# A UNIFIED 4/8/16/32-POINT INTEGER IDCT ARCHITECTURE FOR MULTIPLE VIDEO CODING STANDARDS

*Sha Shen, Weiwei Shen, Yibo Fan, Xiaoyang Zeng*
State Key lab of ASIC and system, Fudan University, Shanghai, China
E-mails: {10110720050, 10110720024, fanyibo, xyzeng}@fudan.edu.cn

**ABSTRACT**

4 or 8-point IDCT are widely used in traditional video coding standards. However larger size (16/32-point) IDCT has been proposed in the next generation video standard such as HEVC. To fulfill this requirement, this work proposes a fast computational algorithm of large size integer IDCT. A unified VLSI architecture for 4/8/16/32-point integer IDCT is also proposed accordingly. It can support the following video standards: MPEG-2/4, H.264, AVS, VC-1 and HEVC. Multiplierless MCM (Multiple Constant Multiplication) is used for 4/8-point IDCT. The regular multipliers and sharing technique are used for 16/32-point IDCT. The transpose memory uses SRAM instead of the traditional register array in order to further reduce the hardware overhead. It can support real-time decoding of 4Kx2K (4096x2048) 30fps video sequence at 191MHz working frequency, with 93K gate count and 18944-bit SRAM. We suggest a normalized criterion called *design efficiency* to compare with previous works. It shows that this design is 31% more efficient than previous work.

***Index Terms***—IDCT, integer transform, HEVC, Multiplierless MCM, multiple standards, video coding

## 1. INTRODUCTION

Discrete Cosine Transform (DCT) [1] has been widely used in the field of video coding due to its ability to concentrate the energy of video residual data into low frequency domain. Floating-point DCT has been adopted by some video coding standards such as MPEG-2 and MPEG-4. In order to avoid the mismatch problem caused by floating-point arithmetic operation, latest video standards like VC-1, H.264 and AVS begin to use integer DCT.

Larger transform size can achieve better result in terms of energy concentration. But the computational complexity will exponentially increase when the transform size of a 2D DCT/IDCT increases. All traditional video coding standards select 4x4 or 8x8 transform size as a tradeoff between coding efficiency and implementation cost.

High Efficiency Video Coding (HEVC) [2] standard is an emerging new video coding standard which is jointly developed by the MPEG and ITU. HEVC is considered as the successor of H.264. The goal of HEVC is to get 50% higher coding efficiency than H.264. In order to achieve this goal, HEVC [3] adopts many new coding tools including the 16x16 and 32x32 2D DCT.

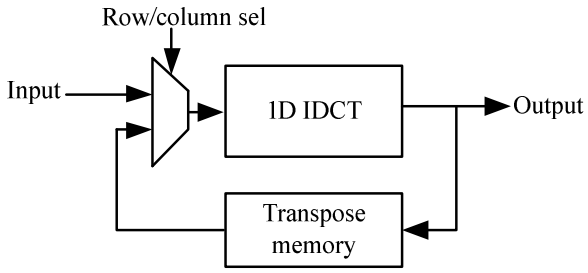


**Fig. 1.** 2D IDCT architecture with row-column decomposition

By using a row-column decomposition approach, we can accomplish a 2D IDCT by two 1D IDCT. Fig.1 shows the diagram of such approach. The 2D IDCT is decomposed into one row 1D IDCT and one column 1D IDCT. The row operation and column operation can share the same hardware resource.

Various VLSI architectures have been proposed for multiple-standard 2D IDCT [4], [5]. All these previous works can only support 4/8-point 1D IDCT. None of them can support 16/32 point IDCT. The multiplication inside 1D IDCT is done by Multiplierless MCM [6]. As we will see in Section 5 of this paper, MCM is proved to be not suitable for large size IDCT in terms of hardware cost. The register array based transpose memory is shown in [7]. Each register bit will be shifted vertically or horizontally at every access. As the transform size increases to 32x32, both the area of the register array and the power caused by bit shifting will become too large to afford.

In order to address the above problems, we propose a unified architecture for 4/8/16/32 1D IDCT. It can support both the existing video coding standards like H.264, AVS, MPEG-2/4, VC-1 and the emerging HEVC standards. The transpose memory is implemented with SRAM to save area and power. Bit shifting is avoided in this design. The regular multipliers and sharing techniques are used for the multiplication circuits of 16/32-point IDCT.

The rest of this paper is organized as follows. In Section 2, a fast computational algorithm of integer IDCT is introduced. Section 3 presents the proposed VLSI architecture of the unified 1D IDCT. The SRAM based transpose memory is introduced in section 4.The result of VLSI implementation and comparisons are shown in Section 5. A conclusion is drawn in Section 6.

978-0-7695-4711-4/12 $26.00 © 2012 IEEE
DOI 10.1109/ICME.2012.7

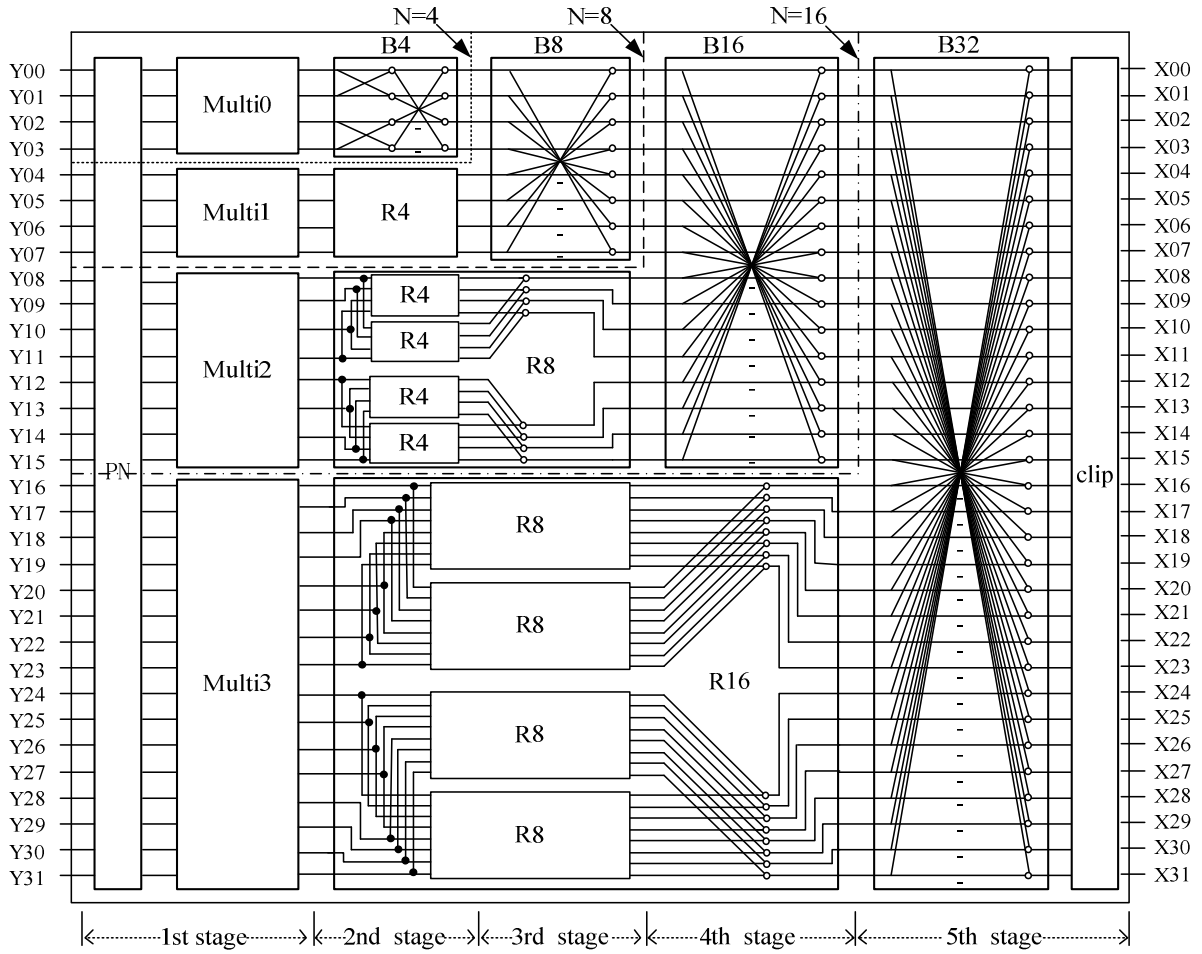


**Fig. 2.** Signal flow graph for N=4, 8, 16, 32-point integer 1D IDCT

## 2. FAST COMPUTATIONAL ALGORITM FOR IDCT

Floating-point 1D IDCT of a discrete function X(k) can be defined as (1)

$$Y(j) = \sum_{k=0}^{N-1} C(k)X(k)\cos\left[\frac{(2j+1)k\pi}{2N}\right] \quad (1)$$

Where

$C(k) = \frac{1}{\sqrt{2}}$ for k=0

$= 1$ for k= 1, 2, 3, …, N-1

A fast computational algorithm for floating-point IDCT has been proposed by [8]. Recursive division and matrix factorization are the core ideas of this fast algorithm. The NxN transform matrix $A_N$ can be shown in a recursive form:

$$[A_N] = [P_N] * \begin{bmatrix} A_{N/2} , & 0 \\ 0 , & R_{N/2} \end{bmatrix} * [B_N] \quad (2)$$

Where $P_N$ is the permutation matrix and $B_N$ is the butterfly operation. The transform matrix $A_N$ is divided into even part matrix ($A_{N/2}$) and odd part matrix ($R_{N/2}$) while matrix $A_{N/2}$ can be further divided in the same fashion. Matrix $R_{N/2}$ can also be factorized and decomposed into several matrices. The signal-flow graph for 4,8,16 and 32-point floating-point DCT is shown in [8].

Floating point IDCT is adopted in the legacy standards like MPEG-2 and MPEG-4 while integer transform is used in the latest video standards like AVS, VC-1 and H.264. Floating point IDCT can easily be done by integer transform if a suitable transform matrix is used. Integer transform reduces the computational complexity. But the orthogonality of transform matrix is spoiled due to the integer approximation of transform coefficients. The Signal-flow graph in [8] cannot be directly applied to integer transform.

After the thorough inspection of various integer transform matrices used in AVS, VC-1, H.264 and HEVC, we can see that Equation (2) can still be applied to integer transform. The integer transform matrix can be recursively divided into smaller matrices. The permutation and butterfly operation remain the same. The only exception is that the generalized method of decomposing the odd part matrix ($R_{N/2}$) can no longer be used for integer transform. So it is possible to design a unified architecture for integer IDCT of different size.

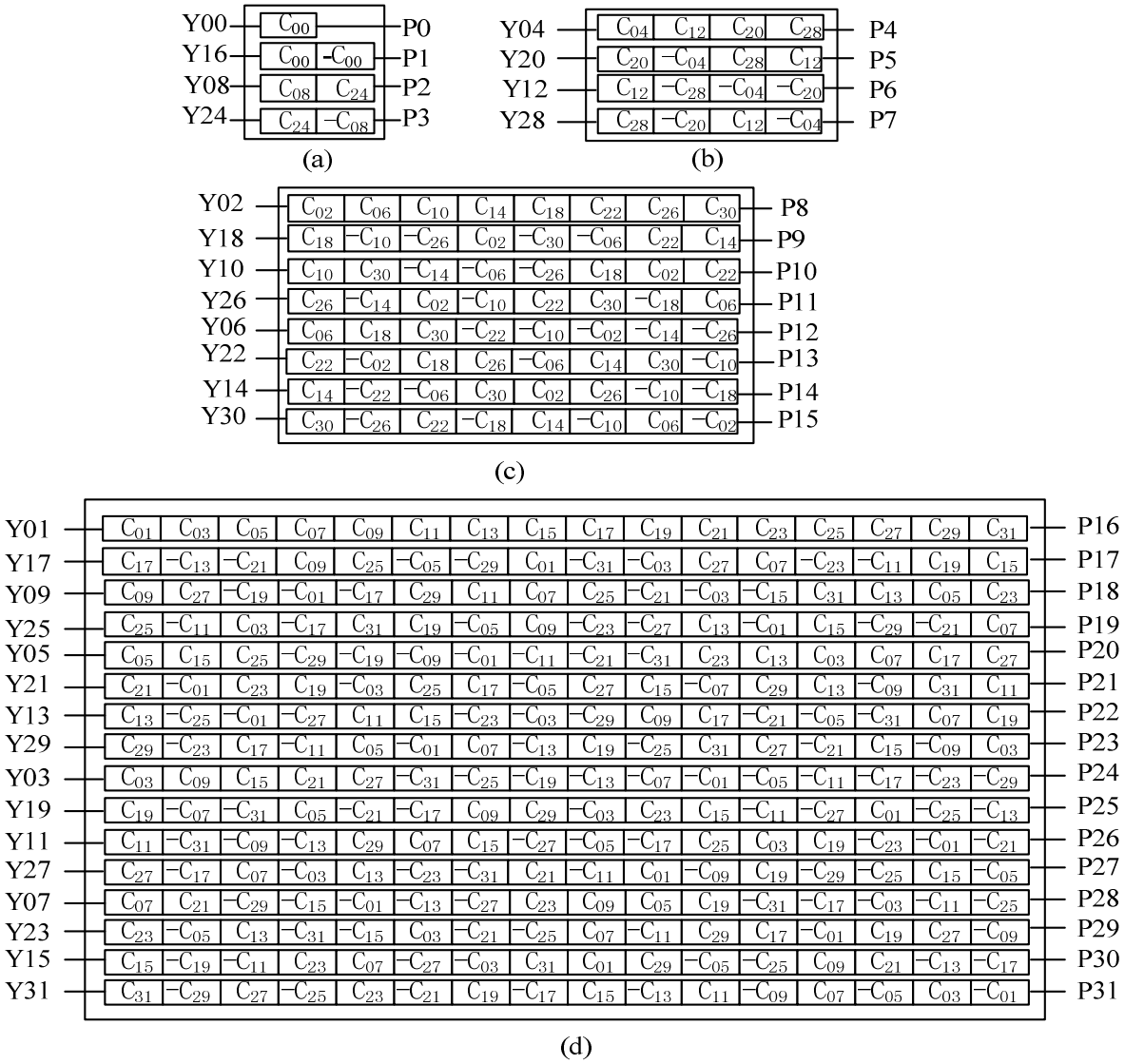


**Fig. 3.** Transform coefficients used in multiplication Block (a) Multi0 (b) Multi1 (c) Multi2 (d) Multi3

A signal flow graph for 32-point integer IDCT is shown in Fig. 2. Signal flow graph for 4/8/16-point integer IDCT are also shown in this figure. 4-point IDCT is marked with dot line. 8-point IDCT is marked with dash line and 16-point IDCT is marked with dot-dash line. It can be clearly seen that the result of small size IDCT can be recursively used for larger size IDCT.

The symbol "。" and "-"in Fig. 2 stand for addition and subtraction. Block $P_N$ is the permutation matrix and it is used to permute the input vector Y. Matrix $P_N$ is defined as:

$$P_N(i,j) = 1 \quad \text{for } i=2*j \text{ or } i=(j-N/2)*2+1$$
$$= 0 \quad \text{otherwise} \qquad (3)$$

Where $0 \le i \le N-1$ and $0 \le j \le N-1$

Block Multi0, Multi1, Multi2, Multi3 are used to perform the multiplication operations of integer transform. The multiplicands are the permuted input signals Y while the multiplicators are the integer approximation of the transform coefficients C(k,j) which are defined in equation (1).The transform coefficients used for block Multi0, Multi1, Multi2, Multi3 are shown in Fig.3. The transform coefficient $C_n$ equals to the value of the transform coefficient C(n*N/32,0), where N is the transform size. For example, the output P3 of Multi0 contains two products of $Y24*C_{24}$ and $Y_{24}*(-C_{08})$, where $Y_{24}$ is the input signal while $C_{24}$ and $C_{08}$ are the transform coefficients defined by a specific video standard. The number of multiplication needed for each block is summarized in Table 1.

Block R4, R8, R16 are the adder trees for the odd part transform matrices. The number of addition for block RN (N=4, 8, 16) is N*(N-1). The diagram of block R4 is shown in Fig. 4 as an example. Sixteen products from Multi1 are divided into 4 groups which are marked as P4, P5, P6 and

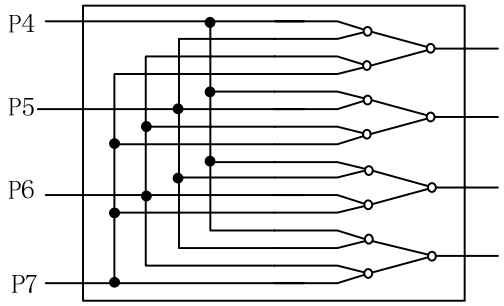


**Fig. 4.** Diagram of adder tree for R4

P7 in Fig. 4. 4 products are added up to get one output. Totally 12 addition operations are needed for Block R4.

**Table 1.** Number of multiplication used in IDCT

| Block name | Multi0 | Multi1 | Multi2 | Multi3 |
|---|---|---|---|---|
| Number of multiplication | 7 | 16 | 64 | 256 |

Block B4, B8, B16, B32 are the butterfly operations as shown in Fig. 2.

All the output values of 4/8/16/32 1D IDCT is rounded and clipped to 16-bit precision so that the same 1D IDCT core can be applied to the row transform as well as the column transform. The rounding and clipping is widely adopted in the video standards which use integer transform. It can also reduce the size of transpose memory.

## 3. VLSI ARCHITECTURE FOR UNIFIED 1D IDCT

In this section, the unified VLSI architecture for 1D integer IDCT is proposed according to the signal flow which is shown in Fig. 2.

The architecture of 4/8/16/32 point 1D IDCT is divided into 5 pipeline stages in order to maximize the working frequency. The permutation and multiplication are performed in stage 1. Butterfly operation B4 and add tree R4 are arranged in stage 2. Partial operation of add tree R8/R16 are also arranged in stage 2. Butterfly operation B8 and partial operation of add tree R8/R16 are arranged in stage 3. Butterfly operation B8 and partial operation of add tree R8/R16 are arranged in stage 3. Butterfly operation B16 and the rest part of add tree R16 are arranged in stage 4. Butterfly operation B32 and the clip module are arranged in stage 5.

The permutation operation can be done by simple hardwires. No additional hardware is needed.

As the transform size increases from 8-point to 32-point, the computation complexity will increase by 16 times. If a 32-point 1D IDCT is done in one single cycle, the hardware cost will also increase 16 times. A direct VLSI implementation of the multiplication circuit and adder trees in Fig. 2 may result in an exponentially increased hardware area.

Previous work [4] has shown that the multiplication and adder tree circuit account for more than 80% of the whole hardware area. An area efficient method must be used to implement the multiplication and addition circuit of 1D IDCT.

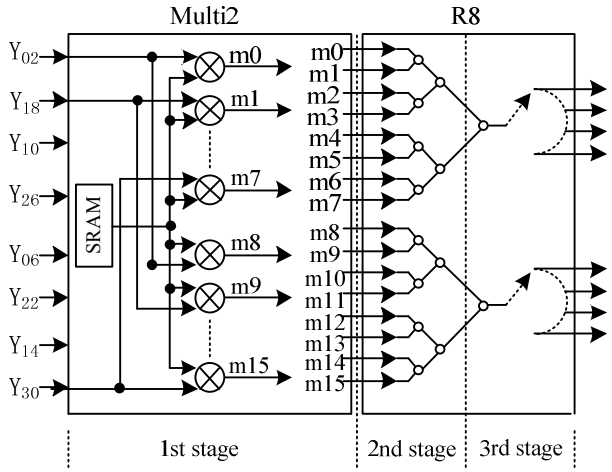


**Fig. 5.** Modified signal flow for Multi2 and R8

In all previous works of integer IDCT, multiplication is implemented by multiplierless Multiple Constant Multiplication (MCM) instead of the regular multiplier (RM). The multiplierless MCM is actually done by adders and shifters. MCM requires less hardware resource than RM if the multiplication of integer 1D DCT is done in one cycle.

However we can see from the result in section 4 that MCM is not suitable for large size IDCT if a single one row (or column) transform of the 1D IDCT is done in multiple clock cycles. The regular multiplier can be fully reused in multiple clock cycles. The more clock cycles are used, the fewer multipliers are needed. But MCM can't achieve such hardware reuse because the number of adders and shifters for MCM is statically determined by the transform coefficients and won't be reduced by more clock cycles.

The throughput of this design is set as 4 pixels/cycle. A 4-point 1D IDCT must be completed in one clock cycle while the 8/16/32-point 1D IDCT can be done in 2/4/8 clock cycles respectively. The multipliers and adders can be reused in different clock cycles. So the number of multipliers and adders in block Multi2, Multi3, R8, R16 can be greatly reduced. The number of multipliers in Multi2 can be reduced from 64 to 16. The number of multipliers in Multi3 can be reduced from 256 to 32.

A modified signal flow for multiplication blocks Multi2/Multi3 and adder tree blocks R8/R16 are used in this design. The signal flow of block Multi2 and R8 is shown in Fig.8 as an example. The 8-bit transform coefficients in Fig.3(c) are stored in SRAM. There are 16 regular multipliers in Multi2 and 14 adders in R8. Multi2 is arranged in pipeline 1 while R8 is arranged in pipeline stage 2 and 3. The 16-point 1D IDCT is done in 4 clock cycles. The input signals remain unchanged in these 4 cycles while 16 transform coefficients come out from the SRAM block. 2

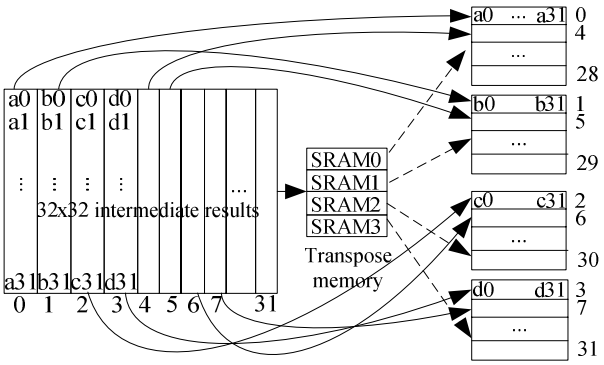


**Fig. 6.** Data mapping in the transpose memory

outputs of adder tree R8 are generated at each cycle. 8 outputs of R8 can be generated in 4 cycles. After the butterfly operation, all 16 outputs of 16-point 1D IDCT are generated in 4 cycles. The width of the SRAM in Multi2 is 128-bit (8x16) and its depth is 4. The signal flow for Multi3 and R16 is also modified in a similar way. Multi3 has 32 multipliers and R16 has 30 adders. The width of the SRAM in Multi3 is 256-bit (8x32) and its depth is 8.

This architecture is very flexible to support a new video standard. When there is a new video standard to be supported, we only need to update the coefficients inside SRAM to support the new video standard. No additional hardware cost is added to the multiplication circuit Multi2 and Multi3.

## 4. SRAM BASED TRANSPOSE MEMORY

One novel feature of this design is that single-port SRAM instead of the traditional register array is used to implement the transpose memory.

The transpose memory is accessed row by row when the results of row IDCT are written into it. It is accessed column by column when the column IDCT module read data from it. It is very easy to access either a row of registers or a column of registers in a register array. But the disadvantage of a register array is its big hardware area and power dissipation. If the data stored in transpose memory have a 16-bit precision, a 4x4 2D IDCT requires a 256-bit register array (4x4x16) which is still affordable. But a 32x32 2D IDCT will require 16384-bit register array (32x32x16). SRAM consumes less silicon area than registers when massive amount of data are to be stored. The register-based transpose memory in [7] also requires bit shifting in the row or column direction at every access. This will also increase the power dissipation.

In order to reduce the hardware cost and power dissipation, we proposed an SRAM based transpose memory. The diagram is shown in Fig. 6. It also shows the way how to store the intermediate output data of the row IDCT core into the 4 single-port SRAM blocks.

In this design, the throughput is set as 4 pixels/cycle. We split the transpose memory into 4 blocks accordingly. Each block consists of an on-chip SRAM. The intermediate output data of row IDCT core is a 32x32 matrix. This matrix can be divided into 32 columns. Each column is marked with a number in Fig.6. At every clock cycle, 4 output data of row 1D IDCT is generated and stored in one of the 4 SRAM blocks. For example, a0, b0, c0 and d0 are the 4 outputs of row 1D IDCT. They are stored in SRAM block 0, 1, 2 and 3 respectively. The transpose operation for a 32x32 data matrix can be done in 256 (32x32/4) cycles. After all the row transforms are done, each column of the matrix is stored as one word of a SRAM block. Column 0,4,8…28 are stored in SRAM block0. Column 1,5,8…29 are stored in SRAM block1. Column 2,6,10…30 are stored in SRAM block2. Column 3,7…31 are stored in SRAM block3.

The width of each SRAM block is 512-bit (16bitx32) and the depth is 8. The size of each SRAM is 4096-bit. Totally 16384 SRAM bits are used in this transpose memory.

## 5. EXPERIMENTAL RESULTS

Verilog HDL is used to implement the proposed design. Synthesized by Synopsys Design Compiler with SMIC 0.13um standard cell library, the unified 1D IDCT core can work at 191MHz with 93K gate count. 2560 SRAM bits are used in the 1D IDCT core. 16384 SRAM bits are used in the transpose memory. The 5-stage pipelined architecture of 1D IDCT core enables this design to achieve a maximum working frequency up to 350 MHz at the cost of 109.2K gate count.

Five kinds of video standards are supported: MPEG-2/4, H.264, VC-1, AVS and the emerging HEVC standard. The HEVC standard is not finalized yet. The transform matrices of HEVC are shown in [3].

**Table 2.** Gate count of multiplication block

| | Multi0 | Multi1 | Multi2 | Multi3 |
|---|---|---|---|---|
| MCM | 2.5K | 7K | 30K | 110K |
| RM | 3.9K | 7.9K | 15.6K | 31K |

Table 2 shows the gate count of multiplication blocks Multi0, Multi1, Multi2 and Multi3. Both the MCM based approach and the regular multiplier based approach are listed in the same table. As we can see from the results that MCM is suitable for 4/8 point IDCT while larger size IDCT should be implemented with regular multipliers. The multiplicand of a regular multiplier is the input signal whose width is 16-bit while the multiplicator is the transform coefficient whose width is 8-bit. The critical timing path is the delay of a regular multiplier.

The numbers of pipeline stages used for 4/8/16/32-point 1D IDCT are 2/3/4/5. The unused pipeline stages are bypassed. One more additional pipeline stage will increase the latency of 1D IDCT by one cycle. The latency between row and column transform of 4-point 1D IDCT is 6 (4+2) cycles. The latency of 8/16/32-point IDCT is 19 (16+3) /68

**Table 3**. Comparison with previous works

| Design | Process | Gate count | SRAM (bit) | Max Speed (MHz) | Supported transform | | | | | | | Normalized Design efficiency |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | MPEG-2/4 | VC-1 | | H.264 | | AVS | HEVC | |
| | | | | | | 4x4 | 8x8 | 4x4 | 8x8 | | | |
| Proposed | 0.13um | 109.2K | 2560 | 350 | √ | √ | √ | √ | √ | √ | √ | 1 |
| Wang[4] | 0.13um | 23.06K | - | 100 | √ | √ | √ | √ | √ | √ | × | 0.42 |
| Qi[5] | 0.13um | 18.00K | - | 100 | √ | √ | √ | √ | √ | × | × | 0.53 |
| Lee[9] | 0.13um | 19.11K | - | 136 | √ | × | √ | √ | × | × | × | 0.69 |

(64+4) /261 (256+5) cycles respectively.

The throughput of this design is fixed at 4 pixels/cycle. A 32x32 block can be processed in 256 (32*32/4) cycles for all different transform sizes. 4-point 1D IDCT can be done in one cycle. 8/16/32-point 1D IDCT can be done in 2/4/8 cycles respectively.

Two proposed 1D IDCT blocks can be used to perform row and column transform simultaneously. In such case, the working frequency needed to support a specific video sequence can be calculated by equation (4):

$$\text{Freq} = \frac{W * H * \text{Format} * \text{fps}}{32 * 32} * (L + 256) \qquad (4)$$

Where W*H is the resolution of video sequence. Format is set as 1.5 for 4:2:0 video and 3 for 4:4:4 video. fps is the video frame rate. L is the latency between the row transform and column transform. (L+256) is the number of cycles needed to process a 32x32 block. The maximum latency is 261 cycles in this design. Our proposed work can easily support the high video resolution of 4Kx2K (4096x2048) @30fps with 4:2:0 format at the working frequency of 191MHz.

All previous designs can only support 4/8-point IDCT while this work can support 4/8/16/32-point IDCT. In order to perform a fare comparison, we introduce a normalized criterion called Design Efficiency (DE). It is defined in equation (5).

$$\text{DE} = \frac{\text{Tp} * \text{MHz} * \text{N}}{\text{Gate\_count} + 10 * \text{SRAM\_size}} \qquad (5)$$

Where Tp is the throughput of 1D IDCT module, which is 4 pixels/cycle for all the designs listed in Table 3. MHz is the maximum work frequency. N is the computation complexity of each pixel and it is equal to the transform size of 1D IDCT. One SRAM bit is equivalent to 10 2-input NAND gates in terms of silicon area.

All the listed designs in Table 3 use 0.13um process. The normalized result in Table 3 shows that our design is 31% more efficient than any previous work.

## 6. CONCLUSION

In this paper, a fast computation algorithm for large size integer IDCT is proposed. All previous works can only support 4/8-point 1D IDCT. This work proposes a unified 4/8/16/32-point 1D IDCT architecture for multiple video standards. The hardware for smaller size integer IDCT can be recursively reused for larger size integer IDCT. SRAM-based transpose memory is used to reduce both hardware area and power dissipation. In order to further reduce the hardware cost, the multiplication of 4/8 point IDCT is implemented with MCM while the multiplication of 16/32 point IDCT is implemented with reusable regular multipliers. The throughput of this design is 4pixels/cycle. It can support 4Kx2K @30fps video (4:2:0 YUV format) at 191MHz working frequency. The gate count is 93K and the size of on-chip SRAM is totally 18944-bit. The 5-stage pipelined architecture enables this design to achieve 350MHz maximum working frequency at the cost of slightly increase silicon area. The normalized criterion called *design efficiency* shows that our design is 31% more efficient than previous work.

## 7. REFERENCES


[1] N. Ahmed, et al., "Discrete cosine transform," *IEEE Trans. Comput.*,pp. 90-93, Jan. 1974.

[2] Joint Collaborative Team on Video Coding (JCT-VC), "WD4: Working Draft 4 of High-Efficiency Video coding", JCTVC-F803.doc.

[3] Joint Collaborative Team on Video Coding (JCT-VC), "HM4: High-Efficiency Video coding (HEVC) Test Model 4 Encoder Description", JCTVC-F802.doc.

[4] K. Wang, J. Chen, W. Cao, Y. Wang, L. Wang and J. Tong, "A Reconfigurable Multi-Transform VLSI Architecture Supporting Video Codec Design," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 7, pp. 432–436, Jul. 2011.

[5] H. Qi, Q. Huang, and W. Gao, "A low-cost very large scale integration architecture for multistandard inverse transform," *IEEE Trans. Circuits Syst. II, Exp. Briefs,* vol. 57, no. 7, pp. 551–555, Jul. 2010.

[6] Y. Voronenko and M. Puschel, "Multiplierless multiple constant multiplication," *ACM Trans. Algorithms*, vol. 3, no. 2, p. 11, May 2007.

[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", *Proc. IEEE ISCAS*, pp. 800–803, May 2003.

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

[9] S. Lee and K. Cho, "Architecture of transform circuit for video decoder supporting multiple standards," *Electron. Lett*., vol. 44, no. 4, pp. 274–275, Feb. 2008