Indonesian Journal of Electrical Engineering and Computer Science
Vol. 6, No. 1, April 2017, pp. 104 ~ 109
DOI: 10.11591/ijeecs.v6.i1.pp104-109

Exploring the Design Space of HEVC Inverse Transforms with Dataflow Programming

Khoo Zhi Yion, Ab Al-Hadi Ab Rahman
Faculty of Electrical Engineering, Universiti Teknologi Malaysia

Abstract

This paper presents the design space exploration of the hardware-based inverse fixed-point integer transform for High Efficiency Video Coding (HEVC). The designs are specified at high-level using CAL dataflow language and automatically synthesized to HDL for FPGA implementation. Several parallel design alternatives are proposed with trade-off between performance and resource. The HEVC transform consists of several independent components from 4x4 to 32x32 discrete cosine transform and 4x4 discrete sine transform.This work explores the strategies to efficiently compute the transforms by applying data parallelism on the different components. Results show that an intermediate version of parallelism, whereby the 4x4 and 8x8 are merged together, and the 16x16 and 32x32 merged together gives the best trade-off between performance and resource. The results presented in this work also give an insight on how the HEVC transform can be designed efficiently in parallel for hardware implementation.

Keywords*: Video Coding, Transform, FPGA, IDCT*

1. Introduction

Nowadays, people can play and manipulate image and video content via various electronic and smart devices. Due to the evolution of the quality of the video from High Definition (HD) to Ultra High Definition (UHD), HEVC is considered as the latest video compression standard, which is accepted by International Telecommunication Union Telecommunication Standardization Sector (ITU-T) [1]. HEVC decoder provides at least 36% higher coding efficiency compared to H.264 or Advance Video Coding (AVC) algorithm ([2, 3]). The decoder with Reconfigurable Video Coding Common Actor Language (RVC-CAL) programming language application by using parallel architecture can be used to optimize further the current series implementation. The inverse HEVC transform, or here known as the xIT block is one of the core components in a HEVC decoder as it performs fixed-point integer transform of the input data stream [2].

There are several advantages of the xIT block, one of them is it does not have dependen- cies, and hence it could be optimized for data parallelism. The parallel implementation can also improve further the computing time of the video processing compared to the one that have been designed in series. The appropriate parallel and serial fusion designs are also able to optimize further the resources utilization [4].

This paper describes the parallel and serial design and implementation of HEVC transform using CAL dataflow actor programming [5]. The high-level dataflow specifications are synthesized automatically to HDL using the appropriate tools [6], and then synthesized at low level to Xilinx FPGAs. The results are analyzed for latency, frequency, and FPGA resource utilizations. Based on the explored designs, we present the one that shows the best trade-off between performance and resource.

2. Background and Related Works

In the hybrid block-based HEVC coding mechanism, the transform are implemented often to the residual signal from RQT. The residual signal of a picture is split into blocks with identical length and width, both with certain integer power of two in the encoder. Every rows and columns can undergo one dimensional transform individually and exhaustively in order to obtain two di- mensional transform results. After that, the resulting transform coefficient will be

Received November 29, 2016; Revised January 25, 2017; Accepted February 15, 2017

pushed to the quantization process to get the quantized transform coefficient at the encoder [4].

Meanwhile, the reverse process occurs at the decoder, which is convert the quantized transform coefficient into the de-quantized transform coefficient by multiplying it with quantization step size. Eventually, the de-quantized values are passed back to the residual block of quantized samples and having increment to the intra- or inter-prediction samples in order to receive the reconstructed block. In addition, the forward matrices is the transpose of the inverse matrices [7]. They are constructed in the way that almost lossless reimplementation compared to the input residual signal without consider the quantization process. Figure 1 illustrates the coding algorithm used by the encoder and the decoder where C is the transform matrix, Qstep is the quantization step size, coeff is the transform coefficient and coeffQ is the de-quantized transform coefficient [4].

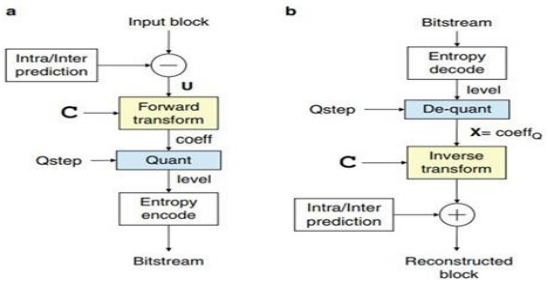

Figure 1. High-level overview of video encoder and decoder

The core transform of HEVC is noted with flexible size from 4x4 to 32x32 with increment in power of two to allow two-dimensional transforms [7]. Although the flexibility provided could enhance the performance of block-based motion-compensated video compression, the method to use it also become more complex [8]. In HEVC, only inverse transform are defined instead of forward transform, so the forward transform are using the exact IDCT in default. The finite precision of inverse discrete cosine transform (IDCT) for all transform sizes are applied rather than the exact IDCT. It is because the main reason to obtain the transform is to disassociate with the input residual block, which is optimally obtained by the Karhunen-Loeve transform (KLT), and hence the exact IDCT could be neglected. The situation is obvious when the alternate transform is utilized in the 4x4 luma intra-picture prediction integer transform, which is conducted by using 4x4 integer transform according to the discrete sine transform (DST) [9]. There are many different approach of IDCT were identify for the core transform in the evolution of HEVC. The pioneer version of HEVC Test Model HM1 exercise the 4x4 and 8x8 blocks transform, and the 16x16 and 32x32 blocks transform by using AVC transform and Chens fast IDCT respectively [10], which contribute to the increasing of complexity of HEVC transform. For instance, the non-flat inverse quantization matrices are experienced by different transform sizes, which require bigger block sizes for huge transform of de-quantization matrices with high application cost, moreover, the difficulty in hardware sharing across various transform capacity will raise the area of the hardware. Besides that, larger transpose buffer size is required to save the transitional results obtained from the first transform stage in two-dimensional transform, which will have to raise the size and bandwidth of the memory [4].

Another problem is encountered with the full factorization structure that needs cascaded multiplier and transitional rounding for 16x16 and 32x32 transforms. It will consequently add the data path dependencies and affect parallel processing performance, as well as extend the bit width for multipliers and accumulators. On the other hand, the increasing of area and the rising of circuit delay will also be faced in hardware with the consequence of restricting the highest frequency available for the inverse transform process [9]. Lastly, a new approach of core transform was accepted so that the difficulties in HM1 design could be overcame as explaining in [11] and to achieve better efficiency in applying the transform to hardware besides SIMD machines. The HEVC core transform matrices were constructed to fulfil the specification of almost orthogonal and identical norm of all basis vectors, identical

symmetry properties as the IDCT basis vectors, approximately the same as IDCT, 16 bit transpose buffer, 8 bit representation of transform matrix components. Furthermore, it employed smaller transform matrices in bigger transform matrices, able to process the multipliers by using 16 bit or less without the implementation of cascade multiplications or intermediate rounding, and also apply the accumulators with less than 32 bit [12].

3. Design Methodology

3.1. Tools and Models

The software used for the design and re-design of the xIT block is the Eclipse Orcc Com- piler [13]. The parallel architectures have been created using dataflow programming, know as the CAL actor language. This high-level description can be synthesized automatically into C program- ming language for software implementation. For the simulation in software, the C programming files were linked into the execution files and generated by using MinGW based on the Make files, which were compiled and made via Cmake GUI.

Similarly with the software method, the various parallel and series designs have been created and then synthesized into HDL for hardware implementation. The low-level hardware synthesis and implementation have been made using Xilinx Vivado into the Artix7 FPGA. The timing and resource utilization reports have also been collected using Xilinx Vivado.

Based on the analysis using the methodology presented in [14] and [15], the original xIT block is able to operate in parallel configuration, which means the inverse transform of 4x4 DST, 4x4 IT, 8x8 IT, 16x16 IT and 32x32 IT could be carried out concurrently and independently. Figure 2 shows the structural block diagram of the original parallel implementation of xIT block.

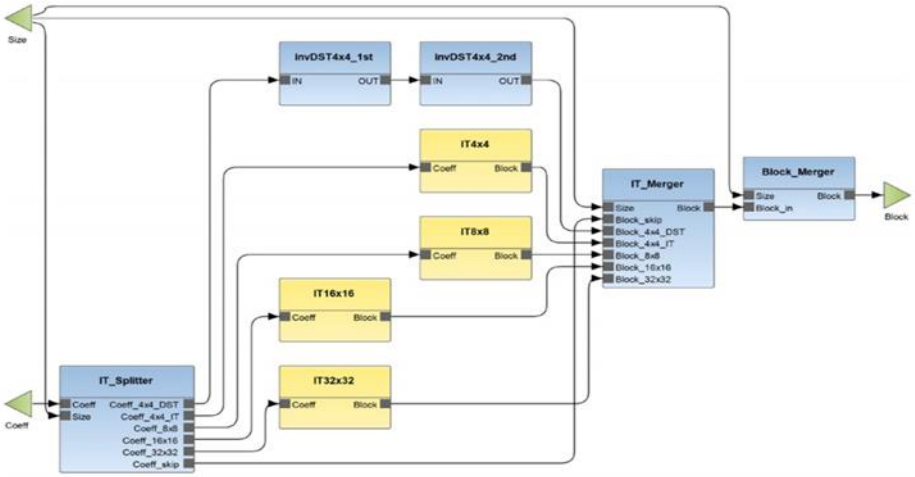

Figure 2. Block diagram of the original fully parallel xIT in the ORCC design environment

A full merging by combining IT4x4 with IT8x8 block into one block and by merging IT16x16 with IT32x32 block into one block, so that the IT could be conducted in series for 4x4 coefficient and 8x8 coefficient, concurrently for 16x16 coefficient and 32x32 coefficient. The finite state machine is implemented into the design with appropriate scheduling to produce the correct results as well as to the original design. The top level of this parallel architecture is shown in Figure 3.

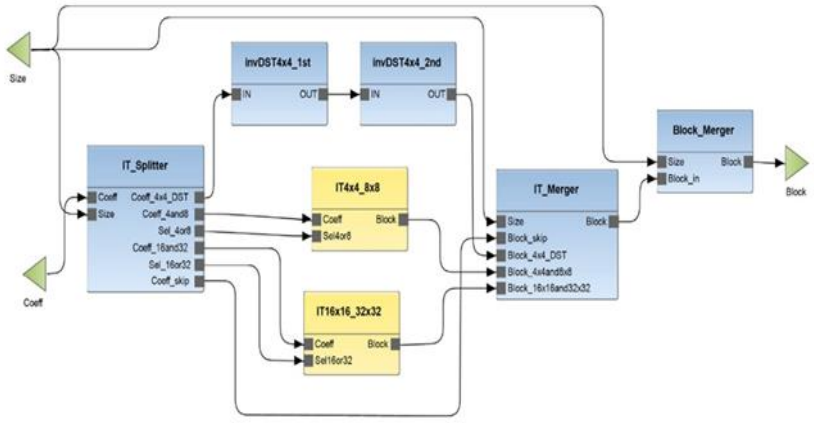

Figure 3. Block diagram of the merged dataflow actors for xIT in the ORCC design environment

The purpose of IT Splitter is to distribute the computation of inverse transform according to the configuration taken from the input of Size. The data of Size is grouped and attributed into different tags to figure out which transform should be conducted to the coefficient, and then further split the coefficient to various operational block from the output of the IT Splitter. The Size is split into different conditional tags in order to accomplish this functionality.

The other designs have been created based on the partial implementation of Full Merging design, such as Parallel xIT Block with Merging of IT4x4 and IT8x8, Parallel xIT Block with Merging of IT16x16 and IT32x32 and the regenerated serial design. The Parallel xIT Block with Merging of IT4x4 and IT8x8 introduced the merging of IT4x4 and IT8x8 blocks in xIT while the Parallel xIT Block with Merging of IT16x16 and IT32x32 introduced the merging of IT16x16 and IT32x32 blocks in xIT.

In terms of functionality, serial xIT is the same as the parallel xIT, but the main difference is the finite state machine scheduling and control. The modelling of serial xIT is performed not only by merging all the CAL codes into a same block CAL codes, but also all the inputs and outputs of the process of different blocks are assigned into different variables.

4. Results and Discussion

Figure 4 shows the graph of time taken to obtain the last results for all xIT by giving the same set of inputs, i.e. the design latency. The time taken to obtain the last results indicated that the time taken to finish up the operation with same inputs. For the original parallel xIT, the time taken for the particular process is 8118ns. On the other hand, the time taken to receive the last results for the xIT with merging IT4x4 and IT16x16, as well as the xIT with merging IT16x16 and IT32x32 are 8264ns and 8242ns respectively. For the xIT with full merging, the time taken to get the last results is 8328ns. Lastly, as expected the serial design shows the most inferior results as expected with 9600ns.

Figure 4. Time taken to obtain all results for all xIT designs

Figure 5 shows the graph of maximum operating frequency for all types of xIT designs. The trend here is quite similar to latency, where the parallel designs show roughly similar values. The serial xIT as expected, shows the most inferior frequency of only 44.78 MHz. This is due to the longer path in the serial design compared to the parallel design.

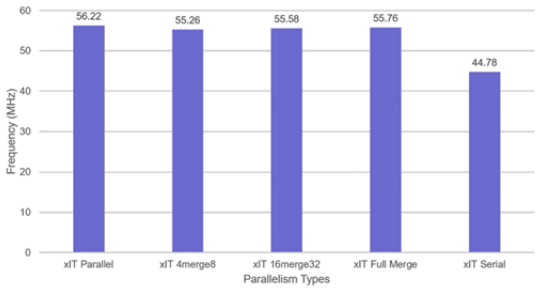

Figure 5. Maximum operating frequency for all xIT designs

Figure 6 shows the graphs of FPGA resource utilization of all xIT designs, with (a) the Lookup Table (LUT), (b) LUTRAM (LUTRAM), (c) the Flip-flop, (d) block RAM, and (e) DSP48. It can be seen that the serial design uses the least resource and the full parallel uses the most as expected.

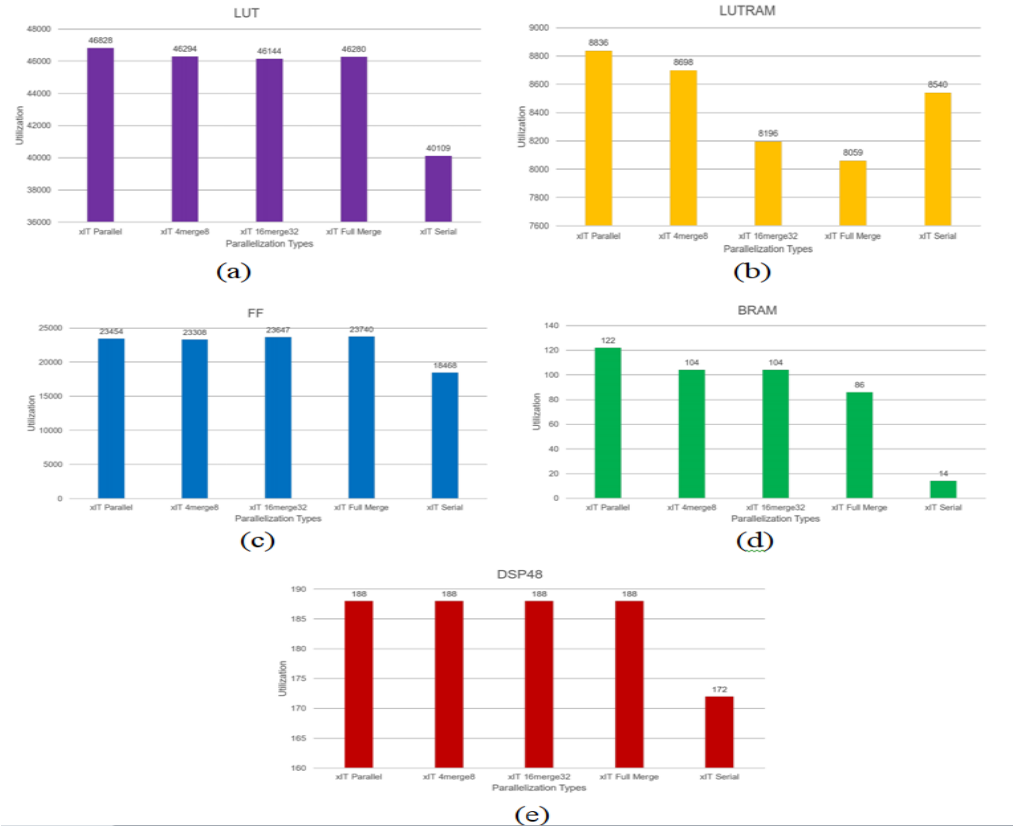

Figure 6. FPGA resource utilization graphs.

Based on the performance and resource results, it can be concluded that the xIT with merging IT4x4 and IT8x8 together, and merging IT16x16 and IT32x32 together leads to the most efficient architecture with the best trade-off. Full parallel consumes the most resource, but with a only a slight increase in performance. Serial implementation on the other hand uses the least resource but with significantly low performance.

5. Conclusion

This paper aims to explore the hardware design space of the HEVC integer transform, and to provide an insight on how the complex transform unit can be efficiently implemented in parallel. Based on the results, the fully parallel design shows the best performance, but not much from an intermediate parallel design with quite significantly less resource. On the other hand, the full serial design uses the least resource, but gives poor performance. The design with the best trade-off is found by merging the IT4x4 and IT8x8 together, and also merging the IT16x16 and IT32x32 together. Future work involves improving further the performance of the transform unit by implementing a multiplierless architecture for the matrix operations.

Acknowledgement

The authors would like to thank the Malaysia Ministry of Education for providing the funds for the work in this paper (Vote no. 4F659).

References

[1] H Sun, D Zhou, S Kimura, S Goto. *An area-efficient 4/8/16/32-point inverse dct architecture for uhdtv hevc decoder*. Visual Communications and Image Processing Confer- ence 2014 IEEE. IEEE, 2014; 1–8.
[2] E Kalali, I Hamzaoglu. *Fpga implementations of hevc inverse dct using high-level synthesis*. Design and Architectures for Signal and Image Processing (DASIP), 2015 Confer- ence on. IEEE, 2015;1–6.
[3] M Viitanen, J Vanne, TD Hmlinen, M Gabbouj, J Lainema. *Complexity analysis of next-generation hevc decoder*. Circuits and Systems (ISCAS), IEEE International Symposium on. IEEE, 2012; 882–885.
[4] V Sze, M Budagavi, G Sullivan, *High Efficiency Video Coding*. Springer. 2014.
[5] J Eker, J Janneck, CAL Language Report: Specification of the CAL Actor Language. University of California-Berkeley. 2003.
[6] E Bezati, S Casale-Brunet, M Mattavelli, J Janneck. Synthesis and optimization of high-level stream *programs*. The 2013 Electronic System Level Synthesis Conference. 2013: 1–6.
[7] M. Narroschke. *Coding efficiency of the dct and dst in hybrid video coding.* Selected Topics in Signal Processing, IEEE Journal. 2013; 7(6): 1062–1071.
[8] IK Kim, J Min, T Lee, WJ Han, J Park. Block partitioning structure in the hevc standard. *IEEE Trans. Cir. and Sys. for Video Technol.* 2012; 22(12): 1697–1706.
[9] PK Meher, SY Park, BK Mohanty, KS Lim, C Yeo. Efficient integer dct architectures for hevc. *IEEE Trans. Cir. and Sys. for Video Technol.* 2014; 24(1): 168–178.
[10] CY Chen, SY Chien, YW Huang, TC Chen, TC Wang, LG Chen. Analysis and architecture design of variable block-size motion estimation for h.264/avc. *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2006; 53(3): 578–593.
[11] TC Chen, YH Chen, SF Tsai, SY Chien, LG Chen. Fast algorithm and architecture design of low-power integer motion estimation for h.264/avc. *IEEE Transactions on Circuits and Systems for Video Technology*. 2007; 17(5): 568–577.
[12] JR Ohm, GJ Sullivan, H Schwarz, TK Tan, T Wiegand. Comparison of the coding efficiency of video coding standards. *IEEE Transactions on Circuits and Systems for Video Technology*. 2012; 22(12): 1669–1684.
[13] M Wipliez, G Roquier, J Nezan. Software Code Generation for the RVC-CAL Lan- guage. *Journal of Signal Processing Systems*. 2009; 1–11.
[14] H Amer, AAHA Rahman, I Amer, C Lucarz, M Mattavelli. *Methodology and technique to improve throughput of fpga-based cal dataflow programs: Case study of the rvc mpeg-4 sp intra decoder. IEE*E Workshop on Signal Processing Systems (SiPS). 2011: 186–191.
[15] AH Ab Rahman, S Casale-Brunet, C Alberti, M Mattavelli. *Design Space Exploration and Refactoring Techniques for Dataflow Programs: MPEG-4 AVC/H.264 Decoder Implementation Case Study*. Design and Architectures for Signal and Image Processing (DASIP), Conference on. 2013: 1–8.