Bit Rate Buffer Control and Optimization of Embedded Video Encoder

Yao Chunlian[1,2,*], Liu Wen[1], Wu hongli[3], Mao Dianhui[1] and Liu Li[1]

[1]School of computer and information engineering, Beijing Technology and Business University, Beijing, China; [2]The Key Laboratory of Advanced Information Science and Network Technology of Beijing, Beijing Jiaotong University, Beijing; [3]Institute of advanced information technology of Beijing, Beijing

Abstract: To satisfy various requirement of embedded video encoder, an embedded video coding system based on TMS320C64xx DSP is designed in this paper. TMS320C64xx DSP (Digital signal processor) is the core of the coding hardware system, and FPGA (Field program gate array) as the co-processor of DSP, which can convert video data into the specified format. For real-time application, we optimize the encoder at three levels: Firstly, at algorithm level, develop some fast algorithm fitting for DSP; secondly, at system level, adjust the architecture of encoder and optimization data transfer with EDMA (Enhance Direct Memory Access); thirdly, at code level, we use linear assembly rewrite key code. In order to avoid overflow and under flow state of bit rate, software FIFO (First Input First Output) is designed, and by the status of FIFO, we can know the state of underflow and overflow of buffer. Experimental results show that the embedded video encoder can compress four channel CIF (352×288) video data real-time.

Keywords: Bit rate control, DSP, optimization, video.

1. INTRODUCTION

With the technology development of network and digital signal processing, more and more people pay much attention to digital image processing and communication technology.

Video signal has the character of large amount of data and can't be stored and transferred directly, so it is very important to study high efficiency multi-media data encoder technology. In recent years, many solutions based on programmable devices, such as video Asics, which has the advantages of small size and low power and the disadvantage of not easy to expand. FPGA has the advantages of small size, low power and high adaptability and disadvantages of the high cost of development. With the advantage of flexibility, expansibility, portability, upgradeable and controllable, General DSP (Digital signal processor) is very suitable for video processing [1-3].

However, the high complex of video encoder and the limitation of chip resource is challenge for the real time compression processing. Therefore, it becomes the critical problem to optimize hardware resource of DSP and software architecture of video encoder to make full use of its benefits [4-7].

At present, there are several video coding standards, such as MPEG-x series standards made by ISO and H.26x series standard made by ITU, and the newest HEVC standard. And H.x series is mainly for transmission application, and Mpeg series is mainly for storage video application. In the embedded video encode system, MPEG-x standards are adopted frequently.

This paper put forward an embedded multi-channel real-time video coding system based on DSP (TMS320C64xx). The system can not only work in real-time but also get high compression ratio, favorable algorithm flexibility and wide application field. With several kinds of optimizing technique, such as encoder algorithm, chip resource, transfer mode and multi-task switch, the system can achieve real-time compression of four channel CIF video data (the resolution is 352×288).

2. VIDEO ENCODER HARDWARE ARCHITECTURE

The hardware system architecture of video encoder is shown in Fig. (**1**), and it consists of two parts, one is acquisition part and the other is encoder part.

Acquisition part: A/D is an analogy/digital chip, it can convert analogy video to digital signal, and FPGA can collect digital video signal from A/D chip, and then transform into the specify form, such as 4:2:0 sample video format, and data transferred to FIFO (First input First output)chip. And FIFO is a memory chip.

The core part of the encoder is TMS320C6416 DSP of TI Corp, it can get data from FIFO and coding video data, and then compression bitrates is stored into SDRAM. TMS320C6416 DSP is a general TI TMS320C6000 DSP and adopts VLIW (Very Long Instruction Word) architecture. TMS320C6416 DSP has a single Instruction stream operation code; the structure of the multiple data within an Instruction cycle can parallel processing Instruction. The bus has a 256 bits program data bus, two 32 bits data bus. Working frequency is 1 GMHZ, with 8M bits on chip storage, integrated with multi-channel synchronous serial port

MCBSP, provides 64 EDMA channel, more than 2 gb/s. SDRAM(Synchronous dynamic random access memory) is a memory chip, it is used to store the video data.

Compressed stream is transferred by the McBSP(Multi channel buffer serial port) of DSP.

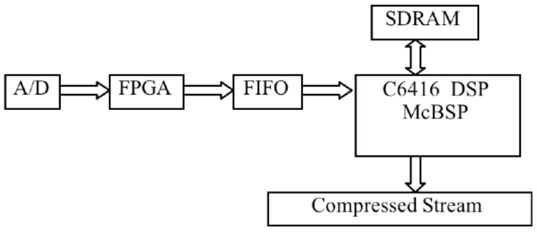

Fig. (1). The hardware system architecture of video encoder.

The hardware design scheme has two advantages: first, good scalability. FPGA chip is not only for the data module Input/output function, but also as a co processor to finish entropy coding, data format transform functions; general DSP as the core processing unit, with the upgrade performance of flexible, easy to upgrade the algorithm; flexible video compression format control, real-time codec requirements.

3. VIDEO ENCODER SOFTWARE OPTIMIZATION

Although the embedded hardware platforms provide sufficient process resource, the embedded platform does not yield good performance. So the software optimization is necessary [8]. In the system, we adopt H.264 standard to encode video [9]. To improve efficiency, high efficiency algorithms have been put forward: Smooth Motion Vector Field Adaptive Search Technique (SMVFAST) [10], DSP-based efficient quantization computation method (EQ4DSP) [11], and Fast Sub-pixel Motion Estimation* (FSME*) [12].

In order to take full advantage of the hardware architecture of the video platform, some system level optimization methods are used, which is described in paper [11], such as changing structure of encoder, using CCS(code compile studio) compiler tools, etc.

Above-mentioned techniques enhance efficiency and improve the performance of the encoder. After implementing algorithm and system level optimization, the performance of the system enhanced remarkably. But it still can't meet the real-time coding requirement; we need to optimize the system further. Optimize the memory resource, with DMA (Direct memory access) channel transferring to improve the speed of transfer data.

3.1. Resource Optimization

In the system, there are three level store architecture (Cache->internal RAM->external SDRAM), and their speed are fast to slow. Because the access speed of DSP to SDRAM is very slow, if DSP take most of its time in accessing external memory, it can't develop its advantage in data processing. We need assign memory resource reasonably.

The two level cache architecture of DSP is used to overcome the bottleneck between high speed CPU and low speed SDRAM. The improvement of the Cache efficiency influences the performance of the whole system greatly.

Key code and data is stored in cache, thus can improve the processing performance of DSP; we optimize the system as follows:

1) Implement GMBL

GMBL (Group of Macro-Block (MB) Lines, which consists of several adjacent MB lines as an encode unit) encoder strategy. The store architecture of encode data is classified as frame level and GMBL level, as shown in Fig. (**2**). Frame level data is stored in external memory and GMBL level data is stored in internal RAM (Random access memory). Thereby, original video data, reference image, midway computation result and all other data needed by coding one GMBL are stored in internal RAM.

The coding system can take full advantage of high-speed internal RAM, reduces the data exchange, and improve the encode efficiency ultimately.

2) Use the memory mode based on Cache

Internal RAM is separated into two-level cache. L2 cache is responsible for transfer program code and data that are stored in SDRAM.L1 cache is responsible for the key code and data used by DSP core.

3) Fixed memory space in internal RAM

is used to store key computational code, such as DCT, IDCT, quantization, IQ, image interpolation, sad calculation, etc. The core code is stored in internal RAM, which can avoid the transfer of code, and reduce the transfer time of DSP.

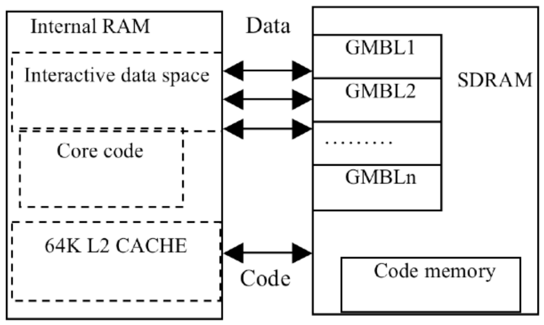

Fig (2). GMBL strategy.

4) Store co-operation code in continuous address and assign the storage space reasonably

In this way, the chance of cache hit can be increased. To avoid cache miss, we further divide big program into submodule, or combine small module into medium module.

Table 1. Cache resource optimization result.

Items	Basket	basket
Cache optimization	No	Yes
code optimization	No	Yes
Clock cycle	934091538	17045757

Table **1** shows the result of resource optimization strategy. When coding 25 frames per second data, resolution is

352×288, the execution efficiency of the cache resource optimization strategy are 54.8 multiple of that of without cache resource optimization.

EDMA transfer optimization:

EDMA of DSP can provide 2Gb/s data transfer bandwidth, and can support 64 channels event transfer and linking/chaining transfer mode. Linking/chaining transfer mode needs 85 configuration parameters. When one event is trigged, link mode permits a serial transfer. When the current channel transfer is finished, the chain mode permits another channel trigged to transfer data. After initializing by CPU, the two transfer mode can transfer data automatically and continuously. During the process of coding, we need interactive data frequently, such as transfer acquisition data and reference data into internal RAM and output reconstruct data to external memory. It is very time-consuming to transfer data by CPU. However, EDMA can implement transfer data in different space without the intervention of CPU.

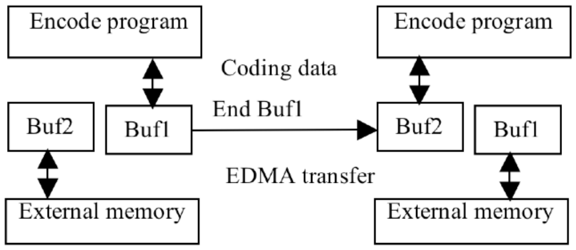

Fig. (3). Dual-buffer architecture.

In order to implement parallel processing, dual-buffer architecture is designed in this paper, which can be shown in Fig. (**3**). We divide internal RAM into two blocks for buffering data. Buf1 and buf2 is used as buffer for input of a line image data, output data and midway computation result. The bing-bang architecture enhances the high-speed process ability of CPU, which makes it possible for CPU and EDMA controller operate the two different buffers at any time. Therefore, it avoids the access conflict and promotes achievement of parallel operation. Test result proves that by using of dual-buffer architecture and chaining function of EDMA, it takes 0.19ms for transferring one frame data with resolution of 352×288. Instead, it will take 10.52ms without using GMBL. Thus data interactive efficiency improves greatly.

3.2. Task Priority Optimization Strategy

In this system, we need implement video encoder, one channel has two tasks, one is input, and the other is coding. The two tasks need one synchronization semaphore. As shown in Fig. (**4**). Input task is composed of three sub-task, initialization and acquisition image and format conversion, and then FPGA chip acquire one frame image and finish the format conversion from 4:2:2 to 4:2:0. The three sub-task finish in sequence. Coding task composes of three functions, initialization, coding image data, and compression data output to FIFO.

In order to utilize the limited resource of CPU reasonably and efficiently, we should arrange the priority of the task by its importance, and guarantee the most important task will gain response in advance. For example, the input task of en-

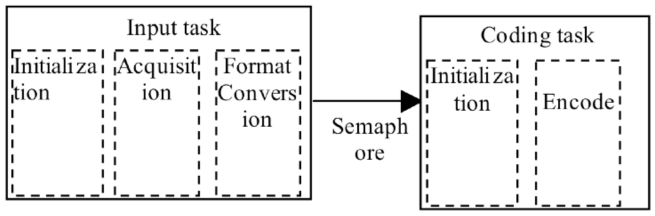

Fig. (4). Task flow.

coder system need receive acquisition data real time, if it has low priority, it will lose data or cause data error, which will seriously affect the performance of the system, and accordingly it must be endowed with high priority.

3.3. Code Level Optimization

By the optimization technology of 3.1-3.2, we need improve the code efficiency father. By profile analysis report of encoder, we find that 20% code modules take 80% run-time in video encoder, so we optimize these modules by using C-code and linear assembly level optimization.

C-Code Level Optimization: Main C-code optimization methods [13, 14] used in the system with the following parts.

1). The first optimization step that can be performed on the DSP platform is to use compiler options. The compiler options include speed / code /file level optimization. To get better performance, we need trade off the improvement of execution speed and incensement of code size.

2). Using intrinsic function to replace complicated C/C++ code and the packed data processing to maximize data throughput. The C6000 compilers support special functions called intrinsic operators, such as _add2, _mpy, _lddw, _mem4_const, etc. those intrinsic operators can be translated into efficient assembly code and can improve the execution speed.

3). Unrolling loop to increase software pipelines. Since the compiler only evaluates the possible loop optimization at the top of loops, we need expand small loops manually and unroll some inner loop.

4). Replace the multiply and division by the logic shift. Multiply can replace with logic left shift and add, division can replace with logic right shift and sub.

5). TI Corp provides two sets of assembly-optimized key functions for data processing, named IMGLIB (image/video processing library) and DSPLIB. By reasonably utilizing available resources and avoiding potential resource conflicts, each function in the two libraries is designed to produce the best performance. Like IMG_fdct_8x8 and IMG_idct_8x8.

Here is a sample of sad16 function with two loops:

Original C code

```
for (j = 0; j < 16; j++) {
  for (i = 0; i < 16; i++) {
   sad += abs(*(ptr_cur + i)- *(ptr_ref + i));
                          }
  ptr_cur += stride;
  ptr_ref += stride;
  }
```

Optimized C code

```
for (j = 0; j <16; j++) {
 sad+=_dotpu4(_subabs4(*ptr_cur,                    _mem4_const(ptr_ref)),
0x01010101);
 sad += _dotpu4 (_subabs4 (*(ptr_cur + 1), _mem4_const(ptr_ref + 4)),
0x01010101);
 sad += _dotpu4 (_subabs4 (*(ptr_cur + 2), _mem4_const(ptr_ref + 8)),
0x01010101);
 sad += _dotpu4 (_subabs4 (*(ptr_cur + 3), _mem4_const(ptr_ref + 12)),
0x01010101);
  ptr_cur += stride>>2;ptr_ref += stride;
  }
```

We optimize the code with unrolling loop and using intrinsic function, _dotpu4, _subabs4, _mem4_const.

6). Linear Assembly Level Optimization

If system can't still satisfy the real-time requirement, we must rewrite kernel C code in linear assembly. The compiler allows writing linear assembly code without being concerned with the pipeline structure or registers assignment.

The CCS compiler assigns registers and uses loop optimization in order to turn linear assembly into highly parallel assembly. The following is a linear assembly sample of sad16 function partly:

```
LDDW * A_srcImg, A_s7654:A_s3210
LDNDW * B_refImg, A_r7654:A_r3210
SUBABS4 A_s7654, A_r7654, A_d7654 SUBABS4 A_s3210, A_r3210, A_d3210
DOTPU4 A_d7654, A_k1, A_s1
DOTPU4 A_d3210, A_k1, A_s0
```

Table **2** lists some results of code optimizations, and fdct use the library of IMGLIB. Experimental results show code optimizations improve the performance notably; the executive cycles of some functions improve more than 100 multiples, such as interpolateH function, use linear assembly optimization strategy can get good result.

3.4. Bit Rate Overflow Control

In the system, multi-channel buffer serial port (McBSP) in DSP is used to send and receive compressed stream.

McBSP in the structure can be divided into a data channel and a control channel. Data channel to complete the data transmitting and receiving; task and control channel complete many tasks including internal clock, generating, frame synchronization signal and selection of multiple channel etc.

In order to control the compressed stream transferred into receiving system, in the paper, the bit-rate overflow control scheme is given in the following part.

The encoder use two buffers for stream output, one is macro-block row level buffer and the other is frame level buffer. Macro block level buffer zone is located in the RAM of DSP, the compressed bit stream for each macro block lines is stored, and it's size is changed with the different application scenarios; frame level buffer zone is located in the SDRAM chip, here, the compressed stream of each frame is stored, which is designed the type of the software FIFO (First in First out). The data compressed by DSP are transferred into the software FIFO.

DMA(Direct memory access)reads data from FIFO to McBSP, through the McBSP shift output. DMA can be set to automatically initialize channel mode, namely each transmission channel of DMA can repeatly transfer the data in FIFO, the DMA channel will be reinitialized, and channel source address registers the starting address for the FIFO. Because the DMA is automatically read the data, without the intervention of CPU, improve the efficiency of the program, but the difficulties in the implementation is to control the possible overflow of the FIFO.

In the coding process, by copying the stream a macro-block row level output buffer to the frame level buffer, here proposed frame level buffer overflow control method.

The first step is to avoid buffer overflow:

Before output each macro block stream to the frame level buffer, check whether the frame level buffer has enough space for the current macro-block stream, once the residual space of the buffer is insufficient, and the encoder will stop code, until there is enough space.

Method for avoiding buffer underflow:

In order to avoid underflow, there are two steps.

firstly given the min value of buffer(such as buffer 10%), after each macro block lines transfer stream to the frame level buffer, check size frame level buffer, if it is less than the lower limit of the buffer, a certain amount of zero byte is filled in the buffer. In the paper, the min value of buffer is determined by the maximum code time of macro-block row.

Secondly, if there was an underflow, it can be detected in time, and make the stream error is reduced to a minimum. When the underflow occurred, the formula (1) error will occur.

Fifo.length=(fifo.rear-fifo.front+fifo.size)%fifo.size (1)

Table 2. Comparison of run cycles on DSP.

Function	Original Code	Optimized c-Code	Linear Assembly	Multiple
transfer_16to8 sub	5523	1835	129	42.81
fdct	9342		220	42.46
sad16	8760	1851	132	66.36
interpolateH	16488	1757	131	125.86

Table 3. System performance of after optimizations.

Sequences	Non-Optimization (Frame/sec)	Optimized (Frame/sec)	Multiple
Basket	2.56	105.3	39.96
Flower	3.8	109.8	28.89
Coastguard	4.1	104.4	25.46
Mother	4.32	110.43	25.56

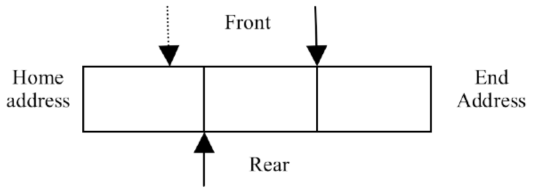

Fig. (5). Underflow of the FIFO.

Fig. (**5**) shows the buffer between before and after the two query state time underflow condition.

Rear points to a location invariant; front position is pointing to the query state last time, the solid line is the latest state.

Obviously, in formula (1) calculation of the buffer length is the maximum, and the actual situation of buffer has been underflow. In each query frame level buffer state, if detected the underflow, a certain amount of.zero byte is immediately filled. And then the fifo.rear is equal to fifo.front. These methods are based on the assumption that between the adjacent two query state time intervals, the change of queue length does not exceed half of the buffer zone.

4. PERFORMANCE TEST AND ANALYSIS

After adopting improved video coding algorithms, system, C-code level and linear assembly optimizations, we compare the encoder performance before and after optimization. The results are shown in Table **2**.

Four channel video signals (CIF resolutions) are captured and input into the system and processed in polling turn. The content of the input video will influence the compression efficiency slightly, so we make several experiments. We choose four CIF sequence, Basket is a moving fast video and Flower is a colourful video and Mother is a relative static video, Coastguard is mid-moving video.

As shown by the experimental results, software optimization implementations studied in Section 3 improve the performance of encoder notably. Our system can satisfy the real-time encoding requirement of four-channel (every channel is 25frame/sec).

CONCLUSION

In this paper, we design an embedded video coding hardware platform based on DSP and then give the efficient video bit rate buffer control method, and in order to improve the performance of the encoder, several optimization have been given, including the store resource, rewrite key code and transfer optimization with EDMA.

Experimental results identify that the system can encode can be fit for real-time application. The system can satisfy the requirements of various embedded multimedia applications, especially those with strict restriction in size, power consumption and multi-channel signal environment, such as surveillance applications.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This study was supported by the National Natural Science Foundation of China (No.61103124) and the Key Laboratory of Advanced Information Science and Network Technology of Beijing (No.XDXX1301).

REFERENCES

[1] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards - Including High Efficiency Video Coding (HEVC)", IEEE Trans. Circuits and Systems for Video Technology, Vol. 22, No. 12, pp. 1669-1684, Dec. 2012.

[2] Lappalainen V. *et al*.. Performance of H. 26L video encoder on general-purpose processor. In: Proceedings of the International Conference on Consumer Electronics, Los Angeles, 2001, 266-267

[3] A. Bahari, T. Arslan, and A. Erdogan, "Low-power hardware architecture for vb sme using pixel truncation," in Proc. 21st Int. Conf. VLSI Design, Hyderabad, Andhra Pradesh, 2008, pp. 389–394.

[4] Hsiu-Cheng Chang, Jia-Wei Chen, Bing-Tsung Wu, Ching-Lung Su, Jinn-Shyan Wang, and Jiun-In Guo, "A Dynamic Quality-Adjustable H. 264 Video Encoder for Power-Aware Video Applications, " IEEE Trans. Circuits Syst. Video Technol., vol. 19, no. 12 pp. 1739-1754, Dec. 2009.

[5] Zhao Bao-jun, *et al.* Implementation of real-time 2D-DCT with FPGA and DSP[J]. Acta Electronica Sinica, 2003, 31(9): 1317-1319. (in Chinese)

[6] Miyazaki T. *et al*.. Real-time software video encoder on multimedia RISC processor. In:Proceedings of the IEEE Workshop on Signal Processing Systems, Cambridge, Massachussetts, 1998, 33-42

[7] Li fanghui, Wang Fei, He Peikun. TMS320C6000 Series DSPs principle and application (second edition). Beijing electrical industry press. 2003

[8] XU Xiao-dong, XU Pei-xia. Optimization of video decoder system based on TMS320 DM642[J]. Journal of Data Acquisition & Processing, 2005, 20(1):91-95. (in Chinese)

[9] Zhong yuzhuo, Wang Qi, He yuwen. Multimedia data compression standard based on object Mpeg-4 and its verify model. Beijing science press. 2000.10

[10] Li Wei, Zhou B, Li B, A Fast Motion Estimation Algorithm Using Adaptive Motion Vector Field Search, Chinese Journal of Computers, 2003, 26(2): 168-173.
[11] Li Wei. Research of Video coding and Dsp-based implementation Ph.D.dissertation]. Beihang University, Beijing 2003
[12] Zhang Jinyin. Block Matching Sub-pixel Motion Estimation and Its Application [M.S.dissertation].Beihang University, Beijing 2004
[13] SPRU190. TMS320C6000 Peripherals Reference Guide[on line]. http://www.ti.com.
[14] SPRU198.pdf. TMS320C6000 programmer's guide[on line]. http://www.ti.com.

Received: September 16, 2014 Revised: December 23, 2014 Accepted: December 31, 2014