

Optimization of Memory Allocation for

H.264/AVC Video Decoder on Digital Signal Processors

Shuai Hu , Zhe Zhang , Mengsu Zhang , Tao Sheng
Department of Electronics and Information Engineering,
Huazhong University of Science and Technology
hushuai@smail.hust.edu.cn
zhangzhe@smail.hust.edu.cn

Abstract

The computing power of microprocessors has exponentially increased in the past few decades, so the support to compute intensive multimedia applications has increased too. With such improved computing power, memory subsystem deficiency becomes the major barrier to support video decoder on the Digital Signal Processor (DSP). H.264/AVC becomes the next generation of video codec for embedded systems. In this paper, our focus is to enhance the decoding performance of H.264/AVC through memory optimization for DSP. The experimental results show that the proposed solutions can improve the H.264/AVC decoding speed by almost 37%. Also, the memory optimization method presented in this paper has been integrated into the developed embedded H.264/AVC video decoder. The decoder can perform real time decoding under eight channels of Common Intermediate Format (CIF) frame size, which is the typical requirement of networked video surveillance applications.

1. Introduction

A proverb says: A picture speaks more than a thousand words. Video messaging, video telephony, and video conferencing have started to enter the marketplace over the past several years and are becoming more and more popular. The demand to run such video applications on embedded devices is growing. The future of the embedded systems running multimedia is video applications. H.264/AVC is the new video coding standard. It can save 25%-45% and 50%-70% of bitrate compared with MPEG-4 Advanced Simple Profile and MPEG-2, respectively [1]. Although motion compensated transform coding is still adopted, many new features are used to achieve

much better compression performance and subjective quality, such as quarter-pixel Motion Estimation (ME) with Multiple Reference Frames (MRF) and Variable Block Sizes (VBS), intra prediction, Context-based Adaptive Variable Length Coding (CAVLC), and in-loop de-blocking filter. The rate distortion optimized mode decision [1] is also included in reference software to improve rate-distortion efficiency.

Applications lead the architecture of embedded devices. For many embedded devices, single-processor implementation is recommended to meet the application's size and power consumption requirement [2]. Due to the improvements in the semi-conductor industry, the required computational power to implement such a device is not an issue. However, due to the growing disparity between the increasing computation power and the memory speed, the amount of traffic from CPU to memory leads to a significant processor/memory speed gap [7]. Also, bus contention becomes a serious problem for the newly proposed multi-core processors like Intel Xeon and AMD Opteron. Using cache(s) may improve the overall performance by dealing with memory bandwidth bottlenecks and bus contention problems. Using data stored in the cache helps cut down bus traffic significantly. Consider an embedded video application with multi-channels Common Intermediate Format (CIF-352×288 pixels), 20-25 frames per second (fps), and 500 kbps data rate, which could be supported by Digital Signal Processors (DSP) with eight highly independent functional units that may take 720M cycles to run an H.264/AVC decoder. So, a two-level internal memory organization for DSP is designed to keep pipeline processing.

The rest of this paper is organized as follows. In section 2, we introduce the related work. Section 3 presents the H.264 decoder architecture. Section 4 shows the two-level internal memory hierarchy

designed for DSP. Finally, we draw our conclusions in Section 5.

2. Related work

Generally speaking, there are two aspects affecting the system the most. One is the computational complexity, and the other is the communication between the processor and the memory. In this paper, we enhance the decoding performance of H.264/AVC through memory optimization of a digital signal processor. Some of the previous researches which have been done in this area are presented in the following paragraph.

In [3], the authors give an overview of high-level complexity analysis and memory architecture of multimedia algorithms. However, if the amount of computation exceeds the maximum ability of processor, multi-processor or low complex algorithm should be considered. In [4], an efficient memory centric design methodology which completes the computational cost with a detailed analysis of data transfers and storage is applied on a simple profile MPEG-4 video decoder. The study [4] has showed how algorithmic optimization is an enabling factor for this methodology. The focus of memory is motivated by the data, which is the dominated nature of video coding algorithms, i.e., the data transfer and storage are the main cost factors in an efficient realization. Cache behavior of multimedia and traditional applications was examined in [5]. The authors in [5] have showed that multimedia applications exhibit higher data miss rate and comparable lower instruction miss rate. The problem related to improving memory hierarchy performance for multitasking data intensive application was addressed in [6]. The authors in [6] have used cache partitioning techniques to find a static task execution order for inter-task data cache misses. Due to the lack of freedom in reordering task execution, this method optimizes the caches more. In this aspect, in this paper, we present a tradeoff between the coding efficiency and the complexity. For the second aspect, it is a common situation that the speed of memory can not catch the speed of processor. Cache can balance the mismatch between them. So a two-level on-chip memory optimization method for DSP is presented in this paper to keep pipeline processing.

3. H.264/AVC Video

The standardization of video compression technology is revolution in the broadcast television, telecommunication, and home entertainment system. H.263 is standardized by ITU-T (International

Telecommunication Union – Telecommunication Standardization Sector) in 1995 and it is widely used in videoconferencing systems. MPEG-4 (Part 2) is standardized by ISO (International Organization for Standardization) in 1998 and it enables a new generation of internet-based video applications. The groups developed these standards, the ISO Motion Picture Experts Group (MPEG) and the ITU-T Video Coding Experts Group (VCEG), have developed new standard that promises to significantly outperform both MPEG4 (Part 2) and H.263, providing high-quality, low bit-rate streaming video. The new standard is called Advanced Video Coding (AVC) and is widely known as H.264/AVC or MPEG-4 Part 10 [11].

In 2001, the Joint Video Team (JVT) was formed including experts from MPEG and VCEG. The main task of JVT was to develop the draft H.26L (VCEG's long-term effort to develop a new standard) into a full International Standard. In 2003, the final drafting work on the first version of the new standard was completed. The outcome is two identical standards: ISO MPEG standard is MPEG4 Part 10 and ITUT standard is H.264/AVC.

3.1. H.264/AVC codec

H.264/AVC standard does not explicitly define CODEC (encoder / decoder pair) like MPEG-4. Instead, H.264/AVC defines the syntax of an encoded video bit-stream together with the method of decoding this bit-stream [12]. The functional elements of an accommodating encoder and decoder are possible to include. Figure 1 shows the decoder functionalities. The functions shown in this figure are likely to be necessary for compliance. In addition, there is scope for considerable variation in the structure of the CODEC. The basic functional elements (such as prediction, transform, quantization, and entropy encoding) are little different from previous standards (MPEG-4 and H.263).

3.2. H.264/AVC decoder

We briefly discuss H.264/AVC decoding algorithm, our target application for this work. Various functionalities in H.264/AVC decoder are shown in Figure 1. The decoder receives an encoded bit-stream from the Network Abstraction Layer (NAL). The data elements are entropy decoded and reordered to produce set of quantized coefficients (X). These are rescaled and inverse transformed to give D'n. Using the header information from the bit-stream, the decoder creates prediction macro-block P, identical to the original prediction P formed in the encoder. P is added to D'n

to produce uF'_n which is filtered to create the decoded macro-block F'_n [12, 13].

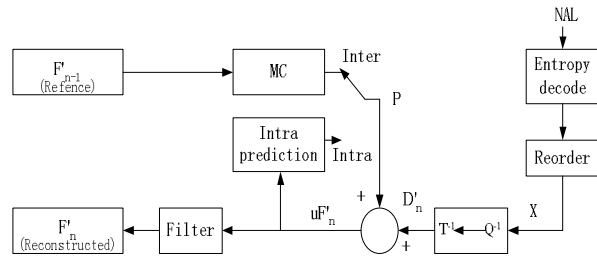


Fig.1. H.264/AVC decoder architecture

It is very important that both encoder and decoder use identical reference frames to create the prediction. Otherwise, the predictions in encoder and decoder may be different, leading to an increased error between the encoder and decoder.

4. Two-level Internal Memory Optimization

In real-time coding system, one serious bottleneck is the speed mismatch between memory and processor. One solution is to use cache efficiently. But by depending only on cache replacement policy by hardware, the decoder can not reach the top coding speed. So, we can analyze the data flow to help cache find out the best mapping position, or manage cache by ourselves. Compile-time data caching decisions have much large effect on the performance [9]. The following parts will introduce the two-level internal memory structure of DSP and the details of the memory organization design for H.264 video coding in turn.

4.1. The Characteristic of TMS320DM642

The TMS320DM642 (DM642) is the first integrated media processor based on the C64x Very Long Instruction Word (VLIW) DSP core [10]. The C64x CPU is optimized for video processing including both 8-way VLIW parallelism and packed data processing within each functional unit. The CPU is complimented by a flexible chip-level architecture that maximizes system options and sustains the core processing capability. The key elements in the DM642 include: Two-level cache architecture, an Enhanced DMA (EDMA) controller, 64-bit external memory interface, three 20-bit video ports, Ethernet MAC, and so on.

4.2. Two-level Internal Memory Structure

The DM642 utilizes two-level cache architecture as shown in Fig.2. The first level, called L1, allows three

parallel single-cycle memory accesses (one instruction fetch and two 64-bit data accesses) by the CPU at clock rates exceeding 720MHz [10]. This ensures a strong performance roadmap over time. The L1 Program (L1P) cache is 16 Kbytes and is directly mapped. The L1 Data (L1D) cache is also 16 Kbytes, but it is a two-way set associative to account for multiple input sources. The second level, called L2, is 256 Kbytes and allows large amounts of data to be brought on-chip. For example, regions of reference data for motion estimation can be retained on-chip across multiple blocks minimizing redundant external I/O. The L2 can be partitioned between cache and Static Random Access Memory (SRAM). Instructions, C variables, and random data accesses are typically cached. Regular data accesses, such as video input and output, are typically done with EDMA requests and use mapped L2 SRAM.

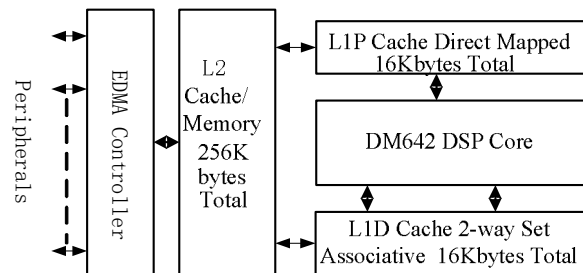


Fig.2. Two-level cache architecture of DM642

4.3. Optimized Cache Sub-system Design

The optimal memory allocation method is based on the Cache-SRAM-SDRAM structure. The cache has the highest speed but the smallest capacity, while the SDRAM is totally opposite of it. The SRAM is in the middle and it negotiates the unbalance of the cache and the SDRAM in speed and capacity. So, it affects the system efficiency greatly. Putting small but frequent reused data types in SRAM is one possible way to protect them from victimization by larger data objects such as video data. The frequently reused data, which are subject of being prematurely ejected from the cache, could then be kept in this space. According to this mode, the SRAM is divided into three sections: The first section is data exchanging section, which is composed of two data buffers. This section can solve the video data storage and exchanging problems. The second section is the core code and variables storage section, which is used to store the frequently called functions such as Inverse Transform, Motion Compensation and relevant variables. This section is used for the frequent data management. The third

section is the rest of the SRAM, which is considered as cache for the other codes and data management.

The data exchanging section is primarily used to solve the access and exchange of video data during H.264 video decoding process. If the reference frame and the current frame are allocated to SRAM, the whole frame buffer will reach 2.4MB. In addition, the source code needs approximately the 150Kbytes storage space, while the other variables need about 300Kbytes storage section. Therefore, the system altogether needs approximately the 2.85MB storage space, which is by far bigger than the 256Kbytes internal memory size that DM642 can provide. So, if decoders take the frame as the unit like the PC decoding mode, all video frames must be laid in SDRAM, which will cause frequent exchanging data between On-chip memory and Off-chip memory during decoding. Since the access to external memory is much slower than the access to internal memory, it further inspires the contradiction between the processing speed of the processor core and the data access speed and it causes the DSP to consume most of the time while accessing to external memory. By this, it is critical to optimize the architecture for H.264 decoding and allocate reasonable data exchanging section. This optimization processes multiple blocks at a time to obtain optimum cache performance and EDMA bandwidth efficiency.

The unit of H.264 decoding is a 16×16 pixel fragment known as a macro-block. The macro-block covers the area of six 8×8 blocks (for 4:2:0 sub-sampling). So that, decoders consider the Group of Macro-Block Lines (GMBL) as it is the decoding unit. In order to reduce the access to SDRAM as much as possible during decoding GMBL, the buffer must be able to store the decoded video data, reference frame data, and the relevant structure variables of each GMBL. In order to process multiple blocks at a time to obtain optimal cache performance and EDMA bandwidth efficiency, a double data buffers, namely BUF1 and BUF2, are used to store the decoded data in turn. Therefore, the decoder allocates almost 173Kbytes size space of SRAM for data exchanging section. The 173Kbytes allocated size space comes from 20.25Kbytes (the size of the original GMBL) plus 98.25Kbytes (the size of reference GMBL) and plus 55Kbytes (the size of structure variables). Therefore, all the ways of storing decoded data may be divided into two kinds: The frame level and the GMBL level. In the optimal memory allocation method, the frame buffers (GMBL1-6) are located in SDRAM, and the GMBL buffers (BUF1-2) are laid in SRAM, as shown in Fig.3.

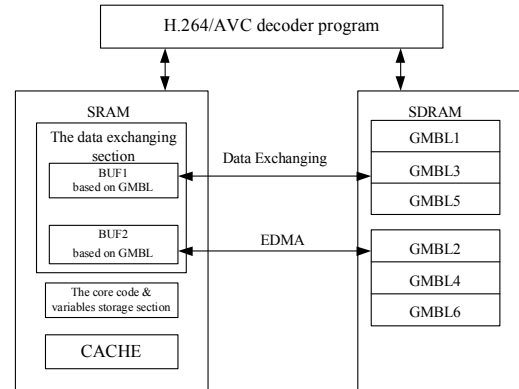


Fig.3. H.264 decoder scheme based on the GMBL structure

Our method for utilizing SRAM to save video data solves the video data storage and the exchange problems. However, unless the data exchange can be realized in time, the performance of the decoder may still not be improved. Therefore, it's necessary to allocate parts of SRAM for storing the relevant variables of code and program. The H.264 decoder has a characteristic of repeating operations and codes, which need to be frequently called. In order to improve the hit rate of L1P cache, the core code storage section needs to be allocated to store the frequently called functions, inverse quantization and so on. Thus, after subtracting data exchanging section, there is only 83Kbytes storage space left in L2 SRAM. In addition, there is a choice of cache setting for selecting the way of data scheduling from SDRAM. By taking into consideration of the contents above, this paper proposes 3 modes:

Mode 1: All the spaces of On-Chip memory are configured to SRAM that is used to store codes, data, and global variables. This mode is called ALL SRAM of L2. As for video algorithms, this mode is feasible because the direction of data streams is clear and scheduling EDMA can complete the data exchange. Yet, it will consume a lot of time once DSP accesses external memory.

Mode 2: The storage space of 83Kbytes size SRAM is divided into two parts: The first one is of 64Kbytes size for cache, while the surplus is for saving codes and data, which are in common use. In this mode, the L2 cache is four-way set associative.

Mode 3: The storage space of 83Kbytes size SRAM is divided into two parts: 32Kbytes size SRAM is set for cache and the surplus is used for saving codes and data. In this way, the access to SDRAM is possible to complete in a high speed taking advantage of the

cache. In this mode, The L2 cache is two-way set associative.

After comparing the performance of the three modes above, it is apparent that the second mode is the optimum one. This mode is called Cache-based memory mode. It separates the 64Kbytes SRAM into L2. Surplus 19Kbytes SRAM is used for storing core codes and variables. Other codes and data, which are scheduled by L2 cache controller, are stored in SDRAM. The DSP/BIOS statistical results are shown in table 3. From this table, we can notice that the proposed method (Cache-based memory mode) can improve H.264/AVC decoding speed by almost 37% compared to the no-memory optimization, with the standard video sequences.

Table 3: The instructions cycles of decoding a frame

Video Sequence	256 Kbytes Cache	64 Kbytes Cache	32 Kbytes Cache	No Cache	Speed Inc. (%)
Foreman	3670	2319	2980	2841	36.8
	784	935	676	189	
Tempete	3878	2439	3117	2970	37.1
	023	276	935	567	

Note: Foreman, Tempete are the typical standard video sequences.

5. Conclusions

Cache memories have strong influence on system performance and are used to fill the processor-memory speed gap. In this paper, an efficient two-level internal memory organization for DSP is designed in detail according to its hardware characteristics and the software characteristics of H.264 video decoder. With the memory optimization, the obtained H.264 decoder can simultaneously decode eight channels CIF video frames on TMS320DM642.

The experimental results presented in this paper show that H.264 decoding performance can be improved by almost 37% compared to the no memory optimization, with the standard video streams. In our future work, we plan to design and implement 720p and 1080i resolution real-time decoder, which can further improve the requirement of high-definition embedded H.264/AVC video decoder.

6. References

[1] T.Wiegand, H.Schwarz, et al, "Rate constrained coder control and comparison of video coding standards," IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 688–703, July 2003.

[2] A.Jerraya, H.Tenhunen, et al, "Multiprocessor Systems-on-Chips," IEEE Computer Society, pp.36-40, July 2005.

[3] M. Ravasi, et al, "High-Abstraction Level Complexity Analysis and Memory Architecture Simulations of Multimedia Algorithms", IEEE Trans. CSVT, Vol. 15, No. 5, pp. 673-684, May 2005.

[4] Denolf, K. De Vleeschouwer, et al., "Memory centric design of an MPEG-4 video encoder", IEEE Trans. CSVT, Vol. 15, No. 5, pp. 609-619, May 2005.

[5] A. Asaduzzaman, I. Mahgoub, et al, "Cache Optimization for Mobile Devices Running Multimedia Applications", Proceedings of the IEEE Sixth ISMSE, Miami,FL, December 2004, pp. 499-506.

[6] A.M. Molnos, B. Mesman, et al, "Data Cache Optimization in Multimedia Applications", Proc. of the 14th Annual Workshop on Circuits, Systems and Signal Processing, Veldhoven, November 2003, pp. 529-532.

[7] E.Rotenberg, "Architectures for Real-Time", www.tinker.ncsu.edu/ericro/research/realtime.htm, 2005.

[8] T. Wiegand, X.Zhang, et al "Long-Term Memory Motion Compensated-Prediction", IEEE Transactions on Circuits and System for Video Technology, pp.70-84, Feb. 1999.

[9] E.D. Greef, et al, "Memory Organization for Video Algorithms on Programmable Signal Processors", in Proceedings of IEEE Int. Conf. on Computer Design, pp. 552-557, Oct. 1995.

[10] Golston, J. "DM642 digital media processor", in Proc. Of SPIE, v 5022 II, pp. 700-706, 2003.

[11] ITU-T Rec. H.264/ISO/IEC 11496-10, "Advanced Video Coding", Final Committee Draft, Document JVTE022, September 2002.

[12] I. Richardson, "H.264 / MPEG-4 Part 10 White Paper", www.vcodex.com, 2002.