ISCAS 2000 - IEEE International Symposium on Circuits and Systems, May 28-31, 2000, Geneva, Switzerland

Performance Analysis and Architecture Evaluation of MPEG-4 Video Codec System

Hao-Chieh Chang, Liang-Gee Chen, Mei-Yun Hsu and Yung-Chi Chang
DSP/IC Design Lab, Department of Electrical Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

Abstract

This paper presents various analyses of computational behavior, namely the number of datapath operations and memory access, on the core profile level 2 (CPL2) of MPEG-4 Video standard. These analyzed data exploit the load distribution and mode selection of the video system. The exploration of data-flow behavior and its derived computation of MPGE-4 video processing algorithms will then drive through an efficient architecture design.

I. Introduction

In the recent years, visual communication becomes very popular and has attracted much attention of researchers as well as customers. Many advanced digital video standards, such as H.263 [1], MPEG-1 [2], MPEG-2 [3], make visual communication become more practical. Lately, rapid evolution of digital multimedia technology directs the multimedia communication service to provide more flexible and powerful functions. MPEG-4 [4] standard is undoubtedly the emerging standard for such multimedia communication trend. In comparison of MPEG-1 and MPEG-2 standards, MPEG-4 standard video part (MPEG-4 Video) is not only to provide a high efficient video coding scheme for transmission but also to represent more active multimedia data with the capability of content-based interactive, universal access and error robustness.

In MPEG-4 Video standard, five different profiles are currently defined to support the coding of various video formats from QCIF to large size video (1920×1088) at a transmission rate ranging from several kbits/s up to 38.4 Mbits/s. Different algorithms and coding modes can be applied in different applications. This implies that high degree of flexibility and more computation power will be demanded for delivering MPEG-4 video. It could demand more processing capability than that today's processors can provide for real-time CPL2 applications or higher specification. A powerful computing engine with proper flexibility is thus required for those applications. Hardwire acceleration (coprocessor) provides a good solution to achieve better computing capability for video applications since data parallelism can be employed. In addition to applying the data parallelism technique, task-pipelining technique can be used by adopting the multithread architecture [5].

This paper will discuss the computational behavior of MPEG-4 Video codec from the system point of view. Computational requirements and behavior of data transfer for a CPL2 MPEG-4 Video codec are analyzed first. Based on the analyzed results, design decisions for efficient codec architecture are derived in order to implement an efficient system architecture for MPEG-4 Video coding.

II. MPEG-4 Video Codec.

MPEG-4 Video employs a tool-based coding scheme that can support various visual communication systems by utilizing different profiles and levels [6]. Such coding scheme allows MPEG-4 a better adaptation to applications of different source data nature and any possible channel environments. Besides, audio/visual contents from different sources can be coded with different scenarios (profiles), quality and dimensionality. This content-based concept allows scene producers to create high flexible composed video scenes and allows users a high degree of interactivity with video contents. Each video object (VO) in a video scene can be divided into many time instances, called video object planes (VOPs). The individual VOP is allowed to have arbitrary shapes. As a consequence, the shape of an object has to be transmitted in addition to its texture. Figure 1 and 2 show the general video encoder and decoder structure for MPEG-4 separately. It is observed that the shape coding tool is provided in this structure beyond the conventional hybrid coding scheme.

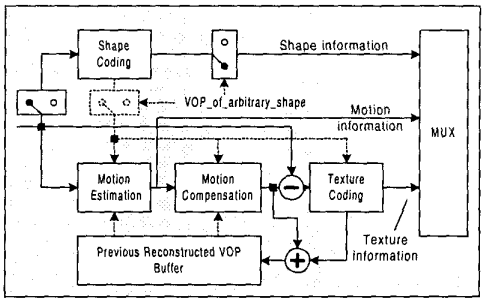

Figure 1. General structure of MPEG-4 Video Encoder

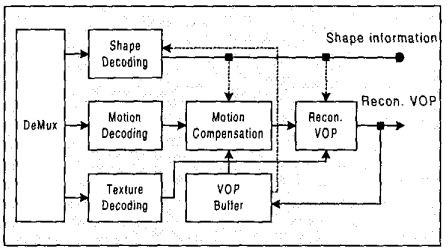

Figure 2. General structure of MPEG-4 Video Decoder

Due to the consideration of regularity, the implementation of MPEG-4 video Codec usually adopts macroblock (MB)-based coding scheme to support the coding of both rectangular video frames and arbitrarily shaped video objects. It is quite reasonable to employ the MB-based approach for conventional video frame coding. However, for the coding of arbitrarily shaped video objects, special processing is required to fit the MB-based approach. In the following, the essential processing for coding arbitrarily shaped video objects are briefly described.

A. Shape Coding

MPEG-4 video adopts the MB-based content-based arithmetic encoding (CAE, [7]) method and MB-based motion estimation to encode the binary shape information of video objects. For coding the shape of I-VOP, intra-mode CAE is applied for all the boundary blocks. While coding the shapes of P-VOP or B-VOP, however, before applying the inter-mode CAE, motion estimation for binary shape is performed to find the best matched macroblock such that inter-CAE could achieve higher compression performance.

B. Motion Estimation/Compensation & Texture Coding

0-7803-5482-6/99/$10.00 ©2000 IEEE

MPEG-4 video adopts a standard 8 × 8 or 16 × 16 pixels MB-based motion estimation and compensation algorithm to perform the temporal prediction coding for inter-frame video sequences. Moreover, several advanced prediction techniques, such as advanced prediction (AP) mode, overlapped block motion compensation (OBMC), unrestricted motion vector (UMV), can be optionally activated. Then, motion vectors and prediction errors are coded by VLC code. For arbitrarily shaped video objects, motion-repetitive padding is applied prior to ME for arbitrarily shaped objects.

The texture information of intra-frames and prediction error of inter-frames are coded using the conventional 8 × 8 block-based DCT. In addition to the intra prediction of DC coefficients, advanced intra block prediction mode (DC/AC prediction) can be applied to remove the remaining redundancy between the DCT coefficients of neighbored blocks.

III. Performance and Complexity Analysis of MPEG-4 Video Codec.

MPEG-4 video codec involves complex data-intensive algorithms. Namely, a very large amount of data will be processed with some similar operations. For this, datapath computation capability and efficiency of data transfer will dominate the system performance of MPEG-4 video codec. In order to understand its computational behavior, we setup a computational model such that computation behavior of MPEG-4 Video codec can be modeled and analyzed. Besides, computer profiling is also employed as a reference for our behavior analysis. This run-time profiling has been performed on the MoMuSys Video FDIS Version 1.0 C implementation [8]. The analyzed results and discussions are describes as follows:

A. Behavior Model

In order to apply the model-based analysis on MPEG-4 video codec, a reference processor model has to be setup.

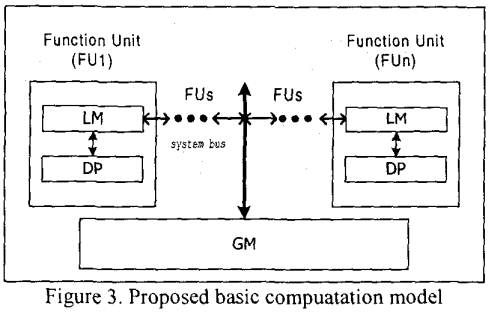

Figure 3. Proposed basic compuatation model

As shown in Figure 3, basically it is a simple computation model composed of several function units, each which comprises a data processor (DP) and a memory module (local memory, LM and global memory, GM). The computational behavior can be modeled as memory access (data transfer between GM and LM) and data processing. Based on this model, we can divide the MPEG-4 video codec into several processing units, each that can deal with one major task of MPEG-4 video coding, such as DCT/IDCT, VLC/VLD, MC, ME...etc. After such a virtual machine has been setup, the computational behavior, i.e., the number of data-path operations, memory access and memory-addressing operations can be calculated by directly analyzing the computing flow of algorithms.

Let us take the inverse quantization (IQ) tool for an example to show how to perform the proposed computational analysis. Figure 4 shows the IQ process adopted in MPEG-4. In the first step of IQ process, the Inverse Quantization Arithmetic, two different quantization methods can be used. Here the MPEG-like method is analyzed. The following shows the formulas for the IQ procedure.

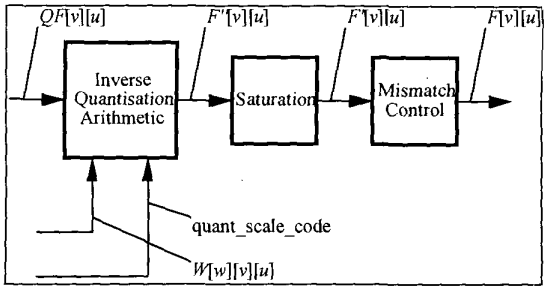

Figure 4. Procedure of inverse quantization defined in MPEG-4 standard

Inverse Quantization (MPEG):

$$F''[v][u]=\begin{cases} 0, & \text{if } QF[v][u]=0 \\ ((2\times QF[v][u]+k)\times W[w][v][u]\times quantiser\_scale)/16, & \text{if } QF[v][u]\neq 0 \end{cases}$$

where:

$$k=\begin{cases} 0 & \text{intra blocks} \\ Sign(QF[v][u]) & \text{non-intra blocks} \end{cases}$$

Saturation:

$$F'[v][u]=\begin{cases} 2^{bits\_per\_pixel+3}-1 & F'[v][u]>2^{bits\_per\_pixel+3}-1 \\ F''[v][u] & -2^{bits\_per\_pixel+3}\leq F'[v][u]\leq 2^{bits\_per\_pixel+3}-1 \\ -2^{bits\_per\_pixel+3} & F'[v][u]<-2^{bits\_per\_pixel+3} \end{cases}$$

Mismatch control:

$$sum=\sum_{v=0}^{v<8}\sum_{u=0}^{u<8}F'[v][u]$$

$F[v][u]=F'[v][u]$ for all u, v except $u=v=7$

$$F[7][7]=\begin{cases} F'[7][7] & \text{if } sum \text{ is odd} \\ \left\{\begin{array}{ll} F'[7][7]-1 & \text{if } F'[7][7] \text{ is odd} \\ F'[7][7]+1 & \text{if } F'[7][7] \text{ is even} \end{array}\right\} & \text{if } sum \text{ is even} \end{cases}$$

In the inverse quantization step, each coefficient in an 8×8 block has to be checked if its value is equal to zero. If so, the reconstructed value is equal to zero, or further calculation has to be performed. In the worst case analysis, the further calculation spends two data-path operations for decision, one for absolution, two for multiplication, two for shift, one for addition and one for table-lookup for each AC coefficient. Low frequency component (DC coefficient) is reconstructed by using one multiplication operation. In the second step, each pixel in the 8×8 block needs two more decision operations for saturation adjustment. Eventually, each block needs 67 extra operations for mismatch control in the final step. According to the above analysis, total data-path operations required by IQ process for CPL2 decoding are 99.93456 MOPS.

Then, we analyze the data transfer behavior while performing the method I of IQ process. A weighting quantization matrix (64 entries) and the quantizer scale, which are decoded by VLD processing unit, are loaded from GM. Assume the bit-width of these data is 16-bit, then totally it has to transfer 65 × 2 bytes to LM of IQ processing unit from GM. In Saturation, each component of the 8×8 block has to be checked if it exceeds the legal range, so 64 × 2 bytes data are loaded from GM to LM (read) and 64 × 2 bytes data are stored back to GM (write). Such situation is the same with the Mismatch Control. In fact,

the data leave in LM can be reused in these processing, which improves the performance of memory access. We will talk about this more in the section IV. Figure 5 summarizes the analyzed results of computational behavior of IQ process. Similarly, the other tools can be analyzed in the same way.

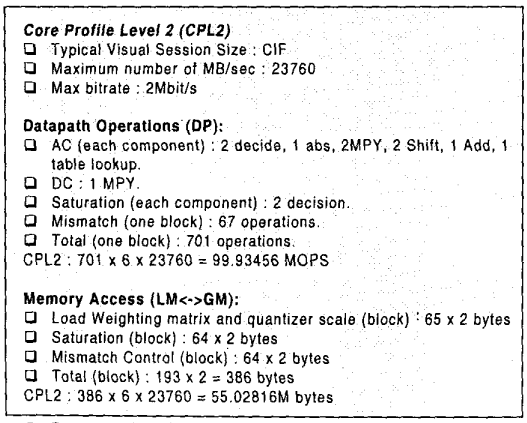

Figure 5. Computational behavior of IQ method 1 based on proposed computation model

By employing this methodology, the computational cost can be easily estimated, including datapath operations and memory access. Besides, behavior of video coding algorithms becomes clear and easy to model. Thus, computational requirement of a real-time CPL2 MPEG-4 video codec could be determined based on this analysis. The results of computational analysis of MPEG-4 video codec are summarized in Table 1 and 2.

B. Run-time Profiling

The software run-time profiling of the non-optimized Momusys C implementation is performed on a general purpose RISC processor, SUN UltraSparc2. GNU profiling tool, gprof, is employed to obtain the statistic of run-time information. In this non-optimized implementation, the sprial-search algorithm with SSDA [9] for motion estimation tool is adopted. It is simulated that the sprial-search with SSDA can achieve around 15%~30% gain. However, this optimized search algorithm still dominates the computation of video coding. (can not real-time) It can be observed that this tool spends most of the execution time, about 85% or even higher. This implies that more efficient algorithms or architectures are required.

IV. Implementation Consideration

As mentioned before, programmability is strongly demanded for supporting various functionality as well as system parameters. Thus, software-oriented implementations should be more suitable for realizing a practical MPEG-4 system. Namely, employing programmable DSPs could be a feasible choice for real implementations. However, the computational requirements for MPEG-4 Video codec exceeds the capability that general-purpose programmable DSP processors can provide today. This strict requirement on computation leads to the development of very powerful computing engines whose performance is much improved by applying several advanced design concepts, such as SIMD [10], VLIW [11],...etc. In these design classics, parallel-computing techniques at various levels are employed in order to achieve higher computation performance. Besides, hardware acceleration (coprocessor) is another good choice for performance enhancement.

In the near future, visual communication will become further functionality-rich. The need of high complexity algorithms for achieving those fantasy functionalities will increase. This requires the development of powerful programmable architecture for supporting more flexibility. Accordingly, DSP design should be scalable for future use. On the other hand, however, several conventional video coding tools are more specified to some dedicated computation, such as Motion Estimation, DCT/IDCT...etc. Moreover, these coding tools are rather computation-intensive. Until now, the computational requirements of these coding tools still dominate the entire computation load. Namely, the system bottleneck would take place over here without optimal design considerations. Design efforts should be spent on these computation-intensive tasks to achieve the maximum optimization. By considering the computation behavior of these tasks, hardware acceleration could be the best solution for implementing these coding tools in terms of system performance. Taking both programmability and computational requirements into considerations, system architecture for MPEG-4 should be configured as two major computation units: a programmable DSP core and a dedicated engine to achieve its maximum performance. Figure 7 shows the proposed system architecture of a multimedia processor that targets MPEG-4 Video coding. The dedicated engine can be optimally designed by applying the design techniques, such as data-reuse and parallel computing, so as to achieve better performance. Table 3 shows different speedups for Motion Estimation tool by different architectures respectively. Since the full-search Motion Estimation algorithm has very good data locality, data-reuse can be applied very efficiently on the systolic array architecture [12]. Thus, they can achieve very much speedup. Generally speaking, the systolic array architecture can be treated as a SIMD processor. On the other hand, although several narrow-scene SIMD DSP processors [10] have been reported that they can achieve much better performance on some video algorithms, such as MC, DCT/IDCT, the speedup is still not enough for real-time full-search Motion Estimation with large search range.

Memory configuration is another important issue for the system design. Therefore, the features of data storage and transfer for video processing tools in MPEG-4 core profile are analyzed so as to design efficient memory system. Table 4 shows the data transfer based on local memory storage analysis for the tools used in MPEG-4. A few tools require large amount of memory storage and data transfer including, Motion Estimation tool, DCT/IDCT tool, Motion Compensation tool and Padding tool. It's clear that if the sufficient local storage is provided, some data transfer overhead can be removed since several tools have data locality within an 8×8 block. Table 4 also shows that motion estimation tool dominates the cost of data transfer in MPEG-4 video processing. This leads to very much memory bandwidth and storage to be consumed. Hierarchical memory system can be employed to remove this problem. For the full-search motion estimation with macroblock size N and search range p, the minimum intermediate local buffer size to achieve minimum main memory access for previous frame data can be derived as the following equations.

Frame memory access: $3\times H(\text{width})\times V(\text{height})$
Intermediate buffer: $\max\{(2p\text{-}1)\times(2p+N\text{-}1), (2N\text{-}1)\times(2p+N\text{-}1)\}$

This can much reduce the main memory access and thus reduce the heavy traffic of system bus.

Based on the various analyses to MPEG-4 video processing described before, we think that efficient system architecture for MPEG-4 video coding should be configured as follows. A powerful dedicated coprocessor for full-search motion estimation, a parallel DSP core for support flexible functions as well as on-chip memories as local data buffers to reduce access from large frame memory.

V. Conclusion

In this paper, the computational behavior of MPEG-4 video codec is

analyzed by utilizing the proposed behavior model and runtime profiling. Based on the computational behavior analyses and data flow exploration, the design space for efficient architecture, including computing units and memory sub-systems is explored. A hybrid system architecture is considered to be the most efficient architecture for MPEG-4 Video coding since the maximum optimization is applied on it based on the complete exploration of computation behavior of the MPEG-4 Video coding.

Reference

[1] "Video Coding for narrow telecommunication channels at < 64 kbits/s," *Draft ITU-T Recommendation H.263*, July 1995.

[2] Didier Le Gall. "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of the ACM*, Vol. 34, No. 4, pp.46-58, April 1991

[3] ISO/IEC/JTC1/SC29/WG11 *Draft CD 13818-2 Recommendation H.262 Committee Draft.*

[4] ISO/IEC JTC1/SC29/WG11, *N2502a, Generic Coding of Audio-Visual Objects: Visual 14496-2, Final Draft IS*, Atlantic City, Dec. 1998.

[5] Jens P. Wittenburg, P. Pirsh and Gerald Meyer, "A Multithreaded Architecture Approach to Parallel DSPs for High Performance Image Processing Applications", *Proc. of IEEE Workshop on Signal Processing Systems*, 1999.

[6] ROB KOENEN, "Profiles and Levels in MPEG-4;Approach and Overview", MPEG HOME WEB, 1999

[7] N. Brady, F. Bossen and N. Murphy, "Context-based Arithmetic Encoding of 2D Shape Sequences", *Proc. of IEEE Conference on Image Processing (ICIP)*, 1997

[8] MPEG-4 VM software, European ACTS project MoMuSys, Apr. 1999.

[9] Tatsuji M., Hiroshi S., Takashi M., and Ichiro K., "Real-Time Software Video Codec with a Fast Adaptive Motion Vector Sesrch", *Proc. of IEEE Workshop on Signal Processing Systems*, 1999.

[10] KOUHEI NADEHARA, HANNO LIESKE and ICHIRO KURODA, "Software MPEG-2 Video Decoder on a 200-MHz, Low Power Multimedia Microprocessor", *Proc. of IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Seattle, May.1998

[11] Dutta S., Wolfe A., Wolf W., O'Connor K.J., "Design Issue for Very-Long-Instruction Word VLSI Video Signal Processors", *VLSI Signal Processing IX*, IEEE Press, pp.95-104, 1996.

[12] Jun-Fu Shen, Liang-Gee Chen, Hao-Chieh Chang and Tu-Chih Wang, "Low Power Full-Search Block-Matching Motion Estimation Chip for H.263+ Video Coding", *Proc. of International Symposium on Circuits and Systems (ISCAS)*, 1999

[13] Chang-Guo Zhou et.al, "MPEG Video Decoding with the UltraSPARC Visual Instruction Set", *Compcon'95*, 5-9 March 1995

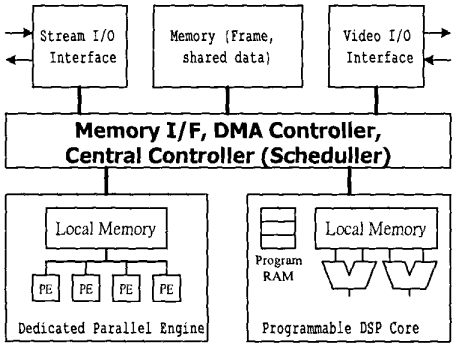

Figure 6. System Architecture of MPEG-4 multimedia system

Tool	Datapath Operation (MOPS)	Percentage (%)	Data Transfer (MBytes)	Percentage (%)
Motion Compensation	61.3129	12.17001535	36.3547	18.64626822
Shape Decoding(CAE)	167.84704	33.31600778	12.1404	6.226791988
IDCT	114.048	22.63742068	57.74304	29.61631403
Inv. Quant	99.93456	19.83603987	55.02816	28.22385636
Inv. Scan	27.37152	5.432980964	18.24768	9.359206255
Padding	21.28896	4.225651861	15.2064	7.799338546
VLD	12	2.381883489	0.25	0.128224605
Total	503.80298	100	194.97038	100

Table 2. Computational behavior of MPEG-4 video decoder tools (CPL2)

	RISC[13]	MMX[13]	SIMD[10]	Systolic Array(64PE) [12]
Required computation (MIPS/Mcycles)	18663.093 (Mcycles)	2395.95	4716.35	0.09569 (Mcycles)
Speedups		7.78	4	195

Table 3 Speedups of different architectures for full-search motion estimation

Tool	Datapath Operation (MOPS)	Percentage (%)	Data Transfer (MBytes)	Percentage (%)
Motion Estimation (FS, -16~+15)	24768.223	90.29387257	13259.981	92.50622553
Motion Compensation	61.3129	0.223519434	36.3547	0.253622994
Shape Coding(CAE) (FS, -16~+15)	2080.58479	7.584882367	778.56768	5.431558115
DCT/IDCT	232.65792	0.848166806	115.48608	0.805670941
Quant/IQ	199.86912	0.728633493	110.05632	0.767791052
Scan/IS	54.74304	0.19956866	18.24768	0.127302143
Padding	21.28896	0.077610034	15.2064	0.106085119
VLC	12	0.043746637	0.25	0.001744087
Total	27430.67973	100	14334.14986	100

Table 1. Computational behavior of MPEG-4 video encoder tools (CPL2)

Video Tool	Data Transfer (LM<->GM) (MBytes)	
	Basic Model	LM for 8x8 block data
Motion Estimation (FS, -16 ~ +15)	13259.981	13259.981
Motion Comp.	36.3547	36.3547
CAE Encoding	778.56768	< 778.56768
CAE Decoding	12.1404	1.1404
DCT	57.74304	18.24768
IDCT	57.74304	18.24768
IS/IQ	18.24768	0
S/Q	18.24768	0
VLC	0.25	0.25
VLD	0.25	0.25
Padding	15.2064	15.2064

Table 4. Amount of data transfer for MPEG-4 video processing in different models