Area-efficient HEVC IDCT/IDST architecture for 8K × 4K video decoding

Hong Liang^{a)}, He Weifeng^{b)}, He Guanghui, and Mao Zhigang

School of Microelectronics, Shanghai Jiao Tong University

a) liang.h.hong@gmail.com

b) hewff@sjtu.edu.cn, Corresponding Author

Abstract: High Efficiency Video Coding (HEVC) is the newest video coding standard beyond H.264/AVC. To more efficiently compress image frames, variable block size DCT/IDCT (from 4×4 to 32×32) as well as 4×4 DST/IDST is employed by HEVC. In this paper, a novel area-efficient IDCT/IDST architecture for Ultra-High Definition (UHD) video applications is proposed. To reduce hardware cost and improve throughput efficiency, a novel resource sharing scheme, a template-based constant multiplication structure and a transpose buffer structure are adopted. Experimental results show that the proposed architecture can address $8K \times 4K$ (7680×4320 , 30 fps) video sequences at 390 MHz with at least a 39.5% gate count savings. Consequently, the proposed architecture offers a cost-efficient solution for future UHD applications.

Keywords: Inverse Discrete Cosine Transform (IDCT), Inverse Discrete Sine Transform (IDST), High Efficiency Video Coding (HEVC), VLSI architecture, video coding

Classification: Integrated circuits

References

- [1] G. J. Sullivan, J.-R. Ohm, W.-J. Han and T. Wiegand: IEEE Trans. Circuits Syst. Video Technol. **22** (2012) 1649. DOI:10.1109/TCSVT.2012.2221191
- [2] M. T. Pourazad, C. Dautre, M. Azimi and P. Nasiopoulos: IEEE Consumer Electron. Mag. **1** [3] (2012) 36. DOI:10.1109/MCE.2012.2192754
- [3] S. Shen, W. Shen, Y. Fan and X. Zeng: Multimedia and Expo (ICME) (2012) 788. DOI:10.1109/ICME.2012.7
- [4] W. Zhao, T. Onoye and T. Song: Proc. IEEE Int. Symp. Circuits Syst. (2013) 1668. DOI:10.1109/ISCAS.2013.6572184
- [5] P. T. Chiang and T. S. Chang: Proc. IEEE Int. Symp. Circuits Syst. (2013) 1006. DOI:10.1109/ISCAS.2013.6572019
- [6] J. Zhu, Z. Liu and D. Wang: Proc. IEEE Int. Symp. Circuits Syst. (2013) 677. DOI:10.1109/ISCAS.2013.6571937
- [7] R. Conceição, J. Cláudio de Souza, Jr., R. Jeske, M. Porto, B. Zatt and L. Agostini: 2014 IEEE International Symposium on Circuits and Systems (2014) 1925. DOI:10.1109/ISCAS.2014.6865537
- [8] E. Kalali, E. Ozcan, O. M. Yalcinkaya and I. Hamzaoglu: IEEE Trans. Consumer Electron. **60** (2014) 754. DOI:10.1109/TCE.2014.7027352
- [9] P. K. Meher, S. Y. Park, B. K. Mohanty, K. S. Lim and C. Yeo: IEEE Trans.

- Circuits Syst. Video Technol. **24** (2014) 168. DOI:10.1109/TCSVT.2013.2276862
- [10] L. Hong, W. He, H. Zhu and Z. Mao: IEICE Electron. Express **10** (2013) 20130210. DOI:10.1587/elex.10.20130210
- [11] Y. Ziyou, H. Weifeng, H. Liang, H. Guanghui and M. Zhigang: 2014 IEEE International Symposium on Circuits and Systems (2014) 2511. DOI:10.1109/ISCAS.2014.6865683

1 Introduction

The HEVC standard [1] is the newest video coding standard developed by MPEG and ITU. By adopting a variety of complex algorithms and flexible coding structures, HEVC provides a 50% greater Bit rate reduction and a higher degree of concurrency when compared to the H.264/AVC standard.

Unlike its predecessors, the HEVC DCT/IDCT tool supports several types of block sizes: 4×4 , 8×8 , 16×16 and 32×32 pels. For a large homogeneous image area, the larger block size of DCT/IDCT may achieve higher compression efficiency in terms of energy compaction and quantization energy reduction. Moreover, HEVC supports the use of the Discrete Sine Transform (DST) and its inverse transform (IDST) in certain intra-prediction modes to obtain higher coding gains on residuals that tend to be larger in value for pixels that are away from the boundaries [2].

Researchers began to pay attention to HEVC DCT/IDCT architecture design in 2012. Based on the traditional butterfly computational structure, S. Shen [3] et al. presented a unified 4/8/16/32-point integer IDCT architecture in 2012 and W. Zhao [4] et al. proposed a multiplierless inverse transform architecture in 2013, for HEVC $4K \times 2K$ video applications. To support different block size IDCT dynamically, P. T. Chiang [5] et al. proposed a reconfigurable inverse transform architecture for HEVC $4K \times 2K$ video decoding in 2013. To support the DCT/IDCT and the Hadamard transform, J. Zhu [6] et al. proposed a fully pipelined transform architecture for $8K \times 4K$ (7680×4320 @30 fps, 4:2:0 format) video applications in 2013. In 2014, R. Conceição [7] et al. proposed a multi-size HEVC IDCT architecture. In the same year, E. Kalali [8] et al. proposed a low energy HEVC IDCT hardware by simplifying the computational complexity of IDCT/IDST algorithm. Moreover, Pramod [9] et al. proposed folded and full-parallel HEVC DCT architectures for $8K \times 4K$ video applications in 2014.

The 1-D transform circuit of above architectures is essentially based on the traditional butterfly computational structure to enable the maximum hardware resource sharing of internal operations. To reduce the hardware costs further, pipelined-based IDCT VLSI architectures for the HEVC standard were proposed in 2013 and 2014 [10, 11] by performing IDCT operations with a fine-grained parallelism.

In this paper, a novel butterfly-based HEVC IDCT architecture is proposed for $8K \times 4K$ video applications by using a novel resource sharing scheme and a template-based constant multiplication structure.

The paper is organized as follows. In Section II, the HEVC IDCT/IDST algorithm is briefly introduced. In Sections III, our proposed area-efficient IDCT/IDST architecture is presented. Experimental results and a performance analysis are discussed in Section IV. Finally, Section V provides concluding remarks.

2 The HEVC IDCT/IDST algorithm

Transform Unit (TU) of HEVC is a basic block that has residual or transform coefficients for integer transform and quantization. The HEVC DCT/IDCT supports four types of TUs: 4×4 , 8×8 , 16×16 and 32×32 pels, and the DST/IDST supports only the 4×4 TU. Based on the texture patterns of image frames, the HEVC codec dynamically chooses the best block type and transform mode (DCT or DST) to improve coding efficiency.

The IDCT can be described as

$$Y = H^T X H \quad (1)$$

where H and X denote the transform coefficient matrix and residual block data. The HEVC IDCT uses a set of integer IDCT transform matrices, including 4×4 , 8×8 , 16×16 and 32×32 . Moreover, the 2D integer inverse transform can be decomposed by row-column decomposition into two One-Dimensional (1-D) inverse transforms. To illustrate the concurrency of the HEVC IDCT algorithm, an 8-point 1D-IDCT is described in (2), where X is the input pixel data and Y is the output data of the 1-D transform. The matrix coefficients $H(i, j)$ ($i, j = 0 \dots 7$) given in (2) have symmetric and anti-symmetric properties.

$$Y = H^T X = \begin{bmatrix} 64 & 89 & 83 & 75 & 64 & 50 & 36 & 18 \\ 64 & 75 & 36 & -18 & -64 & -89 & -83 & -50 \\ 64 & 50 & -36 & -89 & -64 & 18 & 83 & 75 \\ 64 & 18 & -83 & -50 & 64 & 75 & -36 & -89 \\ 64 & -18 & -83 & 50 & 64 & -75 & -36 & 89 \\ 64 & -50 & -36 & 89 & -64 & -18 & 83 & -75 \\ 64 & -75 & 36 & 18 & -64 & 89 & -83 & 50 \\ 64 & -89 & 83 & -75 & 64 & -50 & 36 & -18 \end{bmatrix} \cdot \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} \quad (2)$$

Based on (2), to generate a row of output data (indexed by i ($i = 0 \dots n - 1$)), there are n^2 multiplication operations. Because there is no data dependence among these operations, many architectures have been designed to perform these n^2 operations in parallel by adopting the butterfly computational structure. A butterfly computational structure of a 16-point 1-D HEVC IDCT is shown in Fig. 1.

To reuse internal data, one row or one column data of a residual block are simultaneously launched to calculate the 1-D IDCT results in parallel, as shown in the figure. The butterfly computational structure for 8/4-point integer IDCT is also included in the figure. The 4-point IDCT is marked with a dotted line, and the 8-point IDCT is marked with a dashed line. According to the figure, the 4-point IDCT results can be recursively reused by the 8×8 and 16×16 IDCT. Moreover,

Fig. 1. The butterfly computational structure of 16-point 1-D IDCT.

the 32-point 1-D IDCT has a similar recursive structure. Based on the above computational structure, there are only 8, 24, 88 and 344 multiplication operations for each row of 4/8/16/32-point 1-D IDCT, respectively.

3 The proposed butterfly-based IDCT architecture for 8K × 4K HEVC applications

3.1 The resource sharing scheme of butterfly-based IDCT

The Max. residual Bit rate of IDCT for 8K × 4K video applications is $7680 \times 4320 \times 30 \times 1.5 \times 8$ bit/sec. If the processing rate of the IDCT architecture is 4 pels/cycle, the worst-case real-time work frequency of the circuit is approximately 377.5 MHz. It is feasible to implement this circuit by adopting current popular CMOS technology. As a result, the processing rate of our proposed butterfly-based IDCT circuit is fixed at 4 pels/cycle.

According to Fig. 1, to calculate a row/column of 16-point IDCT, the traditional butterfly computational structure need perform 88 constant multiplications. If the processing rate of the structure is reduced to 4 pels per cycle, it is feasible to perform these 88 multiplication operations in 4 cycles with 22 operations per cycle. As a result, most constant multipliers of the traditional butterfly structure can be removed by a resource sharing scheme.

Taking the 16-point IDCT as an example, the lower left portion of Fig. 1 shows the calculation structure for $O(0)$ to $O(7)$. To generate $O(0)$ to $O(7)$, there are 64 constant multiplication operations with 8 multiplicands of $X(1)$, $X(3)$, $X(5)$, $X(7)$, $X(9)$, $X(11)$, $X(13)$ and $X(15)$. Assuming that the input pixel data of the first cycle are $X(0)$, $X(1)$, $X(2)$ and $X(3)$, $X(1)$ and $X(3)$ will be two of these 8 multiplicands for 16 constant multiplication operations. Consequently, 16 constant multipliers are enough to address these 16 multiplications in parallel.

In next cycle, the input data are $X(4)$, $X(5)$, $X(6)$ and $X(7)$. $X(5)$ and $X(7)$ are also two multiplicands of the new 16 constant multiply operations. In this cycle, the above 16 constant multipliers can be reused by multiplicand $X(5)$ and $X(7)$ for these 16 multiplications. Moreover, these 16 multipliers can also be reused by $X(9)$, $X(11)$, $X(13)$ and $X(15)$. Adopting this reuse scheme, a total of 48 constant multipliers of the traditional 16-point butterfly structure can be removed. Further-

more, constant multipliers for the multiplicand $X(6)$, $X(10)$ and $X(14)$ can also be removed by sharing the multipliers of the multiplicand $X(2)$.

Using the above resource sharing scheme, 60 constant multipliers and many adders of the traditional 16-point, butterfly-based structure can be removed. Adopting the above resource sharing scheme, one-half and seven-eighths of the constant multipliers can be removed from the 8-point and 32-point butterfly computational structures, respectively.

3.2 The proposed butterfly-based 2-D IDCT/IDST architecture

Based on the above resource sharing scheme, we proposed our butterfly-based 2-D IDCT/IDST architecture. Based on the traditional row-column decomposition approach, the proposed 2-D architecture also employs two 1-D IDCT/IDST architectures and a transpose buffer as shown in Fig. 2.

Fig. 2. The 2-D inverse transform architecture.

Fig. 3 shows the details of the 1-D butterfly-based IDCT architecture. The 4-Point IDCT/IDST Unit of Fig. 3 is used to address 4-point IDCT/IDST operations in parallel. The C4, C4+, C16+ and C16 Units are four different types of constant multiplier units used for constant multiplication operations. The C4, C4+, C16+ and C16 Units can address 4, 8, 16 and 16 constant multiplication operations in parallel, respectively. The Butterfly Addition Unit is employed to address 8-, 16- and 32-point butterfly add operations in parallel.

When the size of the current block is 4×4 pels, one row/column of the block data is delivered to the 4-Point IDCT/IDST Unit in each cycle. Then, the 1-D architecture generates four output results, y_0 to y_3 , in every cycle. Under these circumstances, other units are bypassed.

When the current block is 8×8 pels, the proposed architecture requires two cycles to perform one row/column of IDCT operations. In the first cycle, two odd input pixel data are separately delivered to the C4 and C4+ Units for constant multiplication operations. Then, 8 related products are accumulated by the accumulators $\sum y_4 \sim \sum y_7$. Simultaneously, two even input data are delivered to the Serial-to-Parallel Unit. In the second cycle, another two odd input data are delivered to the C4 and C4+ Units, and the related products are accumulated.

Meanwhile, two even input data are delivered to the Serial-to-Parallel Unit, and all four residual data in this unit are then delivered to the 4-point IDCT/IDST Unit in parallel. Then, the internal results of y_0 to y_3 and $\sum y_4$ to $\sum y_7$ are sent to the Butterfly Addition Unit to generate the final 8 output results. Finally, the Parallel-to-Serial Unit outputs these results in two cycles. Meanwhile, these two cycles for data output can be concealed in the next row/column computational cycles for a higher throughput.

Fig. 3. The proposed butterfly-based 1-D IDCT/IDST architecture.

When the current block is 16×16 pels, the proposed architecture requires four cycles to calculate one row/column of a 16-point IDCT. In each cycle, the C4, C4+ and C16+ Units separately obtain one input data from the Residual Input Ports for the constant multiplication operations. Simultaneously, 20 products are generated in every cycle. These products are accumulated by the accumulators $\sum y4$ to $\sum y15$. Moreover, in each cycle, one input data is delivered to the Serial-to-Parallel Unit for data buffering. After four cycles, $y0$ to $y3$ and $\sum y4$ to $\sum y15$ are delivered to the Butterfly Addition Unit and to the Parallel-to-Serial Unit to output these 16 results in the following four cycles.

The calculation process of the 32×32 IDCT is similar to that of the 16×16 IDCT. However, the proposed 1-D architecture requires eight cycles to perform each row/column calculation of the 32-point IDCT. In each cycle, one input residual is allocated to the C4+, C16+ and C16 Units for the constant multiplications. Then, the remaining input data are sent to the C4 Unit and to the Serial-to-Parallel Unit alternately. Under these circumstances, all units of the architecture are used for the 32×32 IDCT.

Fig. 4 illustrates the data flow of the 16- and 32-point IDCT. “S-P Unit” denotes the Serial-to-Parallel Unit. The input pixel data are scheduled as shown in the second row of the figure. Rows 3 to 6 and 11 to 15 of the table show the input data allocation for the 16- and 32-point IDCT, respectively. Furthermore, the data flow of product accumulation in accumulators $\sum y4$, $\sum y7$, $\sum y8$, $\sum y15$, $\sum y16$ and $\sum y31$ are also shown in Fig. 4. When performing the 32-point IDCT, the proposed architecture requires 8 cycles to address one row/column of input pixel data. Using the proposed calculation scheduling scheme, the proposed IDCT architecture can dynamically address IDCT operations for different block sizes with a high hardware utilization as well as a fixed processing rate of 4 pels/cycle.

Cycle	0	1	2	3	4	5
Input data	$x(0)x(1)x(2)x(3)x(4)x(5)x(6)x(7)x(8)x(9)x(10)x(11)x(12)x(13)x(14)x(15)$	$x(16)x(17)x(18)x(19)x(20)x(21)x(22)x(23)$				
16-point IDCT	S-P Unit	$x(0)$	$x(4)$	$x(8)$	$x(12)$	$x(16)$
	C4 Unit	$x(2)$	$x(6)$	$x(10)$	$x(14)$	$x(18)$
	C4 ⁺ Unit	$x(1)$	$x(5)$	$x(9)$	$x(13)$	$x(17)$
	C16 ⁺ Unit	$x(3)$	$x(7)$	$x(11)$	$x(15)$	$x(19)$
	Σy_4	=0	$+89x(2)$	$+75x(6)$	$+50x(10)$	$+18x(14)$
	Σy_7	=0	$+18x(2)$	$-50x(6)$	$+75x(10)$	$-89x(14)$
	Σy_8	=0	$+90x(1)+87x(3)$	$+80x(5)+70x(7)$	$+57x(9)+43x(11)$	$+25x(13)+9x(15)$
	Σy_{15}	=0	$+9x(1)-25x(3)$	$+43x(5)-57x(7)$	$+70x(9)-80x(11)$	$+87x(13)-90x(15)$
32-point IDCT	S-P Unit	$x(0)$	0	$x(8)$	0	$x(16)$
	C4 Unit	0	$x(4)$	0	$x(12)$	0
	C4 ⁺ Unit	$x(2)$	$x(6)$	$x(10)$	$x(14)$	$x(18)$
	C16 ⁺ Unit	$x(1)$	$x(5)$	$x(9)$	$x(13)$	$x(17)$
	C16 Unit	$x(3)$	$x(7)$	$x(11)$	$x(15)$	$x(19)$
	Σy_4	=0	+0	$+89x(4)$	+0	$+75x(12)$
	Σy_7	=0	+0	$+18x(4)$	+0	$-50x(12)$
	Σy_8	=0	$+90x(2)$	$+87x(6)$	$+80x(10)$	$+70x(14)$
	Σy_{15}	=0	$+9x(2)$	$-25x(6)$	$+43x(10)$	$-57x(14)$
	Σy_{16}	=0	$+90x(1)+90x(3)$	$+88x(5)+85x(7)$	$+82x(9)+78x(11)$	$+73x(13)+67x(15)$
	Σy_{31}	=0	$+4x(1)-13x(3)$	$+22x(5)-31x(7)$	$+38x(9)-46x(11)$	$+54x(13)-61x(15)$
					$+67x(17)-73x(19)$	

Fig. 4. The data flow of the proposed 1-D architecture for 16/32-point IDCT.

3.3 The structure of the template-based constant multiplier

According to the HEVC standard, there are 29 different coefficients in coefficient matrix for IDCT and 4 coefficients for IDST. To support these 29 constant multiplications in HEVC IDCT, the proposed architecture employs four constant multiplier units: C4, C4⁺, C16⁺ and C16. Table I shows all the constant coefficients supported by these units. Each multiplier unit has its own constant coefficients, and all constant multiplication operations of each unit are executed in parallel (based on the size of the current block).

Table I. The Constants supported by multiplier units

Multiplier unit	Constant coefficients of HEVC IDCT	Template coefficients
C4	89,75,50,18	1,11,101,1001
C4 ⁺	89,75,50,18,90,87,80,70,57,43,25,9	1,11,101,111,1001
C16 ⁺	90,88,87,85,82,80,78,73,70,67,61,57,54,46,43,38,31,25,22,13,9,4	1,11,101,111,1001
C16	90,88,85,82,78,73,67,61,54,46,38,31,22,13,4	1,11,101,1001

Based on the template multiplication scheme presented in reference [11], the above constant multiplication operations are decomposed into several types of template multiplication operations and several additional add/shift operations. According to the Table, the C4 supports four kinds of constant multiplications, 89x, 75x, 50x and 18x. To generate their template coefficients of these multiplications, 89x, 75x, 50x and 18x can be decomposed into the following template operations:

$$89x = x + x \ll 3 + x \ll 4 + x \ll 6 = (x \times 1001) + (x \times 101) \ll 4 \quad (3)$$

$$75x = x + x \ll 1 + x \ll 3 + x \ll 6 = x \times 11 + (x \times 1001) \ll 3 \quad (4)$$

$$50x = x \ll 1 + x \ll 4 + x \ll 5 = x \ll 1 + (x \times 11) \ll 4 \quad (5)$$

$$18x = x \ll 1 + x \ll 4 = (x \times 1001) \ll 1 \quad (6)$$

As a result, 1, 11, 101 and 1001 can be defined as template coefficients for the constant multiplications of $89x$, $75x$, $50x$ and $18x$. Based on this scheme, all template coefficients of all units are defined as shown in right part of the Table I.

According to the Table I, the C4 and C16 Units as well as the C4+ and C16+ Units support the same template multiplication operations, respectively. The structures of the C4, C4+, C16 and C16+ Units are shown in Fig. 3. Shifters and adders are employed by the Template Units to perform the above template multiplication operations. Adder0~Adder15 are shift-adders employed to perform additional shift and add operations. Fig. 5 shows the detailed structure of the Templates C4, C4+, C16+ and C16.

Compared with traditional constant multipliers, the C4, C4+, C16+ and C16 Units employ 7, 12, 20 and 19 adders to perform 4, 8, 16 and 16 constant multiplications in parallel with 22%, 33%, 62% and 50% adder resource savings, respectively.

Fig. 5. The architecture of the Template unit.

3.4 The structure of the Transpose Buffer

The structure and data mapping scheme of the Transpose Buffer is shown in Fig. 6. The proposed Transpose Buffer is divided into two buffers to buffer and transpose data between two 1-D IDCT/IDST modules, as shown in Fig. 6(a). Adopting a Ping-Pong mode, each buffer is alternately used as a write buffer or read buffer per 256 cycles. In each buffer, there are four single-port memories named from B0 to B3, and the size of each memory is 256 words.

Because the largest size of an IDCT block is 32×32 pels, the 32×32 block is defined as a major processing block. The current block for IDCT is just a sub-block of a major processing block. The major processing block is then partitioned into 64 4×4 blocks and a label is assigned to each 4×4 block in a column scan order. Fig. 6(b) shows the partition structure of the major processing block. According to the position of the current block at the major processing block and the partition structure of the Fig. 6(b), each current block can also be partitioned into 4×4 blocks and each 4×4 block has a unique label.

Fig. 6. The structure and data mapping scheme of the Transpose Buffer.

The two buffers have the same size and each buffer is divided into 64 basic Units. Each Unit has 16 words and it is used to store a 4×4 block according its label. Fig. 6(c) and Fig. 6(d) show the structure of a buffer and the address mapping method between labels and Units.

After the column IDCT, the current block (output data of the column IDCT) is split into 4×4 blocks to be delivered to different Units in accordance with their labels. Fig. 6(e)–(g) is an example to show the data flow of the write buffer and read buffer. To avoid data access conflict, to every output data $y(i, j)$ from the column IDCT, the memory allocation algorithm for the write buffer is described as (7), where $0 \leq i, j \leq 31$.

$$\begin{cases} y(i, j) = B0, & \text{if } (i\%4, j\%4) \in \{(0, 0), (1, 3), (2, 2), (3, 1)\} \\ y(i, j) = B1, & \text{if } (i\%4, j\%4) \in \{(0, 1), (1, 0), (2, 3), (3, 2)\} \\ y(i, j) = B2, & \text{if } (i\%4, j\%4) \in \{(0, 2), (1, 1), (2, 0), (3, 3)\} \\ y(i, j) = B3, & \text{if } (i\%4, j\%4) \in \{(0, 3), (1, 2), (2, 1), (3, 0)\} \end{cases} \quad (7)$$

Fig. 6(e) shows data of an 8×8 block. These data are delivered to the write buffer in a column scan order. However, these data are stored in different Units of the write buffer. Moreover, in each Unit, the data of a 4×4 sub-block are stored in a row scan order. Fig. 6(f) shows the storage format of the blocks labeled (r) and $(r + 1)$ in Unit (r) and Unit $(r + 1)$.

After 256 cycles, the write buffer is full and it is renamed as the read buffer for row IDCT. The data access flow of the read buffer is described in bold type as

shown in Fig. 6(g). The input data for row IDCT may be stored in different Units. It is easy to fetch these data according to their labels. Moreover, the data in each Unit need to be rearranged. The format of the rearranged data in Unit r can be described as:

$$\begin{cases} (y(0,0), y(0,1), y(0,2), y(0,3)) = (B0[s], B1[s+1], B2[s+2], B3[s+3]) \\ (y(1,0), y(1,1), y(1,2), y(1,3)) = (B1[s], B2[s+1], B3[s+2], B0[s+3]) \\ (y(2,0), y(2,1), y(2,2), y(2,3)) = (B2[s], B3[s+1], B0[s+2], B1[s+3]) \\ (y(3,0), y(3,1), y(3,2), y(3,3)) = (B3[s], B0[s+1], B1[s+2], B2[s+3]) \end{cases} \quad (8)$$

Where $(s = r \times 4, 0 \leq r \leq 63)$.

During the IDCT process, four 16-bit intermediate results from the column transform in each cycle are delivered to the write buffer. Moreover, four intermediate results can be read from the read buffer for row IDCT without any access conflict.

4 Experimental results and performance analysis

The proposed 2-D IDCT/IDST architecture has been designed with Verilog HDL and implemented in SMIC 65 nm CMOS process. Fig. 7 shows the chip layout of the architecture. The core size of the butterfly-based 2-D IDCT circuit is $0.68 \text{ mm} \times 0.68 \text{ mm}$ including eight single-port SRAMs. Post simulation results show that the maximum work frequency of the butterfly-based architecture is approximately 435 MHz. The word length of the input/output pixels of 1-D IDCT is 16 bits, and the gate count is normalized by the area of the 2-input NAND gate.

Fig. 7. Layout of proposed 2-D IDCT.

The detailed VLSI costs of the proposed circuit is illustrated in Table II. Ours* is our proposed butterfly-based 2-D IDCT/IDST architecture. The hardware overhead of its 1-D butterfly-based IDCT architecture is 71.2 K gates. The simulation results show that the proposed architecture is able to address real-time HEVC IDCT/IDST of $7680 \times 4320@30 \text{ fps}$ video sequences at 390 MHz on average.

Table III shows hardware cost comparisons of our proposed architecture with other HEVC 1-D IDCT/IDST architectures. To evaluate the processing efficiency of HEVC IDCT architectures in real-time applications, *the throughput efficiency is*

Table II. Hardware costs of the proposed architecture

Architecture	Ours*
Technology	65 nm
Gate Counts	183.6 K
Processing rate (pels/cycle)	4
Max. frequency (Hz)	435 M
Max. throughput (pels/s)	1740 M
Memory size (bits)	2048 × 16
Power (mW)	82.7

Table III. Comparisons with previous HEVC 1-D DCT/IDCT architectures

Design	[6]	[7]	[8]	[10]	Ours*
Function					
IDST				✓	✓
IDCT				✓	✓
Technology	90 nm	90 nm	90 nm	65 nm	65 nm
Max. Speed (MHz)	311	370	150	480	435
Max. Processing Rate (pels/cycle)	32	32	32	4	4
Real-time Frequency (MHz)	47	237.9	150	171	390
Max. Bandwidth (pels/s)	9.95 G	7.6 G	4.8 G	1.92 G	1.74 G
Throughput Efficiency	25.1%	19.6%	7.78%	55.2%	95.7%
1-D Gate Counts (gates)	117.7 k	119.1 k	133.1 k	60.5 k	71.2 k
Performance per unit area	9.1	1.7	1	1.2	1.8

defined as the ratio of necessary throughput for real-time applications over real-time available throughput. The real-time available throughput is equal to the product of the real-time work frequency and the maximum processing rate of the architecture. The real-time necessary throughput is the worst-case throughput of the application. For an $8K \times 4K$ video application (7680×4320 @30 fps, 4:2:0 format), the real-time necessary throughput is approximately 1.51 G pels/second. Concerning the hardware efficiency of a design, we adopt a performance index called *performance per unit area*, which is defined as the ratio of data throughput over the product of hardware cost and work frequency. Assuming that the performance per unit area of the design of [8] is normalized to 1.

The designs of [8] and [10] are focused on $4K \times 2K$ video applications. Compared with the designs of [8], our proposed 1-D IDCT architecture achieves over 10x improvements in throughput efficiency, 0.8x improvements in performance per unit area while consuming only 53.5% of the hardware costs. The design of [10] is the first full-pipelined IDCT/IDST architecture. Compared with the design of [10], Ours* achieves an 73.4% throughput efficiency improvements and 0.49x improvements in performance per unit area.

The designs of [6, 7] and Ours* are focused on $8K \times 4K$ video applications. Compared with our proposed architecture, the design of [6] achieves 4.06x improvements in performance per unit area. However, for $8K \times 4K$ @30 fps applications, its real-time work frequency is just approximately 47 MHz. At the same time, the maximum work frequency is up to 311 MHz. Consequently, performance is not a major bottleneck of this design. Considering the current chip fabrication

technologies, An IDCT/IDST architecture with a higher real-time frequency and lower hardware cost is more valuable.

Compared with the design of [6] and [7], the 1-D architecture of Ours* achieves 2.8x and 3.9x improvement in throughput efficiency, 39.5% and 40.2% reduction in hardware costs, respectively.

Theoretically, the 1-D architecture of Ours* can achieve a throughput efficiency of 100%. As a result, a higher throughput efficiency of our proposed architecture can be achieved via design optimization.

In summary, our 1-D architecture achieves the lowest hardware costs for $8K \times 4K$ real-time video applications. In addition, it is feasible for chip implementation using current, popular CMOS technologies.

5 Conclusion

In this paper, we proposed an area-efficient HEVC 2-D IDCT/IDST VLSI architecture that support a variety of transform sizes, from 4×4 to 32×32 pels. Compared with previous works, our butterfly-based architecture achieves at least a gate count savings of 39.5% (1-D architecture) while retaining a throughput efficiency of 95.7%. Consequently, it offers an area-efficient solution for $8K \times 4K$ applications.

Acknowledgments

This work was supported by the Hi-Tech Research and Development Program (863) of China under Grant No. 2012AA012702.