AN AREA-EFFICIENT 4/8/16/32-POINT INVERSE DCT ARCHITECTURE FOR UHDTV HEVC DECODER

Heming Sun, Dajiang Zhou, Jiayi Zhu, Shinji Kimura, and Satoshi Goto
Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, Japan
terrysun1989@akane.waseda.jp

Abstract—This paper presents a new VLSI architecture for HEVC inverse discrete cosine transform (IDCT). Compared to prior arts, this work reduces hardware cost by 1) reducing computational logic of 1-D IDCTs with a reordered parallel-in serial-out (RPISO) scheme that shares the inputs of the butterfly structure, and 2) reducing the area of the transpose buffer with a cyclic memory organization that achieves 100% I/O utilization of the SRAMs. In the implementation of a unified 4/8/16/32-point IDCT, the proposed schemes demonstrate 35% and 62% reduction of logic and memory costs, respectively. The IDCT implementation can support real-time decoding of 4Kx2K 60fps video with a total hardware cost of 357,250um² on 2-D IDCT and 80,988um² on transpose memory in 90nm process.

Index Terms—HEVC, IDCT, SRAM, area-efficient, video coding

I. INTRODUCTION

High Efficiency Video Coding (HEVC) [1] is the newest video compression standard, and has been approved by ITU-T. HEVC aimed to double the compression ratio compared with H.264. Transform Unit (TU) is a basic unit for HEVC transform. In HEVC, the largest TU size is 32x32 while the largest transform size in H.264 is only 8x8. Larger TU size can improve coding efficiency while the computational complexity dramatically increases. In an HEVC full decoder [2], Inverse DCT (IDCT) engine consumes more hardware resources compared with other processing engines. The large hardware cost of IDCT comes from two parts. First is the logical computational part of 1D IDCT. Second is the transpose memory architecture.


To reduce the hardware cost of transpose memory, [3] and [5] use SRAM instead of register to do the transpose. In [3], four 8 x 512-bit SRAM blocks are adopted. In [5], 32 SRAM blocks with 32 x 16-bit are used.

The new contributions of this paper are listed in the following.

1) To reduce the hardware resources of 1D IDCT logical computation part, we not only adopt the Chen’s algorithm, but also take advantage of the butterfly property. We propose a reordered parallel-in serial-out (RPISO) scheme to share the inputs of butterfly structure. Based on the RPISO scheme, a unified 4/8/16/32 1D IDCT architecture is developed.

2) To save hardware areas of transpose memory, SRAM instead of register is used. [3] and [5] also used SRAM to implement the transpose memory. However, the data width of SRAM is larger than the data parallelism of logical IDCT so that the I/O utilization of SRAM cannot reach 100%. We present a cyclic memory organization that achieves 100% I/O utilization for each SRAM. In addition, to let the row-transform and column-transform work in pipeline, we use two-port SRAM and propose a pipelining schedule without writing and reading address conflict.

The rest of this paper is organized as follows. Section II introduces the HEVC inverse transforms. Section III shows the proposed IDCT architecture design and section IV presents the proposed SRAM-based transpose memory. Section V shows the experimental results, followed by the conclusions in Section VI.

II. HEVC INVERSE TRANSFORM

A 2D IDCT can be decomposed to two 1D IDCT operation as follows:

\[ X = T_{2N}^T \times Y \times T_{2N} \]

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

\[ = (Z \times T_{2N}^T)^T \]

where \( T_{2N} \) is a 2Nx2N transform matrix given by HEVC, Y is the 2Nx2N matrix before 2D IDCT and X is 2Nx2N matrix after 2D IDCT. (Y x T_{2N}) is the first 1D transform which is called as IT1, IT1 result is marked as Z. To store the results of Z, a transpose memory capable of storing (2Nx2N) samples is required. (Z x T_{2N}) is the second 1D transform which is IT2.

III. PROPOSED 1D IDCT ARCHITECTURE

A. Fast Algorithm for HEVC Transform

Chen, et al. [4] decompose a transform matrix to some zero matrixes. This decomposition method is shown as the following equation.

\[ T_{2N} = P_{2N} \times \begin{bmatrix} T_N & 0 \\ 0 & O_N \end{bmatrix} \times B_{2N} \]  

where \( T_{2N} \) is the 2Nx2N transform matrix and \( T_N \) is the NxN transform matrix, which is the even part matrix of \( T_{2N} \), \( O_N \) is the odd part matrix. \( P_{2N} \) is a permutation matrix, and it does not need any arithmetic operations. \( B_{2N} \) is a 2N-point butterfly structure. In fact, \( T_N \) can be further decomposed into \( T_{N/2} \) and...
In the following we use an 8-point 1D IDCT as an example, $Y_n$ is the 8-point inputs and $X_n$ are 8-point outputs. The overall flowchart is demonstrated in Fig. 1.

![Fig. 1. Overall flowchart for 8-point 1D IDCT based on Chen’s algorithm.](image)

**B. Proposed Reordered Parallel-In Serial-Out (RPSIO) Scheme**

The data parallelism of this design is set as 4 pixels/cycle. Therefore, for an 8-point 1D IDCT, two cycles are required to generate 8-point results. Conventionally, 8-point outputs are generated in order. $\{X_0, X_1, X_2, X_3\}$ are generated in the first cycle, and $\{X_4, X_5, X_6, X_7\}$ are generated in the second cycle. The required calculations in this conventional order are listed in the above part of Table I. All the $O_n$ and $E_n$ ($n=0,1,2,3$) samples are required to be calculated in 1st and 2nd cycle.

Due to the property of butterfly structure, $X_n$ and $X_{n-8}$ is computed using the same inputs of the 8-point butterfly structure, as shown in the following:

$X_n = E_n + O_n , \ X_{n-8} = E_n - O_n \ \ (n=0,1,2,3)$ (3)

Therefore, if a pair of outputs sharing the inputs of the butterfly is generated in one cycle, the hardware cost for calculations could be reduced. $\{X_0, X_7\}$ is the pair sharing $E_0$ and $O_0$, $\{X_1, X_6\}$, $\{X_2, X_5\}$ and $\{X_3, X_4\}$ are also pairs sharing the inputs of butterfly. Since the pixel parallelism is 4 pixels/cycle, two-pair outputs are generated in each cycle. In the first cycle, $\{X_0, X_7\}$ and $\{X_1, X_6\}$ are generated. In the second cycle, $\{X_2, X_5\}$ and $\{X_3, X_4\}$ are generated. After we reordered the outputs, the required calculations in each cycle are listed in the lower part of Table I. We can see that only two even results and two odd results are required in each cycle. Half calculations are reduced compared with conventional order.

| TABLE I. REQUIRED CALCULATIONS IN CONVENTIONAL ORDER AND PROPOSED RPSIO SCHEME FOR 8-POINT 1D IDCT |
|---------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
|                                | 1st cycle       | 2nd cycle       | 1st cycle       | 2nd cycle       | 1st cycle       | 2nd cycle       | 1st cycle       | 2nd cycle       |
| Outputs                        | Required        | Outputs         | Required        | Calculations    | Outputs         | Required        | Calculations    |                                |
| Conventional order             | calculations    |                  | calculations    |                  | calculations    |                  | calculations    |                                |
| $X_0$                          | $O_0$           | $X_1$           | $O_1$           |                  |
| $X_1$                          | $E_0$           | $X_2$           | $E_1$           |                  |
| $X_2$                          | $O_1$           | $X_3$           | $O_2$           |                  |
| $X_3$                          | $E_1$           | $X_4$           | $E_2$           |                  |
| Proposed RPSIO scheme          |                  |                  |                  |                  |                  |                  |                  |
| $X_0$                          | $O_0$           | $X_1$           | $O_1$           |                  |
| $X_1$                          | $E_0$           | $X_2$           | $E_1$           |                  |
| $X_2$                          | $O_1$           | $X_3$           | $O_2$           |                  |
| $X_3$                          | $E_1$           | $X_4$           | $E_2$           |                  |

For a 16-point IDCT, it will take four cycles to generate 16-point results. We still reorder the outputs to share the inputs of butterfly structure. Within each cycle, $\{X_{15}, X_{16}\}$ are reordered outputs. For a 32-point IDCT, eight cycles are required to generate the 32-point IDCT results. $\{X_{31}, X_{32}\}$ are reordered outputs in each cycle.

**C. Proposed Unified 4/8/16/32 IDCT Architecture Based on RPSIO Scheme**

According to Chen’s algorithm, a 2N-point IDCT can reuse N-point IDCT as the even engine. This recursively reuses can be adopted in a unified architecture. Besides, the proposed RPSIO scheme is used in the unified architecture.

Fig. 2 gives the detail architecture for an 8-point 1D IDCT which reuses 4-point IDCT (IDCT4) as the even engine. As shown in Table I, by using the RPSIO scheme, only two results from even part are required in either cycle. A multiplexer is used to select 2 out of IDCT4 results. The selection method is shown in Fig. 2. For the odd part of 8-point IDCT, two results are required. The circuit for the Odd Generator is shown in Fig. 2. A multiplexer (Mux_TransCoeff) is used to select the corresponding transform coefficients in either cycle. It has to be noted that the multiplier is multiple constant multiplica-
tions. One input of multiplier is permuted input and the other is transform coefficient.

The overall 4/8/16/32-point unified architecture is shown in Fig. 3. For the 16-point IDCT, 2 results are required from odd part within each cycle. The detail circuits of Odd Generator for 16-point (OG16) are given in Fig. 3. Mux_TransCoeff is used to select the transform coefficients for OG16 in different cycles. The corresponding selection signal (sel_o16) value in each cycle is shown in Fig. 4(A). It is equal to cycle count.

For the even part of a 16-point IDCT, it will reuse the results of 8-point IDCT. sel_e8 and sel_o8 will decide four outputs of 8-point IDCT. From the four results, sel_e16 will select two as the results of even part. The selection signal values in each cycle are shown in Fig. 4(A).

For the 32-point IDCT, eight cycles are required to generate 32-point results. The unification scheme is similar with 16-point IDCT. The selection signal values in each cycle are shown in Fig. 4(B).

![Fig. 4. Selection signals for multiplexers in IDCT16 and IDCT32.](image)

**IV. AREA-EFFICIENT TRANSPOSE MEMORY**

Last section gives the proposed 1D IDCT architecture, it is effective for IT1/IT2. For a 2N-point 2-D inverse transform, IT1 results are generated row by row. After all the IT1 results of 2N rows are generated, the IT1 results are fetched column by column to start IT2. Therefore, the transpose architecture is required to store 2Nx2N pixels.

### A. Data Mapping Scheme for SRAM

SRAM instead of register is used to implement the transpose memory. Four SRAMs are used since the data parallelism for 1D IDCT architecture is 4 pixels/cycle. Within each cycle, four IT1 results are written in memory buffer. The writing pattern is shown in Fig. 5. The IT1 result $X_n$ of row R is marked as $(R,n)$ in the Fig. 5.

Taking TU 32x32 as an example to explain the data mapping scheme for SRAM, it takes eight cycles to write IT1 results of one complete row into memory buffer and read IT1 results of one complete column for IT2. When writing the IT1 results of row0, (0,0) are stored in SRAM0. For row1, (1,0) are stored in SRAM1. For row2, (2,0) are stored in SRAM2. For row3, (3,0) are stored in SRAM3. In this writing pattern, (0,0), (1,0), (2,0) and (3,0) are in four SRAMs and could be read in one cycle. Similarly, $(4^m,n)$, $(4^m+1,n)$, $(4^m+2,n)$ and $(4^m+3,n)$ of the same column n are written in different SRAMs, thus can be read out in one cycle. Every eight cycles, one complete column of IT1 results can be fetched to do IT2.

To make full use of memory buffer, after each column of IT1 results have been read out, that part of available memory is written in new IT1 results. The data mapping mode interchanges when the transpose buffer is full of the IT1 results of the same transform block.

By using this cyclic memory organization method, within each cycle, all I/O ports are utilized for writing and reading. Therefore we can achieve 100% I/O utilization for SRAM.

### B. Pipelining Schedule for the SRAM

IT1 and IT2 are pipelined in our design. Two-port SRAM is adopted so that writing IT1 results and reading results for IT2 are able to work in parallel. To avoid the address conflict of writing and reading, we propose a pipelining schedule for write and read, as shown in Fig. 6. In Fig. 6, from cycle 0 to 7, the IT1 results of row31 in block N (blkN,row31) are written in the SRAM. The writing pattern is explained in Fig. 5, four IT1 results are written in each cycle. Meanwhile, the IT1 re-

![Fig. 5. Proposed data mapping scheme for the SRAM.](image)

![Fig. 6. Proposed pipelining schedule for the write (IT1) and read (IT2).](image)
sults of column 0 in block N (blkN,col0) are read out. Within each cycle, four IT1 results are read out. It has to be noted that when reading the last sample of the first column (31,0) in the cycle 7. This sample has already been written into the SRAM in the cycle 0. Thus it could be fetched in the cycle 7. There is no address conflict between writing (blkN,row31) and reading (blkN,col0). From cycle 8 to 15, we begin to write the IT1 results of the next block N+1, the IT1 results of (blkN+1,row0) are written into the SRAMs. Meanwhile, the IT1 results of (blkN,col1) are read out. The corresponding SRAM address for writing and reading are shown in Fig. 6. We can see that there is also no address conflict between writing and reading. After that, every eight cycles, IT1 results of (blkN+1,rowK) are written in and (blkN,colK+1) are read out for IT2.

V. EXPERIMENTAL RESULTS

We have implemented the proposed method in Verilog HDL. The design is synthesized with TSMC 90nm cell library. The area for 1D IDCT is 178,625 um², which is equivalent to 63.8K NAND gates in 90nm process. So a 2D IDCT with pipelined IT1 and IT2 consumes 357,250 um².

In order to demonstrate the effect of our proposal, we compare with the existing designs for HEVC IDCT architecture. A concept of normalized area (NA) is introduced to do the fair performance comparison, and NA is defined as the following equation.

\[
NA = \frac{\text{Gate}}{\text{MaxThroughput}} = \frac{\text{Gate}}{\text{MaxFreq} \times Tp}
\]

where MaxFreq is the maximum frequency of the design, Tp is the throughput within each cycle, which is equal to the pixel parallelism. Gate means the gate count for the logical calculation part of the IDCT excluding the transpose memory. Note that the throughput in [5] is different for various 2N-point IDCT. In the decoder side, the worst case should be considered. In the worst case, [5] could achieve 1244.8M (77.8M*4*4) pixels/s for a 4-point IDCT. Thus the least parallelism is 4 (1244.8/311) pixels/cycle. The comparison results for a HEVC 2D IDCT architecture are shown in Table II. For the logical IDCT computation part, compared with [2], [3] and [5], we can save at least 35% hardware cost in terms of normalized area. Compare with IDCT architecture in [2], the normalized area could be reduce by 64%.

For the transpose memory, the width of each SRAM is 16bit (one IT1 result) and the depth is 256 (32x32/4). Totally 16384 SRAM bits are used. Total SRAM bits in [3] and [5] are the same as ours. We can achieve much smaller area since the data width is significantly reduced. For the fair comparison of various transpose memory designs, we do the memory compiler by the same process TSMC 90nm. And two-port SRAM is adopted for a 2D IDCT with pipelined IT1 and IT2.

The total area for SRAMs is listed in Table III. By using our proposed memory organization scheme, total SRAM area is 80988 um². At least 62% SRAM area could be saved compared with [3] and [5]. [6] use register to implement a transpose memory, which is not so area-efficient. [2] did not give the detail size of the transpose memory for inverse transform part, so we could not evaluate its memory cost.

### Table II. Comparison with other HEVC 2D IDCT architecture.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology(um)</td>
<td>130</td>
<td>90</td>
<td>40</td>
<td>180</td>
<td>90</td>
</tr>
<tr>
<td>Max Throughput (Mpixels/s)</td>
<td>1400</td>
<td>1244</td>
<td>249</td>
<td>3</td>
<td>636</td>
</tr>
<tr>
<td>Max Speed (MHz)</td>
<td>350</td>
<td>311</td>
<td>N/A</td>
<td>300</td>
<td>311</td>
</tr>
<tr>
<td>Pixel Parallelism</td>
<td>4-32</td>
<td>N/A</td>
<td>121</td>
<td>249</td>
<td></td>
</tr>
<tr>
<td>Gate Count (k)</td>
<td>109.2 x 2</td>
<td>117.5 x 2</td>
<td>71</td>
<td>52.3 x 2</td>
<td>63.8 x 2</td>
</tr>
<tr>
<td>Normalized Area (um²)</td>
<td>156</td>
<td>189</td>
<td>285</td>
<td>164.46</td>
<td>102</td>
</tr>
</tbody>
</table>

1) 249-Mpixels/s is the working throughput. 2) Gate count for 2D IDCT. 3) Normalized Area is calculated by equation (4), the unit for Normalized Area is 1/(MHz*pixel/cycle). 4) The parallelism for 4/16/32-point IDCT is different, and the least parallelism is 4 pixels/cycle for 4-point IDCT.

### Table III. Comparison with other SRAM-based transpose memory.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Bits</td>
<td>16384 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of SRAMs</td>
<td>4</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>Each SRAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Width</td>
<td>512bits</td>
<td>16bits</td>
<td>16bits</td>
</tr>
<tr>
<td>Depth</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Area (um²)</td>
<td>68431</td>
<td>6590</td>
<td>20247</td>
</tr>
<tr>
<td>Total SRAM Area</td>
<td>273724</td>
<td>210880</td>
<td>80988</td>
</tr>
<tr>
<td>N/A</td>
<td>(4*68431)</td>
<td>(32*6590)</td>
<td>(4*20247)</td>
</tr>
</tbody>
</table>

1) The type of SRAM is two-port SRAM. 2) The areas are generated by the same process TSMC90nm.

VI. CONCLUSION

In this paper, an area-efficient 4/8/16/32-point IDCT architecture is presented. Firstly, a RPISO scheme is proposed so that the inputs of the butterfly structure could be shared. 35% area can be saved on the logical part. In addition, the proposed memory organization could achieve 100% I/O utilization and save 62% area compared with previous works. Our design can support real-time decoding of 4Kx2K 60fps video sequence.

ACKNOWLEDGEMENT

The present work was supported in part by the Program for Leading Graduate Schools, “Graduate Program for Embodiment Informatics” of the Ministry of Education, Culture, Sports, Science and Technology.

REFERENCES