Configurable, Low-power Design for Inverse Integer Transform in H.264/AVC *

Muhammad Nadeem
Computer Engineering Lab
Delft University of Technology
The Netherlands
M.Nadeem@tudelft.nl

Stephan Wong
Computer Engineering Lab
Delft University of Technology
The Netherlands
J.S.S.M.Wong@tudelft.nl

Georgi Kuzmanov
Computer Engineering Lab
Delft University of Technology
The Netherlands
G.K.Kuzmanov@tudelft.nl

ABSTRACT

The inverse integer transform is one of the compute-intensive processing units in an H.264/AVC video decoder. In this paper, we propose a configurable, low-power design for the inverse integer transform in an H.264/AVC decoder for video processing applications running on battery-powered electronic devices such as mobile phone. The proposed design is based on a data-driven computation algorithm for the inverse integer transform. It efficiently exploits the zero-valued coefficients in the input blocks to reduce dynamic power consumption. The area-requirement for the hardware implementation of the proposed design is reduced by designing configurable processing units to share the same hardware resources on the device. The proposed design is described in VHDL and is synthesized under 0.18μm CMOS standard cell technology. The experimental results show that the proposed design consumes significantly less dynamic power (up to 80% reduction) when compared with existing conventional design for the inverse integer transform, with a small area-overhead (approximately 2K gates).

Keywords

H.264/AVC, inverse integer transform, configurable, low-power and area-efficient design.

1. INTRODUCTION

The Advanced Video Coding standard H.264/AVC, also known as MPEG-4 part 10, is jointly developed by ITU-T VCEG and ISO/IEC MPEG [1]. Like other previous video coding standards, the H.264/AVC utilizes video coding algorithms based on block-based motion compensation and transform-based spatial coding framework. However, the H.264/AVC video coding standard outperforms previous video coding standards (e.g., H.263, MPEG-2, MPEG-4) in terms of bit-rate reduction. It provides similar subjective video quality as that of MPEG-2 with at least 50% more reduction in bit-rate [2], [3] and up to 30% better compression when compared with the H.263+ and the MPEG-4 Advanced Simple Profile (ASP) [4]. The improved coding efficiency achieved in H.264/AVC is not a result of any single feature in the new standard; rather it is a combined result of a number of advanced coding tools at the expense of increased complexity. Among many other innovations, the H.264/AVC utilizes a DCT-based integer transform instead of the popular 8×8 DCT used in previous video coding standards. This new DCT-based 4×4 integer transform can be computed exactly in integer arithmetic to avoid the inverse transform mismatch problems [5].

The H.264/AVC video coding standard has already been adopted for a wide range of applications, from low bit-rate mobile video to high-definition TV. The applications running on battery-powered electronic devices, such as video play-back on mobile phones, still require significant efforts in lowering the dynamic power consumption to support real-time video decoding.

The H.264/AVC baseline decoder is approximately 2.5 times more time complex than the H.263 baseline decoder [6]. According to the analysis of the run-time profiling data of H.264/AVC baseline decoder sub-functions, the inverse integer transform is among the top 3 compute-intensive processing functions [6] and is on the critical path of the decoder. The H.264/AVC video decoder uses multiple transforms namely 4×4 inverse integer transform (for luma and chroma blocks), 4×4 inverse Hadamard transform (for luma DC blocks) and 2×2 inverse Hadamard transform (for chroma DC blocks). The 4×4 inverse integer transform is, however, the most frequently executed sub-function among these various transforms [6].

In recent years, many researchers proposed a number of optimized algorithms to compute the transforms used in H.264/AVC. The major focus of the research has been to develop fast algorithms for the transform unit. Liu, et al., in [7], used a parallel register array to realize the transpose operation for the 2-D 4×4 forward integer transform. In [8], Cheng, et al., proposed a high-throughput 4×4 integer

transform architecture with no transpose memory requirement. Similarly in [9] - [11], a number of fast algorithms for the integer transform were proposed. Since H.264/AVC video standard make use of multiple transforms, therefore, recently the research focus has been shifted to the design of a unified transform block to support multiple transforms in H.264 [12]- [16]. The motivation for such efforts is to reduce the chip-area requirement for the hardware implementation of these transforms units by sharing area between them.

Most of these algorithms for the inverse integer transform assume that all of the 4×4 input block data items are non-zero. Consequently, these algorithms perform a constant number of operations per input block to compute the inverse integer transform for the given block of data. Therefore, the appropriate actions for All Zero blocks (AZB) or DC-only blocks are to be taken at the higher layer in the decoder implementation.

In digital video coding, however, it is very common that a substantial number of spatial high frequency transformed coefficients of residual block are quantized to zero. A typical transformed and quantized residual block consists of few non-zero coefficients in the low-frequency region and mostly zero-valued coefficients in the high-frequency region. For the inverse integer transform, while decoding the incoming video bit-stream, it is already known for the given input block of data which transformed coefficients are non-zero and shall contribute to the pixel values for the reconstructed block. Therefore, a considerable amount of computations can be saved, leading to a significant reduction in the dynamic power consumption by exploiting this information while developing an algorithm for the inverse integer transform.

In this paper, we focus on a low-power realization of the inverse integer transform and provide the following specific contributions:

- Derivation of data-driven algorithm for inverse integer transform with variable number of addition operations. The new algorithm is derived from the fast 1-D inverse integer transform.
- Architecture for the proposed algorithm using a hardware sharing approach to reduce area requirement for its hardware implementation.

The organization of the paper is as follows. The proposed algorithm with its justification is presented in Section 2. Section 3 describes the architecture for the proposed algorithm, whereas Section 4 provides the implementation results. Finally, Section 5 concludes this paper.

2. PROPOSED DATA-DRIVEN ALGORITHM FOR INVERSE INTEGER TRANSFORM

In this section, we first propose to categorize the 4×4 input data block into a set of 4 different types along with its justification. Later a data-driven algorithm for the inverse integer transform, using the proposed 4 different types of input block, is described in detail.

A typical transformed block after the quantization process consists of only a few non-zero coefficients. The actual distribution of coefficients values in a block after the quantization process, among various factors like video signal itself and encoding type, etc., heavily depends on the *Quantization parameter* (Qp) used for encoding of this block. Though the coarser quantization parameter provides better compression ratio because of less number non-zero blocks after quantization process, the quality of compressed video suffers in terms of PSNR value. This is the reason that the primary responsibility of the rate control in an encoder is to allocate bits in a video frame such that the generated compressed video bitstream meets the available bandwidth constraint while providing the best possible video quality. Consequently, as the Qp varies from a finer value to a coarser one, the number of non-zero blocks after the quantization process increases. This is also depicted in Figure 1 for a number of video test sequences (with 352×288 CIF resolution) encoded using a set of fixed Qp values.

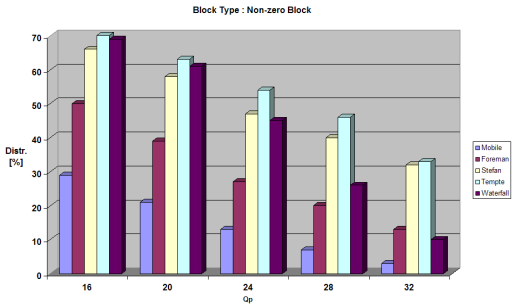

Figure 1: Non-zero input blocks distribution.

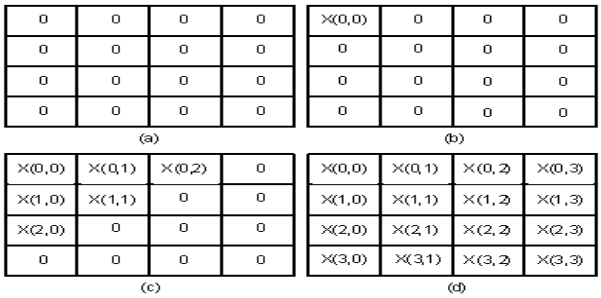

Figure 2: Proposed block types (a) All-zeros block, (b) DC-only block , (c) ULT block, and (d) Normal block.

The typical distribution of non-zero values within a quantized block is such that it contains few non-zero coefficients in the low-frequency region and mostly zero-valued coefficients in the high-frequency region. In our work, we propose to categorize the input blocks into a set of 4 different types namely All-Zero blocks (AZB), DC-only blocks, Upper Left Triangular (ULT) blocks and Normal blocks. An example of such blocks is depicted in Figure 2. The All-Zero block (AZB) type can be easily justified from the non-zero block distribution in Figure 1 as for the test video sequences, the profile data suggests that the percentage of All-zero blocks varies from about 30% to 75% for typical Qp values from 16 to 32. Similarly, the percentage distribution of DC-only blocks, Upper Left Triangular blocks and Normal blocks, within the non-zero blocks is depicted in Figures 3, 4, and 5, respectively, for the same video test sequences encoded using the same set of Qp values.

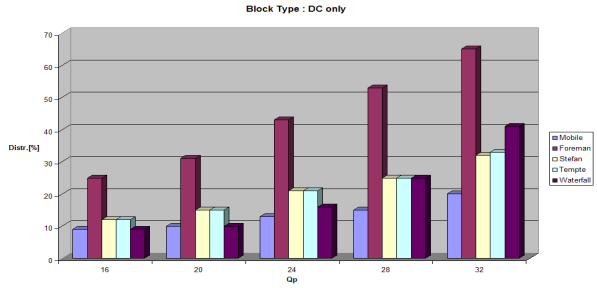

Figure 3: Percentage distribution of DC-only block type.

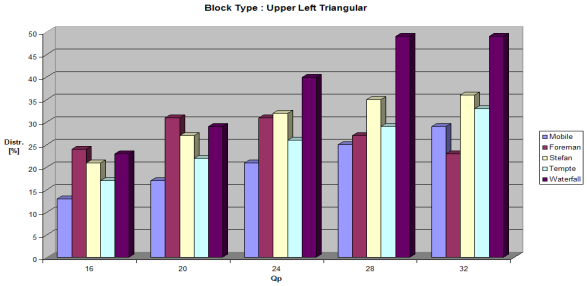

Figure 4: Percentage distribution of Upper Left Triangular block type.

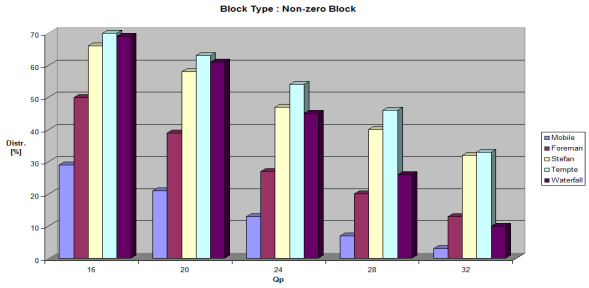

Figure 5: Percentage distribution of Normal block type.

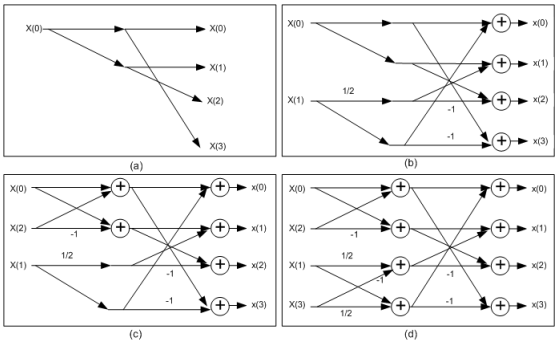

Figure 6: Data flow diagram for (a) M1, (b) M2, (c) M3, and (d) M4 cases.

The data flow diagram for the 1-D inverse integer transform with four input coefficients is depicted in Figure 6(d). One can derive the data flows for the 1-D inverse integer transform for the cases when only one, two, or three coefficients are non-zero in any given input data vector by removing the corresponding data flow part for the zero-valued coefficients in Figure 6(d). The resulting data flow diagram for the 1-D inverse integer transform with only one, two, or three non-zero coefficients is depicted in Figures 6(a), (b), and (c), respectively. The number of addition operations for the 1-D inverse integer transform, therefore, is reduced by 100%, 50%, and 25%, respectively, for the three specified cases.

We further denote the signal flow for the 1-D inverse integer transform with one, two, three, or four non-zero coefficients in Figures 6(a)-(d) by processing units M1, M2, M3, and M4 respectively. Based on these processing units, the functional block diagram for the 2-D inverse integer transform for the DC-only, the Upper Left Triangular block, and the Normal block is depicted in Figures 7(a)-(c), respectively. The "Cx", in Figure 7, represents the column vector in the input block "X", whereas "Rx" denotes the corresponding 2-D inverse transformed row-vector for the given input block "X". The "Pr" block represents the row-column permutations to realize the transpose between two 1-D inverse integer transform blocks. The number of additions for the Normal block are 64 for the 2-D inverse integer transform (Figure 7c and Figure 6d). This number is reduced to 34 (Figure 7b and Figures 6a- 6c) and 0 (Figure 7a and Figure 6a) for the Upper Left Triangular block and the DC-only block, respectively.

The straight-forward hardware realization of the data-driven algorithm for the inverse integer transform is an implementation using independent processing units for the Normal blocks, the Upper Left Triangular blocks and the DC-only blocks. For All-zero, DC-only and Upper Left Triangular blocks, the reduction in adders implies less signal activity in the circuit and, therefore, lower dynamic power consumption. This is, however, achieved at the cost of additional chip-area (with $34 + 0 = 34$ additional adder units and input registers) for independent implementation of the Upper Left Triangular block and the DC-only block processing units along with control circuitry. The number of additional adders (34) for the Upper Left Triangular block processing unit can be identified using Figures 6 and 7. The area-overhead for the inverse integer transform unit can be reduced by sharing the hardware resources between the independent processing units by designing configurable units.

3. CONFIGURABLE DESIGN FOR THE PROPOSED DATA-DRIVEN INVERSE INTEGER TRANSFORM ALGORITHM

In this section, the design for the data-driven inverse integer transform unit along with basic configurable units is described.

The top level organization for the data-driven inverse integer transform unit is depicted in Figure 8. The unit takes the 4x4 input block "X", along with the Block Type information determined during the entropy decoding process. If the input block is an AZB or a DC-only block, the control unit routes it to the top level processing block (Figure 8) which is essentially a bypass stage as no further processing is required for such blocks. The value for all the data items in the output block is either zero or the DC value for the AZB or the DC-only block types, respectively.

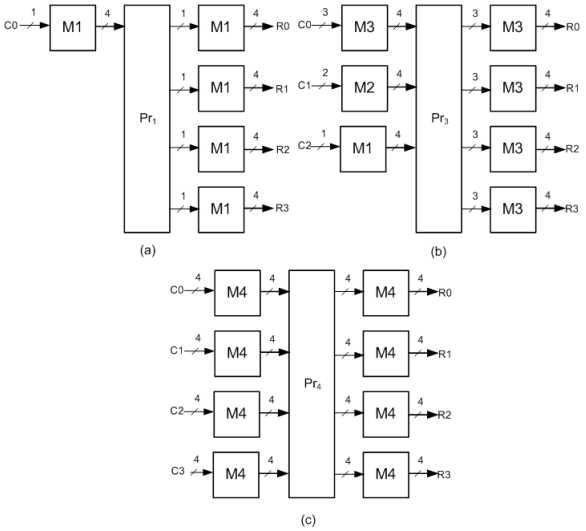

Figure 7: Functional block diagrams (a) DC-only blocks, (b) ULT blocks, (c) Normal blocks.

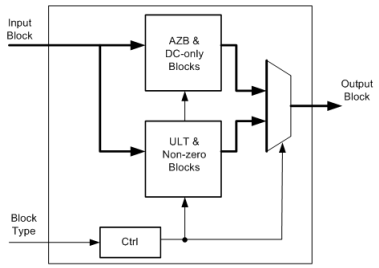

Figure 8: Top level organization of inverse integer transform unit.

The Upper Left Triangular and Non-zero blocks are routed to the lower processing block in Figure 8. The internal organization of this block is depicted in Figure 9. Since the processing block M1- M3 are derived from M4 (Figure 6) and have the similar structure, therefore, we can design a configurable processing units (CM14, CM24, and CM34) with overlapped functionality to reduce the hardware resource requirement for its implementation. The configurable processing units (CM14, CM24, and CM34) as the name suggest can be configured to provide processing for either (M1, M4), (M2, M4), or (M3, M4) using the appropriate control signal. The internal architecture for these configurable units is depicted in Figures 10(a) - 10(c). Therefore, no additional (34) adders are required anymore because of configurable processing units. Furthermore, the input registers (in CM24) are also shared among processing for data vectors with 2 and 4 non-zero coefficients.

4. DESIGN EVALUATION

The design proposed in this paper for a data-driven computation of the low-power inverse integer transform was described in VHDL and synthesized by Synopsys Design Compiler (v2002, rev. 05) for a maximum operating frequency of 166 MHz with 0.18μm CMOS standard cell technology (v1.5). The implementation was verified by simulation results generated using ModelSim 6.5 and comparing with that of the JM13.2 reference software [17].

Furthermore, VCD data was generated to record signal activity using ModelSim tool for a number of video test sequences. Subsequently, this VCD data was used to estimate the dynamic power consumption. Since the dynamic power consumption for the inverse integer transform unit using video test sequence data is not provided in the literature, we implemented a parallel 2-D inverse integer transform based on architecture proposed in [11] as a reference for the same technology and the maximum operating frequency. The dynamic power consumption was estimated for a number of video test sequences with fine (Qp = 16) and coarse (Qp = 32) quantization parameter values and is illustrated in Figures 11- 14. Table I provides the area requirement for the proposed design in terms of equivalent gate count.

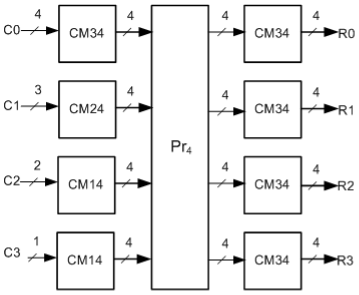

Figure 9: Functional block diagram: Configurable inverse integer transform unit.

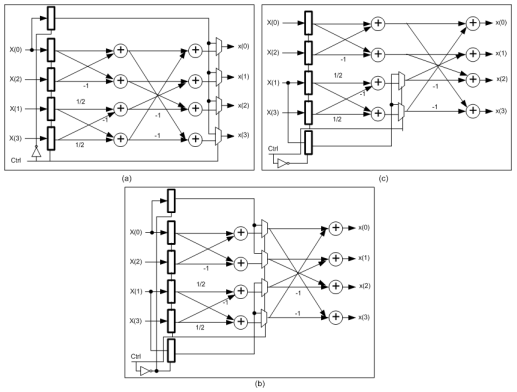

Figure 10: Data flow diagram (a) CM14, (b) CM24, and (c) CM34.

Table 1: Comparison table.

	Ref	Proposed
Technology [μm]	0.18	0.18
Frequency[MHz]	166	166
Area [K gates]	7.5	9.6

The test results suggest that dynamic power consumption is greatly affected by the Qp value chosen and a significant reduction (up to 80%) can be achieved by using a data-driven computation algorithm for the inverse integer transform. The number of operations to compute the inverse integer transform is variable and depends on the input block type in contrast to the conventional inverse integer transform algorithm with constant number of operations for all types of input blocks. This result in a significantly reduced signal activity in the hardware circuit and, therefore, lower dynamic power consumption. The area overhead for such an

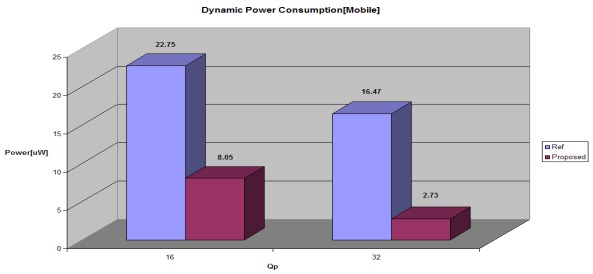

Figure 11: Dynamic power consumption comparison for "Mobile-CIF" video sequence.

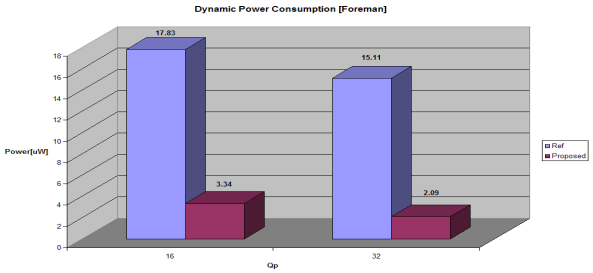

Figure 12: Dynamic power consumption comparison for "Foreman-CIF" video sequence.

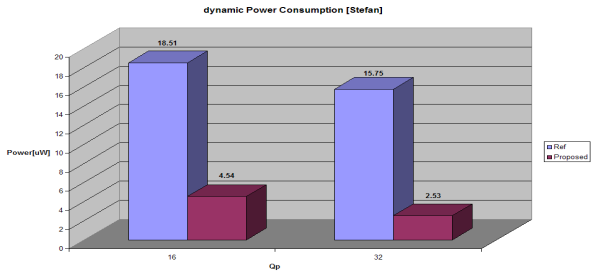

Figure 13: Dynamic power consumption comparison for "Stefan-CIF" video sequence.

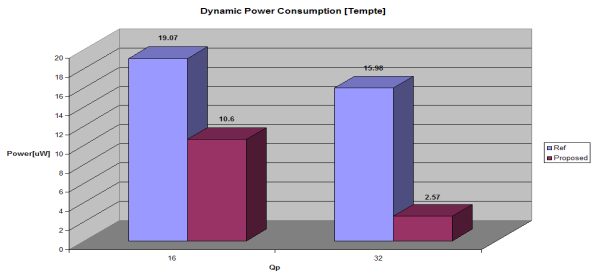

Figure 14: Dynamic power consumption comparison for "Tempte-CIF" video sequence.

algorithm was reduced by designing a configurable processing unit to share the hardware resources.

5. CONCLUSION

This paper presented a configurable, low-power design for the inverse integer transform in H.264/AVC. The dynamic power consumption is drastically reduced by employing an input block-type aware algorithm with variable number of operations for the computation of the inverse integer transform. This algorithm takes advantage of significant number of zero-valued transformed and quantized coefficients in a typical input block. Additionally, the area overhead was reduced by designing basic configurable processing blocks in order to share the hardware resources (adders) for different input block types. The experimental results show that the proposed design consumes significantly less dynamic power (up to 80% reduction) when compared with existing conventional design for the inverse integer transform, with a small area overhead (approximately 2K gates).

6. REFERENCES

[1] ITU-T Rec. H.264 and ISO/IEC 14496-10:2005 (E) (MPEG-4 AVC). "Advanced Video Coding for Generic Audiovisual Services", 2005.

[2] ISO/IEC JTC1/SC29/WG11. "Report of the Formal Verification Tests on H.264/AVC". In *doc. N6231, Waikoloa, USA*, 2003.

[3] G. Sullivan, P. Topiwala, and A. Luthra. "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions". In *SPIE Conference On Applications of Digital Image Processing*, vol. 558, pp. 454–474, 2004.

[4] J. Ostermann, J. Bormans, P. List, D. Maroe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi. "Video Coding with H.264/AVC: Tools, Performance and Complexity". In *IEEE Circuit and Systems Magazine*, vol. 4, pp. 7–28, 2004.

[5] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky. "Low-complexity Transform and Quantization in H.264/AVC". In *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 598–603, 2003.

[6] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro. "H.264/AVC Baseline Profile Decoder Complexity Analysis". In *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 704–716, 2003.

[7] L. Z. Liu, Q. Lin, M. T. Rong, and J. Li. "A 2-D Forward/Inverse Integer Transform Processor for H.264 based on Highly-Parallel Architecture". In *IEEE International Workshop on System-on-Chip for Real-Time Applications*, pp. 158–161, 2004.

[8] Z. Y. Cheng, C. Chen, B. D. Liu, and J. F. Yang. "High Throughput 2-D Transform Architectures for H.264 Advanced Video Coders". In *Proceeding IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 1141–1144, 2004.

[9] T. C. Wang, Y. W. Huang, H. C. Fang, and L. G. Chen. "Parallel 4×4 2D Transform and Inverse Transform Architecture for MPEG-4 AVC/H.264". In *Proceeding IEEE International Symposium on Circuits and Systems*, pp. 800–803, 2003.

[10] K. H. Chen, J. I. Guo, and J. S. Wang. "A High-Performance Direct 2-D Transform IP Design for MPEG-4 AVC/H.264". In *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 16, pp. 472–483, 2006.

[11] R. C. Kordasiewics, and S. Shirani. "ASIC and FPGA Implementations of H.264 DCT and Quantization Blocks". In *IEEE International Conference on Image Processing* (ICIP), vol. 3, pp. 1020–1023, 2005.

[12] C. P. Fan. "Cost-effective Hardware Sharing Architectures of Fast 8×8 and 4×4 Integer Transforms for H.264/AVC". In *IEEE Asia Pacific Conference on Circuits and Systems*, pp. 776–779, 2006.

[13] W. Hwangbo, and C. M. Kyung. "A Multitransform Architecture for H.264/AVC High-Profile Coders". In *IEEE Transactions on Multimedia*, vol. 12, no. 3, pp. 157–167, 2010.

[14] C. Y. Huang, L. F. Chen, and Y. K. Lai. "A High-speed 2-D Transform Architecture with Unique Kernel for Multi-standard Video Applications". In *IEEE International Symposium on Circuits and Systems*(ISCAS), pp. 21–24, 2008.

[15] Y. Li, Y. He, and S. Mei. "A Highly Parallel Joint VLSI Architecture for Transforms in H.264/AVC". In *Journal of Signal Processing Systems*, vol. 50, no. 1, pp. 19–32, 2008.

[16] H. Y. Lin, Y. C. Chao, C. H. Chen, B. D. Liu, and J. F. Yang. "Combined 2-D transform and Quantization Architectures for H.264 Video Coders". In *IEEE International Symposium on Circuits and Systems*(ISCAS), pp. 1802–1805, 2005.

[17] Reference Software for H.264 codec JM 13.2. http://iphome.hhi.de/suehring/tml/index.htm.