reuse the 16-point

![Fig. 3 Proposed architecture for EE8, EO8 and O8 engines by using the RPISO scheme in the 8-point IDCT.](image1)

![Fig. 4 Proposed architecture for 8-point 1D IDCT.](image2)

![Fig. 5 Overall flowchart for 16-point 1D IDCT based on Chen's algorithm.](image3)

![Fig. 6 Selection signal values for multiplexers and the reordered outputs in each cycle for IDCT8 and IDCT16.](image4)
IDCT as the even engine. A multiplexer with the selection signal sel_e32 is used to select two results from 16-point IDCT. sel_e16 and sel_o16 can decide the outputs from 16-point IDCT. For the odd part of 32-point IDCT, an O32 can generate one result. O32 is composed of 16 MCMs and 15 adders, as shown in Fig. 7. sel_o32 is used to select the constant coefficients from LUT32. The values for all the selection signals in each cycle are shown in Fig. 6(B). Eight cycles are required to generate 32-point results. The overall architecture for 8/16/32-point IDCT is shown in Fig. 7.

It has to be noted that in the RTL level, we implement the multiplication directly (* operator in Verilog). However, the multiplier in odd engine is multiple constant multiplier (MCM) which can be implemented as the adder tree by the Design Compiler tool. Figure 8(A) gives the inputs of MCM in O32, one constant input is selected from an 8to1 multiplexer and the variable is odd-index input of 32-point IDCT. Similarly, for the multiplier in O16, one constant input is selected from a 4 to 1 multiplexer, and the variable is odd-index input of 16-point IDCT. Moreover, the circuit of O32 is shown in Fig. 7. 16 multipliers are processed in parallel. The number of multipliers does not influence the critical path. According to the timing report by Design Compiler, the critical path is shown in Fig. 7. We can see that the critical path does not include the O32.

It should be noted that in our design, within each cycle, four final outputs of IDCT can be generated regardless of the TU size. The register is not used to store the intermediate results, and the 4-point IDCT does not adopt the RPISO scheme since it may be generated within one cycle; thus, there is no need to reorder the outputs. The proposed unified architecture only supports the 8/16/32-point IDCT. Because the 4-point IDCT incurs a small hardware cost, we implemented the 4-point IDCT individually. The reason for doing so is that the overhead required to embed a 4-point IDCT into the unified architecture is higher than the direct implementation of a 4-point IDCT.

4. Area-Efficient Transpose Memory

As shown in Eq. (1), a complete 2D IDCT is composed of IT1 and IT2. IT2 can only start after getting a complete column of IT1 results. However, the IT1 results are obtained row by row. Therefore, we require transpose memory stores the IT1 results.

4.1 Data Mapping Scheme for SRAM

To reduce the hardware cost, we use SRAM to realize the transpose memory. The IT1 result is stored in the SRAM. The parallelism in our design is 4 pixels/cycle, so four SRAMs were used. To make full use of the transpose memory, we propose to use two data mapping modes. Taking TU 32 × 32 as an example, the mapping scheme for mode 0 is shown in Fig. 9. In Fig. 9, (m, n) indicates the IT1 result X_{m,n} (m, n = 0, 1, 2, ..., 31). The samples in the different columns are marked using various colors. The column 4k+1 (k = 0, 1, 2, ..., 7) is shaded in white, the column 4k+2 is shaded in light gray, the column 4k+3 is shaded in dark gray, and the column 4k+4 is shaded in black. In each cycle, four IT1 results are written into SRAMs. Note that we write X_{4k+1,n}, X_{4k+2,n}, X_{4k+3,n}, and X_{4k+4,n} in four SRAMs so that they can be read out in one cycle. For example, X_{0,0}, X_{1,0}, X_{2,0}, and X_{3,0} are written in four SRAMs. Therefore, we can read these four samples within one cycle. Similarly, X_{4k,0}, X_{4k+1,0}, X_{4k+2,0}, and X_{4k+3,0} are written in four SRAMs. We can read four IT1 results of the first column within one cycle. After eight cycles, the IT1 results of the first column can be fetched. After we obtain the IT1 results of the first column, IT2 can begin. Every eight cycles, the IT1 results of one complete column are read out for IT2.

When the transpose memory is full of the IT1 results of the same transform block N, the IT1 results of the next transform block N+1 are written in from the next cycle. Meanwhile, the data mapping mode interchanges. In the other mapping scheme mode, X’_{4k,n} (the IT1 results of block N+1) is stored in the same address as X_{m,n} of block N. The detailed mapping scheme for mode 1 is shown in Fig. 10. Similarly, we can read IT1 results of one complete column within eight cycles.
We aim to design a transpose memory that can store the IT1 results for various transform sizes on the decoder side. The data mapping scheme used to store the $32 \times 32$ IT1 results is given in Fig. 9 and Fig. 10. To store the $32 \times 32$ IT1 results in four SRAMs, the depth of each SRAM is $256 (32 \times 32 / 4)$. If the transform size is smaller, the IT1 results are stored in the corresponding address in the SRAM. For example, when writing the $16 \times 16$ IT1 results, $X_{m,n}$ ($m = 0, 1, 2, \ldots, 15$) is written in the corresponding position as shown in Fig. 9 and Fig. 10. One quarter of the transpose memory is used for transform block $16 \times 16$. When storing the $8 \times 8$ and $4 \times 4$ IT1 results, the proportion of the transpose memory that is used is only $1/16$ and $1/64$, respectively.

By using the proposed data mapping scheme, within each cycle, all the input/output ports of SRAM are used for writing/reading. Therefore, we can achieve 100% I/O utilization of each SRAM. In addition, the data width of SRAM is significantly reduced compared with previous work, so the hardware area can be reduced. The detail comparison result is shown in Sect. 5.2.

4.2 Pipeline Schedule for the SRAM

The type of SRAM does not influence the proposed data mapping scheme in Sect. 4.1. No matter a single-port or two-port SRAM is adopted, the I/O utilization of each SRAM can always achieve 100% by using our proposed data mapping scheme.

If a single-port SRAM is used as a transpose memory, write and read cannot work in parallel. IT1 is performed at first and the results are written in the memory. After that, the IT1 results are read out and IT2 is performed. IT1 and IT2 can share one 1D IDCT architecture. The overall architecture is shown in Fig. 11(A).

To make a fully pipelined 2D IDCT architecture, two-port (1r1w) SRAM is used in our design. Writing IT1 results in memory and reading IT1 results from memory can perform in parallel. IT1 and IT2 require the 1D IDCT hardware resource, respectively. The overall architecture is shown in Fig. 11(B).

The processes for IT1 and IT2 are pipelined in our design. In this way, the writing of the IT1 results and reading results for IT2 can be done in parallel. The read and write address for each SRAM should be different in each cycle. To avoid this address conflict, we propose a pipelining schedule for a two-port SRAM, as shown in Fig. 12.

Taking $TU 32 \times 32$ as an example, from cycles 0 to 7, the IT1 results of row 31 in block N (blkN, row 31) are written in the SRAM. Assume that the data mapping scheme
5. Experimental Results

We used Verilog HDL (VHDL) to implement the proposed design. The design was synthesized with a TSMC 90nm cell library. As mentioned previously, there are two critical problems that lead to the significant hardware cost. One is the logical computation part, and the other is the transpose memory part. We analyze the results from these two perspectives.

5.1 Comparison of the Logical Computation of IDCT

In our design, a 4-point IDCT is not included in the unified architecture. The reason is explained at the end of Sect. 3. The individual gate count for a 4-point IDCT is 2.4 k, and the gate count for the 8/16/32-point unified IDCT is 63.9 k. Therefore, an overall gate count of 66.3 k is required for a 1D IDCT design that can support the entire transform size on the decode side. The concept of the normalized area (NA) is introduced to obtain a fair performance comparison, and normalized area by maximum throughput is defined by the following equation.

\[
\text{Normalized area by maximum throughput} = \frac{\text{Gate}}{\text{MaxFreq} \times Tp} = \frac{\text{Gate}}{\text{MaxFreq} \times \text{MaxThroughput}}
\]

where MaxFreq is the maximum frequency of the design and Tp is the throughput within each cycle, which is equal to the pixel parallelism. “Gate” refers to the gate count for the logical calculation part for the IDCT excluding the transpose memory.

In order to demonstrate the effectiveness of our proposed design, we performed a comparison with those in references [9]–[13], which describe the low-cost designs for the HEVC IDCT. The comparison result is shown in Table 1. It should be noted that the throughput in [10], [11], and [13] is different for various 2N-point IDCTs. On the decoder side, the worst-case scenario should be considered. For example, in [11], the authors achieve 4/8/16/32 pixels/cycle for a 4/8/16/32-point IDCT, respectively. In the worst case, when the TU size is always 4 × 4 on the decoder side, the parallelism is 4 pixels/cycle. The smallest parallelism is used to calculate the normalized area. For the logical IDCT computation part, compared with [9]–[13], we can reduce the gate count by more than 25%.

For the 1D DCT architecture in [17], the pixel parallelism is 16 pixels/cycle. In fact, for the same function, the architecture with higher pixel parallelism can share more computations and hardware resources. So the architecture with higher pixel parallelism usually has less normalized area than one with lower pixel parallelism. The normalized area by parallelism is defined as follows.

\[
\text{Normalized area by parallelism} = \frac{\text{Gate}}{\text{pixel parallelism}}
\]
Table 1  Comparison with other HEVC 2D IDCT architectures.

<table>
<thead>
<tr>
<th>Design</th>
<th>Technology (nm)</th>
<th>Max Speed (MHz)</th>
<th>Pixel Parallelism</th>
<th>Gate Count (k)²</th>
<th>Normalized Area by Eq. (11)²</th>
<th>Normalized Area by Eq. (12)²</th>
<th>Supporting Video Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shen [9]</td>
<td>130</td>
<td>350</td>
<td>4</td>
<td>109.2 x 2</td>
<td>156</td>
<td>54.6</td>
<td>4Kx2K 30fps</td>
</tr>
<tr>
<td>Shen [10]</td>
<td>130</td>
<td>191</td>
<td>4-8</td>
<td>54.1 x 2</td>
<td>141.62</td>
<td>27.05</td>
<td>4Kx2K 30fps</td>
</tr>
<tr>
<td>Zhu [11]</td>
<td>90</td>
<td>311</td>
<td>4-32</td>
<td>117.7 x 2</td>
<td>189.23</td>
<td>58.85</td>
<td>4Kx2K 60fps</td>
</tr>
<tr>
<td>Park [12]³</td>
<td>180</td>
<td>300</td>
<td>2.12</td>
<td>52.3 x 2</td>
<td>164.46</td>
<td>49.34</td>
<td>4Kx2K 30fps</td>
</tr>
<tr>
<td>Chiang [13]</td>
<td>90</td>
<td>270</td>
<td>1.38-2.67</td>
<td>63.8 x 2</td>
<td>342.46</td>
<td>92.46</td>
<td>4Kx2K 30fps</td>
</tr>
<tr>
<td>Meher [17]</td>
<td>90</td>
<td>N/A</td>
<td>16</td>
<td>131 x 2</td>
<td>N/A</td>
<td>16.38</td>
<td>8Kx4K 60fps</td>
</tr>
<tr>
<td>Proposed 4x</td>
<td>90</td>
<td>312</td>
<td>4</td>
<td>66.3 x 2</td>
<td>106.25</td>
<td>33.15</td>
<td>4Kx2K 60fps</td>
</tr>
<tr>
<td>Proposed 8x</td>
<td>90</td>
<td>312</td>
<td>8</td>
<td>88.2 x 2</td>
<td>70.67</td>
<td>22.05</td>
<td>8Kx4K 60fps</td>
</tr>
<tr>
<td>Proposed 16x</td>
<td>90</td>
<td>312</td>
<td>16</td>
<td>113 x 2</td>
<td>45.27</td>
<td>14.13</td>
<td>8Kx4K 60fps</td>
</tr>
</tbody>
</table>

1) Park [12] only supports 16/32-point IDCT.
2) The parallelism for 4/8/16/32-point IDCT is 4 / 8 / 4 / 4 pixels/cycle, respectively.
3) The parallelism for 4/8/16/32-point IDCT is 4 / 8 / 16 / 32 pixels/cycle, respectively.
4) The parallelism for 4/8/16/32-point IDCT is 2 / 2.67 / 2 / 1.38 pixels/cycle, respectively.
5) Gate count for 2D IDCT.
6) The unit for Normalized Area by max. throughput is 1/(MHz*pixel/cycle). If the parallelism is different for various IDCT size, the least parallelism is used to calculate in Eq. (11).
7) The unit for Normalized Area by parallelism is 1K/(pixel/cycle). If the parallelism is different for various IDCT size, the least parallelism is used to calculate in Eq. (12).
8) Reference [17] synthesizes the architecture with the desired timing constraint for 8Kx4K 60fps. The corresponding operating frequency is 187MHz.

Table 2  Comparison of the hardware cost for each transform block.

<table>
<thead>
<tr>
<th>Transform size</th>
<th>Gate Count (k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-point</td>
<td>7.2</td>
</tr>
<tr>
<td>8-point</td>
<td>19.6</td>
</tr>
<tr>
<td>16-point</td>
<td>31.1</td>
</tr>
<tr>
<td>32-point</td>
<td>54.1</td>
</tr>
<tr>
<td>Proposed</td>
<td>2.4</td>
</tr>
<tr>
<td>4-point</td>
<td>8.6</td>
</tr>
<tr>
<td>8-point</td>
<td>24.3</td>
</tr>
<tr>
<td>32-point</td>
<td>47.8</td>
</tr>
</tbody>
</table>

32-point IDCT can take advantage of RPISO scheme to save the hardware cost.

The results are shown in Table 1. We can see that the design with higher pixel parallelism consumes much less normalized area than the design with lower pixel parallelism. Compared with [17], our IDCT architecture with parallelism of 16pixels/cycle consumes less normalized area by parallelism than [17]. The reason is partially because that we use RPISO scheme to reduce the calculations of butterfly inputs for 32-point IDCT.

It has to be noted that in [17], the author synthesized the 1D architecture with the desired timing constraint for 8 K x 4 K 60 fps. The corresponding frequency is 187 MHz (7680 x 4320 x 60 x 1.5 / 16). Therefore, we think 187 MHz mentioned in [17] is the operating frequency.

The results in Table 1 show a comparison of our results with other unified architectures for HEVC IDCT. Using the RPISO scheme, it is possible to realize savings in area. Each transform block is also implemented individually. The results are shown in Table 2. In [10], the author presented the hardware area for each transform block. Compared with [10], we can realize savings in the required hardware resources for each transform block by using the RPISO scheme.

In addition, it is obvious that the unification method proposed in Sect. 3.3 can result in reduced hardware overhead. The overall gate count for an individual transform block is 83.1k (2.4 + 8.6 + 24.3 + 47.8), which is greater than the 66.3 K of the proposed unified architecture.

5.2 Comparison of the Transpose Architecture

For the transpose memory part, four SRAM blocks are used since the parallelism is 4 pixels/cycle. To store the IT1 results of a 32 x 32 transform block, the width of each SRAM is 16 bit (one IT1 result) and the depth is 256 (32 x 32 / 4). A total of 16384 SRAM bits were used. To store the IT1 results of the smaller transform, a part of the transpose memory is used. The detailed data mapping scheme was given in Sect. 4.

In [9] and [11], the authors also used SRAM to realize the transpose architecture, and the total SRAM bit is the same as our design. However, we can achieve much smaller area since the width of our SRAM is significantly reduced compared with [9] and [11]. To perform a comparison with various transpose memory designs, we implemented a memory compiler using the same process TSMC 90 nm. A single-port SRAM is used in [9]. However, a 1D IDCT architecture in [9] can also be adopted in a fully pipelined 2D IDCT architecture when using a two-port SRAM. To have a fair comparison of all the SRAM-based transpose memory designs, we used a two-port memory compiler to generate the area. The total area for the SRAMs is shown in Table 3. Using our proposed narrower and deeper SRAMs, the total SRAM area was 80988 µm². Therefore, we realized a savings of at least 62% in the SRAM area compared with [9] and [11]. In [12] and [13], the authors used registers to implement a transpose memory based on the traditional method, which is not so area-efficient. The author in [10] did not develop the transpose architecture.

In summary, we have developed an area-efficient SRAM-based transpose architecture.

6. Conclusions

In this paper, we presented a low-cost multi-size IDCT ar-
architecture for HEVC. To reduce the required amount of hardware resources, the RPISO scheme was proposed. In this scheme, the final outputs are reordered so that we can reduce the redundant inputs of butterfly and the required calculations for butterfly inputs in each cycle. Based on the RPISO scheme, we presented a unified architecture for an 8/16/32-point IDCT. Using Chen’s algorithm, the N-point architecture was reused in the 2N-point architecture. We used an SRAM instead of a register to realize the transpose memory. We proposed the data mapping scheme and pipelining schedule for the SRAM so that the data width of the SRAM would be significantly reduced. The results show that we can realize savings of about 25% in the area for the logical computational part, and more than 60% in the area for the transpose memory part. Our design can therefore support real-time decoding of a 4 K × 2 K 60 fps video sequence.

Acknowledgments

This research was partly supported by Grand-in-Aid for Scientific Research (MEXT KAKENHI), STARC and the Program for Leading Graduate Schools, “Graduate Program for Embodiment Informatics” of the Ministry of Education, Culture, Sports, Science and Technology.

References


Heming Sun received the B.E. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2011, and received the M.E. degree from Waseda University and Shanghai Jiao Tong University, in 2012 and 2014, respectively, through a double-degree program. He is currently pursuing the doctor degree in Waseda University. His interests are in algorithms and VLSI architectures for multimedia signal processing.

Dajiang Zhou received B.E. and M.E. degrees in electronic engineering from Shanghai Jiao Tong University, China, in 2005 and 2008, respectively. In 2010 he earned his Ph.D. degree in engineering from Waseda University, where he is currently an assistant professor. Dr. Zhou’s major interests are in VLSI architectures and algorithms for multimedia signal processing. He has published over 50 technical papers on journals and conferences including IEEE JSSC, IEEE and IEICE Transactions, ISSCC and VLSI Circuits Symposium, etc. He holds 9 patents in Japan and China. He was a recipient of several awards including the best student paper award of VLSI Circuits Symposium 2010, the design contest award of ISLPED 2010, and the 2010 Chinese Government Award for Excellent Students Abroad.

Peilin Liu received the D.Eng. degree from the University of Tokyo, Tokyo, Japan, in 1998. She was a Researcher with the University of Tokyo, Tokyo, Japan, in 1999. From 1999 to 2003, she was a Senior Researcher with the Central Research Institute of Fujitsu, Tokyo, Japan, where her research topics included multimedia processing, IC design, and high-performance processor architecture. She joined Shanghai Jiao Tong University, Shanghai, China, in 2003, and is currently a Professor with the Department of Electronic Engineering. She directs the MediaSoc Lab and is responsible for a series of projects, such as HDTV SoC platform development and high-performance portable audio/video systems.

Satoshi Goto received the B.E. and the M.E. Degrees in Electronics and Communication Engineering from Waseda University in 1968 and 1970 respectively. He also received the Dr. of Engineering from the same University in 1978. He joined NEC Laboratories in 1970 where he worked for LSI design, Multimedia system and Software as GM and Vice President. Since 2002, he has been Professor, at Graduate school of Information, Production and Systems of Waseda University at Kitakyushu. He served as GC of ICCAD, ASPDAC, VLSI-SOC,ASICON and ISOCC and was a board member of IEEE CAS society. He is IEEE Life Fellow and IEICE Fellow. He is Visiting Professor at Shanghai Jiao Tang University and Tsinghua University of China and Member of Science Council of Japan.