IEICE TRANSACTIONS

on Electronics

VOL. E100-C NO. 6
JUNE 2017

A PUBLICATION OF THE ELECTRONICS SOCIETY

The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER

A High-Throughput and Compact Hardware Implementation for the Reconstruction Loop in HEVC Intra Encoding

Yibo FAN[†a)], ***Member***, **Leilei HUANG**[†], **Zheng XIE**[†], ***and*** **Xiaoyang ZENG**[†], ***Nonmembers***

SUMMARY In the newly finalized video coding standard, namely high efficiency video coding (HEVC), new notations like coding unit (CU), prediction unit (PU) and transformation unit (TU) are introduced to improve the coding performance. As a result, the reconstruction loop in intra encoding is heavily burdened to choose the best partitions or modes for them. In order to solve the bottleneck problems in cycle and hardware cost, this paper proposed a high-throughput and compact implementation for such a reconstruction loop. By "high-throughput", it refers to that it has a fixed throughput of 32 pixel/cycle independent of the TU/PU size (except for 4×4 TUs). By "compact", it refers to that it fully explores the reusability between discrete cosine transform (DCT) and inverse discrete cosine transform (IDCT) as well as that between quantization (Q) and de-quantization (IQ). Besides the contributions made in designing related hardware, this paper also provides a universal formula to analyze the cycle cost of the reconstruction loop and proposed a parallel-process scheme to further reduce the cycle cost. This design is verified on the Stratix IV FPGA. The basic structure achieved a maximum frequency of 150MHz and a hardware cost of 64K ALUTs, which could support the real time TU/PU partition decision for 4K×2K@20fps videos.

key words: *reconstruction loop, discrete cosine transform (DCT), inverse discrete cosine transform (IDCT), quantization (Q), de-quantization (IQ), high efficiency video coding (HEVC)*

1. Introduction

HEVC standard is the successor of H.264/AVC standard, which continues to adopt the hybrid coding technology based on blocks, but as a new coding standard, it achieves an average gain of 39.3% in terms of BD-BR savings compared with H.264/AVC. It is also summarized that HEVC was designed to be applicable for almost all existing H.264/MPEG-AVC applications, while putting emphasis on high-resolution video coding [1].

With this emphasis, a complicated quad-tree coding structure is adopted by HEVC. To be more specific, the basic processing unit in HEVC is called coding tree unit (CTU) which contains one luma coding tree block (CTB) and two chroma CTB. The size of luma one can be set to 16×16, 32×32 or 64×64. In general situation, it always takes 64×64 to fully explore the performance of HEVC. CTU plays the role of root node of the CU quad-tree while CU plays the role of root node of the TU quad-tree and PU. When the luma CTB is set to a 64×64 block, which is a normal case as mentioned, the size of TU can vary among 4×4, 8×8, 16×16 and 32×32; while the size of PU can vary among 4×4, 8×8, 16×16, 32×32 and 64×64 with 35 possible prediction modes in intra prediction. Although several fast mode decision designs have been proposed, still a considerable amount of candidate PU modes, PU partitions or TU partitions are needed to be traversed by the reconstruction loop.

It can be inferred that the reconstruction loop in intra prediction has become a bottleneck in cycle and hardware cost. Cycle cost origins from the data dependency and the traverse process mentioned above. The former one makes the pipelining operation between TU/PU meaningless and the latter one greatly increases the total amount of data to be processed. Hardware cost origins from 4 different TU sizes and a maximum TU size of 32×32. The former one makes the support to multiple sizes necessary and the latter one directly leads to a high cost in calculation logics and transpose memories.

In order to solve these two problems, this paper proposed a high-throughput and compact hardware implementation for the reconstruction loop in HEVC intra encoding. By the word "high-throughput", it refers to that it has a fixed throughput of 32 pixels per cycle independent of the TU/PU size (except for 4×4 TUs). By the word "compact", it refers to that it fully explores the reusability between DCT and IDCT as well as that between Q and IQ. This design is verified on Stratix IV FPGA. The basic structure achieved a maximum frequency of 150MHz with 64K ALUTs.

This section mainly gives a brief introduction and the rest of this paper is organized as follows. In Sect. 2, motivations, challenges and main contributions are provided with related background and previous works. In Sect. 3, both the top-level architecture and detailed implementations of related modules are illustrated with pictures, equations and necessary explanations. As to the integration of this design, it is discussed in Sect. 4, which includes the calculation of overall cycle cost and practical scenarios to use it. Section 5 gives some comparison data between the proposed design and other implementations. Finally, Sect. 6 concludes this paper.

2. Reconstruction Loop in HEVC Intra Encoding

Related background and previous works would be provided in this section, followed by motivations, challenges and main contributions.

Manuscript received May 20, 2016.
Manuscript revised November 24, 2016.
[†]The authors are with the State Kay Lab of ASIC & System, Fudan University, Shanghai, 200433 China.
a) E-mail: fanyibo@fudan.edu.cn (Corresponding author)
DOI: 10.1587/transele.E100.C.643

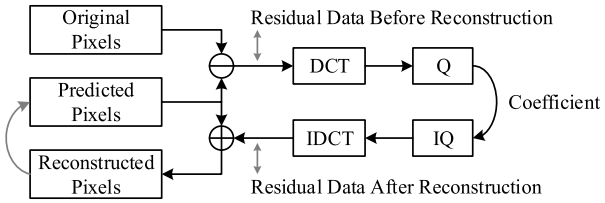

Fig. 1 Reconstruction loop in HEVC

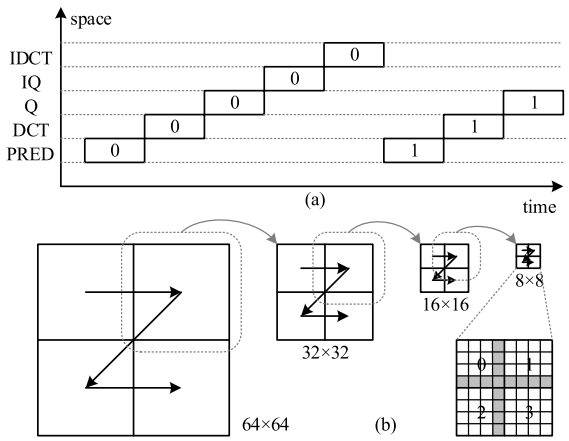

Fig. 2 (a) Space-time diagram for the regular reconstruction loop. (b) The traverse operation in HEVC

2.1 Related Background

Reconstruction loop here refers to four modules, discrete cosine transform (DCT), inverse discrete cosine transform (IDCT), quantization (Q) and de-quantization (IQ). In fact, a prediction module would be needed to form such a loop, but design of this module is not covered in this paper.

As shown in Fig. 1, the residual data calculated from original pixels and predicted pixels are sent to DCT and Q to generate coefficients needed by CABAC. Meanwhile, these coefficients are sent to IQ and IDCT to generate the reconstructed data. The reason to call this process as a loop is that in order to keep the consistence between encoding and decoding, the predicted data should be generated based on the reconstructed data. In other words, there is a data dependency between the predicted data and the reconstructed data.

For intra prediction, this data dependency would affect the coding speed to a great extent, because the reconstructed data needed is from neighboring TUs. Only after the reconstruction towards the former TU is done, the reconstruction towards the current TU could be started, and only after the reconstruction towards the current TU is done, the reconstruction process towards the next TU could be started. This situation is shown in a more visual way by the space-time diagram in Fig. 2. It shows that the prediction operation towards TU 1 is launched after the finish of IDCT towards TU 0 because the reconstructed data located in the last column of TU 0 is needed to do prediction to TU 1. In the same way, prediction to TU 2 needs the last row of both TU 0 and TU 1, while the prediction to TU 3 needs the last column of TU 2. Namely, all the reconstruction process for these TUs need to wait for the finish of their former one. To make things worse, the traverse operation shown in Fig. 2 (b) leads to a great data amount to be processed, which makes the reconstruction loop become a bottleneck in cycle cost.

Of course, cycle cost can be reduced by doing predictions based on original pixels, but it will greatly increase the BD-bitrate, which leads to a poor encoding performance.

As to other existing methods of fast PU mode decision, PU partition decision or TU partition decision, they [2]–[4] can only give a coarse range for possible candidates. A narrower scope can reduce the cycle cost to some extent but the data amount to be processed is still too huge.

Besides the cycle cost, the hardware cost is also a bottleneck problem to be solved. As mentioned before, the maximum TU size in HEVC is up to 32×32, which directly leads to a high cost in calculation logics and transpose memories, not mention that a total of 4 different TU sizes need to be supported.

2.2 Previous Works

Regarding to these difficulties, several papers have focused on related modules and put forward many valuable ideas or implementations.

For DCT/IDCT, Conceicao et al. [5] and Jeske et al. [6] designed a one-dimension (1-D) DCT design for 16×16 TUs and a two-dimension (2-D) IDCT design for 32×32 TUs separately, however they are somehow unpractical to use because they do not support other TU sizes. Later, Park et al. [7] proposed three 2-D DCT structures suitable for all TU sizes, but the reusability between structures for different sizes is not considered, which makes the overall gate count too large. Shen et al. [8] put forward a unified 2-D IDCT design for all sizes and realized the reuse between these sizes, while the throughput is only 4 pixel/cycle, which is not enough for HD applications. Base on the previous works, Meher et al. [9] came up with a new solution, which explored the reusability in DCT for different sizes and succeeded in raising the throughput to 32 pixels per cycle. Unfortunately, the IDCT function is not supported and the maximum frequency is too low.

For transpose memory, Zhao et al. [10] and Langemeyer et al. [11] put forward two SDRAM-based architectures, which, of course, is not suitable for HD applications, considering the access latency from the on-chip processor to the off-chip memory. Later, Bojnordi et al. [12] and Jang et al. [13] designed two SRAM-based architectures which managed to realize the transpose function by dividing SRAMs into several banks, but, either the depth or the width of these banks is not appropriate, which makes the area efficiency and throughput unsatisfying. Based on these work, Shang et al. [14] and Zhu et al.'s [21] proposed a new architecture, which could provide a maximum throughput of 32 pixels each cycle and succeeded in optimizing the depth

and width of banks by adopting a diagonal data mapping method. However, their throughput would decrease with TUs sizes. Thus, when it works with the 2-D DCT design proposed by Meher et al. [9], some throughput of DCT module will be wasted for small TUs. Unfortunately, Meher et al. [9] did not give satisfying suggestions about how to fully utilize his design as well.

For Q/IQ, almost no paper explicitly gave detailed designs, not mention the reuse architecture of them. But the lack of related papers may just come from the simplicity of Q/IQ.

2.3 Motivations and Challenges

As listed above, several papers have proposed many valuable designs but almost none of them has put their design into a practical situation, which leads to a result that these designs may seem good individually but become somehow meaningless when combined together. On the contrary, this paper not only focuses on the module itself but also pays close attention to the interaction between them as well as the practical scenarios to use them. To better embody this attention, detailed motivations and challenges are provided in the following part.

Firstly, it can be concluded from the previous introduction that the reconstruction loop in HEVC intra encoding is a bottleneck in cycle cost and hardware cost.

Secondly, the cycle cost would be much more urgent because it directly determines whether this HEVC encoder can support processing HD videos in real time or not. As mentioned before, the cycle cost origins from the data dependency and the traverse process. The data dependency can be broken by doing traverse based on original pixels, but the BD-bitrate would increase a lot; while the traverse can be reduced by adopting some fast algorithms, but the candidates left are still considerable. As a result, it becomes natural to solve the cycle problems by raising the processing ability towards a single TU/PU. To achieve this, the proposed implementation studies not only the data process speed of each module but also the data exchange between them. In this manner, the high throughput of each individual module would be truly meaningful.

Thirdly, the hardware cost of the reconstruction loop is also a big problem because the processing unit of HEVC is no longer as small as that in H.264. Also as mentioned above, the hardware cost origins from a maximum processing size of 32×32 and 4 different sizes adopted. To solve this problem, the proposed design fully explored the reusability not only in but also between each module. The former reusability is very obvious and already adopted by many designs while the latter one may need some explanations.

As shown in Fig. 2 (a), DCT, Q, IQ and IDCT could not work simultaneously because of the data dependency, thus it is feasible to reuse DCT as IDCT and reuse Q as IQ. As shown in Fig. 3, in time slice 1, DCT/IDCT module plays the role of DCT, and in time slice 2, Q/IQ module acts as Q, while in time slice 3, it acts as IQ, and finally DCT/IDCT

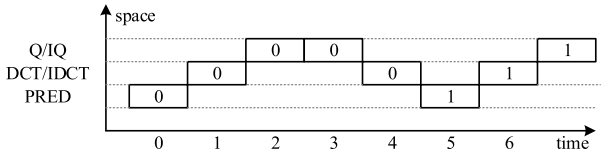

Fig. 3 Space-time diagram for the proposed reconstruction loop

module plays the role of IDCT in time slice 4.

Of course, for PU mode decisions, the situation is different, because the prediction, transform and quantization towards different modes of the same PU can be pipelined, or even be paralleled. But since mode decision can be made based on coefficients, these modules could still be reused as shown in Figs. 17-18. Detailed discussions will be given in Sect. 4.4.

2.4 Application Context and Main Contributions

The application contexts of this design include

i. reconstruction only
ii. TU partition decision
iii. PU partition decision
iv. PU partition decision + PU mode decision

Contributions of this paper include

i. proposing a DCT/IDCT architecture, throughput of which is 32 pixel/cycle independent of the TU/PU size (except for 4×4 TUs)
ii. proposing an SRAM-based transpose memory to cooperate with the above throughput
iii. proposing a DCT/IDCT-reused, Q/IQ-reused architecture based on the application context
iv. analyzing the practical scenarios to present an universal formula to calculate cycle cost of such a reconstruction loop with data dependency and communication cost considered
v. proposing two architecture with different throughput and hardware reuse rate: a basic one and a pipelined one
vi. proposing a dedicated data path for 4×4 TUs to provide a throughput of "32+" pixel/cycle based on the above formula

3. Detailed Implementation

In this section, detailed implementations would be given in the order of top-level architecture, 1-D DCT/IDCT, transpose memory, Q/IQ and design integration.

3.1 Top-Level Architecture: The Basic Structure

This design is composed of three modules, 1-D DCT/IDCT, transpose memory and Q/IQ, which is described in Fig. 4. The basic structure could fulfill the same task as the one in Fig. 1 by adding two more MUXes.

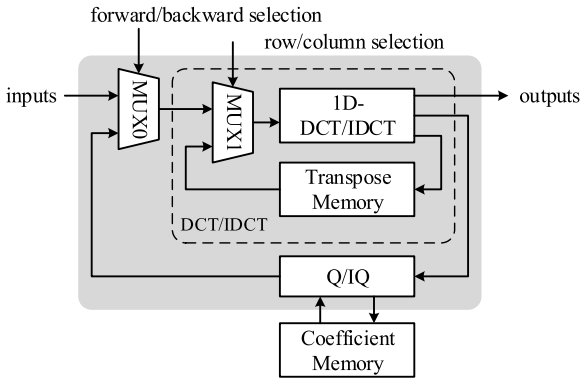

Fig. 4 Brief architecture of the proposed reconstruction loop

MUX0 is used to select data source between Q/IQ and inputs, which determines the current behavior of DCT/IDCT and Q/IQ is forward or backward. MUX1 is used to select data source between MUX0 and transpose memory, which determines the current behavior of 1-D DCT/IDCT is a row transformation or a column one. It may need to be clarified that the DCT/IDCT in this design refers to MUX1, 1-D DCT/IDCT and the transpose memory together.

3.2 1-D DCT/IDCT Design

To explain the detailed implementation of 1-D DCT/IDCT design, some basic features of the DCT in HEVC standard are analyzed here, which can be inferred from Eq. (1)~(2).

$$A_{\mathrm{N}} = P_{\mathrm{N}} \times \begin{bmatrix} A_{\mathrm{N/2}} & 0 \\ 0 & R_{\mathrm{N/2}} \end{bmatrix} \times B_{\mathrm{N}} \tag{1}$$

$$A_{\mathrm{N}}^{\mathrm{T}} = B_{\mathrm{N}}^{\mathrm{T}} \times \begin{bmatrix} A_{\mathrm{N/2}}^{\mathrm{T}} & 0 \\ 0 & R_{\mathrm{N/2}}^{\mathrm{T}} \end{bmatrix} \times P_{\mathrm{N}}^{\mathrm{T}} \tag{2}$$

where, A_{N} denotes for the N×N transformation matrix, P_{N} and B_{N} denote for the permutation matrix and butterfly operation matrix as described in Eq. (3)~(4).

$$P_{\mathrm{N}}(i,j) = \begin{cases} 1\ , & i = 2 \times j \text{ or } i = (j - \mathrm{N}/2) \times 2 + 1 \\ 0\ , & \text{else} \end{cases} \tag{3}$$

$$B_{\mathrm{N}} = \begin{bmatrix} \mathrm{I}_{\mathrm{N/2}} & \widetilde{\mathrm{I}_{\mathrm{N/2}}} \\ \widetilde{\mathrm{I}_{\mathrm{N/2}}} & -\mathrm{I}_{\mathrm{N/2}} \end{bmatrix} \tag{4}$$

where, $\mathrm{I}_{\mathrm{N/2}}$ and $\widetilde{I_{\mathrm{N/2}}}$ denote for the identity matrix and the opposite diagonal identity matrix respectively. As to $R_{\mathrm{N/2}}$, it is made up of the left-half part of the odd rows in A_{N}, while $A_{\mathrm{N}}^{\mathrm{T}}$, $R_{\mathrm{N}}^{\mathrm{T}}$, $B_{\mathrm{N}}^{\mathrm{T}}$ and $P_{\mathrm{N}}^{\mathrm{T}}$ denote for the transposed matrix of A_{N}, R_{N}, B_{N} and P_{N} respectively. It is easy to find that $A_{\mathrm{N/2}}$ is contained in A_{N} after decomposition. By taking advantage of this, 1-D DCT can be implemented in a simple way shown in Fig. 5. In this figure, modules marked with BE_32, BE_16 and BE_8 play the role of butterfly operation matrixes B_{32}, B_{16} and B_8, namely, B_{N} in Eq. (1). Similarly, AE_4, RE_4, RE_8 and RE_16, PE_8, PE_16 and PE_32 play the role of $A_{\mathrm{N/2}}$, $R_{\mathrm{N/2}}$, P_{N} part.

To complete the DCT transformation, modules marked with white color alone would be enough. Detailed structure could also be referred from Zhu et al.'s paper [21].

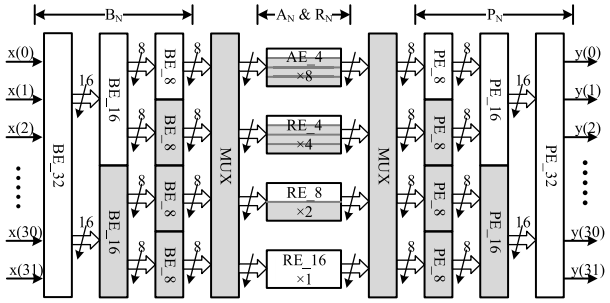

Fig. 5 Structure of 1-D DCT with a fixed-throughput of 32 pixel/cycle

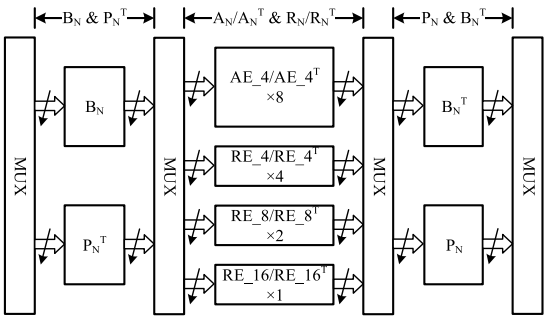

Fig. 6 Structure of 1-D DCT/IDCT

In this paper, in order to keep a fixed throughput of 32 pixel/cycle independent of TU sizes, some extra modules are added, which are marked with gray color.

To integrate 1-D IDCT, two possible solutions could be adopted. One is to reuse all the A_{N}, P_{N}, B_{N} and R_{N} part, the other is to reuse A_{N} and R_{N} part only. The former one would lead to a more compact structure, however, too many MUXes would be needed to rearrange the calculation order, because B_{N} is executed firstly in 1-D DCT process while P_{N} is executed firstly in 1-D IDCT process, which may lead to a sticky timing problem. On the other hand, the latter solution could not only avoid the potential timing issue but also save almost the same hardware cost, considering most of the cost is occupied by A_{N} and R_{N}. A simple schematic diagram for such a structure is shown in Fig. 6.

$B_{\mathrm{N}}^{\mathrm{T}}$ and $P_{\mathrm{N}}^{\mathrm{T}}$ are designed in the same manner as B_{N} and P_{N} described in Fig. 5. As to the reuse of $A_{\mathrm{N}}/A_{\mathrm{N}}^{\mathrm{T}}$ and $R_{\mathrm{N}}/R_{\mathrm{N}}^{\mathrm{T}}$, it utilized the feature of corresponding matrixes. For example, values of matrixes R_4 and R_4^{T} are listed in Eq. (5)~(6).

$$R_4 = \begin{bmatrix} 18 & 50 & 75 & 89 \\ -50 & -89 & -18 & 75 \\ 75 & 18 & -89 & 50 \\ -89 & 75 & -50 & 18 \end{bmatrix} \tag{5}$$

$$R_4^{\mathrm{T}} = \begin{bmatrix} 18 & -50 & 75 & -89 \\ 50 & -89 & 18 & 75 \\ 75 & -18 & -89 & -50 \\ 89 & 75 & 50 & 18 \end{bmatrix} \tag{6}$$

The absolute value in the corresponding position is equal

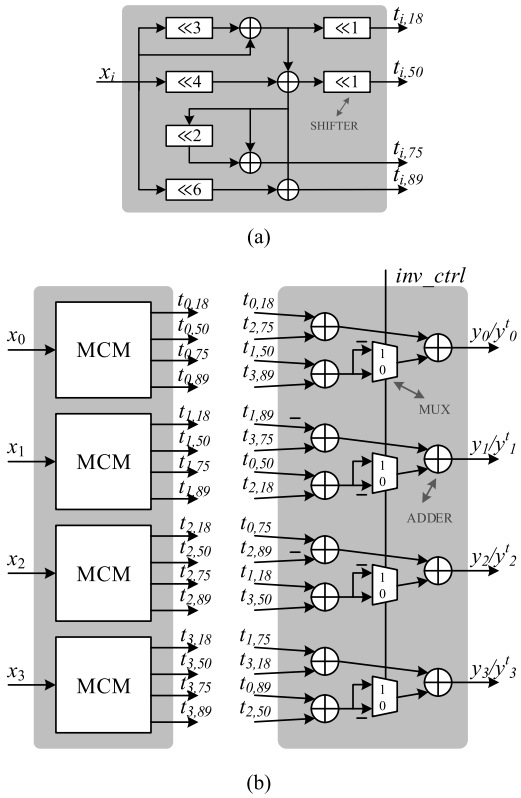

Fig. 7 (a) Design of the corresponding MCM. (b) Structure of R_4/R_4^{T}

and the only difference between R_4 and R_4^{T} is just the sign. It is natural to implement them by multi-constant multipliers (MCM) which shares hardware resources between different constant multipliers (CM). As shown in Fig. 7, one MCM with four constant multiplier of 18, 50, 75 and 89 are designed because any row or any column is always made up of these four values. As to the sign, it is calculated later to reduce more hardware cost. Equation (7) shows the transforming result y of $R_4 x$ and the transforming results y^t of $R_4^{\mathrm{T}} x$. Here, x refers to a 4×1 column vector; x_0, x_1, x_2 and x_3 denote for the zeroth, first, second and the third element of vector x respectively; $y_0 \ldots y_3$ and $y_0^t \ldots y_3^t$ have similar meanings.

$$\begin{aligned}
y_0 &= +(+18x_0 + 75x_2) + (+50x_1 + 89x_3)\\
y_0^t &= +(+18x_0 + 75x_2) - (+50x_1 + 89x_3)\\
y_1 &= -(+50x_0 + 18x_2) + (-89x_1 + 75x_3)\\
y_1^t &= +(+50x_0 + 18x_2) + (-89x_1 + 75x_3)\\
y_2 &= +(+75x_0 - 89x_2) + (+18x_1 + 50x_3)\\
y_2^t &= +(+75x_0 - 89x_2) - (+18x_1 + 50x_3)\\
y_3 &= -(+89x_0 + 50x_2) + (+75x_1 + 18x_3)\\
y_3^t &= +(+89x_0 + 50x_2) + (+75x_1 + 18x_3)
\end{aligned} \tag{7}$$

Base on Eq. (7), R_4/R_4^{T} can be unified into one simple structure as shown in Fig. 7 (b). It can be seen from the same figure, reuse is achieved at a very low cost, which is the same for R_8/R_8^{T} and $R_{16}/R_{16}^{\mathrm{T}}$.

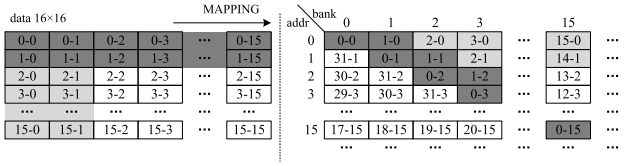

Fig. 8 Shang et al.'s mapping method [14]

3.3 Transpose Memory

Not like the one under H.264 standard, the maximum size of the transformation block in HEVC is as large as 32×32, which is too costly to be stored with registers. But, simply storing the intermediate data with SRAMs in a normal way would lead to an unbearably low throughput because of the read and write features of SRAMs.

As mentioned, an SRAM-based high-throughput solution is already proposed by Shang et al. [14]. However, when it deals with TUs of other sizes, Shang's mapping method would lead to access conflicts. An example of 16×16 TUs is shown in Fig. 8, which uses the horizontal and vertical position to number the pixels, for example, the pixel positioned in row 2 column 5 is marked with 2-5. Column access is marked with lighter gray, while row access is marked with darker gray. Thus, it is obvious to see that both write and read towards 2 lines of a 16×16 TU would cause conflicts, like pixel 1-0 and 0-1 when a column access is launched towards column 0 and 1, or pixel 1-2 and 0-3 when a row access is launched towards row 0 and 1. In another word, this mapping method could not provide a throughput of 32 pixel/cycle when it deals with 16×16 (or other smaller) TUs.

To fully utilize the throughput, a new mapping method is proposed in this paper. Similar to Shang et al.'s method, this method still uses 32 banks, but it could provide a throughput of 32 pixel/cycle for all PU sizes.

To fulfill this target, pixels need to be divided into different groups depending on the TU sizes. To be more specific, when it deals with 32×32 TUs, the proposed method regards pixels in a 1×1 square as one group. When it deals with 16×16 TUs, the proposed method regards pixels in a 2×2 square as one group, for example, pixel 0-0, 0-1, 1-0 and 1-1 would belong to one group in this case. When it deals with 8×8 TUs, it regards pixels in a 4×4 square as one group in a similar way. In a mathematical description, the group of pixel i-j is determined by ⌊j/(32/N)⌋ and ⌊i/(32/N)⌋, where N still denotes the TU size. Groups with the same ⌊i/(32/N)⌋ or the same ⌊j/(32/N)⌋ value would belong to the same access, thus they must be arranged in different banks to avoid access conflicts. One of the feasible mapping method is given here. Groups with the same ⌊j/(32/N)⌋ are arranged to the same address, and each group occupies $(32/\mathrm{N})^2$ continuous banks with an offset of

$$\left(\left(\left\lfloor i/\left(\frac{32}{\mathrm{N}}\right)\right\rfloor + \left\lfloor j/\left(\frac{32}{\mathrm{N}}\right)\right\rfloor\right) \% \left(\frac{\mathrm{N}^2}{32}\right)\right) \times \left(\frac{32}{\mathrm{N}}\right)^2$$

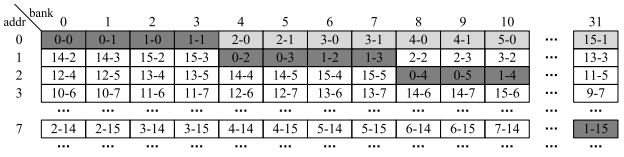

Fig. 9 Proposed mapping method for 16×16 TUs

Fig. 10 Proposed mapping method for 8×8 TUs

In this way, both the groups with the same ⌊i/(32/N)⌋ value and the groups with the same ⌊j/(32/N)⌋ value would be allocated to different banks in a similar way like Shang's mapping method.

As to the pixels in each group, they are arranged from the offset given above according to

$$\left(i\%\left(\frac{32}{\mathrm{N}}\right)\right)\times\left(\frac{32}{\mathrm{N}}\right)+\left(j\%\left(\frac{32}{\mathrm{N}}\right)\right)$$

In this way, all the $(32/\mathrm{N})^2$ pixels in one group can be easily mapped to the $(32/\mathrm{N})^2$ continuous banks without conflicts. Based the above derivation, the mathematic relationship between the pixel position and the mapping position is concluded by Eq. (8), where, $Addr_{i,j}$ and $Bank_{i,j}$ denotes for the address and bank to access pixel i-j.

$$\begin{aligned} Addr_{i,j} &= \left\lfloor j/\left(\frac{32}{\mathrm{N}}\right)\right\rfloor \\ Bank_{i,j} &= \left(\left(\left\lfloor i/\left(\frac{32}{\mathrm{N}}\right)\right\rfloor+\left\lfloor j/\left(\frac{32}{\mathrm{N}}\right)\right\rfloor\right)\%\left(\frac{\mathrm{N}^2}{32}\right)\right)\times\left(\frac{32}{\mathrm{N}}\right)^2 \\ &\quad+\left(i\%\left(\frac{32}{\mathrm{N}}\right)\right)\times\left(\frac{32}{\mathrm{N}}\right)+\left(j\%\left(\frac{32}{\mathrm{N}}\right)\right) \end{aligned} \tag{8}$$

The corresponding mapping for 16×16 TUs is illustrated with Fig. 9. Taking pixel 14-5 in a 16×16 TU as an example, $Addr_{i,j}$ should equal to 2 (= floor(5/(32/16))), and $Bank_{i,j}$ would equal to 5 (= $(7+2)\%8\times4+0\times2+1\times1$).

The corresponding mapping for 8×8 TUs is illustrated with Fig. 10. Taking pixel 6-4 in a 8×8 TU, $Addr_{i,j}$ should equal to 1 (= floor(4/(32/8))), and $Bank_{i,j}$ would equal to 8 (= $(1+1)\%2\times16+2\times4+0\times1$), The one of 32×32 TUs are not illustrated here, because for 32×32 TUs, the proposed mapping method is in fact identical with Shang et al.'s [14]. Or more precisely, Shang et al.'s mapping way can be considered as a special circumstances of the proposed method.

3.4 Q and IQ Design

Either Q or IQ could be concluded into the same equation as expressed by Eq. (9), which makes the reuse quite easy.

$$Q_{out} = (Q_{in}\times Q_{coe} + Offset) \gg Shift \tag{9}$$

For quantization,

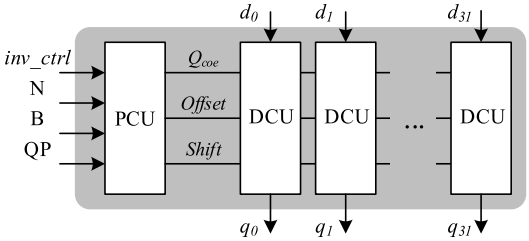

Fig. 11 Simplified structure of Q/IQ with a bandwidth of 32 pixel/cycle

$$\begin{aligned} Q_{coe} &= \mathrm{F}\,(\mathrm{QP}\%6) \\ offset &= 171 \ll (12+\mathrm{QP}/6-\mathrm{M}-(\mathrm{B}-8)) \\ Shift &= 21+\mathrm{QP}/6-\mathrm{M}-(\mathrm{B}-8) \end{aligned} \tag{10}$$

For de-quantization,

$$\begin{aligned} Q_{coe} &= \mathrm{G}(\mathrm{QP}\%6) \ll (\mathrm{QP}/6) \\ Offset &= 1 \ll (\mathrm{M}-2+(\mathrm{B}-8)) \\ Shift &= 1 \ll (\mathrm{M}-1+(\mathrm{B}-8)) \end{aligned} \tag{11}$$

where, Q_{in} and Q_{out} denote the input and output data of Q/IQ; Q_{coe} denotes for the quantization or de-quantization coefficient, which is generated by look-up table F and G; *Offset* denotes for the compensation part; M equals to $\log 2(\mathrm{N})$; N refers to the TU size; B refers to the internal bit depth. The detailed mapping relation of table F and G can be referred from HEVC standard, so they are not listed here. Besides, the final truncation process towards de-quantization results is excluded from Eq. (9) as well.

For the same TU, these parameters are shared, namely, Q_{coe}, *Offset* and *Shift*. A module called parameter calculation unit (PCU) is designed to generate these parameters as shown in Fig. 11, where signal inv_ctrl is used to determine whether the current behavior is quantization or de-quantization. With this reused PCU, a unified module called data calculation units (DCU) is designed to calculate Q_{out} from these parameter and Q_{in}. In another word, it plays the role of Eq. (9). This reduction is considerable, noticing 64 DCUs would be needed, if Q and IQ were not reused.

4. Integration Considerations

As mentioned above, the proposed design not only focuses on the inner structure of the sub-modules but also pays close attention to the interaction between them and, most importantly, the practical scenarios to use them.

4.1 Interaction between Modules and Data Mapping

For interaction part, problems about data exchange are mainly discussed here.

In this paper, problems about data exchange are divided to two different types. One type is data exchange inside the proposed design, namely, the interaction between 1-D DCT/IDCT and Q/IQ. The other type is data exchanges outside the proposed design, namely, the interaction between

the proposed design and other modules.

The former one is solved by the mapping method introduced in Sect. 3.3, otherwise the throughput will be reduced with TU sizes.

The latter one is also solved by a different mapping method proposed by us [15], otherwise the throughput will be dragged by the data exchange outside the proposed design.

Besides, an easily-neglected conflict should be mentioned here, which occurs between Q and IQ. In traditional way, Q and IQ are separate from each other, which means once the quantized coefficient is figured out, they could be immediately sent to IQ. However, in the proposed structure, Q and IQ are reused, thus Q/IQ module is still executing quantization and will not be able to do de-quantization at the same time. Due to this reason, these coefficients needs to be buffered like the intermediate data in DCT/IDCT. Fortunately, these coefficients are needed by CABAC, so they are already buffered.

4.2 Practical Scenarios

For practical scenarios, problems about pipeline stages are mainly discussed here.

Pipeline stages directly influence two key parameters, the cycle cost and the maximum frequency, which, however, are conflicted with each other. If the cycle cost is fewer, then more cycle margin would be left for other modules, but the maximum frequency will drop, which may decrease the cycle margin in return. Advanced HEVC designs usually achieve a relatively high maximum frequency, like Liu et al.'s [16], Jayakrishnan et al.'s [17], Zhou et al.'s [18] and other related works. Then, as a part of encoder, reconstruction loop naturally works under the same frequency with other modules. To better reach the balance between cycle cost and working frequency, the relationship of cycle cost and pipeline stages is calculated and given here. Where L_X denotes for the cycle cost between the first input and the first output of module X; module X can be 1-D

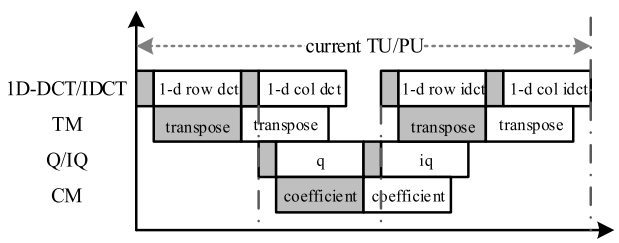

Fig. 12 Space-time diagram to process one TU/PU (PU modes decided)

$$
\begin{aligned}
&L_{\text{1D\_IDCT}} = L_{\text{1D\_DCT}} \\
&L_{\text{IQ}} = L_{\text{Q}} \\
&L_{TM} = L_{CM} = \text{N}^2/\text{T} - 1 \\
&C = L_{\text{1D\_DCT}} + L_{\text{TM}} + L_{\text{1D\_DCT}} + L_{\text{Q}} + L_{\text{CM}} + L_{\text{IQ}} \\
&\quad + L_{\text{1D\_IDCT}} + L_{\text{TM}} + L_{\text{1D\_IDCT}} + \text{N}^2/\text{T} - 1 \\
&\quad = 4 \times L_{\text{1D\_DCT}} + 2 \times L_{\text{Q}} + 4 \times \text{N}^2/\text{T} - 4
\end{aligned} \tag{12}
$$

DCT/IDCT, Q/IQ, transpose memory (TM) and coefficient memory (CM); C denotes for the total cycle cost for the reconstruction loop to process one TU/PU; N denotes for the TU/PU size; T denotes for the throughput. The corresponding space-time diagram is given in Fig. 12.

For PU partition decision, two assumptions are announced here.

i. PU mode decision is already done.
ii. 64×64 PUs are not considered (For 64×64 PU, the diagram is different from others because the largest TU size is only 32×32. As suggested by Pastuszak et. al. [28], 64×64 PU is not discussed in this paper as well.

In this way, situation is almost the same for PU partition decision and TU partition decision.

4.3 TU/PU Partition Decision and Parallel Process

According to Eq. (12), the corresponding cycle cost to do TU/PU partition decision is illustrated in Fig. 13 and listed in Table 1. Attention should be paid on the fact that most of the cycle is spent on the traverse towards 4×4 TU/PUs, for example, 5120 in 8000 for Solution A or 6144 in 9360 for solution A'. However, the hardware cost of a 4×4 2-D DCT/IDCT occupies just a small proportion in the overall cost.

Based on the above analysis, an individual channel for 4×4 TU/PUs is adopted. In another word, it parallels the process to 4×4 TU/PUs with other TU/PUs. This trick is feasible because there is no data dependency between the current 4×4 TU/PU and the bigger TU/PUs it belongs to. Thus, the reconstruction process can be rearranged like the one shown in Fig. 14. The corresponding cycle cost is also listed in Table 1, which achieves a speedup of 29%. However, it is obvious to see that there are so many pipeline bubbles during the traverse process, which makes the hardware utilization ratio not satisfying.

The reason why this solution has a low utilization ratio can be inferred from Eq. (12). According to Eq. (12), for larger TU/PUs, most of the cycle cost is occupied by $4 \times \text{N}^2/\text{T}$, which is determined by the TU/PU size and throughput, while for smaller TU/PUs, $4 \times L_{\text{1D\_DCT}} + 2 \times L_{\text{Q}}$

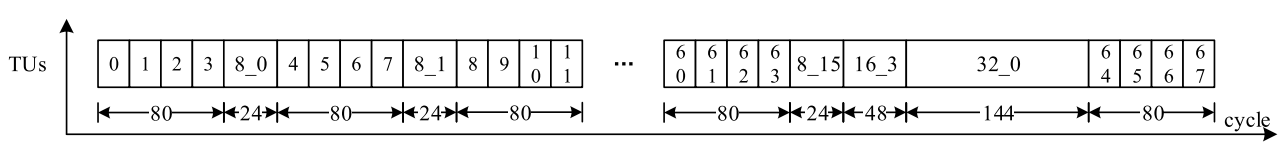

Fig. 13 Cycle cost without parallel process (solution A)

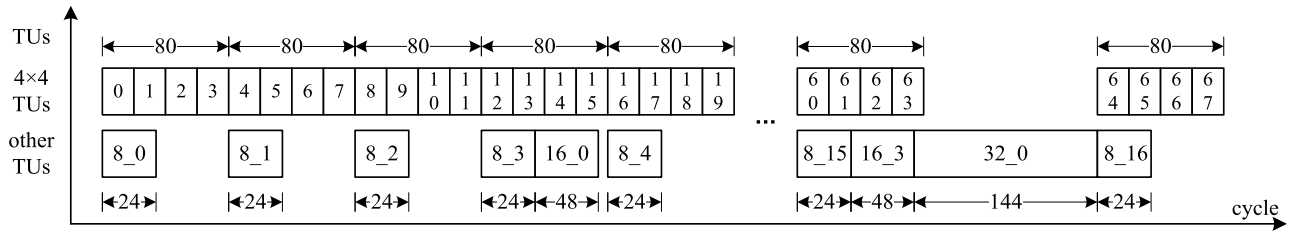

Fig. 14 Cycle cost with parallel process before optimization (solution B)

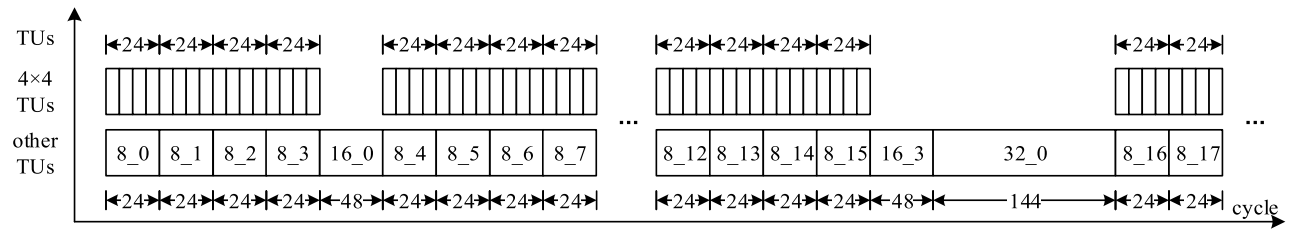

Fig. 15 Cycle cost with parallel process after optimization (solution C)

Table 1 Cycle cost in reconstruction loop

Sol	N	T	L_{1D_DCT}	L_Q	C	Cycle	Freq
A	4	**16**	4	2	20	8000	324
	8	32	4	2	24		
	16	32	4	2	48		
	32	32	4	2	144		
B	4	**16**	4	2	20	5664	239
	8	32	4	2	24		
	16	32	4	2	48		
	32	32	4	2	144		
C	4	**16**	1	1	6	2880	117
	8	32	4	2	24		
	16	32	4	2	48		
	32	32	4	2	144		
A'	4	**16**	5	2	24	9360	379
	8	32	5	2	28		
	16	32	5	2	52		
	32	32	5	2	148		
B'	4	**16**	5	2	24	6672	270
	8	32	5	2	28		
	16	32	5	2	52		
	32	32	5	2	148		
C'	4	**16**	1	1	6	3216	130
	8	32	5	2	28		
	16	32	5	2	52		
	32	32	5	2	148		

Column "Cycle" lists out the total cycle needed to traverse a LCU. Only y component is considered. Column "Freq" lists out the required frequency to encoding 4K×2K @ 20fps videos. Solution X and X' adopt the same structure, but L_{1D_DCT} and L_Q is different (X = A, B, C).

is dominant, which is determined by pipeline stages. As a result, the cycle costs to process one 4×4 TU/PU and one 8×8 TU/PU is almost the same. However, the calculation complexity of 4×4 transform is far low than the other transforms, which makes a pipeline stage of 4 or 5 unnecessary. In order to balance the cycle cost, the dedicated data path for 4×4 TUs is redesigned to the one shown in Fig. 16, where the dotted line indicates the pipeline stages. In other words, L_{1D_DCT} and L_Q are reduced to 1 cycle according

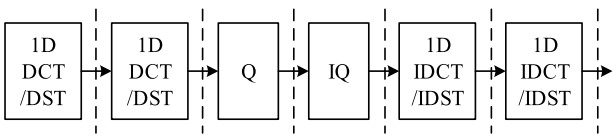

Fig. 16 Redesigned dedicated 4×4 TU/PU data path

to the calculation complexity. The cycle cost diagram is shown in Fig. 15, which achieves a speedup of about 65% as listed in Table 1. Several points may need to be discussed here. Firstly, DST is also integrated in this data path. Secondly, these modules are also highly reused, thus only one 1-D DCT/IDCT/DST/IDST and one Q/IQ module is used. Thirdly, it is no use integrating transpose memory because all of the data needed is generated in one cycle and can be directly sent to the next stage.

4.4 PU Mode Decision and the Pipelined Structure

If PU modes are considered, the space-time diagram in Fig. 12 would change into the one shown in Fig. 17, where, TM and CM can be omitted, because they do not occupy extra cycles thanks to the mapping method given in Sect. 3.3; PRED denotes for prediction module; MD denotes for mode decision module; N_{md} denotes for the amount of candidate modes. The corresponding cycle cost formula is given in Fig. 17 as well, according to which, 5 modes can be supported @ 398MHz as listed in Table 2 solution D.

To support more prediction modes, a pipelined structure is proposed here, which takes advantages of the fact that no data dependency exists when PRED does predictions to the same PU with different PU modes. In this structure, one extra 1-D DCT and transpose memory module is added to realize the pipeline. The corresponding space-time diagram and cycle cost formula are given in Fig. 18, according to which, 13 modes could be supported @ 409MHz as listed

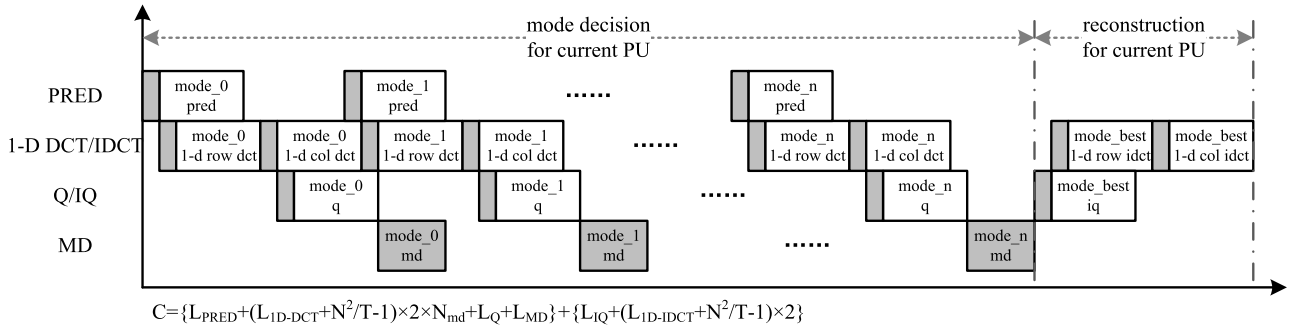

Fig. 17 Space-time diagram to process one PU (PU candidate modes considered) (solution D)

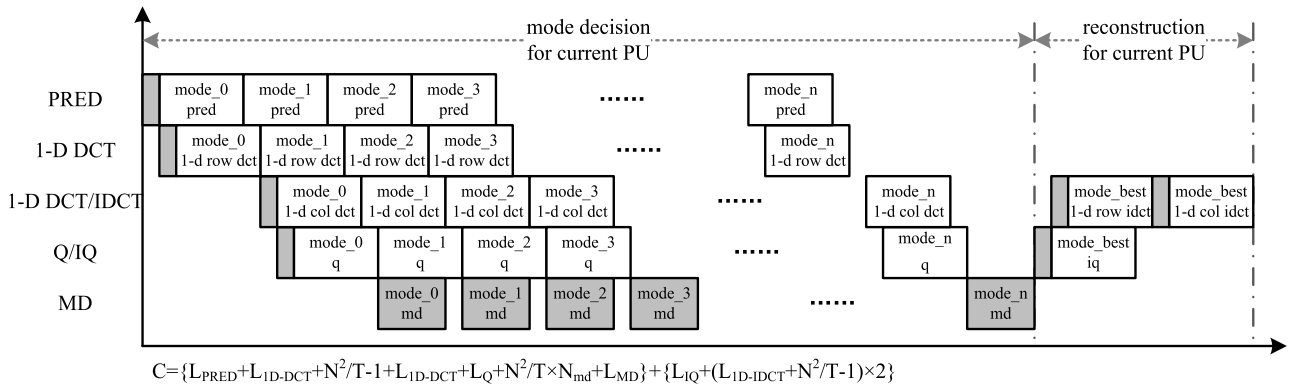

Fig. 18 Space-time diagram to process one PU (the second structure) (PU candidate modes considered) (solution E)

Table 2 Cycle cost in reconstruction loop

S	N	T	$L_{1D\ DCT}$	L_Q	L_{PRED}	L_{MD}	C	N_{md}	Cyc	Freq
D	4	16	4	2	1	1	16	5	9816	398
	8	32	4	2	5	5	86			
	16	32	4	2	5	5	158			
	32	32	4	2	5	5	446			
E	4	16	4	2	1	1	21	13	10092	409
	8	32	4	2	5	5	63			
	16	32	4	2	5	5	159			
	32	32	4	2	5	5	543			
F	4	16	/	/	/	/	16	16	9900	401
	8	32	5	2	5	5	69			
	16	32	5	2	5	5	183			
	32	32	5	2	5	5	639			

Only y component is considered; Column "N_{md}" lists out the amount of candidate modes; "Cycle" lists out the total cycle cost to search from 4×4 to 32×32 PUs with N_{md} modes for each possible PUs; "Freq" lists out the required frequency to encoding 4K×2K@20fps videos;

in Table 2 solution E.

It is obvious to see that the 4×4 data path for solution E is not well balanced because 21×4=84, which is far bigger than 63. However, because of the low calculation complexity of 4×4 transforms, 1-D row DCT, 1-D column DCT and Q could be put into one pipeline stage, which is the same for IQ, 1-D row IDCT and 1-D column IDCT. If the cycle cost is still not balanced, an extra dedicated 4×4 data path could be added to help the balance. The cycle cost of a balanced solution is listed in Table 2 solution F, according to which, 16 modes can be supported @ 401MHz.

For solution D-F, both L_{PRED} and L_{MD} are assumed to be 1 cycle for 4×4 PUs and 5 cycles for other PUs.

Table 3 Synthesis results of the proposed reconstruction loop

N	Module	Max. Freq.	ALUTs	reuse rate
1	1-D row DCT	177.6 MHz	29311	
2	1-D col DCT	191.4 MHz	24497	
3	1-D row IDCT	177.0 MHz	28812	
4	1-D col IDCT	190.4 MHz	26093	
5	1-4 reused	161.2 MHz	47746	56.08 %
6	Q	209.2 MHz	5953	
7	IQ	208.7 MHz	5182	
8	6-7 reused	189.4 MHz	6198	44.34 %
9	transpose memory	211.6 MHz	8131	
10	basic structure	149.9 MHz	64070	
11	pipelined structure	154.0 MHz	96658	

5. Comparison

This design is verified on the Stratix IV FPGA, results of which are listed in Table 3.

Both DCT/IDCT, Q/IQ engines with reuse and engines without reuse are implemented to get the corresponding maximum frequency and ALUTs.

Basic structure (Sol.D) occupies 64K ALUTs @ 150MHz, containing a reused DCT/IDCT, a reused Q/IQ, a

transpose memory, a dedicated 4×4 PU path and some glue logics, while the pipelined structure (Sol.F) occupies 97K ALUTs @ 154MHz, containing an extra 1-D row DCT and transpose memory.

5.1 Comparisons in Data Buffering

As mentioned previously, most papers only focus on their designs alone and rarely pay attention to the data exchange between modules, thus the high-throughput feature of their designs may be easily dragged by the data exchange. Based on the above situation, this paper proposed an SRAM-based data mapping method to buffer the intermediate data inside DCT or IDCT. Combined with our previously-proposed data mapping method to buffer the data around DCT/IDCT [15], this solution can provide a throughput of 32 pixel/cycle based on just several SRAMs. On the contrary, although Meher et al.'s design [9] has the same throughput, it is implemented by registers which would occupy too much area; while other designs [10]–[14] could not provide the same throughput, though their designs are based on SRAMs too. It should be pointed out that it is not totally negative for register-based designs. According to Zhu et al.'s work [21], a register-based design with clock gating technique can save more power. However, clock gating in such a low granularity may be disfavored by FPGA designs because the clock resources are relatively precious.

Table 4 Comparisons in data buffering

Design	THROUGHPUT	Implementation
Meher et al.'s [9]	32 pixels/cyc	Registers
Zhao et al.'s [10]	/	SDRAM
Langemeyer et al.'s [11]	/	SDRAM
Bojnordi et al.'s [12]	4 pixels/cyc	SRAM (8 banks)
Jang et al.'s [13]	2 pixels/cyc	SRAM (4 banks)
Shang et al.'s [14]	N pixels/cyc	SRAM (32 banks)
Zhu et al.'s [21]	N pixels/cyc	SRAM (32 banks)
Proposed Design	32 pixels/cyc	SRAM (32 banks)

N refers to the TU size

5.2 Comparisons in DCT/IDCT

Several papers have proposed brilliant DCT/IDCT designs as listed in Table 5. Since the proposed design is implemented on FPGA, comparison would be made between FPGA designs.

Conceicao et al. [5] realized a 2-D IDCT design, throughput of which is 32 pixels per cycle, however only size 32×32 is supported. A maximum frequency of 43.62MHz is also too low, which, of course, would lead to a low cost. In a similar way, Jeske et al.'s [6] and Martuza et al.'s [20] designs only support size 16×16 or size 8×8.

Darji et al.'s [26] design supports all TU sizes, but they are not integrated together. Most importantly, for size 32×32, the maximum frequency of this design is 23.73MHz only.

Arayacheeppreecha et al.'s [27] design integrates all TU sizes, but a throughput of 8 pixel/cycle is not satisfying as well.

Kalali et al.'s [23] design has a higher maximum frequency and throughput, as a result of which, it consumes 34K (LUTs).

Finally, the proposed DCT/IDCT part in the proposed design occupies 48K ALUTs, but it could support both 1-D DCT and 1-D IDCT, all TU sizes, a throughput of 32 pixels/cycle and a maximum frequency of 161.2MHz using the same Stratix IV technology with Conceicao et al.'s [5].

Hardware cost of the reused part in DCT modules is listed in Table 6, including module AE_4, RE_4, RE_8 and RE_16 as mentioned in Sect. 3.2. It can be seen from this table that the hardware cost of 4×4 TU/PU data path is very low.

6. Conclusion

In HEVC encoder, the reconstruction loop in intra encoding

Table 5 Comparisons in DCT/IDCT

Design	Function	Size	Platform	Technology	Throughput	Gate Count	Freq
Park [7]	1-D DCT	4/8/16/32*	ASIC	0.15 um	16 pixel/cycle	127 K / 105 K	94 M
Shen [8]	2-D IDCT	4/8/16/32	ASIC	0.13um	4 pixel/cycle	134 K	350 M
Meher [9]	2-D DCT	4/8/16/32	ASIC	90 nm	32 pixel/cycle	208 K / 347 K	187 M
Budagav [19]	2-D DCT/IDCT	4/8/16/32	ASIC	45 nm	32 pixel/cycle	156 K	250 M
Zhu [21]	2-D DCT/IDCT**	4/8/16/32	ASIC	90 nm	N pixel/cycle	412 K / 320 K	311 M
Zhao [22]	2-D DCT	4/8/16/32*	ASIC	45 nm	N pixel/cycle	206 K	333 M
Yao [24]	1-D IDCT	4/8/16/32	ASIC	65 nm	1 pixel/cycle	40 K	500 M
Dias [25]	2-D DCT	4/8/16/32	ASIC	90 nm	8/16/32/32	328 K	400 M
Conceicao [5]	2-D IDCT	32	FPGA	Stratix IV	32 pixel/cycle	28 K (ALUTs)	43.62 M
Jeske [6]	1-D IDCT	16	FPGA	Stratix III	/	5 K (ALUTs)	87.60 M
Martuza [20]	2-D DCT	8	FPGA	Virtex 4	1 pixel/cycle	706 (LUTs)	/
Kalali [23]	2-D IDCT	4/8/16/32	FPGA	Virtex 6	N pixel/cycle	34 K (LUTs)	150 M
Darji [26]	1-D DCT	4/8/16/32*	FPGA	Spartan 3E	/	12 K (LEs)	23.73 M
Arayacheeppreecha [27]	1-D DCT	4/8/16/32	FPGA	Spartan 3A	8 pixels/cycle	15 K (LUTs)	205 M
Proposed Design	1-D DCT/IDCT	4/8/16/32	FPGA	Stratix IV***	32 pixel/cycle	48 K (ALUTs)	161.2 M

* this design could support 4/8/16/32 but they are not unified together; ** this design could support DCT/IDCT but they are not unified together;

*** the specific type is EP4SGX530KH40C2; N refers to the TU size

Table 6 ALUTs cost of reused part in DCT modules

Function / Module	DCT	IDCT	DCT/IDCT	reuse rate
AE_4	267	319	411	29.86 %
RE_4	750	754	860	42.82 %
RE_8	2959	2972	3380	43.01 %
RE_16	11083	11083	12672	42.83 %

is heavily burdened to choose the best partitions or modes for them. In order to solve the bottleneck problems in cycle and hardware cost, this paper not only focuses on the module itself but also pays close attention to the interaction between them as well as the practical scenarios to use them. Based on the above studies, a high-throughput and compact implementation is proposed for application contexts including reconstruction only; TU partition decision; PU partition decision; PU partition decision + PU mode decision. Several contributions at module, interaction and system level are made to achieve the throughput of "32+" pixel/cycle and a satisfying hardware cost. This design is verified on the Stratix IV FPGA. The basic structure achieved a maximum frequency of 150MHz and a hardware cost of 64K ALUTs, which could support the real time TU/PU partition decision for 4K×2K@20fps videos.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61674041, in part by the STCSM under Grant 16XD1400300, in part by the State Key Lab of ASIC & System under Grant 2015MS006.

References

[1] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky, and O. Hadar, "Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders," 2013 Picture Coding Symposium (PCS), pp.394–397, Dec. 2013.

[2] H. Liu and Y. Jie, "Fast HEVC intra-prediction mode decision based on conditional selection with hybrid cost ranking," 2015 IEEE Workshop on Signal Processing Systems (SiPS), Hangzhou, pp.1–6, 2015.

[3] J. Tariq and Sam Kwong, "Hybrid Fast Intra Mode Decision and Early Termination of Prediction Unit (PU) Splitting for HEVC," 2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Kowloon, pp.1782–1786, 2015.

[4] X. Shang, G. Wang, T. Fan, and Y. Li, "Fast CU size decision and PU mode decision algorithm in HEVC intra coding," 2015 IEEE International Conference on Image Processing (ICIP), Quebec City, QC, pp.1593–1597, 2015.

[5] R. Conceicao, J.C. Souza, R. Jeske, M. Porto, J. Mattos, and L. Agostini, "Hardware design for the 32×32 IDCT of the HEVC video coding standard," 2013 26th Symposium on Integrated Circuits and Systems Design (SBCCI), pp.1–6, Sept. 2013.

[6] R. Jeske, J.C. de Souza, G. Wrege, R. Conceicao, M. Grellert, J. Mattos, and L. Agostini, "Low cost and high throughput multiplierless design of a 16 point 1-D DCT of the new HEVC video coding standard," 2012 VIII Southern Conference on Programmable Logic (SPL), pp.1–6, March 2012.

[7] S.Y. Park and P.K. Meher, "Flexible integer DCT architectures for HEVC," 2013 IEEE International Symposium on Circuits and Systems (ISCAS), pp.1376–1379, May 2013.

[8] S. Shen, W. Shen, Y. Fan, and X. Zeng, "A Unified 4/8/16/32-Point Integer IDCT Architecture for Multiple Video Coding Standards," 2012 IEEE International Conference on Multimedia and Expo (ICME), pp.788–793, July 2012.

[9] P.K. Meher, S.Y. Park, B.K. Mohanty, K.S. Lim, and C. Yeo, "Efficient Integer DCT Architectures for HEVC," IEEE Trans. Circuits Syst. Video Technol., vol.24, no.1, pp.168–178, Jan. 2014.

[10] B. Tu, D. Li, and C. Han, "Two-dimensional image processing without transpose," 2004 7th International Conference on Signal Processing, 2004. Proceedings. ICSP '04. vol.1, pp.523–526, 31 Aug.-4 Sept. 2004.

[11] S. Langemeyer, P. Pirsch, and H. Blume, "Using SDRAMs for two-dimensional accesses of long 2n ×2m-point FFTs and transposing," 2011 International Conference on Embedded Computer Systems (SAMOS), pp.242–248, July 2011.

[12] M.N. Bojnordi, N. Sedaghati-Mokhtari, O. Fatemi, and M.R. Hashemi, "An Efficient Self-Transposing Memory Structure for 32-bit Video Processors," 2006 IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2006, pp.1438–1441, Dec. 2006.

[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] Q. Shang, Y. Fan, W. Shen, S. Shen, and X. Zeng, "Single-Port SRAM-Based Transpose Memory With Diagonal Data Mapping for Large Size 2-D DCT/IDCT," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.22, no.11, pp.2422–2426, Nov. 2014.

[15] Y. Fan, L. Huang, Y. Bai, and X. Zeng, "A Parallel-Access Mapping Method for the Data Exchange Buffers around DCT/IDCT in HEVC Encoders Based on Single-Port SRAMs," IEEE Transactions on in Circuits and Systems II: Express Briefs, vol.62, no.12, pp.1139–1143, 2015.

[16] Z. Liu, D. Wang, H. Zhu, and X. Huang, "41.7BN-pixels/s reconfigurable intra prediction architecture for HEVC 2560×1600 encoder," 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp.2634–2638, May 2013.

[17] P. Jayakrishnan, P.V.A. Lincy, and R.M. Niyas, "A high speed real time multi-bin CABAC encoder for ultra high resolution video," 2013 International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICE-CCN), pp.207–210, March 2013.

[18] N. Zhou, D. Ding, and L. Yu, "On hardware architecture and processing order of HEVC intra prediction module," 2013 Picture Coding Symposium (PCS), pp.101–104, Dec. 2013.

[19] M. Budagavi and V. Sze, "Unified forward+inverse transform architecture for HEVC," 2012 19th IEEE International Conference on Image Processing (ICIP), pp.209–212, Sept. 30 2012-Oct. 3 2012.

[20] M. Martuza and K. Wahid, "A cost effective implementation of 8×8 transform of HEVC from H.264/AVC," 2012 25th IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), pp.1–4, April 29 2012-May 2 2012.

[21] J. Zhu, Z. Liu, and D. Wang, "Fully pipelined DCT/IDCT/Hadamard unified transform architecture for HEVC Codec," 2013 IEEE International Symposium on Circuits and Systems (ISCAS), pp.677–680, May 2013.

[22] W. Zhao, T. Onoye, and T. Song, "High-performance multiplierless transform architecture for HEVC," 2013 IEEE International Symposium on Circuits and Systems (ISCAS), pp.1668–1671, May 2013.

[23] E. Kalali, E. Ozcan, O.M. Yalcinkaya, and I. Hamzaoglu, "A low energy HEVC Inverse DCT hardware," 2013 IEEE Third International Conference on Consumer Electronics Berlin (ICCE-Berlin), ICCEBerlin 2013, pp.123–124, Sept. 2013.

[24] Z. Yao, W. He, L. Hong, G. He, and Z. Mao, "Area and throughput efficient IDCT/IDST architecture for HEVC standard," 2014 IEEE International Symposium on Circuits and Systems (ISCAS), pp.2511–2514, June 2014.

[25] G. Pastuszak, "Hardware Architectures for the H.265/HEVC Discrete Cosine Transform," IET Image Processing, vol.9, no.6, pp.468–477, June 2015.
[26] A.D. Darji and R.P. Makwana, "High-performance multiplierless DCT architecture for HEVC," 2015 19th International Symposium on VLSI Design and Test (VDAT), pp.1–5, June 2015.
[27] P. Arayacheeppreecha, S. Pumrin, and B. Supmonchai, "Flexible input transform architecture for HEVC encoder on FPGA," 2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), pp.1–6, June 2015.
[28] G. Pastuszak and A. Abramowski, "Algorithm and Architecture Design of the H.265/HEVC Intra Encoder," IEEE Trans. Circuits Syst. Video Technol., vol.26, no.1, pp.210–222, Jan. 2016.

Yibo Fan received the B.E. degree in electronics and engineering from Zhejiang University, China in 2003, M.S. degree in microelectronics from Fudan University, China in 2006, and Ph.D. degree in engineering from Waseda University, Japan in 2009. From 2009 to 2010, he worked as an assistant professor in Shanghai Jiaotong University, and currently, he is the associate professor in the college of microelectronics of Fudan University. His research interesting includes image processing, video coding and associated VLSI architecture.

Leilei Huang received the B.S. degree in Microelectronics and Solid Electronics from Fudan University, Shanghai, China, in 2014. He is currently pursuing toward the M.S. degree in Microelectronics and Solid Electronics from Fudan University. His research interests include VLSI design, algorithms and corresponding VLSI architectures for multimedia signal processing.

Zheng Xie received the B.S.degree in Integrated Circuit Design and Integrated System from TianJing University in 2013 and the M.S. degree in Integrated Circuit Engineering from Fudan University,shanghai, China, in 2015. His research interests include VLSI design and algorithms optimization for multimedia signal processing and wireless communication.

Xiaoyang Zeng received the B.S. degree from Xiangtan University, Xiangtan, China, in 1992, and the Ph.D. degree from Changchun Institute of Optics, Fine Mechanics, and Physics, Chinese Academy of Sciences, Changchun, China, in 2001. From 2001 to 2003, he was a Postdoctoral Researcher with Fudan University, Shanghai, China. Then, he joined the State Key Lab of ASIC and System, Fudan University, as an Associate Professor, where he is currently a Full Professor and the Director. His research interests include information security chip design, system-on-chip platforms, and VLSI implementation of digital signal processing and communication system.