SIMD Acceleration for HEVC Encoding on DSP

Yongfei Zhang[1*], Rui Fan[2], Chao Zhang[1], Gang Wang[1], and Zhe Li[3]
[1] Beijing Key Lab of Digital Media, School of Computer Science and Engineering, Beihang University, Beijing, China, 100191
[2] China Academy of Electronic and Information Technology, Beijing, China, 100041
[3] Shandong Province Key Laboratory of Wisdom Mine Information Technology, Shandong University of Science and Technology, Qingdao China, 266590
*Corresponding Author: Yongfei Zhang (E-mail: yfzhang@buaa.edu.cn Tel: +86-1082314108)

***Abstract*— As the new generation video coding standard, High Efficient Video Coding (HEVC) significantly improves the video compression efficiency, which is however at the cost of a far more computational payload than the capacity of real-time video applications and general purpose processors. In this paper, we focus on the SIMD-based fast implementation of the HEVC encoder over modern TI Digital Signal Processors (DSPs). We first test the DSP-based HEVC encoder and indentify the most time-consuming encoding modules. Then SIMD instructions are exploited to improve the parallel computing capacity of these modules and thus speed up the encoder. The experimental results show that the proposed implementations can significantly improve the encoding speed of the DSP-based HEVC encoder, with a speedup ratio of 8.38-87.32 over the original C-based encoder and 1.59-6.56 over o3-optimization enabled encoder.**

***Index Terms*— HEVC, Encoder, DSP, SIMD**

I. INTRODUCTION

High Efficiency Video Coding (HEVC)[1], the latest video compression standard developed by the joint collaborative team on video coding (JCTVC), can significantly improve the coding performance compared its predecessor H.264/AVC[2], which is however achieved at a much improved computational cost of up to 2-10 times higher computational complexity, which makes it quite difficult to apply in real-time video applications[3].

Considering the high coding efficiency and pervasive applications of HEVC, low-complexity thus low-power-consumption HEVC encoder/decoder is urgently needed [3-6]. Besides plenty of algorithm-level optimizations for the encoding modules have been proposed to speed up either the encoder or the decoder [4-6], much attention has also been paid on the code/instruction level optimization to further and more significantly reduce the computational complexity[7-12].

Since HEVC decoder is much less computational intensive than the encoder, there are many literatures reporting the decoder implementations for HEVC, either on general purpose processor CPU/GPUs [7-10], FPGAs[11] or digital signal processors[12-13]. However, few research has been conducted on the code/instruction level optimization implementation for the more important and time-intensive HEVC encoders [14-17], most of which are on CPUs/GPUs.

This work was partially supported by the National Key R&D Program of China (Grant No.2016YFC0801001), the NSFC Key Project (No. 61632001) and the National Natural Science Foundation of China (No 61502278, 61772054). This paper is partially done when Rui Fan and Zhe Li were with Beijing Key Lab of Digital Media, School of Computer Science and Engineering, Beihang University, Beijing, China, 100191.

Programmable processors such as multi-core digital signal processors are especially good at computational intensive tasks with very low power consumption, which make it very competitive and promising solution to help reduce the high computational burden and put it into real-time video applications. However, up to the best of our knowledge, [17] is the only work which addressed the encoder implementation on DSPs, which however only tackles the implementation of most simple SAD and SSE on DSPs and leave the most time-consuming encoding modules, such as inter and intra prediction, DCT/IDCT(Inverse/ Discrete Cosine Transform) and Sample Adaptive Offset(SAO), untouched.

In this paper, to address the problem of optimized implementation of HEVC video encoder on the powerful main-stream TI TMS320C6678 DSPs[18], various Single-instruction-multiple-data (SIMD) optimizations are explored to optimize the most time-consuming encoding modules of the DSP-based HEVC encoder. Up to the best of our knowledge, this is the first of this kind to comprehensively implement the encoder using SIMD. With our careful design, the encoding speed of the DSP-based HEVC encoder is significantly improved, which will help make possible to achieve real-time implementation of HEVC for practical video applications.

The rest of this paper is organized as follows. Section 2 analyses and identifies the most time-consuming encoding modules of the DSP-based HEVC encoder. After a brief introduction of the Single Instruction Multiple Data (SIMD) in the chosen DSP, the SIMD acceleration design and implementation for these modules are elaborated in Section 3. Experimental results are shown in Section 4. Finally, conclusions are drawn in Section 5.

II. COMPLEXITY ANALYSIS OF DSP-BASED HEVC ENCODER

In order to identify the most time-consuming modules in the DSP-based HEVC encoder, we analyze the execution time of our HM16.0 [19]-based embedded HEVC encoder on TI TMS320C6678 DSP. The first 100 frames of the video sequences in Class B (1080p) are encoded with a QP of 32 under Low Delay P main configuration. The rest of the configurations are kept unchanged [20]. The average execution time of the major modules of the DSP-based encoder is shown in Fig. 1. As can be seen, although it is slightly different from that in HM, the most time-consuming modules are Inter prediction (mainly half/quarter pixel

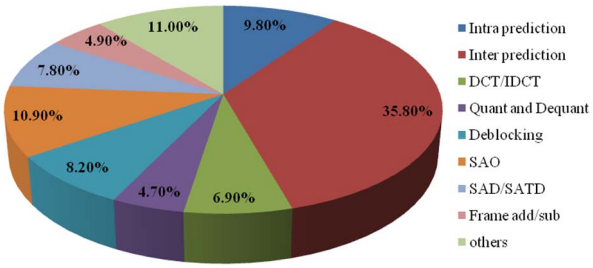

Fig. 1 Execution Time Analysis of DSP-based HEVC Encoder

Interpolation), Intra prediction, DCT/IDCT, Quantization/De-quantization, deblocking, SAO, SAD/SATD, Frame add/sub. As will be shown in Section III and IV, he computational performance of the first four modules can be effectively improved by using SIMD methods.

III. Proposed Scheme

To improve the video coding speed, several critical modules will be written in linear assembly language using the powerful SIMD instructions available in TI TMS320C6678 DSPs. The main idea is to improve the parallelism of data processing by using SIMD instructions.

In this section, we first give a brief introduction of the Single Instruction Multiple Data instruction sets available in the TI TMS320C6678 DSP. And we then elaborate the SIMD acceleration design and implementation for these time-consuming encoding modules identified in Section II.

A. C66x CorePac[21-22]

The TMS320C6678 is an eight-core, high-performance DSP with both fixed-point and floating-point precision capabilities. C66x CorePac is based on a Very Long Instruction Word (VLIW) architecture, which differs from Reduced Instruction Set Computing (RISC) or Complex Instruction Set Computing (CISC) architectures by having multiple execution units which can execute several instructions in parallel. The C66x CorePac has two identical data paths, A and B, each with four unique functional units (M, L, S, and D). The M unit performs multiplication operations, while the L and S units handle addition, subtraction, logical, branching, and bitwise operations. The D unit is responsible for load/store and address calculations. All the functional units provide vector-processing capabilities using the SIMD instruction set included in the C66x CorePac. The SIMD instructions can operate on up to 128-bit vectors providing data-level parallelism within each core. With L, M, and S units on the two data paths, each core can perform eight single-precision or two double-precision multiply-add operations in one cycle. The TMS320C6678 also provides thread-level parallelism by scheduling application on the eight available cores, which is has been implemented in our DSP-based HEVC encoder and thus out of the scope of this paper.

B. Inter Prediction

As is well known [1], inter prediction is the most powerful encoding techniques to remove the inter-frame redundancy. However, as shown in Fig. 1, inter prediction is responsible for about one thirds of the computational burden of the whole HEVC encoder. Furthermore, the major computation of inter prediction lies in the interpolation for quarter-pixel motion estimation (ME) and compensation (MC). Thus, we mainly focus on the SIMD implementation here.

In HEVC, since quarter-sample precision is used for ME/MC, 7-tap or 8-tap filters are used for interpolation of fractional-sample positions of the luminance component, (compared to six-tap filtering of half-sample positions followed by linear interpolation for quarter-sample positions in H.264/MPEG-4 AVC)[1], which make the implementation different and more complicated.

Taking the design for the luminance component as an example, the interpolation filter in HEVC is composed of two steps, namely horizontal filtering first and then vertical filtering. Please refer to Ref. [1] for more details of the interpolation as well as filter coefficients.

Since storage and access of the source data, i.e., the integer pixels for horizontal and vertical interpolation are quite different, the implementation of the horizontal and vertical interpolation will be elaborated separately as below.

Some of the notations used are defined as below. Let A-H denote the value of luminance component of the eight integer pixels used for interpolation while C0-C7 8-tap interpolation filter coefficients. Ai is the pixel in row i and column A. It is the intermediate result after horizontal interpolation filter, and Ii is 14 bit precision to ensure the accuracy. X is the final filtered pixel through vertical interpolation whose precision is the same to the raw data.

1) Horizontal Interpolation

The proposed implementation of luma horizontal interpolation filter is shown in Fig. 2. Firstly, eight reference frame pixels on the horizontal direction are loaded into a register pair which is composed of two adjacent 32-bit registers. Then the interpolation filter is divided into two ways, and each way completes four-pixel interpolation filters. Finally, the two-way sums results are added together, taking into consideration the interpolation offset and the 14-bit luma horizontal filter result is obtained. SIMD instructions involved as in Fig. 2 are specified as follows.

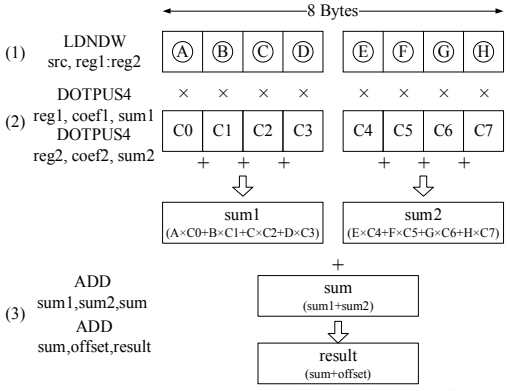

Fig. 2. Horizontal interpolation filter for inter prediction