ICSP2006 Proceedings

Cache optimization for an embedded MPEG-4 video decoder

Hongxing Guo, Tao Sheng, Weiping Sun, Jingli Zhou, Shengsheng Yu
Department of Computer Science and Engineering, Huazhong University of Science and Technology,
Division of Data Storage System of Wuhan National Laboratory for Optoelectronics,
WuHan, HuBei, 430074 China
E-mail: wtguohx@hust.edu.cn

Abstract

Video decoders are an important component in modern networked multimedia systems. TMS320DM642 is suitable for implementation of embedded video decoder. However, the high data rate, large sizes, and distinctive memory access patterns of MPEG-4 video decoders exert a particular strain on cache. This paper proposes an optimization method to allocate the memory of TMS320DM642. In Cache-SRAM-SDRAM structure, the SRAM negotiates the unbalance of the cache and the SDRAM in speed and capacity. Due to its importance, a cache-based memory allocation mode is proposed to make full use of the SRAM. According to this mode, the SRAM is divided into three sections: the data exchanging section, the core code and variables storage section, and the rest of the SRAM which is set as cache for the other codes and data management. The experiment results show the proposed solutions can improve the MPEG-4 decoding speed by almost 25%. This cache optimization method has been integrated into the developed embedded MPEG-4 video decoder. The decoder can perform real time decoding with bitstreams coded under two channels 4CIF or eight channels CIF frame size which is the typical requirement of networked video surveillance applications nowadays.

1. Introduction

Embedded digital video, particularly for video surveillance and streaming media, is an emerging application whose adoption is inevitable in light of demand for applications and the increasing capabilities of semiconductor technology. The MPEG-4 is emerging as the predominant video and audio format for these applications. 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 TMS320DM642 with eight highly independent functional units that may take 600M cycles to run an MPEG-4 decoder to support this application. Unlike the encoder, the decoder is explicitly specified by the MPEG-4 standard, leaving few choices for the system designer regarding what algorithms to employ. The high data rate, large sizes, and distinctive memory access patterns of MPEG-4 exert a particular strain on cache [1]. While miss rates are acceptable, they generate significant excess cache-memory traffic. Multimedia applications seriously suffer due to cache inefficiency from dropped frames, blocking, or other annoying artifacts. It is important to understand the decoding algorithm to improve performance through Cache-SRAM-SDRAM sub-system optimization.

This paper will propose a cache-based memory allocation mode to optimally utilize the SRAM. In Section 2, the MPEG-4 video decoding algorithm is summarized. The Architecture of the MPEG-4 decoder is discussed in Section 3. Optimized cache sub-system designs are explained in Section 4. Finally, Conclusions are given in Section 5.

2. The MPEG-4 video decoding algorithm

This section reviews the essentials of MPEG-4 Video compression techniques. MPEG-4 achieves the compression of video data using two orthogonal lines of attack [2]. In the spatial domain, MPEG-4 extracts redundancies from each individual image, operating on a frame by frame basis. In the temporal domain, the algorithm operates between frames, taking advantage of visual content common to adjacent frames. First, each frame is broken down into 8 × 8 pixel image fragments known as blocks. With spatial compression, each block is filtered through the discrete cosine transform and quantized, and then undergoes variable length coding. Temporal compression, which is most relevant for cache performance, uses motion estimation, constructing frames from pieces of other frames (known as reference frames), translated as a group from their location in the source image. This information is stored as a motion vector representing the translation, and a difference block, requiring far fewer bits than the original image fragment. The unit of motion is a 16 × 16 pixel fragment known as a macro block, covering the area of four 8 × 8 blocks. Image information not present in reference frames is encoded spatially on a macro block by macro block basis. Difference blocks and motion vectors are further compressed with variable-length coding. The reconstruction of images from motion estimation data is known as motion compensation.

For MPEG-4 simple profile, the motion estimation and compensation are organized using two different types of frames. I (intra) frames contain a spatially compressed image without motion-compensated elements. P (predicted) frames are built primarily of pixels from the closest previous I or P frame. The ability to reference data in past frames provides additional opportunities for compression, but introduces data dependencies.

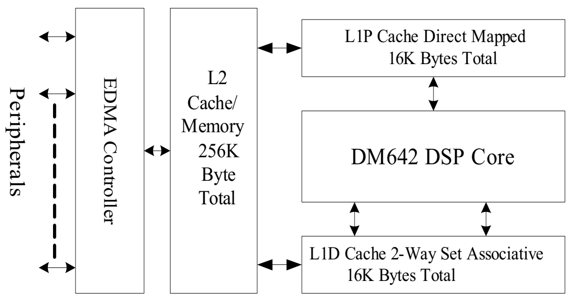
3. The Architecture of the MPEG-4 decoder

3.1. The characteristic of TMS320DM642

The DM642 is the first integrated media processor based on the C64x VLIW DSP core [4]. The C64x CPU is optimized for video processing including both 8-way VLIW parallelism and packed data processing (SIMD) within each functional unit. The CPU is complimented by a flexible chip-level architecture that maximizes system options and sustains the core processing capability. Key elements in the device 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

3.2 Two-level cache architecture

The DM642 utilizes two-level cache architecture as shown in Fig.1. The L1 cache allows 3 parallel single-cycle memory accesses (one instruction fetch and two 64-bit data accesses) by the CPU at clock rates exceeding 1 GHz [4]. This ensures a strong performance roadmap over time. The L1 program (L1P) cache is 16 Kbytes and is direct mapped. The L1 data (L1D) cache is also 16 Kbytes but is two-way set associative to account for multiple input sources. A second level unified data and instruction L2 memory 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 SRAM. Instructions, C variables, and random data accesses are typically cached. Regular data accesses such as streaming video input and output are typically done with EDMA requests and use mapped L2 SRAM.



Peripherals
EDMA Controller
L2 Cache/ Memory 256K Byte Total
L1P Cache Direct Mapped 16K Bytes Total
DM642 DSP Core
L1D Cache 2-Way Set Associative 16K Bytes Total

Figure1. Two-level cache architecture of DM642

3.3 The architecture of the decoder

The MPEG-4 decoder based on DM642 is formed by three parts--receive task, decode task and output task. DSP/BIOS Timers and Benchmarking Tools are used to take count of time consumed by decoding a frame. The EDMA-I/O transfers and buffers encoded video data from the network interface to SDRAM. DSP decodes and writes the video streams into the SDRAM through its cache hierarchy. The DSP reads the encoded data from and writes the decoded video into the SDRAM. Another EDMA-I/O transfers the decoded video from SDRAM to the display peripheral. The focus of this paper is on cache sub-system optimization for DSP running MPEG-4 video decoding algorithm only.

3.4 Cache design parameters

Various design parameters need to be considered while designing the cache sub-system. The system performance is evaluated in terms of the following cache parameters: cache sizes, associativity, and cache levels.

Cache size: The first most significant design parameter is cache size. Both L1 and L2 cache sizes are usually increased by factors of two (example: 16 K, 32 K, 64 K, etc). For MPEG-4 decoding, the cache-memory traffic is a function of cache size and increasing sizes show improvement, but for very large caches the improvement may not be significant. Cache memory has cost and space constraints, so the decision of how large a cache to implement in a system is critical [6].As for the DM642, the L1 program (L1P) cache is 16 Kbytes as well as the L1 data (L1D) cache. A second level unified data and instruction L2 memory is 256 Kbytes and allows large amounts of data to be brought on-chip.

Associativity: Better performance can be achieved by increasing the level of associativity of smaller caches. Changing from a direct-mapped cache to a 2-way set-associative may reduce memory traffic by as much as 50% for small caches. Set sizes of greater than 4, however, show minimal benefit across all cache sizes [6]. As for the DM642, the L1 program (L1P) cache is direct mapped. The L1 data (L1D) cache is two-way set associative to account for multiple input sources.

Cache levels: L2 cache between on-chip L1 and SDRAM may significantly reduce the CPU utilization and that may improve the overall performance. In general, addition of L2 decreases the bus traffic and memory latency [1]. The DM642 utilizes two-level cache architecture, namely L1P, L1D and L2 cache.

A number of articles have been published on cache optimization for DSP running multimedia applications. Simulation results show MPEG-4 decoding performance can be enhanced by cache optimization [3]. It is also observed that miss rates decrease considerably as L2 size is increased from 256 KB to 2 MB [3]. Similarly, the miss rates decrease significantly as the associativity level is increased from 2- to 4-way.

4. 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 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. One possible way to protect small but frequently reused data types from victimization by larger data objects, like video data, is to put them in SRAM. Data that are subject to 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 data exchanging section, composed of two data buffers, is to solve the video data storage and exchanging problems, the core code and variables storage section, used to store the frequently called functions such as IDCT and relevant variables, is for the frequent data management, and the rest of the SRAM is set as cache for the other codes and data management. Additionally, the code structure is properly modularized, the code fragments in teamwork are adjusted to be stored continuously, and the related data are stored together. These actions are to increase the hit rate of the cache in the SRAM.

4.1 The allocation of data exchanging section

The data exchanging section is primarily used to solve the access and exchange of video data during MPEG-4 video decoding process. The goal is to realize the system which can simultaneously decode eight channels CIF video frames on DM642, but the size of eight-channels CIF frames buffer is 8×352×288×1.5=1215KB. If the reference frame and the current frame are allocated to SRAM during decoding, the whole frame buffer will reach 2.4MB, in addition the source code needs approximately the 150KB storage space, the other variables need about 300KB storage section. Therefore the system altogether need approximately the 2.85MB storage space, which is bigger than the 256KB internal memory size by far which DM642 can provide. Therefore, if decoders take the frame as the unit like the PC decoding mode, all video frames must be laid in SDRAM, which will causes to exchange data frequently between On-chip memory and Off-chip memory during decoding. Because the access to external memory is much slower than the access to internal memory, which further inspires the contradiction between the processing speed of the processor core and the data access speed and causes the DSP to consume most of time when accessing to extern memory. Therefore it is critical to optimize the architecture for MPEG-4 decoding and allocate reasonable data exchanging section, which processes multiple blocks at a time to obtain optimum cache performance and EDMA bandwidth efficiency.

The unit of MPEG-4 decoding is a 16 × 16 pixel fragment known as a macro block, covering the area of six 8 × 8 blocks(for 4:2:0 subsampling), and many video processing algorithms are based on the macro-block/block .Therefore decoders take the GMBL（Group of Macro Block Lines）as the decoding unit. In order to reduce the access to SDRAM as far 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 ping-pang data buffer, namely BUF1 and BUF2, is used to store the decoded data in turn. Therefore decoders allocate almost 173KB size space of SRAM for data exchanging section, namely 20.25KB (the size of original GMBL) +98.25KB (the size of reference GMBL) +55KB (the size of structure variables) to CIF frame. Therefore, all the ways of storing decoded data may be divided into two kinds: the frame level and the GMBL level as shown in Fig.2. In the optimal memory allocation method, the frame buffer is located in SDRAM and the GMBL buffer is laid in SRAM.

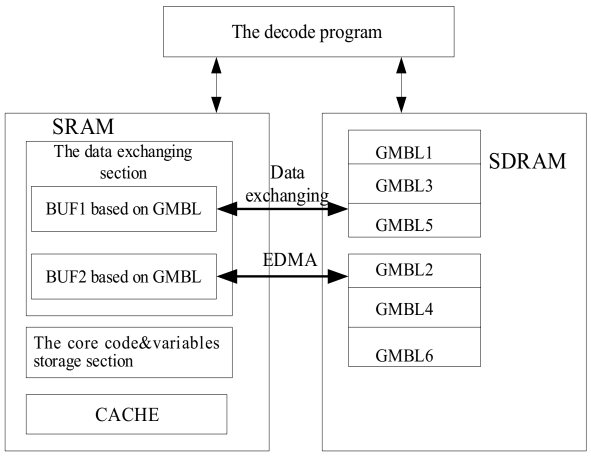


Figure2. The MPEG-4 decoder scheme based on the GMBL structure

4.2 The allocation of core code and variables storage section and Cache setting

The method that utilizes SRAM to save video data is to solve the video data storage and exchanging problems. However unless the data exchange can be realized in time, the performance of the decoder still may not be improved. Therefore, it’s necessary to allocate parts of SRAM for storing the relevant variables of code and program. The MPEG4 decoder is 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 such as IDCT, inverse quantization and so on. Thus, after subtracting data exchanging section, there is only left 83KB storage space in L2 RAM. In addition, there is a choice of cache setting for selecting the way of data scheduling from SDRAM. Similarly, the miss rates decrease significantly as the associativity level is increased from 2- to 4-way. In 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. For video algorithms, this mode is feasible because the direction of data streams is clear and the data exchange can be completed by scheduling EDMA. Yet, it will consume a lot of time once DSP accesses external memory.

Mode 2: The storage space of 83KB size SRAM is divided into two parts with one of 64KB size for cache, 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 83KB size SRAM is divided into two parts with one of 32KB size for cache, the surplus is for saving codes and data which are in common use. In this way, the access to SDRAM is possibly completed in a high speed taking advantage of the cache. In this mode, The L2 cache is two-way set associative.

Compared with mode 1, both mode2 and 3 has a higher miss rates when CPU read block which could waste lots of time because of the impossibility of manual scheduler resulted from the invisible map program. In addition, the miss rates decrease significantly as the associativity level is increased from 2- to 4-way. The DSP/BIOS statistical results are shown in table 1.

Table 1: the instructions cycles of decoding a frame

Standard video streams	256KB Cache	64KB Cache	32KB Cache	No Cache	Improved percentage
Foreman	3670784	2756759	2980676	2841185	24.9%
Akiyo	2948577	2252713	2394244	2267456	23.6%
Tempete	3878023	2900761	3117930	2970566	25.2%

After the performance comparison of three modes above, the second mode is finally the optimum one and is used in this paper which is called Cache-based memory mode. It separates the 64KB SRAM into L2 levels and surplus 19KB SRAM is used for store core codes and variables .Other codes and data which is scheduled by L2 cache controller is stored in SDRAM, just as is shown in Fig.2.

Additionally, to have a higher performance, the code structure should be properly modularized, the code fragments in teamwork are adjusted to be stored continuously, and the related data are stored together. These actions are to increase the hit rate of the cache in the SRAM.

5. Conclusions

Cache memories have strong influence on system performance and are used to fill the processor-memory speed gap. In this paper, a cache-based memory allocation mode is proposed to optimally allocate the limited on–chip memory of TMS320DM642 according to its hardware characteristics and the software characteristics of MPEG-4 video decoders. With the cache sub-system optimization and the latter code optimization, the obtained MPEG-4 decoders can simultaneously decode eight channels CIF video frames on DM642. The experiment results show that MPEG-4 decoding performance can be improved by almost 25% compared to the no cache optimization, with the standard video streams.

Cache-oriented architectural enhancements like configuring the reasonable size of L1 on-chip memory, optimizing MPEG-4 decoder procedure, and prefetching video data can further improve the performance of the MPEG-4 decoders. These techniques will be investigated in the future.

Acknowledgment

This work was partly supported by the Natural Science Foundation of Hubei Province (No. 2005ABA250) and the Creative Group Project of the Natural Science Foundation of Hubei Province (No. 2005ABC005).

The authors would like to thank Xia Xiaojian, Chen Boming, and Zhao Mian for their essential suggestions, feedback, and technical information. In addition, the authors thank the reviewers for their helpful comments and constructive criticism.

References

[1] Asaduzzaman A., Mahgoub I., Kalva H. et al. Cache Optimization for Mobile Devices Running Multimedia Applications, In: Proceedings of the Sixth IEEE International Symposium on Multimedia Software Engineering (ISMSE 2004), 2004, 499–506

[2] Chase J. G. and Pretty C, Efficient Algorithms for MPEG-4 Video Decoding, University of Canter-bury, New Zealand, TechOnLine Publication Date: Dec. 17, 2002

[3] Asaduzzaman A. and Mahgoub I. Evaluation of Application-Specific Multiprocessor Mobile System, Proceedings of the 2004 Symposium on Performance Evaluation of Computer Telecommunication Systems (SPECTS 2004), 2004, 751–758

[4] Golston, J. *DM642 digital media processor*, Proceedings of SPIE - The International Society for Optical Engineering, v 5022 II, 2003, 700-706

[5] Molnos A. M., Heijligers M. J. M., Cotofana S. D., *et al. Data Cache Optimization in Multimedia Applications*, Proceedings of the 14th Annual Workshop on Circuits, Systems and Signal Processing, ProRISC 2003, 529–532

[6] Soderquist P. and Leeser M. *Optimizing the Data Cache Performance of a Software MPEG-2 Video Decoder*, ACM Multimedia 97—Electronic Proceedings, 1997

[7] Kulkarni C., Catthoor F., DeMan H. *Hardware cache optimization for parallel multimedia applications*, Proceedings of the 4th International Euro-Par Conference on Parallel Processing table of contents, 1998, 923–932

[8] Jeremiah G., Satish A., Ratna R., *Optimized Video Decoder Architecture for TMS320C64x DSP Generation* Proceedings of SPIE The International Society for Optical Engineering, 2003, 719-726