Advanced Engineering Forum *Online: 2012-09-26*
ISSN: 2234-991X, Vols. 6-7, pp 416-421
doi:10.4028/www.scientific.net/AEF.6-7.416

High-speed Video Encoding System Based on H.264 and DSP

Chou-Chen Wang, Zhi-Hao Huang, Han-Yen Chen, Jyun-Liang Li

Department of Electronic Engineering, I-Shou University, Kaohsiung, Taiwan, R.O.C.
E-mail: chchwang@isu.edu.tw

Keywords: Video coding, H.264, motion estimation, Embedded system

Abstract. In this paper, a high-speed H.264 encoder based on ADI BF548 Blackfin DSP is proposed. In order to speed up the process of motion estimation (ME) module in H.264, we propose a two-step bit-transform-based normalized partial distortion search (TSB-NPDS) algorithm for fast ME by using the characteristics of pattern similarity matching errors. An initial standard compliant raw-C encoder has been optimized in speed for target processor. In addition, the parallelism between algorithm execution and data movement has been fully exploited using DMA. Experimental results demonstrate that the encoding rate can reach above 30 fps as using QCIF video.

Introduction

In H.264 encoding system, motion estimation (ME) module is a core process within a video image coding scheme, because it enables the transmission of video signals while using a lower bit rate. Recently, efficient block matching algorithms (BMA) have been adopted in modern video coding standards such as MPEG-1/2/4 and H.261/263/264. From the experimental results conducted in [1], we can find that the ME module consumes 60% (1 reference frame) to 80% (5 reference frames) of the total encoding time of H.264 codec. To speed up H.264 encoder, we adopts a two-step bit-transform-based NPDS (TSB-NPDS) algorithm to improve the searching efficiency of ME module.

Non-programmable encoders offer enough computational power at a low cost but cannot cope with the quick evolution of the video encoding algorithms. On the other hand, the latest generation of digital signal processors (DSPs) [2-4] can support very flexible encoders like the multi-format one proposed in [5] at a relative low cost. As a result, optimization of applications to the maximum possible extent is necessary.

Fast ME modules

The full search algorithm (FSA) is the most straightforward BMA for ME module, which provides an optimal solution by matching all the candidate blocks inside a search window. However, the computational complexity of FSA is always too high for real-time applications. Therefore, many fast ME modules have been proposed to reduce the computation of FSA [6-7]. Among these improved methods, the normalized partial distortion search (NPDS) [6] is a good approach. But, there exists some problems in NPDS such that the searching speed cannot be still satisfied in practical applications. To overcome the problem of NPDS, we adopt a two-step bit-transform-based NPDS (TSB-NPDS) algorithm to improve the searching efficiency of ME module.

NPDS. The ME module is to obtain a motion vector (MV) for a target macro block (MB) by using the block matching technique, which minimizes a measure of matching distortion between the target MB in the current frame and a candidate MB within a search window in a reference frame. The displacement between the candidate MB with the smallest distortion and the target MB will be selected as the resulting MV. One of the most frequently used criteria to measure the matching distortion is the sum of absolute difference (SAD). The SAD between a target MB at position (x, y) in the current frame (I_t) and a candidate MB at position $(x+u, y+v)$ in the reference frame (I_{t-1}) is defined as follows:

$$D(x,y;u,v) = \sum_{i=0}^{15}\sum_{j=0}^{15}\left|I_t(x+i,y+j) - I_{t-1}(x+i+u,y+j+v)\right| \quad (1)$$

where $I_t(\cdot,\cdot)$ and $I_{t-1}(\cdot,\cdot)$ represent pixels intensity.

Based on halfway-stop idea, the partial distortion of the NPDS is defined as a group of pixels' distortions instead of a single pixel's distortion. Thus, the block distortion $D(x,y;u,v)$ is divided into 16 partial distortions d_p, where each partial distortion consists of 16 points spaced equally between adjacent points. The pth partial distortion is defined as

$$d_p(x,y;u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3}\left|I_t(x+4i+s_p,y+4j+t_p)-I_{t-1}(x+4i+s_p+u,y+4j+t_p+v)\right| \quad (2)$$

where (s_p,t_p) is the offset of the upper left corner point of the pth partial distortion from the upper left corner point of the candidate block. The pth accumulated partial distortion is defined as

$$D_p(x,y;u,v)=\sum_{i=1}^{p}d_i(x,y;u,v) \quad (3)$$

The NPDS matches all the search points inside the search window as that in the FSA. During each block matching, the NPDS compares each accumulated partial distortion D_p with the normalized minimum distortion defined as

$$D_{norm}=pD_{min}/16 \quad (4)$$

where D_{min} is the current minimum distortion. The comparison starts from $p=1$ and proceeds toward $p=16$, and the comparison is stopped if the normalized partial distortion of the candidate motion vectors (CMVs) is greater than D_{norm}. At the end of comparison (i.e. $p=16$), if D_{16} is smaller than D_{min}, then this CMV becomes the new current minimum point. By comparing the normalized partial distortion against D_{norm}, computational complexity is reduced by high rejection of impossible CMVs at early stage.

Proposed TSB-NPDS. From the study of NPDS, we find that the proper matching scan is an important factors that affect the searching speed to eliminate the impossible candidate MV. However, the NPDS uses the evenly dithering order which takes account of pixels being evenly distributed on the block. Therefore, the speedup ratio to find MVs is limited due to uniform calculation order.

To speed up NPDS, we make use of the fact that the image blocks with lower pattern similarity have larger matching errors on average. Therefore, the motivation of the proposed algorithm is to use two significant features of image block pattern extracted by 2BT from a MB to find the impossible candidates faster. In addition, the key idea of the proposed method is to design a simple and low-complexity hardware machine to calculate approximate distortion order of each candidate block. For simplicity, we take the mean value as the threshold variables after several experiments in our work. 2BT can be expressed as

$$B_1(x+4i+s_p,y+4j+t_p)=\begin{cases}1 & if\ I_t(x+4i+s_p,y+4j+t_p)\ge\mu_p\\ 0 & otherwise\end{cases} \quad (5)$$

$$B_2(x+4i+s_p,y+4j+t_p)=\begin{cases}1 & if\ I_t(x+4i+s_p,y+4j+t_p)\ge\mu_p+0.5\mu_p\\ 0 & otherwise\end{cases} \quad (6)$$

where $i=0,1,2,3$, $j=0,1,2,3$, $B_1(\cdot,\cdot)$ and $B_2(\cdot,\cdot)$ represent the resulting two bit-plans of the sub-block, and μ_p represents the pth mean value of the sub-block in a MB. Thus, B_1 chooses the mean threshold to extract binary direction pattern and B_2 chooses the other threshold further offset half of mean to pick out binary contrast pattern. If the bit-plane pattern of a sub-block is dissimilar to that of another sub-block, the two blocks are unlikely to have similar image characteristics, and then result in a large matching error.

With the above-observation, we propose a two-step binary pattern matching scan algorithm based on these characteristics to achieve more computational reduction of NPDS. In the first step, the Hamming distance between two bit-planes of sub-blocks in the target MB and the candidate MB of the initial searching point is used as the measurement of binary direction pattern similarity. The Hamming distance H_1 is defined as

$$H_1 = \sum_{i=0}^{3}\sum_{j=0}^{3} B_1^t(x+4i+s_p, y+4j+t_p) \oplus B_1^{t-1}(x+4i+s_p, y+4j+t_p) \tag{7}$$

where $B_1^t(\cdot,\cdot)$ and $B_1^{t-1}(\cdot,\cdot)$ denote the binary value of the $(x+4i+s_p, y+4j+t_p)\,th$ pixel of the bit-plane of the 4×4 sub-block in the current frame and the reference frame, respectively; and $\oplus$ denotes the modulo-2 addition (XOR logic operation). The Hamming distance for each sub-block between the target MB and the candidate MB of the initial searching point is carried out, and the values are sorted in descending order. Therefore, the H_1 is first used to determine the order of matching priority. In addition, it is noted that the different contrast between gray-levels in the block maybe have the same direction patterns such that result in the same bit-plane and H_1. In order to further separate the pattern similarity of gray-level between two sub-blocks with the same H_1, the binary contrast pattern B_2 is adopted in the second step. The Hamming distance H_2 of the second step is defined as

$$H_2 = \sum_{i=0}^{3}\sum_{j=0}^{3} B_2^t(x+4i+s_p, y+4j+t_p) \oplus B_2^{t-1}(x+4i+s_p, y+4j+t_p) \tag{8}$$

Then, the values of H_2 for the same H_1 are re-sorted in the descending order.

The first step mainly computes edge directional differences using binary pattern matching for all pixels in the center block of the search window, that is to say the one with the null candidate motion vector. The magnitudes of Hamming distances of sub-blocks in a MB are sorted in descending order. In the second step, if the sub-blocks have the same Hamming distances, then the H_2 magnitudes of these sub-blocks are resorted in descending order. The positions are first sorted in descending order according to our proposed two-step matching scan method, and then used to determine the order of matching priority for all the candidate blocks in the search window. According to the arranged sub-block, we find the best MV using the same search procedure as the NPDS.

Figure 1 shows that the average partial distortions versus calculation order determined by the proposed two-step bit-transform-based matching scan. We can find that matching errors is proportional to the Hamming distances between two blocks. This indicates that larger matching errors can be obtained by calculating matching distortions of the image area with large Hamming distance. Therefore, we can prove that it is true the lower calculation order the higher matching distortion in our method. The first partial distortions are much larger than those for evenly dithering order.

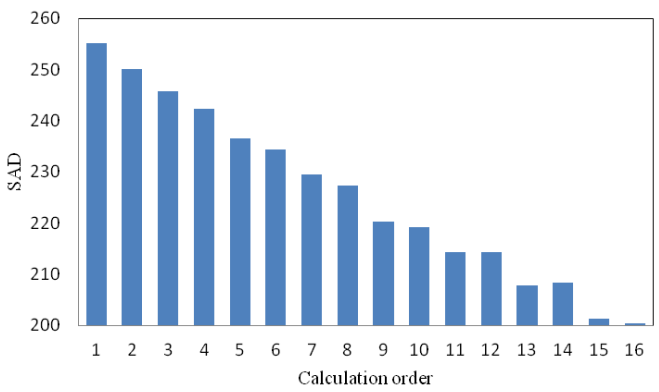


Fig. 1 The SAD versus calculation order determined by the TSB-NPDS matching scan.

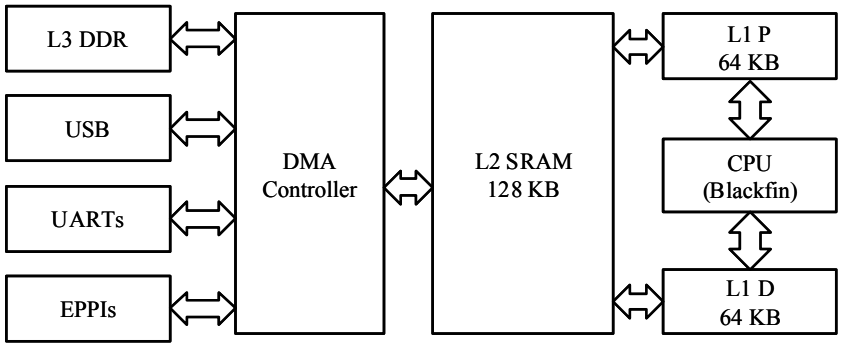


Fig. 2 The architecture of ADI BF548 DSP.

The architecture of ADSP-BF548

A simplified block diagram of the DSP internal architecture is shown in Fig. 2. The CPU is a Blackin processor with a performance of up to 4800 MIP@600 MHz. ADI BF548 is a 16/32-bit processor and gives high performance for low power consumption, which makes it attractive for embedded applications. A 192 KB internal SRAM can be divided into 64 KB level-1 cache for program (L1P) and 64 KB for data (L1D). A leve-2 cache (L2) unified for data/program is 128 KB. A 64 MB external memory DDRRAM level-3 (L3) can be accessed through a dedicated interface using a 64-bit data interface. The other peripherals are a dynamic memory access (DMA) controller, video ports, an Ethernet port (EMAC), an output audio interface and several general-purpose I/O pins. The DMA controller allows moving data between memory and peripherals.

BF548 is based on RISC principles, and the normalized instruction set and related decode mechanism are much simpler. This simplicity results in a high instruction throughput and impressive real-time interrupt response from a small and cost-effective chip. Pipelining is employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory. The memory interface has been designed to allow the performance potential to be realized without incurring high costs in the memory system. Unlike traditional video embedded solutions that utilize two processor cores to provide video functionality, the BF548 provides a convergent solution in a unified core architecture that allows voice and video signal processing concurrent with RISC MCU processing to handle network and user-interface demands. This unique ability to offer full functionality on a single convergent processor provides for a unified software development environment, faster system debugging and deployment, and lower overall system cost.

Table 1: The BP profile information of the encoding process.

Module Name	**Time Consume in Module**
Motion Estimation (ME)/ Motion Compensation (MC),	79.19%
Intra Prediction	10.14%
Integer Cosine Transform (ICT)/inverse ICT (IICT), Quantization (Q)/Inverse Quantization (IQ)	8.06%
Entropy Coning	2.04%
Others	0.67%

H.264 Encoding Implementation

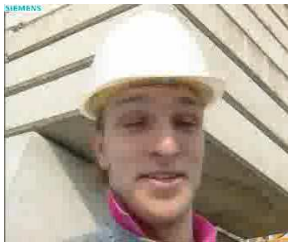
The encoder implements BP of H.264 video coding standard [8]. The starting point was a standard compliant C encoder fully tested first in a PC environment and moved o the DSP environment afterwards. This initial code was optimized to increase the execution speed in about two orders of magnitude.

H.264 Encoding Procedure. H.264 encoding modules including ME/ motion compensation (MC), intra prediction, quantization (Q)/inverse quantization (IQ), integer cosine transform (ICT)/inverse ICT (IICT), and entropy coding. Table 1 gives the H.264 BP information of the encoding modules [9]. The most consuming processes of encoder include ME/MC and ICT/IICT modules as summarized in Table 1. It is evident from Table 1 that the ME/MC module occupies most time in encoder. Therefore, we apply the proposed TSB-NPDS to the ME module to speed up the encoding process. To reduce the computational load, we use C level optimization and hardware specific optimization technique like intrinsic operator and memory management. In addition, there will be considerable improvement in performance if the ICT/IICT can be optimized to the maximum possible extent. ICT/IICT module is generally applied to a block of 8×8 pixels by using Chen's algorithm [10]. There is high spatial and temporal correlation when video sequence is applied to multimedia applications, this will result in very few significant coefficients (non-zero coefficients) after ICT while encoding. The number of non-zero coefficients reduces further after quantization. Instead of generalized 8×8 IICT, depending on the number of non-zero coefficients, IICT can be applied to only the significant valued coefficients. This optimizes IICT to a great extent since default 8×8 IICT will involve unnecessary multiplications and additions for the zero valued coefficients.

Use of Memory. The DSP memory usage in the decoder for processing data in ADI BF548 DSP is optimally allocated according the proposed structure. The reference pictures are stored in internal memory L2 to speedup the encoding procedure. REF, ICT_COEFFS, Q/IQ, DC/AC_COEFFS, VLD_COEFFS and REC buffers are located in internal memory L1D (internal SRAM is configured as 128 KB). The assigned addresses for L1 and L2 are from 0xffb00000 to 0xff800000 and from 0xfeb00000 to 0xfeb20000, respectively.

Code Optimization. In order to reach real-time operation, additional optimization steps have been carried out as follows:

1. The higher computational cost functions have been moved to internal memory.
2. Also frequently accessed data have been moved internal memory.



(a) Forman

(b) Carphone

(c) Test

Fig. 3 Test video sequences.

Table 2: Complexity of various module in H.264 encoder

H.264 Encoder Module	M Cycle/sec	
	Non-Optimized	Non-Optimized
ME/MC	504	102
Intra Prediction	122	22
ICT/IICT and Q/IQ	108	15
Entropy Coning	46	10

Table 3 Performance of high-speed H.264 encoder using QCIF sequences

Bitrate / PSNR / Sequence	M Cycle/sec		Average PSNR (dB)	
	384 kbps	512 kbps	384 kbps	512 kbps
Foreman	288	424	34.2	34.8
Carphone	292	402	38.2	39.1
Test	312	464	33.8	34.6

3. Frequent arithmetic operations have been coded using intrinsic (pseudo-assembler) instructions.
4. The core of the Q/IQ, ICT/IICT, VLD and ME/MC has been also coded intrinsic.
5. The reference frame has been loaded to L2 memory.

Test Results

The proposed high-speed H.264 encoding system is tested for different sequences representing typical video conferencing content including "Forman", "Carphone" and "Test" as shown in Fig. 3. The "Test" video sequence is actually taken from the Digital Video (DV). The sequences are encoded at QCIF (176×144) resolution, 30 fps and at bit-rates of 384 and 512 kbps. A PC running Visual DSP++ v5.0 has been used to carry out the simulation profiles and implemented on the ADSP-BF548 EZ-KIT Lite. Module complexity of optimized H.264 encoder is shown in Table 2 for Foreman sequence. The objective quality of encoded sequence is measured in terms of peak signal to noise ratio (PSNR) and the processing power required is measured in Mcycles as shown in Table 3. The PSNR loss for optimized encoder is about 0.03 to 0.12 db. For the CIF video sequence, it still needs to improve the encoding rate due to the limitation of memory.

Conclusions

In this paper, we propose high-speed H.264 BP video encoder based on BF548 DSP. The implementation is well suited for videophone applications, as one core of DSP can perform encoding. The optimization techniques presented in this work can be effectively used for other programmable processors with similar architecture and instruction set.

References

[1] Z. Chen, P. Zhou, Y. He, "Fast integer pel and fractional pel motion estimation in for JVT", JVT-F017r1.doc, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Awaji, Island, Dec. (2002).

[2] Philips Semiconductors. Next Media Processors. Available online at http://www.nxp.com/products/nexperia/home/products/mediaprocessors/index.html

[3] Analog Devices. Blackfin processors. Available online at: http://www.analog.com/processors/blackfin/index.html

[4] Texas Instruments. C6000 DSPs. Available online at: http://focus.ti.com/paramsearch/docs/parametricsearch.tsp?family=dsp&sectionId=2&tabId=1941&familyId=1398

[5] Y. S. Tung et al.: "DSP-Based Multi-Format Video Decoding Engine for Media Adapter Applications", IEEE Trans. on Consumer Electronics, vol. 51 (2005), p. 273

[6] C. K. Cheung and L. M. Po, "Normalized partial distortion search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 10 (2000), no. 3, p. 417

[7] C. C. Yang, G. L. Li, M. C. Chi and M. J. Chen "Prediction error prioritizing strategy for fast normalized partial distortion motion estimation algorithm," IEEE Trans. Circuits Syst. Video Technol., vol. 20 (2010), no. 8, p. 1150.

[8] I. E Richardson, H.264 and MPEG-4 Video Compression. John Wiley & Sons, September 2003. ISBN 0-470-84837-5.

[9] S. Y. Chien, Y. W. Huang, C. Y. Chen, H. H. Chen and L. G.. Chen, "Hardware architecture design of video compression for multimedia communication systems," IEEE Communications Magazine, p. 123, Aug. (2005)

[10] D. LeGall: "MPEG: A Video Compression Standard for Multimedia Applications," Communications of ACM, vol. 34 (1991), no. 4, p. 46