Embedded System for Video Coding with Logic-Enhanced DRAM and Configurable Processor

Toshiyuki KAYA†, Ryusuke MIYAMOTO††, Takao ONOYE†,††, and Isao SHIRAKAWA†

†Dept. Information Systems Eng.
Osaka University
2-1 Yamada-Oka, Suita,
Osaka, 565-0871 Japan

††Dept. Communications & Computer Eng.
Kyoto University
Yoshida-Honmachi, Sakyo-Ku,
Kyoto, 606-8501 Japan

Abstract— A novel approach of embedded systems for video coding is introduced with the main theme focused on logic-enhanced DRAM and configurable processor. This approach is aiming at reducing high computational costs and frequent memory accessing, which embedded systems are suffering with in the execution of video coding. According to the software execution analysis, large size functions with intensive memory accesses are tuned to be executed by the logic-enhanced DRAM while small size functions repeatedly called are to be executed by dedicated instructions, which are newly introduced in the configurable processor. The proposed system can speed up H.263 video coding algorithm 7.4 times in comparison with the conventional embedded processor based system.

1 Introduction

Video coding algorithms incur considerable amount of computations, and hence generally dedicated hardwares have been used to execute these algorithms through more than the past decade. With the recent progress of semiconductor technology, high performance processors are now capable of executing video coding in realtime.

On the other hand, low power consumption and low hardware cost are indispensable to an embedded system for mobile application. For such kind of systems, high performance processors can not be employed, and hence ASICs are mainly used to attain high cost performance.

In both of these situations, efficient use of memory accessing bus is of the most important issue in order to process huge image frame. A number of approaches to accelerate memory accessing speed has been proposed. While RDRAM[1] by Rambus Inc. attempts to attain high maximum transmission speed by access interleaving technique, VC-SDRAM[2] does to raise average transmission speed by placing high speed large register files called *"virtual channel"* between memory cell and I/O interface.

Also, as another approach to execute video processing efficiently, functional memory architecture is proposed[3]. This architecture equips functional unit on the inside of memory cell and executes highly parallel computation with avoiding data transfer between processor and memory. However, due to the difference of process technology for logics and memories, this architecture usually suffers from low integration efficiency and low accessing speed.

Motivated by this, the present paper proposes a novel architecture of embedded systems dedicated to video coding, which mainly consists of logic-enhanced DRAM and configurable processor. Referring to the software execution analysis, large size functions with intensive memory accesses are executed in the logic-enhanced DRAM while small size functions ubiquitously appear are executed with dedicated instructions, which are newly introduced in the configurable processor. H.263[4] video coding algorithm is used as a benchmark of performance measurement. The proposed system executes the algorithm 7.4 times as fast as the normal embedded processor based system.

2 System Architecture

2.1 Software Analysis

A number of hybrid video coding algorithms based on motion compensation (MC) and discrete cosine transform (DCT), such as H.263 and MPEG-4, requires intensive computation, which causes system performance degradation and high power consumption in the cases being executed on the embedded processor-memory system.

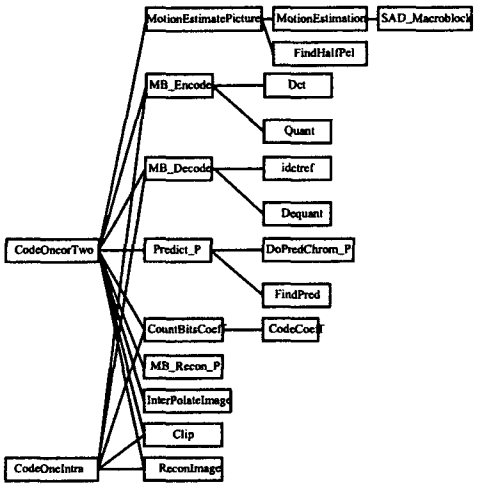

Figure 1: Caller-callee relation of H.263 algorithm.

First, we analyzed the number of cycles needed to execute the H.263 video coding by using an embedded processor and its instruction set simulator. Figure 1 and Table 1 show the caller-callee relation of the algorithm and simulation results. According to the simulation results, the ratio of sub-processes of H.263 coding to total process of that were found to be 44.06% for integer-pel motion estimation (ME), 23.70% for DCT

15.48% for inverse DCT (IDCT), 3.90% for quantization, 2.70% for half-pel ME, and 10.16% for the other sub-processes, respectively.

Table 1: Software analysis by instruction set simulator.

Function Name	#Cycles	#Called
CodeOneOrTwo	1,157,878,407	10
MotionEstimatePicture	579,777,572	10
MotionEstimation	538,315,074	990
SAD_Macroblock	441,639,674	775,380
FindHalfPel	32,980,159	985
CodeOneIntra	63,809,604	1
MB_Encode	343,265,117	1,089
Dct	289,479,768	6,534
Quant	47,598,542	6,534
MB_Decode	199,517,965	796
idctref	189,114,447	4,776
Dequant	5,550,868	4,776
Predict_P	22,692,378	985
DoPredChrom_P	6,406,993	985
FindPred	9,156,728	985
CountBitsCoeff	22,841,658	796
CodeCoeff	22,274,554	2,205
MB_Recon_P	10,425,330	990
InterpolateImage	9,052,086	10
Clip	5,641,019	1,089
ReconImage	13,625,044	1,089

3 Organization

Based on this simulation result, we have constructed an embedded system providing an efficient video coding performance. As is shown in Fig. 2, the system consists of an embedded RISC processor with extended instruction set, a DRAM enhanced with a video-processing unit (VPU) for the video coding, and memory interface specialized to handle these modules. Details of these module are described in the following sections.

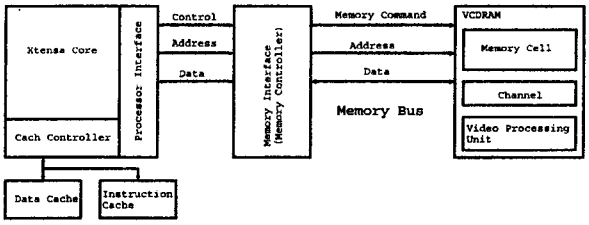

Figure 2: Overall block diagram of the embedded system for video coding.

4 Acceleration by Configurable Processor

In this system, the Xtensa[5] configurable processor core by Tensilica Inc. is employed, which is designed to easily match specific application requirements; i.e. designers can not only flexibly configure its base architecture, but also add new instructions to the original instruction set by using Tensilica Instruction Extension (TIE) language. To speed up quantization and half-pel ME, which are repeatedly called small size functions, we newly introduced 7 instructions (6 for quantization and 1 for half-pel ME), which are illustrated in Figs. 3 and 4, respectively. The number of gates for additional hardwares are 900.4 gates and 270.6 gates, respectively.

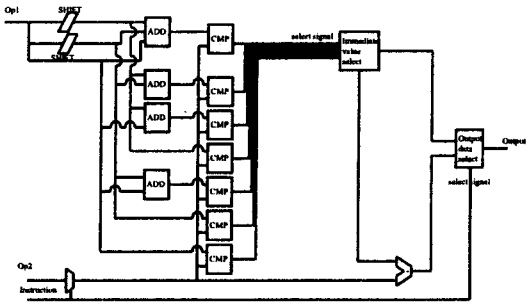

Figure 3: Additional instruction for quantization.

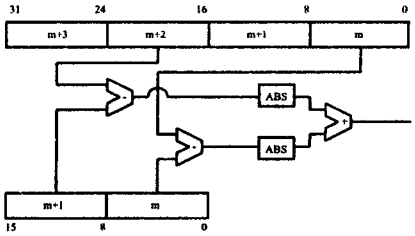

Figure 4: Additional instruction for half-pel ME.

5 Logic-enhanced DRAM Acceleration

Our logic-enhanced memory architecture, which accelerates integer-pel ME, DCT, and IDCT, is based on the virtual channel DRAM (VCDRAM) by Elpida Memory Inc. The VCDRAM provides efficient data throughput for multiple memory masters by accessing through the *virtual channels*, which equips a set of 16 high-speed registers between memory cell array and I/O buffer.

Taking advantage of ability for multi-tasking access to the channels, the newly introduced VPU can operate in parallel with other memory accesses, as illustrated in Fig. 5. Executive instructions of the VPU issued by the processor are converted into some memory commands in the memory interface and input through the command and address pins of VCDRAM, as is the case with other conventional VCDRAM commands. Once an instruction is given, the VPU operates it with the data stored in the channels until it finishes without any other commands. Therefore, intensive memory accessing to load large amount of data to the processor are taken away, and hence results in the reduction of the total cost of the video coding. In addition to this, since the processor can issue other load/store instructions even though the VPU is running, the performance overhead on the processor for the video coding by the VPU left quite small.

The integer-pel ME block of the VPU estimates a "reference macroblock" similar to the "current macroblock", which is now going to be encoded, from the

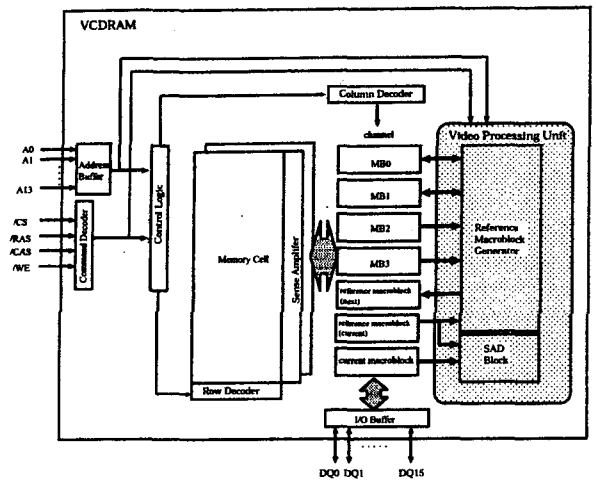

Figure 5: VCDRAM with the video processing unit (VPU).

previous frame, where each macroblock is with 16 × 16 pixels.

This ME block are mainly concerned with two processes, namely, a calculation of a sum of absolute differences (SAD) between the two macroblocks stored in channels, and generation of a reference macroblock. These two processes are executed in parallel so as to attain high throughput. The size of memory channel is compatible with that of a macroblock. Thus we use totally 7 channels for ME, 4 channels for search area, 1 for current macroblock, and 2 for generation of a reference macroblock. Figures 6 and 7 illustrate organization when executing ME and architecture of SAD functional unit, respectively. Once a command is input from the memory interface, the ME block conducts integer-pel ME for a quarter of the search area, i.e. to complete an ME process for one macroblock, 4 command inputs are needed.

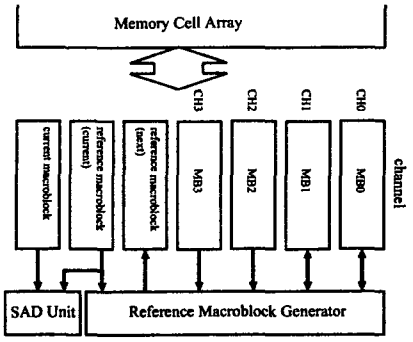

Figure 6: Motion estimation by logic-enhanced VCSDRAM.

DCT incurs many of multiplications since it is generally based on a matrix calculation for pixel data. Therefore the VPU carry out multiplications using a channel as a table containing sub-products, based on so-called *distributed arithmetic*. In this way, IDCT can be realized by replacing the contents of the table from those of DCT, and thus as a result the quantity of the hardware can be saved. It should be noticed here that a channel can also be used as transposition memory between two 1-D DCT/IDCTs. Block diagram of the DCT unit is depicted in Fig. 8.

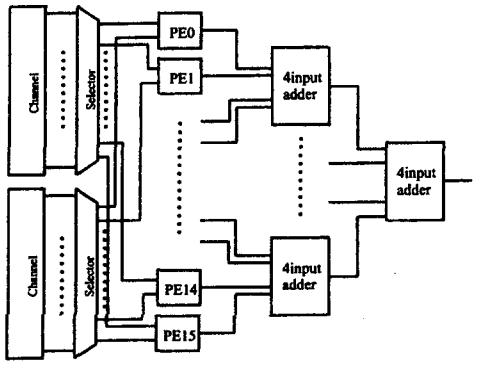

Figure 7: SAD functional unit.

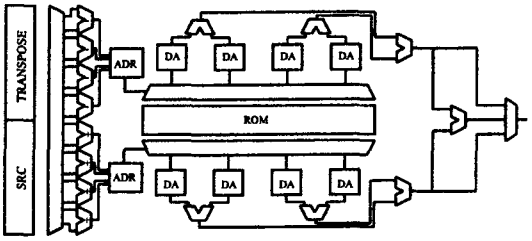

Figure 8: DCT functional unit.

With these units, integer-pel ME for one macroblock spends 15,376 cycles, and DCT/IDCT for one block (8 × 8 pixels) are conducted in 206 cycles. The VPU can operate at 133MHz, and the ME block occupies 7,731 gates and the DCT/IDCT block does 6,545 gates by 0.18 μm CMOS technology.

6 Memory I/F

The dedicated memory interface bridges the gap of the interface between the Xtensa and logic-enhanced VCDRAM, and provides *non-blocked* access from one to another. The Xtensa communicates with external modules through 9-bit control and 32-bit memory address signals all the time, thus instructions for the VPU must be issued in the same way as conventional load/store requests.

As its organization is illustrated in Fig. 9, the memory interface decodes the instruction and generates VCDRAM's commands referring to the states of the memory and the VPU. During the execution of video coding instruction, the memory interface observes the VPU constantly, and reports the completion of the computation to the processor through an interrupt signal. This interface also manages the virtual channels, which play a role of *cache* laid in the memory module. However, some of them have to be locked not to be accessed,

when they are used by the VPU. Thus the memory interface adopts a fully associative cache whose block can be flexibly replaced.

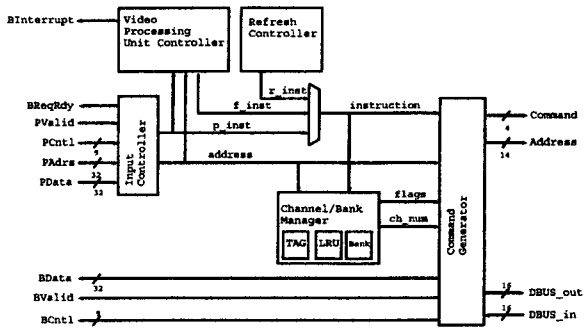

Figure 9: Memory interface block diagram.

The throughput of the memory interface to handle a processor request, i.e. the cycle count needed for accepting another one, depends on the request category and the status of the VCDRAM.

A 32-bit write request needs [2 + channel_miss_latency] cycles, and an n-bit read request needs [$n/16$ + 6 + channel_miss_latency] cycles, where n indicates the cache line width of the Xtensa configured either of 128, 256, or 512 bits. "channel_miss_latency" is the cycle count required for a channel miss transaction, which costs at least 2 cycles and in the worst case 18 cycles.

As for video coding instructions, an integer-pel ME instruction for 1 macroblock needs [110 + 8*restore_latency + execute_time] cycles, a DCT or an IDCT for 1 block does [29 + 4*restore_latency + execute_time] cycles. In these equations, "restore_latency" indicates the time required for writing back the contents of the channel to the memory cell if they have been changed, and equals to 11 cycles. Although "execute_time" is the time that the instruction is actually executed on the VPU, there is no need to include it in the overhead on the memory interface since the interface can accept conventional memory accesses for unlocked channel during this period.

As a result overheads on the memory interface is summarized in Table 2. This interface module can operate at 133MHz, and the number of gates is 10,888 gates by 0.18 μm CMOS technology.

Table 2: Overhead on memory interface.

Instruction	Best case (cycles)	Worst case (cycles)
32-bit write	2	20
n-bit read	$n/16+6$	$n/16+24$
integer-pel ME	110	198
DCT or IDCT	29	73

7 Performance Evaluation

Table 3 shows required cycle counts for H.263 video coding in the case of the proposed system and the conventional system, where the numbers of cycles for encoding 11 frames are indicated. With the proposed system, the cycle count required for integer-pel ME, DCT, IDCT, quantization, and half-pel ME are reduced to 7.56%, 1.26%, 1.41%, 13.25% and 53.04%, respectively, compared to the conventional system. Consequently, the total computational load for the H.263 video coding is reduced to 13.5%, which can be executed in an embedded applications.

Table 3: Cycles counts for H.263 video coding.

	Conventional (k cycles)	Proposed (k cycles)
ME	546,798	41,315
DCT	289,479	3,646
IDCT	189,114	2,665
Q	47,598	6,307
HalfPel	32,980	17,492
MC	31,744	31,744
VLC	22,841	22,841
IQ	5,551	5,551
others	29,691	29,691
Total	1,195,799	161,252

8 Conclusion

We have proposed a novel embedded system for video coding by using logic-enhanced DRAM and configurable processor. This approach is reducs high computational costs and frequent memory accessing, which embedded systems are suffering with in the execution of video coding. Based on the software execution analysis, large size functions with intensive memory accesses are tuned to be executed by the logic-enhanced DRAM while small size functions repeatedly called are to be executed by dedicated instructions, which are newly introduced in the configurable processor. The proposed system can successsfully accelerate H.263 video coding algorithm 7.4 times in comparison with the conventional embedded processor based system.

References

[1] Rambus Inc.: *http://www.rambus.com.*

[2] Y. Yabe, N. Nakamura, Y. Aimoto, M. Motomura, Y. Matsui, and Y. Asakura: "A next generation channeled-DRAM architecture with direct background-operation and delayed channel-replacement techniques," *Symp. VLSI Circuits Digest of Tech. Papers*, 2000.

[3] K. Tamaru: "The trend of functional memory development." *IEICE Trans. Electronics*, vol. E76-C, no. 11, pp. 1545-1554, 1993.

[4] ITU-T Rec. H.263: "Video coding for low bitrate communication," International Standard, 1998.

[5] Tensilica Inc.: *http://www.tensilica.com.*