Porting a H264/AVC Adaptive in Loop Deblocking Filter to a TI DM6437EVM DSP

Abdellah Skoudarli[1], Mokhtar Nibouche[2], and Amina Serir[1]

[1] USTHB-Faculty of Electronic and Informatic Laboratory of Image Processing and Radiation, BP 32 El Alia Bab Ezzouar Alger, Algeria
[2] Frenchay Campus, Coldharbour Lane, Bristol BS16 1QY, UWE, UK
askoudarli@usthb.dz, mokhtar.nibouche@uwe.ac.uk, aserir@hotmail.com

Abstract. Complementary units in the form encoders and decoders are generally involved in video compression standards. Both the encoder and the decoder integrate an adaptive deblocking filter, which is very beneficial in preserving and enhancing the video quality. Deblocking filters are extremely popular in improving the visual quality of decoded frames in the H.264/AVC video coding standard. The prime goal of the current paper is to efficiently implement a H.264/AVC adaptive deblocking filter using the Texas Instruments DM6437EVM DSP processor. The adopted approach requires an initial identification of the portions of the algorithm wherein parallel processing can be exploited. The functions are then re-written and the instructions rearranged using the features of the targeted hardware architecture. The adaptive deblocking algorithm was optimised and ported to a DM6437EVM DSP platform. A quick comparison shows that the optimised code is a 32 % better, in terms of speed, than the non-optimised code.

Keywords: H.264/AVC, Filtering, Adaptive Deblocking Filter, DM6437EVM DSP, C/C++ optimization.

1 Introduction

H.264/AVC is the latest video compression standard jointly developed by the ISO and ITU [1][2]. The standard achieves the best encoding performance in terms of video quality and compression ratio than its predecessor by adopting a number of new techniques including, variable block size based motion estimation in inter mode prediction, multiple directions of intra prediction, quarter-pel accuracy in motion estimation, multiple reference frames, weighted prediction, rate distortion estimation and highly adaptive in loop deblocking filter. Both the encoder and the decoder must apply the normative deblocking filter at block boundaries. The standard specifies that the filter should be applied within the motion compensation loop, and as such, the filter is often referred to as a "loop filter".

Deblocking filters are used to improve the visual quality of decoded frames in the H.264 video coding standard [1]. These filters attempt to remove the artifacts

A. Elmoataz et al. (Eds.): ICISP 2012, LNCS 7340, pp. 122–130, 2012.

produced by block-based operations, which consists of 4x4 DCT blocks and motion compensation prediction. Although these deblocking filters help tremendously in improving the subjective and objective quality of the output frames, they are generally computationally intensive. In fact, even after the tremendous efforts that have been made to optimise the speed of these filtering algorithms, unfortunately, they still easily account for one third of the computational complexity of a decoder [1]. This complexity is mainly due to the high adaptivity of the filter, which requires conditional processing on the block edge and sample levels. These are known to be very time consuming and present a real challenge for parallel processing in DSP hardware.

In embedded, real-time video applications, the implementation of the H.264/AVC requires high performance, low power consumption and low cost, as well as a level of flexibility, which can be very beneficial in relation to these requirements.

Complexity analysis shows that loop filtering uses 5% and 33% of the execution time of the encoder and of the decoder, respectively. Since the filtering process is normative, it can be accelerated by processor-dedicated parallel processing instructions. DSP processors, such as the TI DM6437EVM, are specialised platforms for fast execution of specific numerical operations like multiplications and additions and as such are excellent targets for implementing loop filtering algorithms.

The remainder of this paper is organised as follows: Section 2 presents an overview of the adaptive deblocking filter algorithm. Section 3 provides a brief description of the DM6437EVM DSP. Section 4 describes the optimisation approach and section 5 summarises the experimental results. Section 6 is dedicated to the conclusion.

2 Adaptive Deblocking Filter

2.1 Deblocking Filters

There are a number of deblocking algorithms that have been proposed for reducing the block artifacts in block DCT based image compression with minimal smoothing of true edges, as illustrated in Figure 1.b. Three of the most popular techniques include:

- Projection On Convex Sets (POCS)
- Weighted Sum of Symmetrically Aligned Pixels
- Adaptive Deblocking Filter.

The POCS based iterative algorithm [3] is implemented as a two stage process. The first stage involves the band limiting of the image by low pass filtering. Then, the image is transformed to obtain the transform coefficients, which are then subjected to quantisation constraints.

In the Weighted Sum of Symmetrically Aligned Pixels [4], the value of each pixel in the picture is recomputed with a weighted sum of itself and the other pixel values, which are symmetrically aligned with respect to block boundaries.

In case of the Adaptive Deblocking Filter algorithm [5], the deblocking process is separated into two stages. In the first stage, the edge is classified into different boundary strength with the pixels along the normal to an edge. In the second stage, different filtering scheme is applied according to the strengths obtained in stage one. The algorithm flow in each of these algorithms is highly iterative either at the pixel, block or edge level.

There are two main methods for implementing deblocking filters for video codecs. They can be implemented either as post filter or loop filter, with tradeoffs inherent in either implementation. The loop filter is normative in the H.264/AVC standard and provides better visual quality and rate distortion performance [5].

(a) (b)

Fig. 1. (a) Original Foreman image (b) Reconstructed Foreman image without filtering

2.2 ADF Algorithm

H264/AVC uses an adaptive deblocking filter that operates on horizontal and vertical block edges within the prediction loop in order to remove artefacts caused by both the 4x4-block based transform and the coarse quantization of the transform coefficients. The filtering is based on 4x4 block boundaries, in witch two pixels on either side of the boundary may be updated using different filter. The filter adjusts its filter strength adaptively according to the image local characteristics, leading to better image quality [5].

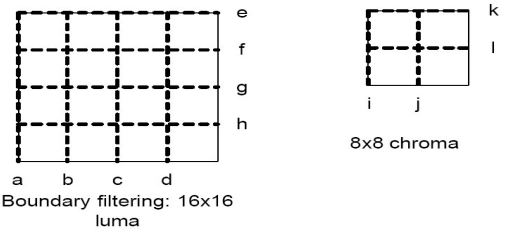

Fig. 2. Edges to be filtered in luma and chroma components

The deblocking filter is applied to both luma and chroma components separately. For each macroblock (MB), vertical edges are filtered from left to right; horizontal edges are processed from top to bottom, as illustrated in figure 2. The filtering is performed on MBs of a picture in a raster scan fashion. The filter should be applied to all 4x4 block edges of a picture, except the edges at the boundary frame, or any edges for which the filtering is disabled under certain conditions by a special flag, as described in the flowchart of figure 3.

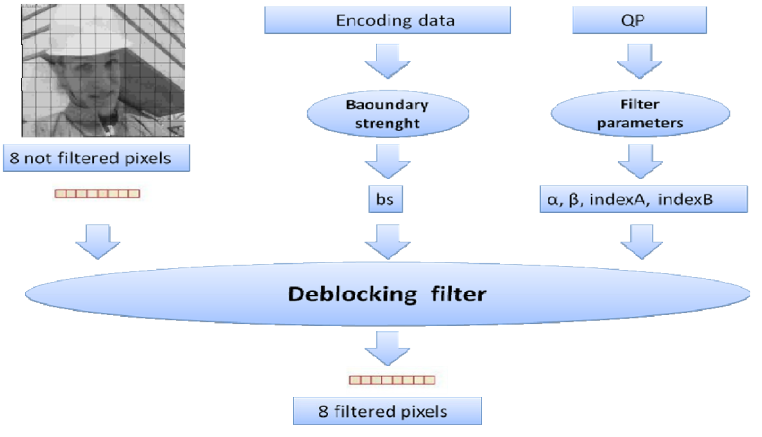

Fig. 3. Filtering Process

Depending on the coding type of the 4x4 blocks and their position within the array, a boundary strength (BS) is assigned to each edge [5]. This parameter determines the strength of filtering to be applied as shown in Table 1. ,

Table 1. Conditions for Determining a BS Value

BS	Rule
4	One of the blocks is intra and the edge is a macroblock a macroblock edge;
3	One of the blocks is intra;
2	One of the blocks has coded residuals;
1	Difference of block motion ≥ 1 luma sample distance;
1	Motion compensation from different reference frames;
0	Otherwise.

The filter is turned on or off, for each pixels across each line based on the values of p1, p0, q0 and q1 and two thresholds Alpha α(QP) and Beta β(QP), which have values depending on the quantisation parameter QP of the current frame.

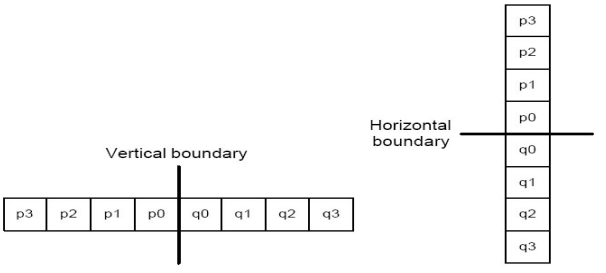

Fig. 4. Samples Across a 4x4 block Horizontal/Vertical Luma or Chroma edge

The filtering is applied to p0 and q0 only if these conditions are true:

bs =! 0
abs($p_0 - q_0$) < α(QP)
abs($p_1 - p_0$) < β(QP)
abs($q_1 - q_0$) < β(QP) with β(QP)< α(QP)

The filtering is extended to p1 and q1 respectively if the further conditions are also true:

$$\text{abs}(p_2 - p_0) < \beta(QP) \text{ or } \text{abs}(q_2 - q_0) < \beta(QP)$$

The length of the filtering is also determined by the sample values over the edge, which determines the "activity parameters". These parameters determine whether none, one or two pixels on either side of the edge are modified by the normative filter. Consequently, this analysis assesses the likelihood of an edge in the image being natural, or the result of a block based transform.

The equations calculating pixel values are defined in [2]. The equations can be classified into five categories, according to the BS values, as follows:

$$p2 + p1 + p0 \quad (1)$$

$$p2 + 2 \times p1 + 2 \times p0 \quad (2)$$

$$3 \times p3 + 3 \times p2 + p1 + p0 \quad (3)$$

$$2 \times p1 + p0 \quad (4)$$

$$(p0 + q0 + 1) >> 1 \quad (5)$$

The filter is "stronger" where there is likely to be significant blocking distortion (high values of BS=4)

2.3 Complexity

The filtering algorithm is very complex. This is due to the highly adaptive nature of the algorithm of the deblocking filter, as well as to the huge quantity of pixel data to be read from memory and processed [7][8][9].

The filtering process consists of these two principal tasks:

First: **Get Strength**, which involves a large number of conditional branches. Filtering decision can be made from multiple data in parallel, in a way that pixels can be packed to operate simultaneously.

Second: **Loop Filtering** with multi-tap filter applied to the edge pixels in the decoded frame. Some optimisation techniques can be adopted conveniently to get a significant reduction.

Before performing the optimisation, the complexity of the filtering algorithm is analysed. The complexity can be sumarised in the following four points:

The ADF is highly adaptive.

It is applied to each edge of all 4x4 luma and chroma blocks in a MB.

It can updates three pixels in each direction where the filtering takes place.

In order to be applied to an edge, the related pixels in the current and neighboring 4x4 blocks must be read from memory and processed.

3 Overview of DM6437EVM DSP

The DaVinci™ TMS320DM6437 Digital Video Development Platform (DVDP) is a high performance video DSP processor from Texas Instruments. It is based on the third generation high performance, advanced VelociTI, TI's very long instruction word (VLIW) [10]. With performance of 4800 million of instructions per second at the clock rate 600 MHz, the DM6437EVM core offers solution to high performance DSP programming challenges. The DM6437EVM has an application–specific hardware logic, on chip memory, and additional on chip peripherals.

The DM6437EVM core uses a two level cache-based architecture [11]. The level 1 program memory cache (L1P) consists of a 32 Ko memory space and the level 1 data (L1D) consists of 80 Ko memory space. The level 2 memory cache (L2) consists of a 128 Ko memory space that is shared between program and data space. L2 memory can be configured as mapped memory, cache or combined memory.

Existing C6x DSPs support various instructions to execute packed operations between two registers. These operations are very useful for video processing [12]. Figure 5 illustrates the TMS320 DM6437EVM block diagram.

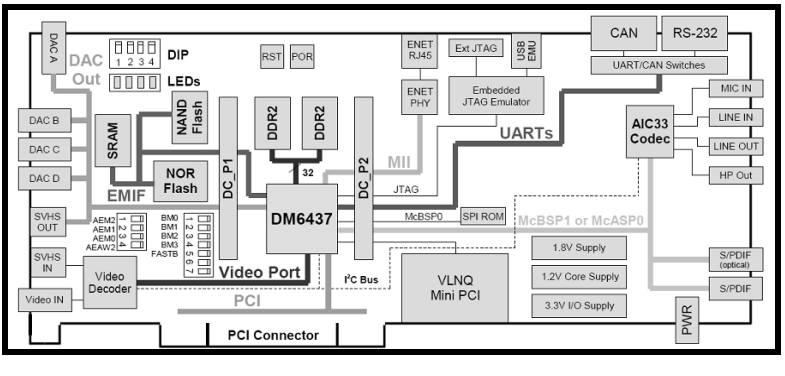

Fig. 5. TMS320 DM6437EVM block diagram

4 The Proposed Optimisation Approach

The proposed optimisation approach realises substantial gains from the performance of the C/C++ code by refining the code in terms of areas execution time, code size and memory access.

4.1 Using Intrinsics to Replace Complicated C Code

The C6000 compiler provides intrinsic, special functions that map high-level operations directly to the inline C64xx instructions to speed up the C codes [12]. All instructions that are note easily expressed in C codes are supported as intrinsics. For example, the intrinsic operator "_abs" calculate the saturated absolute value.

4.2 Using Word Access to Operate on 16-Bit Data Stored in the High and Low Parts of a 32-Bit Register

In order to maximize data throughput, it is often desirable to use a single load or store instruction to access multiple data values consecutively located in memory. For example, C6x have instructions with associated intrinsics, such as "_add2()", "_mpyhl()", "_mpylh(), etc that operate on the 16-bit data stored in the high and low parts of 32-bit register [12]. When operating on a stream of 16-bit data, word accesses can be used to read two 16-bit values at a time, and then another C6x intrinsic is used to operate on the data. Ideally, we would like to get all units simultaneously operating on all individual instructions. This parallelism is still hard to achieve by the compiler and may still need hand coding in some cases.

4.3 Memory Management

The memory management becomes very important as the DSP has a small amount of fast internal memory. Using internal memory to store instructions and data helps in increasing the processing speed. Generally, each 4x4 block in a MB has 4 edges, then each pixel in 4x4 block may be read or updated four times before the 4x4 block is filtered completely. Since the pixels of a MB (256 luma and 128 chroma pixels) are accessed frequently during the filtering process, they are stored in the internal memory, leading thus to a reduction of memory access.

5 Experimental Results

To evaluate the effectiveness of the proposed optimised algorithm, the adaptive deblocking filter was implemented on DM6437EVM platform. A TI DM6437EVM development environment including target board and Code Composer Studio 3.3 profile tools was set up [13]. Furthermore, system level optimisation methods were adopted according to TI's technical documentation [12].

The optimsed ADF was ported to a DM6437EVM achieving a speed performance of 32% in comparison to a direct implementation (no optimisation). The performance of the optimizing approach has been measured on Foreman and Paris QCIF video sequences, using three different QP values. The performances, in terms of video quality, with DSP implementation are shown in the table 2.

(a) (b)

Fig. 6. Performance of the deblocking filter for a highly compressed image (QP=38) (a) Reconstructed Foreman Image without Filtering. (b) Reconstructed Foreman Image with Filtering

(a) (b)

Fig. 7. Performance of the deblocking filter for a highly compressed image (QP=38) (a) Reconstructed Paris Image without Filtering. (b) Reconstructed Paris Image with Filtering

Table 2. PSNR of non filtered and filtered Foreman and Paris images with QP=28,33,38

QP	Image	PSNR Image non filtered	PSNR Image Filtered
28	Foreman	33,92	34,78
	Paris	30,27	30,92
33	Foreman	32,44	33,07
	Paris	30,06	30,62
38	Foreman	31,69	32,21
	Paris	29,58	30,23

6 Conclusion

In this paper, a DSP based specific method to decrease the adaptive deblocking filter module complexity in the H.264/AVC encoder and decoder is proposed. The implementation of the adaptive deblocking filter on DM6437EVM DSP using code optimisation reduces the module cycles consumption by 32%. The losses, in terms of video quality, are minimal compared to the non-optimised implementation.

The results presented in this paper show that the same image quality, but with less computing time can be obtained through optimisation for a target specific implementation. As perspective, the specific optimisations of this module, exploiting the architecture features of the DM6437EVM will be carried out. This will allow a speeding up of the filtering process for real time applications.

References

1. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Transactions on Circuits and Systems for Video Technology 13(7), 560–576 (2003)
2. Draft ITU-T Recommandations and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ISO/IEC/14496-10 (E) AVC) (July 2004)
3. Zakhor, A.: Iterative procedures for reduction of blocking effects in transform image coding. IEEE Transactions on Circuits and Systems for Video Technology 2(1), 91–95 (1992)
4. Averbuch, A.Z., Schlar, A., Donoho, D.L.: Deblocking of Block-Transform Compressed Images Using Weighted Sums of Symmetrically Aligned Pixels. IEEE Transactions on Image Processing 14(2), 200–212 (2005)
5. List, A., Joch, A., Lainema, J., Bjontegaard, A., Karczewicz, M.: Adaptive Deblocking Filter. IEEE Transactions on Circuits and Systems for Video Technology 13(7), 614–619 (2003)
6. Lin, H.C., Wang, Y.J.: Cheng. K.T., Yeh, S.Y., Chen, W.N., Tsai, C.N., Chang, T.S., Hung, H.M.: Algorithm and DSP Implementation of H.264/AVC. In: ASPDAC, pp. 742–749 (2006)
7. Lin, H.C., Wang, Y.J., Cheng, K.T., Yeh, S.Y., Chen, W.N., Tsai, C.N., Chang, T.S., Hung, H.M.: SIP Approach for Implementation of H.264/AVC. Journal of Signal Processing Systems 50(1), 53–67 (2008)
8. Warrington, S., Shojania, H., Sudharsanan, S., Chan, W.Y.: Performance Improvement of the Deblocking Filter Using SIMD Instructions. In: ISCAS 2006 (2006)
9. Major, A., Nousias, I., Khawan, S., Milward, M., Yi, Y., Arslan, T.: H.264/AVC In Loop De-Blocking Filter Targeting a Dynamically Reconfigurable Instruction Cell Based Architecture. In: IEEE 2nd NASA/ESA Conference on Adaptive Hardware and Systems (2007)
10. Texas Instrument, The New TMS320C64x Architecture Enhancements over the TMS320C62x
11. Texas Instrument, TMS320DM6437 Evaluation Module Technical Reference (2006)
12. Texas Instrument, TMS320C6000 Programmer Guide (2001)
13. Texas Instrument, TMS320C6000 Code Composer Studio Tutorial (1999)