REFERENCES

[1] J. Mietzner, R. Schober, L. Lampe, W. H. Gerstacker, and P. A. Hoeher, "Multiple-antenna techniques for wireless communications—A comprehensive literature survey," *IEEE Commun. Survey Tuts.*, vol. 11, no. 2, pp. 87–105, Jun. 2009.

[2] M. O. Damen, A. Chkeif, and J.-C. Belfiore, "Lattice code decoder for space-time codes," *IEEE Commun. Lett.*, vol. 4, no. 5, pp. 161–163, May 2000.

[3] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[4] S. Bäro, J. Hagenauer, and M. Witzke, "Iterative detection of MIMO transmission using a list-sequential (LISS) detector," in *Proc. IEEE Int. Conf. Commun.*, May 2003, pp. 2653–2657.

[5] K.-W. Wong, C.-Y. Tsui, R. S.-K. Cheng, and W.-H. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proc. IEEE ISCAS*, May 2002, pp. 273–276.

[6] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.

[7] C. Studer, A. Burg, and H. Bolcskei, "Soft-output sphere decoding: Algorithms and VLSI implementation," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 2, pp. 290–300, Feb. 2008.

[8] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner, "K-best MIMO detection VLSI architectures achieving up to 424 Mbps," in *Proc. IEEE ISCAS*, May 2006, pp. 1151–1154.

[9] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 3, pp. 491–503, Mar. 2006.

[10] S. Chen, T. Zhang, and Y. Xin, "Relaxed K-best MIMO signal detector design and VLSI implementation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 3, pp. 328–337, Mar. 2007.

[11] M. Shabany and P. G. Gulak, "Scalable VLSI architecture for K-best lattice decoders," in *Proc. IEEE ISCAS*, May 2008, pp. 940–943.

[12] H.-L. Lin, R. C. Chang, and H.-L. Chen, "A high-speed SDM-MIMO decoder using efficient candidate searching for wireless communication," *IEEE Trans. Circuit and Syst. II, Exp. Briefs*, vol. 55, no. 3, pp. 289–293, Mar. 2008.

[13] M. Shabany and P. G. Gulak, "A 0.13 μm CMOS 655 Mb/s 4×4 64-QAM K-best MIMO detector," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2009, pp. 256–257.

[14] C.-H. Yang and D. Marković, "A flexible DSP architecture for MIMO sphere decoding," *IEEE Trans. Circuit Syst. I, Reg. Papers*, vol. 56, no. 10, pp. 2301–2314, Oct. 2009.

[15] C.-H. Liao, T.-P. Wang, and T.-D. Chiueh, "A 74.8 mW soft-output detector IC for 8×8 spatial-multiplexing MIMO communications," *IEEE J. Solid-State Circuits*, vol. 45, no. 2, pp. 411–421, Feb. 2010.

[16] M. Shabany, K. Su, and P. G. Gulak, "A pipelined scalable high-throughput implementation of a near-ML K-best complex lattice decoder," in *Proc. IEEE ICASSP*, Mar./Apr. 2008, pp. 3173–3176.

[17] B. M. Hochwald and S. Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[18] C.-J. Huang, C.-W. Yu, and H.-P. Ma, "A power-efficient configurable low-complexity MIMO detector," *IEEE Trans. Circuit Syst. I, Reg. Papers*, vol. 56, no. 2, pp. 485–496, Feb. 2009.

[19] P.-Y. Tsai, W.-T. Chen, X.-C. Lin, and M.-Y. Huang, "A 4×4 64-QAM reduced-complexity K-best MIMO detector up to 1.5 Gbps," in *Proc. IEEE ISCAS*, May/Jun. 2010, pp. 3953–3956.

[20] L. Liu, F. Ye, X. Ma, T. Zhang, and J. Ren, "A 1.1-Gb/s 115-pJ/bit configurable MIMO detector using 0.13-μm CMOS technology," *IEEE Trans. Circuit Syst. II, Exp. Briefs*, vol. 57, no. 9, pp. 701–705, Sep. 2010.

[21] C.-A. Shen and A. M. Eltawil, "A radius adaptive K-best decoder with early termination: Algorithm and VLSI architecture," *IEEE Trans. Circuit Syst. I, Reg. Papers*, vol. 57, no. 9, pp. 2476–2486, Sep. 2010.

[22] T.-W. Kim and I.-C. Park, "Small-area and low-energy K-best MIMO detector using relaxed tree expansion and early forwarding," *IEEE Trans. Circuit Syst. I, Reg. Papers*, vol. 57, no. 10, pp. 2753–2761, Oct. 2010.

[23] M. Shabany and P. G. Gulak, "A 675 Mbps, 4×4 64-QAM K-best MIMO detector in 0.13 μm CMOS," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 135–147, Jan. 2012.

Single-Port SRAM-Based Transpose Memory With Diagonal Data Mapping for Large Size 2-D DCT/IDCT

Qing Shang, Yibo Fan, Weiwei Shen, Sha Shen, and Xiaoyang Zeng

***Abstract*— This brief describes a new method to implement the single-port SRAM-based transpose memory for large size discrete cosine transform (DCT)/indiscrete cosine transform (IDCT) which are used in the latest video coding standard, such as high efficiency video coding. Instead of shift-register array or multiport SRAM, only single-port SRAM is used in the proposed design. A new diagonal data mapping scheme is proposed to reduce the number of SRAM banks used to implement the transpose memory. This design can be flexibly extended to support DCT/IDCT of different transform sizes and different data throughput rates. To support larger size DCT/IDCT, only the depth of SRAM needs to be increased. To support different data throughput rate, multiple SRAM banks are well organized according to the required throughput. Row access and column access can be perfectly supported under single port SRAM. The equivalent gate count per bit (EGC) of proposed approach is less than two, which is much more efficient than the previous method. It is suitable for real-time processing of the video with the resolution up to 1080P HD or even higher.**

***Index Terms*— Discrete cosine transform (DCT)/indiscrete cosine transform (IDCT), high efficiency video coding (HEVC), single-port SRAM, transpose memory.**

I. INTRODUCTION

Discrete cosine transform (DCT) is considered as the suboptimal transform due to its good ability to concentrate the signal energy. 2-D DCT/ indiscrete cosine transform (IDCT) have been widely used in almost every block-based image or video coding standards such as JPEG, MPEG-1/2/4, ITU-T H.264/MPEG-4 AVC, VC-1, AVS, and the latest video coding standard high efficiency video coding (HEVC). Traditional image or video coding standards like JPEG, MPEG-1/2/4 use 8×8 floating point 2-D DCT while latter standards like H.264, VC-1, AVS, and HEVC use $4 \times 4/8 \times 8$ integer DCT to reduce computational complexity and avoid the data mismatch between the encoder and the decoder. To further improve the coding efficiency, 16×16 and 32×32 2-D integer DCT are also used in HEVC.

Various methods have been proposed to implement floating point or integer 2-D DCT/IDCT. They can be categorized into two types: the row–column decomposition method (RCDM) or the direct 2-D method. In [1], a fast algorithm is proposed to implement $N \times N$ floating-point 2-D DCT by using the direct 2-D method. In [2], another optimized direct 2-D is used to implement 8×8 floating-point 2-D DCT. However, the fast algorithms used in [1] and [2] cannot be

Manuscript received May 29, 2013; revised September 21, 2013; accepted December 9, 2013. Date of publication January 2, 2014; date of current version October 21, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant 61306023, in part by the Specialized Research Fund for the Doctoral Program of Higher Education under Grant 20120071120021, in part by STCSM under Grant 13511503400, and in part by the National High Technology Research and Development Program (863) under Grant 2012AA012001.

The authors are with the State Key Laboratory of ASIC and System, Fudan University, Shanghai 201203, China (e-mail: 11212020039@fudan.edu.cn; fanyibo@fudan.edu.cn; 10110720024@fudan.edu.cn; 10110720050@fudan.edu.cn; xyzeng@fudan.edu.cn).

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

Digital Object Identifier 10.1109/TVLSI.2013.2295116

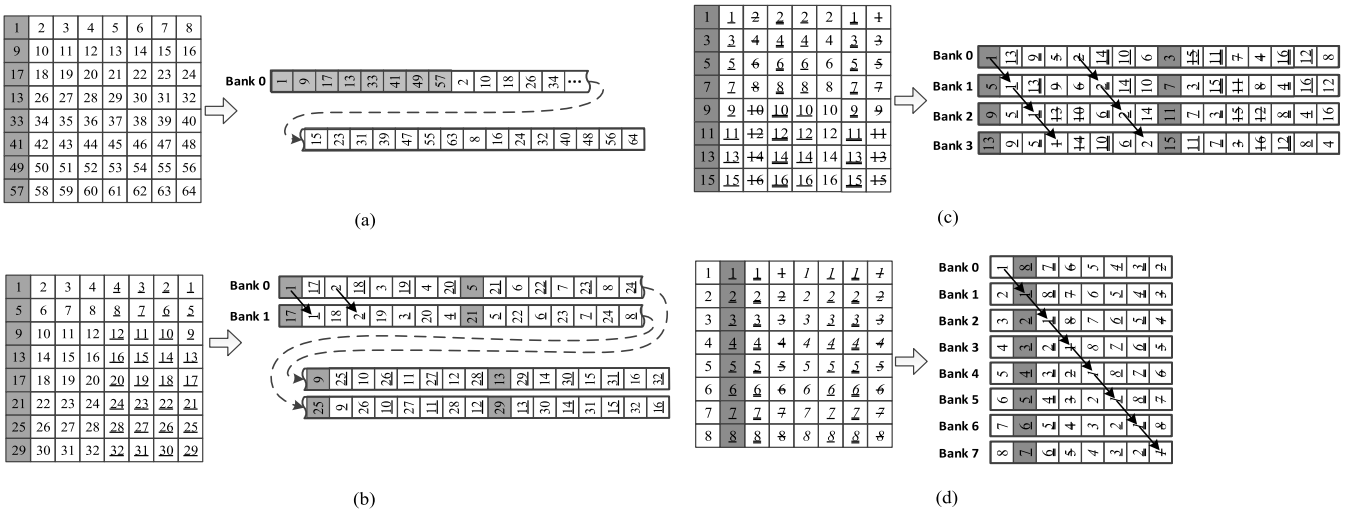

Fig. 1. New spiral data mapping scheme for 8 × 8 2-D IDCT with different write throughput (WTP). (a) RTP/WTP = 1-sample/cycle. (b) RTP/WTP = 2-sample/cycle. (c) RTP/WTP = 4-sample/cycle. (d) RTP/WTP = 8-sample/cycle.

applied to integer 2-D DCT because the transform matrix of integer 2-D DCT is no longer orthogonal and the integer transform matrix cannot be decomposed in the same way as the floating-point one. A new direct 2-D method for 4 × 4 integer DCT of H.264 is proposed in [3]. However, the hardware cost of the computational logic and the interconnection in these designs will increase for larger exponentially size DCT, which make the VLSI implementation unaffordable in case of 32 × 32 2-D transform. RCDM is a more cost-effective way to implement large size 2-D transform.

The process of RCDM-based 2-D DCT/IDCT can be decomposed into two separate steps. First, the row-direction 1-D DCT/IDCT is performed for the input signal. Then the column-direction 1-D DCT/IDCT is performed for the intermediate results of the row-direction transform.

According to different VLSI implementation, RCDM can also be categorized into two types: transpose memory-based RCDM and transpose memory free RCDM. A systolic array without matrix transposition is proposed to compute floating-point 2-D DCT/IDCT in [4]. The hardware cost of this design is in proportion to N^2 for $N \times N$-point 2-D DCT/IDCT. Therefore, it is not suitable for large size 2-D transform. Compared with [4], a more area efficient architecture without transpose memory is proposed in [5]. However, the designs in [5] can only support floating-point transform. The trigonometric decomposition presented in [5] cannot be applied to integer 2-D DCT/IDCT. Transpose memory-based RCDM can support both floating point and integer 2-D DCT/IDCT with a relative low-hardware cost. Many designs using transpose memory-based RCDM have been proposed to support floating-point or integer 2-D DCT/IDCT in [6]–[14]. The transpose memories in these designs are implemented in different ways. A shift-register array-based transpose memory is proposed for floating-point 8 × 8 2-D IDCT in [6]. The same shift-register array architecture is also used in [7] to implement the integer 2-D 4 × 4 DCT/IDCT of H.264. The register array can be shifted in both row and column direction. The intermediate results of row transformation are shifted into the register array in row direction. After the row transform is completed, the results stored in register array are shifted out in column direction.

The experimental results in Section III will show that the register array is not area efficient for large size transpose memory if compared with SRAM. During each access to the register array, each bit of the register array needs to be shifted in horizontal or vertical direction. The register array is also not power efficient. The so-called double buffers architecture is proposed for 8 × 8 2-D IDCT in [8]. Four triple-port register files are used in this design. In [9] and [10], an SDRAM-based architecture is proposed for large size matrix transposition. Due to the large and unpredictable access latency of SDRAM, this architecture is not suitable for high performance real-time video processing systems. A self-aligned transpose memory is proposed for MPEG-4 shape-adaptive DCT (SA-DCT) in [11]. Single-port SRAM is used in this design. The transpose memory is physically divided into four banks. The data throughput of this transpose memory is only two samples per clock cycle. Higher throughput is desired for the regular DCT/IDCT used in video coding process. A dual-port SRAM-based transpose memory is proposed in [12] and [13], and the transpose memory in [12] or [13] is divided into several banks. The more banks are used in the transpose memory, the more hardware area overhead will occur. The data mapping in transpose memory can be optimized to reduce the number of SRAM banks. A single-port SRAM-based transpose memory is used in [14], but for the 32 × 32 IDCT/DCT, it's write through is only 4-sample/cycle. In addition, the width of the SRAM in [14] is 512 bit, which is not area efficient.

To address the above problems, we proposed a single-port SRAM-based transpose memory. The proposed architecture is flexible to support DCT/IDCT of different transform sizes and different data throughputs. Larger size transform can be easily supported by the increase of SRAM depth. The transpose memory is physically divided into several banks according to the required data throughput. Higher data throughput can be achieved by dividing the transpose memory into more banks and the proposed diagonal data mapping scheme.

The rest of this brief is organized as follows. In Section II, the proposed architecture for the transpose memory is described in more details. Experimental results and comparison are given in Section III. Finally, the conclusion of this brief is drawn in Section IV.

II. Proposed Architecture for Single-Port SRAM-Based Transpose Memory

Compared with the register array, it is a more cost-effective approach to implement large size transpose memory by SRAM. The disadvantage of SRAM is that SRAM can only be accessed in row direction. The transpose method in [6]–[8] cannot be applied to the SRAM-based transpose memory because the both row and column

accesses to the memory block are required in these designs. The intermediate results of row transform are written into the transpose memory in row direction and are read out by column transform in column direction.

In this section, we will propose a new diagonal data mapping scheme. The intermediate results of the transform are written into the transpose memory in diagonal direction instead of row direction. When the column transform starts, the intermediate results in a column could be fetched out from different SRAM banks in parallel. The explicit data transposition is avoided by rearrange the read/write order. The similar method is also used in the motion estimation search window memory [15]. However, the propose data mapping scheme is more universal and flexible, it can be used in the transpose memory with different throughput and size.

As an example, the transpose memory architectures which can support 8 × 8 2-D IDCT with four different read and write throughputs are shown in Fig. 1. The read and write throughputs of the architectures shown in Fig. 1(a)–(d) are 1/2/4/8-sample per cycle, respectively. The arrows in Fig. 1 indicate the data mapping direction.

The data mapping scheme for the transpose memory whose write throughput is 1-sample per cycle is shown in Fig. 1(a). The numbers (1–64) in the left side of Fig. 1(a) are the cycle numbers at which the corresponding intermediate results are generated. For a 2-D 8 × 8 IDCT, all eight 8-point 1-D row transforms are done in 64 cycles. The depth of the SRAM used in Fig. 1(a) is 64-word. The width of each word is 16-bit (assuming the width of an intermediate result is 16-bit and one intermediate results is stored in one word). At cycle 1, the first result is generated and stored in the SRAM word with address 0. The second result is generated in cycle 2 and it is stored in the word with address eight. After 64 clock cycles, eight intermediate results (1, 9, 17, 25, 33, 41, 49, and 57) are stored in consecutive SRAM words with addresses ranging from 0 to 7. Eight intermediate results (2, 10, 18, 26, 34, 42, 50, and 58) are stored in consecutive SRAM addresses (8–15). Other intermediate results are also stored in the same way as shown in Fig. 1(a). In such a writing order, the data transposition is achieved. During the column transform process, one SRAM word can be read out at each cycle.

The architecture for the transpose memory whose read/write throughput is 2-sample per cycle is shown in Fig. 1(b). If the butterfly based fast algorithm is used for 1-D IDCT, two results of row transform are generated at each cycle. These two results are symmetric along with the column direction, which is shown in the left diagram of Fig. 1(b). The numbers (1–32) means the clock cycle number at which the corresponding results are generated. The two results need to be stored into SRAM in diagonal direction within one cycle. However, only one SRAM word can be accessed in any single clock cycle due to the intrinsic physical structure of SRAM. To accommodate this requirement, the transpose memory is divided into two banks which are shown in the right side of Fig. 1(b). The width of each SRAM bank is 16-bit (width of one intermediate results) while the depth is 32-word. At each clock cycle, two row transform results are separately stored into the corresponding SRAM banks. To distinguish the two results which are generated in the same cycle, one number is marked by an underscore while the other is not changed. At cycle 1, two results (1 and 1) are generated. The result 1 is stored in the SRAM bank 0 with word address of 0 and the result 1 is stored in the SRAM bank 1 with word address of 1. The two results are stored in a spiral direction. After 32 cycles, all the 64 results are stored in two separate SRAM banks and each bank contains 32 results. When column transform is in progress, the eight results in the same column are fetched out in four cycles. As shown in Fig. 1(b), the eight intermediate results (1, 5, 9, 13, 17, 21, 25, and 29) in first column are shown in gray blocks. Results 1, 5, 9, and 13 are stored in SRAM bank 0 and the corresponding addresses are 0, 8, 16, and 24, respectively. Results 17, 21, 25, and 29 stored in SRAM bank1 and the corresponding addresses are 0, 8, 16, and 24, respectively. The eight intermediate results in the same column can be fetch out in four cycles so that the throughput is 2-sample/cycle.

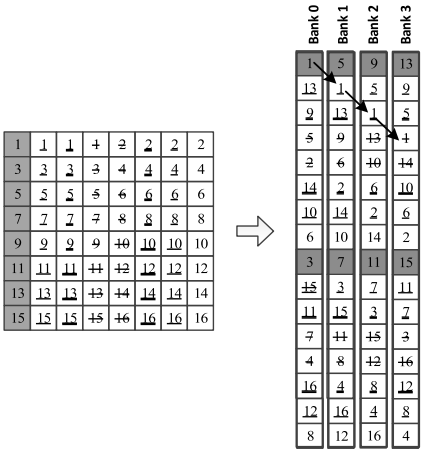

Fig. 2. New data mapping scheme for 8 × 8 2-D DCT with 4-sample/cycle write throughput.

The same data mapping scheme can be easily extended to any different throughput and any large size transform. To support the write throughput of TP-sample per clock cycle (where TP is an integer), the transpose memory needs to be physically divided into TP banks. The architectures to support 8 × 8 2-D IDCT with 4/8-sample/cycle read/write throughput are shown in Fig. 1(c) and (d). Four SRAM banks are used in Fig. 1(c) and eight SRAM banks are used in Fig. 1(d). To distinguish the multiple output results generated in the same clock cycle, the numbers in Fig. 1(c) and (d) are marked with underscore, double underscore, delete line, or italic font. As shown in Fig. 1(d), the eight results in the second column are marked as gray blocks. Each result is stored into one of the eight different SRAM banks, so eight results can be easily fetched out in one cycle.

The transpose memory for 2-D DCT can be implemented in the same way. An example of the new data mapping scheme is shown in Fig. 2 for 8 × 8 2-D DCT. The read and write throughput of this architecture is 4-sample/cycle and four SRAM banks are used. The width of each SRAM bank is 16-bit and the depth is 16-word. It can be extended to support different throughputs in the same way as that of IDCT.

Larger size transform can also be easily supported by increasing the depth of each SRAM bank. If the transpose memory is supposed to support 16 × 16 2-D transform and the read and write throughput is set as 4-sample/cycle, four SRAM banks are used to implement this transpose memory. The intermediate results stored in the transpose memory are assumed to have a precision of W-bit. The width of the each SRAM is W-bit and the depth is 16 × 16/4, where four is the number of SRAM bank (or the throughout). A more generalized architecture can be proposed in such a way. To support an $N \times N$ 2-D transform with a TP-sample/cycle read/write throughput, the number of SRAM banks used to implement the transpose memory is TP. The width of each SRAM bank is W-bit and the depth is N^2/TP.

Two typical VLSI architectures for a transpose memory-based DCT/IDCT are shown in Fig. 3. Fig. 3(a) shows that the row transform can share the same hardware resource with the column

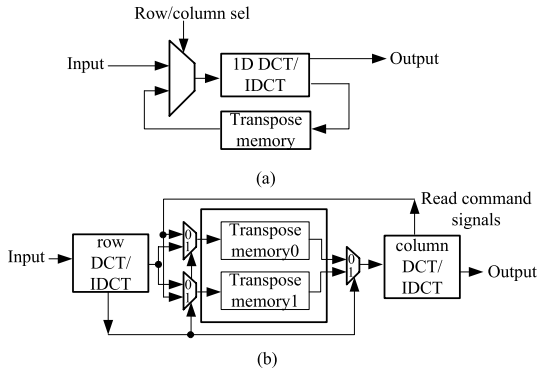

Fig. 3. Transpose memory-based 2-D DCT/IDCT. (a) Shared 1-D transform architecture. (b) Pipelined row/column transform architecture.

TABLE I
AREA COMPARISON BETWEEN SHIFT-REGISTER ARRAY AND SRAM-BASED TRANSPOSE MEMORY (GATE COUNT)

	Transpose memory size			
	4x4	8x8	16x16	32x32
Shift-register	2.86K	10.08K	38.4K	148.9K
Dual-port SRAM (shared buffer)	3.79K	8.67K	24.64K	61.76K
Single-port SRAM (shared buffer)	2.73K	5.98K	13.49K	32.52K
Dual-port SRAM (ping-pong buffer)	7.58K	17.34K	49.28K	123.52K
Single-port SRAM (ping-pong buffer)	5.46K	11.96K	26.98K	65.04K

transform. The row transforms are performed first and the intermediate results are stored into the transpose memory. After all the row transforms are completed, the same transform hardware can be used to perform the column transform.

III. EXPERIMENTAL RESULTS AND COMPARISON

The experiment is performed to compare the hardware cost of the different transpose memory. The shift-register array architecture proposed in [6] and [7] is implemented in Verilog HDL and synthesized with the SMIC 0.13 μm standard cell library. The single/dual-port register file generators provided by VeriSilicon are used to generate the required SRAM banks for SMIC 0.13 μm process. Each intermediate result stored in the transpose memory is assumed to be 16-bit and the silicon areas for $4 \times 4/8 \times 8/16 \times 16/32 \times 32$ shift-register arrays are shown in Table I. The throughput of the shift-register array and SRAM arrays are 4, 8, 16, and 32 samples/cycle for the size of $4 \times 4/8 \times 8/16 \times 16/32 \times 32$, respectively. If the SRAM array is used in a ping-pong buffer-based structure, two same SRAM arrays are needed, and the gate count is double. The unit of the data in Table I is gate count. The silicon area of both single-port and dual-port SRAM-based transpose memory are shown in Table I. The word width of the SRAM used in this table is 16-bit and the SRAM depth is set as N^2/TP, where N is the transpose memory size and the TP is the throughput. Table I shows the single-port SRAM-based transpose memory is more efficient than shift register or dual-port-based transpose memory in the shared 1-D transform architecture. When pipelined row/column transform architecture is adopted, the single-port SRAM-based transpose memory is more efficient in the situation of large size transformation (bigger than 8×8).

The hardware cost comparison of the shift register or SRAM-based 32×32 transpose memory is shown in Fig. 4. The EGC is used to

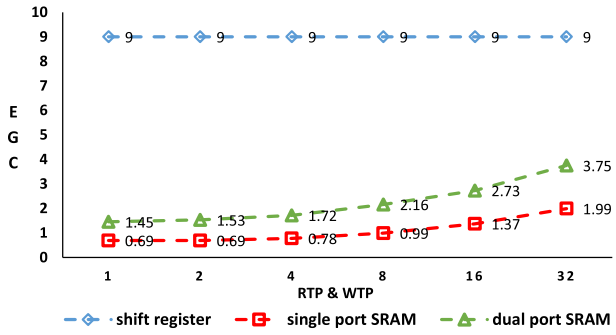

Fig. 4. Hardware cost of 32×32 transpose memory with different RTP (read throughput) and WTP (write throughput).

measure the efficiency of transpose memory

$$\text{EGC} = \frac{\text{The Transpose Memory Gate Count}}{\text{N} * \text{N} * \text{BitWidth}} \tag{1}$$

where the $N * N$ is the transpose memory size, the BitWidth is the bit width of one point. The EGC of 32×32 shift register array is 9 (148.9 K/($32 \times 32 \times 16$)). The EGC of SRAM-based transpose memory will increase significantly with higher throughput. The EGC of single-port SRAM is 0.69/0.69/0.78/0.99/1.37/1.99 with throughput at 1/2/4/8/16/32-sample/cycle. The EGC of dual-port SRAM is 1.45/1.53/1.72/2.16/2.73/3.75, respectively. The single-port SRAM is more efficient as Fig. 4 showing. For a $N \times N$ transpose memory, the throughput ranges from 1 to N^2/D, where D is the minimum SRAM depth. The minimum SRAM depth of VeriSilicon memory compilers is four. Therefore, the maximum write throughput supported by the proposed $N \times N$ transpose memory is $N^2/4$.

The comparison with previous works is summarized in Table II. The SDRAM-based transpose memory in [9] or [10] does not require on-chip SRAM or register. However, its throughput is unstable due to the unpredictable access latency of SDRAM. The EGC of $4 \times 4/8 \times 8/32 \times 32$ shift register array is 11.2/9.8/9. In this proposed design, the EGC of the single-port SRAM-based transpose memory is listed in Table II. The gate count of SRAM control logic is about 0.4 K. Ten different configurations of the proposed design are used and ten EGC results are given in this table. The area of transpose memory in [11]–[13] is not reported. Thus, the VeriSilicon memory generators are used to generate the required SRAM banks in [11]–[13] accordingly. The area is estimated by using the area reports of the memory compiler. The EGC of each bit in [11]/[12]/[13] is 5.1/8.5/5.7. The results show that this proposed design is more area efficient. To evaluate the hardware efficiency for different TP and memory size, the equivalent throughput (ET) is proposed

$$\text{ET} = \frac{\text{TP}}{\text{EGC}}. \tag{2}$$

The larger ET means the better hardware efficiency. As shown in Table II, our proposed one-port SRAM transpose memory is more hardware efficient than the previous methods.

The transpose memory is divided into eight banks with the throughput of 4-sample/cycle in [12]. In [13], the transpose memory is divided into four banks with the throughput of 2-sample/cycle. If the new data mapping scheme proposed in this brief is used, the number of SRAM banks is four for the transpose memory in [12] and two for the transpose memory in [13]. The less SRAM banks are used, the less hardware resource is consumed. Less SRAM banks will also facilitate the place and routing process of backend design and helps to achieve better timing and power performance. In [14], the transpose memory is divided into four banks with

TABLE II
COMPARISON WITH PREVIOUS WORKS

	Memory type	Memory size	Gate count	EGC	ET	# of SRAM banks	WTP/RTP (1)
[6]	Register	8x8	10.08k	9.8	0.82	-	8/8
[7]	Register	4x4	2.86k	11.2	0.36	-	4/4
[8]	3-port SRAM	8x8	10.03k	9.80	0.20	4	2/2
[9]	SDRAM	8x8	-	-	-	-	-
[10]	SDRAM	8x8	-	-	-	-	-
[11]	1-port SRAM	8x8	5.18k	5.10	0.39	4	2/2
[12]	Dual-port SRAM	8x8	8.66k	8.50	0.47	8	4/4
[13]	Dual-port SRAM	8x8	5.86k	5.70	0.35	4	2/2
[14]	1-port SRAM	32x32	60.40k	3.70	1.08	4	4/32
This design[1]	1-port SRAM	8x8	1.40k	1.36	0.74	1	1/1
This design[2]	1-port SRAM	8x8	2.03K	1.98	1.01	2	2/2
This design[3]	1-port SRAM	8x8	3.37k	3.29	1.21	4	4/4
This design[4]	1-port SRAM	8x8	5.98k	5.84	1.37	8	8/8
This design[5](2)	1-port SRAM	32x32	11.31k	0.69	1.45	1	1/1
This design[6]	1-port SRAM	32x32	11.31k	0.69	1.45	2	2/2
This design[7]	1-port SRAM	32x32	12.78k	0.78	5.12	4	4/4
This design[8]	1-port SRAM	32x32	16.17k	0.99	8.08	8	8/8
This design[9]	1-port SRAM	32x32	22.43k	1.37	11.68	16	16/16
This design[10]	1-port SRAM	32x32	32.52 k	1.99	16.08	32	32/32

(1) WTP/RTP means write/read throughput
(2) The Memory compiler cannot generate 16bits x 1024words 1-port SRAM, replaced by two 16bits x 512words 1-port SRAM

the throughput of 4-sample/cycle, but the width of the SRAM is 512 bit. In proposed data mapping scheme, the width of SRAM banks are fixed in 16 bit. The small width helps to achieve better area efficiency.

If row transform block and column transform block work in a pipelined fashion, the required throughput of the transpose memory can be calculated by the following equation:

$$\mathrm{TP} = \frac{\mathrm{W} * \mathrm{H} * \mathrm{fps} * \mathrm{Format}}{\mathrm{MHz}} \tag{3}$$

where $W \times H$ is the resolution of the video sequence. fps is the frame rate. Format is set as 1.5 for 4:2:0 and 3 for 4:4:4 video. MHz is the working frequency. To support a full-HD video (1920 $\times$ 1080@30 fps) with 4:2:0 format, the throughput of the transpose memory is 1-sample/cycle at the working frequency of 94 MHz. Each different working frequency (47/24/12 MHz) requires a corresponding minimum throughput (2/4/8-sample/cycle). The throughput of 4/8-sample/cycle can meet the requirement of most video coding applications at a reasonable working frequency. For example, the quad HD video format (4 K $\times$ 2 K@30 fps) can be well supported if the throughput is set as 4-sample/cycle and the pipelined architecture in Fig. 3(b) operates at 90 MHz working frequency.

IV. CONCLUSION

In this brief, we proposed a single-port SRAM-based transpose memory which can be applied to large size 2-D DCT/IDCT. The hardware cost can be significantly reduced in comparison with the shift-register array-based transpose memory. A new diagonal data mapping scheme is also proposed to support different throughput. The number of SRAM banks used in the transpose memory can also be reduced by this new data mapping scheme, which can also reduce the hardware cost. This architecture is also flexible to support any larger size transpose memory by increasing the depth of SRAM.

REFERENCES

[1] N. I. Cho and S. U. Lee, "Fast algorithm and implementation of 2-D discrete cosine transform," *IEEE Trans. Circuits Syst.*, vol. 38, no. 3, pp. 297–305, Mar. 1991.

[2] Y. P. Lee, T. H. Chen, L. G. Chen, M. J. Chen, and C. W. Ku, "A cost-effective architecture for 8×8 two-dimensional DCT/IDCT using direct method," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 3, pp. 459–467, Jun. 1997.

[3] C. P. Fan, "Fast 2-dimensional 4×4 forward integer transform implementation for H.264/AVC," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 3, pp. 297–305, Mar. 2006.

[4] Y. T. Chang and C. L. Wang, "New systolic array implementation of the 2-D discrete cosine transform and its inverse," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 2, pp. 150–157, Apr. 1995.

[5] D. Gong, Y. He, and Z. Cao, "New cost-effective VLSI implementation of a 2-D discrete cosine transform and its inverse," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 4, pp. 405–415, Apr. 2004.

[6] J. R. Choi, W. J. Hur, K. K. Lee, and A. S. Kim, "A 400 MPixel/s IDCT for HDTV by multibit coding and group symmetry," in *IEEE 43rd Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 1997, pp. 262–263.

[7] T. C. Wang, Y. W. Huang, H. C. Fang, and L. G. Chen, "Parallel 4×4 2D transform and inverse transform architecture for MPEG-4 AVC/H.264," in *Proc. IEEE ISCAS*, May 2003, pp. 800–803.

[8] T. Masaki, Y. Morimoto, T. Onoye, and I. Shirakawa, "VLSI implementation of inverse discrete for MPEG2 HDTV video decoding cosine transformer and motion compensator," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 5, pp. 387–395, Oct. 1995.

[9] B. Z. Tu, D. Li, and C. D. Han, "Two-dimensional image processing without transpose," in *Proc. 7th Int. Conf. Signal Process.*, 2004, pp. 523–526.

[10] S. Langemeyer, P. Pirsch, and H. Blume, "Using SDRAMs for two-dimensional accesses of long $2^n \times 2^m$-point FFTs and transposing," in *Proc. Int. Conf. Embedded Comput. Syst.*, Jul. 2011, pp. 242–248.

[11] K. B. Lee, H. C. Hsu, and C. W. Jen, "A cost-effective MPEG-4 shape-adaptive DCT with auto-aligned transpose memory organization," in *Proc. Int. Symp. Circuits Syst.*, vol. 2. May 2004, pp. 777–780.

[12] M. N. Bojnordi, S. M. Naser, O. Fatemi, and M. R. Hashemi, "An efficient self-transposing memory structure for 32-bit video processors," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, Dec. 2006, pp. 1438–1441.

[13] Y. F. Jang, J. N. Kao, J. S. Yang, and P. C. Huang, "A 0.8 μ 100-MHz 2-D DCT core processor," *IEEE Trans. Consum. Electron.*, vol. 40, no. 3, pp. 703–710, Aug. 1994.

[14] S. Shen, W. Shen, Y. Fan, and X. Zeng, "A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards," in *Proc. IEEE ICME*, Jul. 2012, pp. 788–793.

[15] T.-C. Chen, Y.-H. Chen, S.-F. Tsai, and L.-G. Chen, "Architecture design of low power integer motion estimation for H.264/AVC," in *Proc. IEEE ICASSP*, vol. 3. May 2006, p. 3.