MPEG-4 FGS Encoder Design for an Interactive Content-aware MPEG-4 Video Streaming SOC

Yung-Chi Chang, Chih-Wei Hsu, and Liang-Gee Chen
DSP/IC Design Lab
Department of Electrical Engineering and Graduate Institute of Electronics Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.
{watchman, jeromn, lgchen}@video.ee.ntu.edu.tw

***Abstract*— In this paper, the computational complexity of MPEG-4 Fine Granularity Scalability (FGS) coding is analyzed to explore an efficient FGS implementation for video streaming applications. With the proposed hardware-oriented optimization approaches, a hardwired FGS block-level processing core is proposed to provide a cost-effective solution. It can be integrated into an existing MPEG-4 coding system to form an interactive video streaming system. By the proposed FGS coder with reordered coding flow, the streaming server can adaptively decide quality-enhanced region by selective enhancement according to both object information from encoding side and user-defined region from receiver side. The proposed hardware core can support FGS profile level 5, frame size 720x576, 30Hz, for real-time streaming applications at 54 MHz.**

I. Introduction

The objective of video streaming is to allow users to view the video content while downloading instead of after totally receiving [1] [2]. Its principal goal is to reliably deliver high quality video when dealing with unknown and dynamic bandwidth, delay jitter, and loss rate [3]. MPEG-4 have standardized streaming video profile [4] and recommended Fine Granularity Scalability (FGS) as the core technique for video streaming applications. FGS enhancement layer is generated using bit-plane coding and it can be truncated to any amount of bits according to the real channel conditions. FGS coding scheme performs predictions only from base layer at cost of lower coding efficiency, so there is no drift error when part or all the enhancement bitstream are corrupted or lost. The more bits of enhancement layer are received and decoded at the decoder side, the higher quality of the reconstructed video results.

The server in a streaming system is responsible for transmitting video bitstreams with proper bit rate according to the network conditions and user preference [5]. The concept of prioritized transmission has been introduced in some work for more efficient streaming [6] [7]. However, either the enhancement is decided at the encoder side and cannot be changed at transmission time, or the FGS bitstream syntax needs to be modified. The proposed content-aware transmission model is introduced as shown in Fig. 1. The enhancement is performed at the server side. The video content providers can still provide video analysis information to be used at the server side to enhance specific video content. The users also can feedback

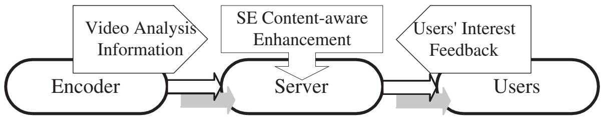

Fig. 1. Proposed Transmission Model.

the most interested part, and server can respond it in real-time. Since the selective enhancement is performed at the server side, the enhancement bitstream generated from encoder needs reproduction, i.e. decoding and re-encoding. This causes large computation overhead at server. To avoid the overhead, the FGS coding flow in MPEG-4 VM should be rearranged, and the FGS coding core should be implemented such that it can be integrated into an existing MPEG-4 encoding SOC easily. With our design, all operations for FGS enhancement at encoder side are now block-based, which is conformed to the base layer encoder.

This paper is organized as follows. In Sec. II, the function and coding flow of MPEG-4 FGS are described and analyzed first. Several proposed hardware-oriented optimization approaches are discussed in Sec. III. A cost-effective solution for implementing FGS encoder is exploited in Sec. IV. Sec. V provides the conclusions.

II. Analysis on MPEG-4 FGS Coding

A. Functional Description

FGS codes a video sequence into two layers. The base layer is supposed to be received and decoded completely at the decoder side. The enhancement layer is with bit-plane coding and can be truncated at any point since this results in only some data loss in less significant bit-planes. It can be referred to [5] for more details of the FGS functionality. Fig. 2 shows the detailed FGS coding flow in MPEG-4 VM [8]. It is divided into block- and picture-level operations.

In block-level operations, MSB position of each block is found first from the residues. Zigzag scan and word-to-bit-plane conversion are performed in the following. Finally Symbol Formation unit generates (RUN, EOP) symbols for each bit-plane and stores them into temporary buffer with necessary information, such as MSB position of that block, to facilitate the picture-level bitstream formation. In picture-level

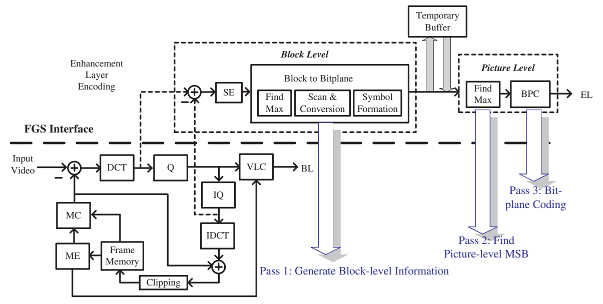

Fig. 2. FGS Coding Flow in MEPG-4 VM Implementation.

TABLE I
INSTRUCTION PROFILING ANALYSIS. (UNIT: MIPS)

	Arithmetic	Control	Load/Store	Others	Total
FGS	910	374	1,151	350	2,785 (13.64%)
Total	4,630	3,393	10,058	2,330	20,411 (100%)

operations, picture-level MSB information must be obtained first, which results in one picture-level passing by looking up the stored MSB information of all blocks. When entering bit-plane coding (BPC), moving through the same significant position of the picture from highest bit-plane downward, it transforms the stored symbols into variable-length bitstream and packs them into ultimate enhancement bitstream.

There are two advanced features for prioritized transmission and further improve the visual quality of FGS enhanced video: ”frequency weighting” (FW) [9] and ”selective enhancement” (SE) [10] techniques. FW exerts higher transmission priority for lower frequency components, and SE is the same concept as FW but now performed in spatial domain. The shifting weightings for SE are specified on MB basis.

B. *Profiling Results*

The FGS enhancement function can be integrated into the original MPEG-4 base-layer encoder SOC independently with some other aids either by system software or dedicated hardware implementation. So, we must decide what kind of implementation is most suitable for FGS by estimating its required computational power. We perform the instruction level profiling of FGS in an MPEG-4 encoding system using MPEG-4 Verification Model(VM) software [8]. The required instruction analysis for FGS is shown in Table. I, and the required memory bandwidth is shown in Table. II. The profiling condition is for foreman sequence, CIF format(352x288) and 30 fps, at 448MHz UltraSparc-II.

It is shown that when FGS function is integrated into an MPEG-4 encoder SOC targeting real-time applications,

TABLE II
MEMORY BANDWIDTH PROFILING ANALYSIS. (UNIT: MBYTES/SEC)

	Load	Store	Total Bandwith	Percentage(%)
FGS	2,799	618	3,418	10.26%
Total	25,002	8,298	33,300	100%

it consumes more than 2.7 GIPS computational power and 3.4 GBytes memory access bandwidth, which is 13.64% and 10.26% of the system resources, respectively. In addition, 41.3% of instructions are consumed for Load/Store operations and up to 80% memory bandwidth is spent on data loading. This statistics show that implementing FGS characterizes massive data processing and huge memory access bandwidth.

C. *Coding Flow Analysis*

The FGS coding flow in MPEG-4 VM [8] can be divided into block- and picture-level operations, and a temporary buffer is used to bridge these two parts of operations. The intermediate data after block-level processing will be stored in external memory. These data are loaded in order to find picture-level maximal bit-plane level.

In our simulation, SCAN, where zigzag scan and word-to-bit-plane conversion are included, Symbol Formation (SF) unit, and BPC take about 80% computation and memory bandwidth of the overall FGS encoding operation. The reason is that a general-purpose processor is not suitable for the bit-level operations due to its word-based sequential processing property. Thus, a dedicated hardware design for these block-level operations is more efficient in case FGS becomes a bottleneck for the whole SOC. In picture-level operations, picture-level MSB information must be obtained first, which results in an extra picture-level passing. Only the final bitstream picking and packing operations demand sequential passing order, from higher bit-plane to lower bit-plane. As to bitstream formation, VLC table lookup can be performed in parallel. The parallelism follows block-based raster-scan order, which implies rearranging this part into a block-level core is possible and more efficient.

As to the temporary buffering data type, in MPEG-4 VM it is (RUN, EOP) symbols for each bit-plane that are stored in memory. From system viewpoint, the amount of occupied system bus for memory access is more critical than the allocated external memory size. The most suitable temporary buffering data type will be evaluated to improve the system performance.

III. HARDWARE-ORIENTED OPTIMIZATION APPROACHES

To achieve a cost-effective implementation of FGS block-level processing core, several hardware-oriented optimization algorithms are proposed as follow.

A. *Global Maximum Keeping (GMK)*

In MPEG-4 VM implementation, it spends one picture-level pass to get the picture-level MSB information, which is ”Find Max” unit in the picture-level operation, as shown in Fig. 2. However, the work of finding out the picture-level MSB information can be performed in passing during block-level processing. After finding out one block-level MSB, the picture-level MSB, which we define global maximum here, can be continually updated at the same time. With GMK, the redundant picture-level pass is saved. It contributes much to the reduction of both memory access bandwidth and memory

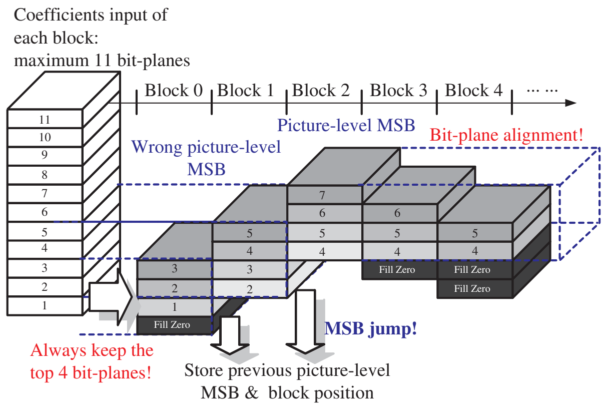
Coefficients input of each block: maximum 11 bit-planes
Block 0
Block 1
Block 2
Block 3
Block 4
… …
Picture-level MSB
Wrong picture-level MSB
Bit-plane alignment!
Fill Zero
MSB jump!
Always keep the top 4 bit-planes!
Store previous picture-level MSB & block position

Fig. 3. Concept of DBPA.

storage unit. All the block-level MSB information needn't be stored or fetched for finding out the picture-level MSB.

B. *Dynamic Bit-Plane Adaptation (DBPA)*

In MPEG-4 FGS profile [4], maximum four coded bit-planes for enhancing one frame are supported. The enhancement layer with four bit-planes covers a wide range of bit-rate variations. Although the coding flow is divided into separate block-level and picture-level operations, only the information of the top four bit-planes for each block needs to be saved at block-level regardless of the picture-level MSB. To keep only four bit-planes in process is the main concept of the DBPA approach. This reduces the required computational complexity and implementation cost.

Refer to Fig. 3, DBPA functions as follows. Each input block has maximum 11 bit-planes due to the dynamic range of the magnitude of the DCT coefficients. The block-level MSB information is extracted first and the number of bit-planes for that block is determined. GMK algorithm continually updates the picture-level MSB as soon as a block-level MSB is generated. Only the information of the top four bit-planes for each block is dynamically kept according to the current block-level MSB and finally aligned according to the picture-level MSB. The lower bit-plane information can be even replaced with zero, which means they don't have to be processed when the picture-level MSB is found out. According to our experiments, up to 30% computation can be reduced. With DBPA combined with GMK, the block-level MSB of all the blocks need not be saved because the picture-level MSB is gradually updated and only four bit-planes are in processing. The only side information has to be kept is the MSB jump. Since the jump can only occur from 4 to 11, there are few information to be stored.

C. *Coding Flow Reordering (CFR)*

There are three candidates to estimate the most proper temporary buffering type, including bit-plane raw data, (RUN, EOP) symbol, which is adopted in MPEG-4 VM, and partial bitstream. The bit-plane raw data are just the residues obtained from taking the difference between DCT coefficients and the dequantized ones. However, proper packing should be carried out such that the raw data form independent bit-planes. As to partial bitstream, since each bit-plane in one block can perform VLC table lookup according to the significant position in that block regardless of picture-level MSB, this work can be advanced to perform at block-level to generate partial bitstream, which is semi-finished version of the ultimate enhancement bitstream. This is the main concept of CFR approach.

The final enhancement bitstream is generated by picking and packing the right partial bitstreams. With CFR approach, it stores bitstream-level information of each block into the external memory. In average, the external bandwidth required by bit-plane raw data is 2.78 MBytes/sec, while that for partial bitstream is about 1.81 MBytes/sec. Since the partial bitstream is generated by VLC, the amount of data to be stored is much less. The variable-length bitstream should be packed into the hardware system bus bandwidth units and a header is required to keep the information about it.

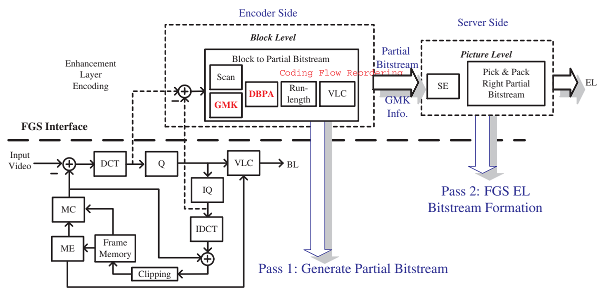

Fig. 4. Proposed Rearranged FGS Coding Flow.

D. *Picture-level Processing*

As to generating the ultimate enhancement bitstream of FGS, it requires one more picture-level processing from the MSB bit-plane of the frame to lower bit-plane to pick and pack all the partial bitstream in order. All required information is generated in GMK and DBPA units so partial bitstreams can be aligned in proper significant order. Sign is properly inserted once the partial bitstreams are generated. The required CBP information is also ready in the partial bitstream header to benefit the packing procedure. With the proposed optimization algorithms, a dedicated hardware is responsible for all the block-level operations, including Scan, Symbol Formation, and VLC coding in BPC, which are the main workload of FGS module. With block-level hardware acceleration, the picture-level processing is left simple and sequential in nature to be performed by the system software of the encoder. However, since the enhancement bitstream is tailored at the server side to meet the users' conditions, this simple task of picking and packing partial bitstreams can even be moved to server side to generate the final bitstream according to the bit allocation plans. Fig. 4 is our proposed rearranged coding flow for FGS with hardware-oriented optimizations.

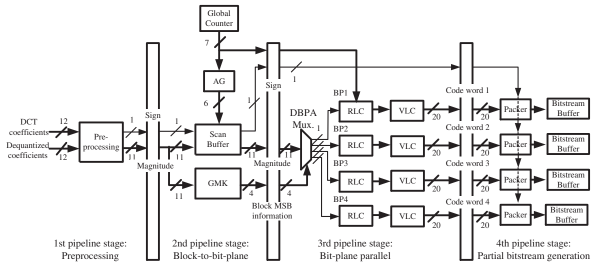

Fig. 5. Architecture of the Proposed Block-level Processing Core.

TABLE III
CHIP FEATURES.

Technology	TSMC 0.25um CMOS 1P5M
Package	128 CQFP
Die Size	1.9851 mm x 1.9851 mm
Core Size	0.769 mm x 0.769 mm
Gate Count	24,287
Transistor Count	118,280
Clock Rate	54 MHz
Power Consumption	23.925mW @ 2.5V, 54MHz

IV. ARCHITECTURE DESIGN AND IMPLEMENTATION

Fig. 5 is the architecture of our proposed hardware processing core for FGS block-level operations. Preprocessing unit takes the residues between DCT coefficients and dequantized ones. Sign and magnitude are generated in this stage. In the 2nd stage, the MSB information is extracted by GMK during the scan procedure. In the 3rd and 4th stage, the DBPA multiplexer selects the top four bits and there are four sets of subsystems working in parallel to generate partial bitstreams. Each subsystem includes independent run-length coders, different VLC tables, and bitstream packers, for each of the top four bit-planes. Finally, all the partial bitstreams are stored in the partial bitstream buffers.

When integrating the FGS block-level processing core into our previous work [11], which is responsible for the base layer texture coding, the partial bitstreams for one MB can be generated within 1,000 cycles. So, our proposed hardware core can support FGS profile level 5, frame size 720x576, 30Hz, for real-time streaming application at 54 MHz, which demands that one MB should be finished in 1,111 cycles. Compared with software implementation, this specification can be achieved at cost of 11 GIPS, our proposed hardware core is a cost-effective solution. Our design is fabricated by TSMC 0.25um 1P5M CMOS process. The chip works correctly under 54MHz. The measured power consumption is 23.925mW. Fig. 6 is the chip die photo and Table. III is the chip features.

V. CONCLUSIONS

A content-aware video streaming system based on MPEG-4 FGS is presented in this paper. We introduce a transmission model for such applications. The FGS coding flow is re-arranged and combined with selective enhancement to achieve content-aware requirement. We also design a cost-effective FGS block-level processing core for MPEG-4 encoder SOC with several optimization approaches, such as GMK and DBPA. It is fabricated and verified under 54 MHz ,which can support MPEG-4 FGS profile level 5 real-time encoding and streaming applications. With this core, the server can adaptively decide quality-enhanced region according to encoding side and receiver side, and an interactive user-defined feedback scheme is realized.

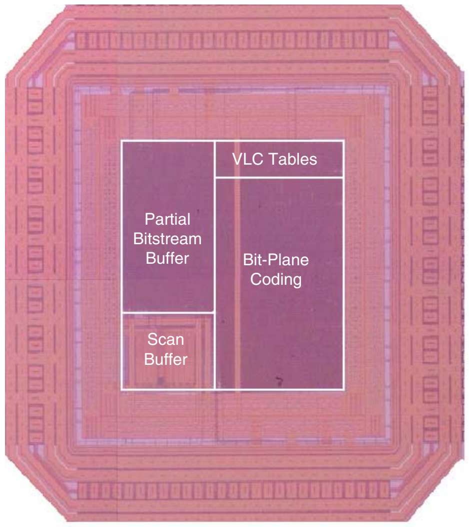

Fig. 6. Chip Die Photo.

REFERENCES

[1] G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman, and Y. A. Reznik, "Video coding for streaming media delivery on the internet," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 269–281, Mar. 2001.

[2] D. Wu, Y. T. Hou, W. Zhu, Y. Q. Zhang, and J. M. Peha, "Streaming video over the internet: Approaches and directions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282–300, Mar. 2001.

[3] J. G. Apostolopoulos, W. T. Tan, and S. J. Wee, "Video streaming: Concepts, algorithms, and systems," HP, Tech. Rep. HPL-2002-260, Sept. 2002.

[4] *ISO/IEC 14496-2:1999/FDAM4: Streaming video profile*, ISO/IEC JTC1/SC29/WG11 Final Draft Amendment N3904, Jan. 2001.

[5] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, Mar. 2001.

[6] M. van der Schaar and Y. T. Lin, "Content-based selective enhancement for streaming video," in *IEEE International Conference on Image Processing (ICIP)*, 2001, pp. 977–980.

[7] Y. He, S. Yang, and Y. Zhong, "Block-based fine granularity scalable video coding for content-aware streaming," in *IEEE International Conference on Image Processing (ICIP)*, vol. 2, 2002, pp. 45–48.

[8] "MPEG-4 VM FPDAM1-FPDAM4," MoMuSys, Dec. 2000.

[9] H. Jiang and G. M. Thayer, "Using frequency weighting in FGS bit-plane coding for natural video," ISO/IEC JTC1/SC29/WG11 MPEG99/M5489, Dec. 1999.

[10] W. Li, F. Ling, and X. Chen, "Fine granularity scalability in MPEG-4 for streaming video," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 1, 2000, pp. 299–302.

[11] C. W. Hsu, W. M. Chao, Y. C. Chang, and L. G. Chen, "Texture coder design of MPEG-4 video by using interleaving schedule," in *IEEE International Conference on Multimedia and Expo (ICME)*, vol. 2, 2002, pp. 157–160.