Reduction of CPU Computation Load Based on OpenCL for Speech Codec

Rong-San Lin*, *Member, IEEE*, Yung-Chiang Wei[+] and Rong-Xian Zhang

*Southern Taiwan University of Science and Technology ,

[+]Far East University

***Abstract*-- The G.723.1 codec has been widely used and implemented for real-time Voice over Internet Protocol (VoIP). The implementation should consider the computational power for real-time encoder. This paper aims to reduce the computational load of CPU-based G.723.1 codec implementation. The G.723.1 encoding can be parallelized using Graphical Processing Units (GUP) to further reduce the computation time. With the assistance of GPU handling, the CPU could have extra computing power to perform other operations. Nowadays, each computer or handheld mobile device will have a general low-cost graphics card with GPU. So using the GPU as an auxiliary device does not increase the hardware cost, but makes good use of the computer hardware resources.**

The overall evaluation results present that the average perceptual evaluation of speech quality score is slightly reduced by 0.012 and the method we proposed can reduce the computational loading of CPU by about 22.3% relative to the original CPU only. Furthermore, an objective speech quality assessment validates that the proposed method can provide comparable speech quality to the original coding method.

Keywords: *OpenCL, GPU, Speech Codec, Parallelized, Reduced Computation Load.*

I. INTRODUCTION

For the applications of Voice over IP (VoIP) communication system, the G.723.1 encoder [1], which is based on the principles of linear prediction analysis-by-synthesis (AbS), has been widely used on the internet. The typical coder structure of G.723.1 coder definitely achieves high speech quality and low bit rate. Nevertheless, the obvious drawback from this structure is that the encoder requires much more computational complexity for adaptive codebook (Acbk) search. Besides, fixed codebook coding also consumes huge computation power. With the cost-effective implementation of Samsung's DSP chip, Lee and Park et al., [2] make great efforts to interpret the general distribution of computation load in the entire process of G.723.1 codec. For more detailed information, it is illustrated in Table 1 that adaptive codebook search constitutes more than 24.9% of the computation consumption in the G.723.1 encoding process. In order to reduce the codebook search computation loading, some fast search approaches have been proposed from recent literatures. Lin *et al.* [3] proposed two efficient candidate schemes to reduce the search complexity for both of the adaptive codebook and fixed codebook. Jung et al., [4] also presented a one-order closed-loop pitch predictor to predict both pitch-gains and pitch-lag considered in the fifth-order closed-loop pitch predictor.

TABLE I. Cycles for each algorithmic block of G.723.1

	Function description	cycles
Encoder	LPC	0.3MIPS
	pitch	1.1 MIPS
	LSP	1.6 MIPS
	Filtering	1.9 MIPS
	Adaptive Codebook (Acbk)	5.1 MIPS
	Fixed Codebook	12 MIPS(MP-MLQ) 9 MIPS(ACELP)
Decoder		1.5 MIPS
Total		23.5 MIPS(MP-MLQ) 20.5 MIPS(ACELP)

Million Instructions Per-Second (MIPS)

II. ANALYSIS COMPUTATION COMPLEXITY OF THE SPEECH CODEC

Firstly, we use Visual studio Profile to analyze the overall computation complexity for all functionalities in performing G.723.1 codec. As shown in Fig.1, the function of DotProd (Dot product) requires about 52.2% of the computational complexity relative to the overall complexity in the entire codec. This function from our algorithm is suitable for GPU parallel processing for the purpose of increasing the encoding speed. However, the largest computation load for speech encoding functionalities to execute multiplication and addition operations are computation of adaptive codebook (Find_Acbk). Therefore, we transfer the heavy computation load to the GPU processors, and it is the most effective approach for our consideration. In this paper, we propose that the computation of DotProd function that can be parallelized in this part of speech coding process, and can be transferred to GPU processors. Ideally, the amount of computation burden for CPU in G.723.1 encoder can be reduced by about 24.9%.

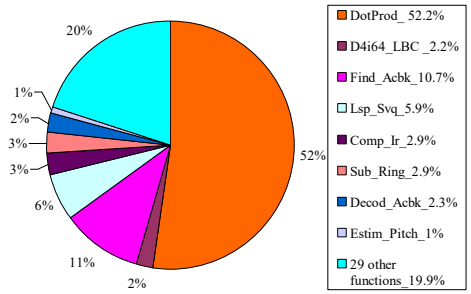

Fig. 1. The distribution of computation load for the codec process of the G.723.1.

III. THE PROPOSED METHOD

The G.723.1 encoder takes operations on frames of 240 samples each, each frame is divided into 4 subframes of 60 samples each. Following this specification, we have analyzed the computation load of the Acbk coding module used in the G.723.1 encoder in terms of the number of multiplication/accumulation (MAC) operations for every subframe. There are two approaches favorable to computation process with OpenCL in parallel computation for signal processing, in case of performing adaptive codebook (Find_Acbk) algorithm. The first is to calculate the cross products of the target signals and the predicted signals, energies of the predicted signals, and the between crosses among the fifth-order predicted signals. Each subframe speech encoder has to call DotProd function for computing cross products, which require 300 multiplication-addition operations. Similarly, 600 multiplication-addition operations are required when calling DotProd function to compute crosses. Furthermore, the demanded computation are 3600(3x1200) and 4800 (4x1200) multiplication-addition operations, respectively. Accordingly, these three parts require 4200 multiplication-addition operations on average for each subframe. This paper uses an OpenCL work item to calculate the Dot product operation for 60 samples each, and based on frequency of Dot product operation to allot those to one dimension OpenCL work group. Under this strategy, we can assign the 4200 multiplications and additions to the GPU at a time.

Secondarily, we search for index of the adaptive gain codebook (Gid) and adaptive pitch lag (Lid). In this part to compute DotProd, the length of multiplication and addition operation is 20 samples. Based on open loop pitch lag and encoding bit rate, adaptive codebook (Find_Acbk) algorithm in G.723.1 exploits a fifth-order pitch predictor to estimates the pitch lag and gains simultaneously. In detail, pitch predictor gains are vector quantized using two gain-codebooks (GB) with 85 or 170 entries, and each gain-vector has 20 elements. The pitch predictor gains are obtained with 85 or 170 entries under the high bit rate and 170 entries under the 5.3kbps rate. The 170-entry codebook is reused and common to both rates. For the 6.3kbps rate, if open loop pitch lag (L0) is less than 58 for both subframes 0 and 1 or if open loop pitch lag (L2) is less than 58 for both subframes 2 and 3, then the 85 entry codebook is used for the pitch gain quantization. Otherwise, we use the 170-entry codebook to quantize the pitch gain.

Therefore, pitch gains encoder have to call 3x85, 3x170, 4x85 or 4x170 times of Dot product, respectively, based on different Gain table that established different length of one dimension OpenCL work group. Each OpenCL work item is responsible for computing 20-samples Dot product of multiplication and addition operations. Therefore, there may be four kinds of operation for each subframe, including 3x85x20, 3x170x20, 4x85x20, and 4x170x20 multiplications and additions. Hence, for each subframe of speech coding, an average of about 8925 multiplications and additions are required. This paper proposes to pass this part of the calculations to GPU processing. With this scheme, the computation burden for CPU to execute G.723.1 encoding is reduced by approximately 22.3%, that is, about 52500 multiplication and addition operations. Finally, we adopted the ITU-T P.862 recommendation [5] to measure the objective speech qualities for six test speech files. Based on our proposal, the experimental results are shown in Fig. 2, when comparing to the Perceptual Evaluation of Speech Quality (PESQ) values of the proposed method with the only using CPU encoding procedure.

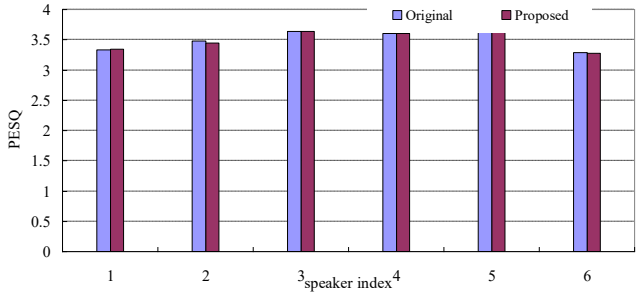

Fig. 2. PESQ of the original codec approach compared with the proposed method.

IV. CONCLUSIONS

This paper proposes that the adaptive codebook encoding procedure can be parallel computed with GPU to handle DotProd function. The simulation results indicate that the average PESQ value is degraded slightly by 0.012. Besides, our proposed method can reduce average computation load of the CPU by about 22.3% relative to the scheme using only CPU encoding computation with the slight penalty of negligible perceptually degradation.

ACKNOWLEDGMENT

This work was supported by the Ministry of Science and Technology of the Republic of China under research contract MOST 105-2221-E-218 -023 -MY2.

REFERENCES

[1] ITU-T Rec. G.723.1, *Dual Rate Speech Coder for Multimedia Communications at 5.3 and 6.3 kbit/s*, Mar. 1996.

[2] S. M. Lee, S. Park and Y. Jang, "Cost-effective Implementation of ITU-T G.723.1 on A DSP Chip", Proceedings of 1997 IEEE International Symposium on Consumer Electronics, pp. 31-34, December 1997.

[3] R. S. Lin and J. Y. Wang, "Efficient Candidate Scheme for Fast Codebook Search in G.723.1", *IEICE Trans. Information and Systems*, vol.E95-D, no.1, pp.239-246, Jan. 2012.

[4] S. K. Jung, K. T. Kim, Y. C. Park and H. G. Kang, "A Fast Adaptive-Codebook Search Algorithm for G.723.1 Speech Coder", *IEEE Signal processing letters*, vol. 12, no.1, pp. 75-78, January 2005.

[5] ITU-T Rec. P.862, *Perceptual Evaluation of Speech Quality (PESQ), an Objective Method for End-to-end Speech Quality Assessment of Narrowband Telephone Networks and Speech Codecs*, February 2001, http://www.itu.int/rec/T-REC-P.862-200102-I/en.