Real-time Implementation of G.729.1 Coder on ARM9E Processor for Wideband VoIP Communication

WoonSeob So, InKi Hwang, KiJong Koo, SeungHan Choi, DoYoung Kim and HyunJoo Bae
Electronics and Telecommunications Research Institute
138 Gajeongno, Yuseong-gu, Daejeon, 305-700, Korea

Abstract

In this paper, we describe the real-time implementation of ITU-T G.729.1 coder to run on ARM9E processor core. We translated most functions of the coder including basic and arithmetic operations C into assembly. G.729.1 is a scalable speech and audio coder for wideband telephony applications. This coder covers the full energy of human speech, and is compatible with G.729 widely used in speech communications. The optimized coder has the complexity of about 54MCPS at 32kbit/s. And it takes 11.5ms to execute the coder on the target system. By applying the coder in the Internet phone, we confirmed the wideband VoIP communications.

Key Words

G.729.1, Wideband Codec, ARM9E, VoIP, IP phone

1. Introduction

Recently, the speech communication services over the Internet environment require better quality of service than the existing telephony and the interoperability with the existing speech terminals in the consumer electronics field. To meet these requirements it is necessary to develop new Internet appliances with high quality and interoperability with the present devices like Voice over IP (VoIP) phone. For this reason, we considered the G.729.1 [1] wideband speech and audio coder which covers the full energy of human speech, and is compatible with G.729 [2] widely used in speech communications. To implement the coder in real-time on a single processor economically and efficiently we selected the ARM9E processor core [3] having DSP enhanced extensions. One of the most important benefits of the ARM9E solution, and a significant advantage, is that all the required processing can be performed on the ARM9E as a standalone processor.

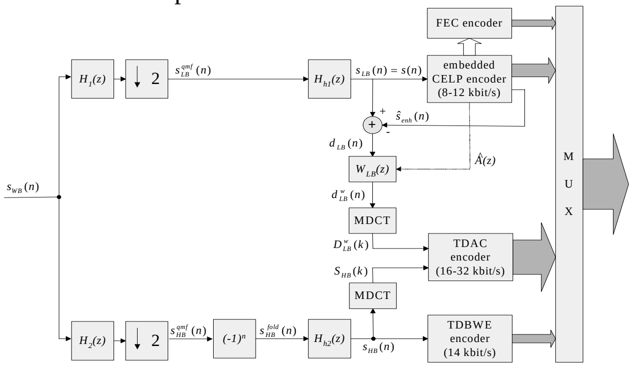
In this paper, we describe the real-time implementation of ITU-T G.729.1 coder using optimization process to run the coder on the ARM9E processor core. Most functions of the coder including basic and arithmetic operations were translated C into assembly language so as to minimize processing power in the processor core. As a result of this work we reduced the execution time of the coder about 80% and confirmed the wideband speech communications actually.

This paper describes how the real-time implementation of the G.729.1 coder has been implemented and how the results of the coder have been verified. Firstly, the implementation of results of the G.729.1 wideband coder is described in introduction. In section 2, the overview of the G.729.1 coder and its major functions will be discussed. And section 3 describes how the real-time implementation was processed. In section 4, how the tests were executed to verify the modified coder will be described. Finally, in section 5, the conclusion will be reached.

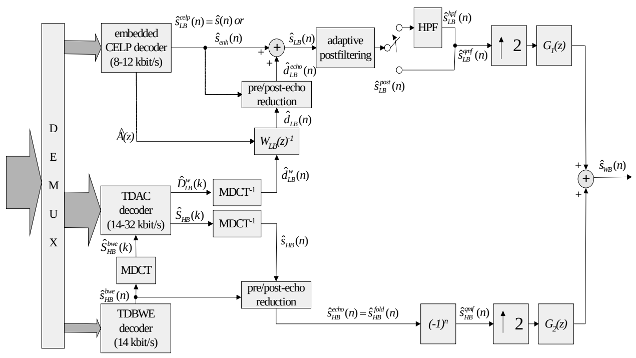
2. Overview of G.729.1 Wideband Coder

G.729.1 is an 8-32 kbit/s scalable wideband (50-7,000 Hz) speech and audio coder. The encoder input and decoder output are sampled at 8 and 16kHz. The bitstream is structured into 12 embedded layers with a core layer interoperable with G.729. This coder is designed to operate with a digital signal sampled at 8 or 16kHz followed by conversion to 16-bit linear PCM for the input to the encoder. Similarly, the format of the decoder output is 16-bit linear PCM with a sampling frequency of 8 or 16kHz. This coder algorithm is based on a three-stage embedded coding structure: embedded Code-Excited Linear Predictive (CELP) coding of the lower band (50-4000 Hz), parametric coding of the higher band (4000-7000 Hz) by Time-Domain Bandwidth Extension (TDBWE), and enhancement of the full band (50-7000Hz) by predictive transform coding referred to as Time-Domain Aliasing Cancellation (TDAC).

G.729.1 operates on 20 ms frames. However, the embedded CELP coding stage operates on 10 ms frames. As a result two 10 ms CELP frames are processed per 20 ms frames. To be consistent with the G.729, using 10 ms frames and the 5 ms subframes, the 20 ms frames are referred to as superframes.



(Fig. 1) Structure of the G.729.1 Encoder



(Fig. 2) Structure of the G.729.1 Decoder

3. Real-time Implementation on ARM9E Processor

3.1 Performance analysis of the coder

Before the real-time implementation of the coder on ARM9E processor core, it is necessary to classify the coder functions that require much more complexity than others and to analyze the performance finding the part which optimization has to be intensively accomplished. Generally, in performance analysis, there are some methods: the method of calculating the Weighted Million Operations Per Second (WMOPS) which gives the weighted value at the coder about the basis and control operations and estimates the complexity, the method of using the profile information about the coder, and the method of measuring the execution time of the optimized coder on the target system actually. As to the WMOPS, there is more the meaning of the estimation about the complexity of the coder before optimization. The profiling results indicate the information about the calling frequency of the coder functions and execution time of the functions. Because it takes too long to check the execution time until the final coder is produced, it is necessary to analyze the WMOPS complexity and profile information before optimization.

To calculate the function's WMOPS of the coder in detail, we modified the released G.729.1 coder which was programmed to calculate the whole WMOPS by default. After compiling and executing the encoder and decoder, we got the WMOPS value of each function. The overall complexity of the coder is about 35.79WMOPS based on the basic operators of ITU-T Software Tool Library [4]. The profiling information stands for that which part of the coder requires more time to run, so it can be used to analysis so big or complicated program. As to the profiler, there are several kinds of programs but the target system using the coder operates in the Linux OS, we used the gprof which is the GNU profiler [5]. After execution the gprof we can get a plat profile and a call graph. A plat profile has the information of how many times each function was called and how long it took to execute each function. A call graph indicates the information of the function call relationship of each function and the execution time of the called functions.

According to the results of the performance analysis, table 1 shows each block's priority to be optimized of the coder.

Table 1 Optimization Target Blocks of G.729.1 Coder

Block		Priority
Encoder	CELP2S_ACELP_code_A_OTH	H
	CELP2S_ACELP_code_2NDLAYER	H
	Pitch_fr3	H
	Pitch_ol	H
	Qua_lsp	H
	Az_lsp	M
	Syn_filt2	M
	Syn_filt	H
	Residu2	M
	AutoCorrLPC	L
	Qua_gain	L
	Pred_lt_3, Int_qlpc	M
	Convolve	L
	TDAC_quantif	H
	TDAC_allocbit	H
	TDAC_mdct	H
Decoder	G729_pst_ltp	H
	TDBWE_frequency_envelope_shaping	H
	TDBWE_generate_excitation	H
	G729_Pred_lt_3	M
	G729_Int_qlpc	M
	TDAC_inv_mdct	H
	TDAC_mdct	H
	MAIN_QMF_syn	M
Others	MAIN_lp3k	M

3.2 Characteristics of ARM9E Processor Core

The ARM9E processor core has the V5TE architecture, which includes DSP instructions to support signal processing algorithms efficiently. ARM's DSP extensions broaden the suitability of the ARM CPU family to applications that require intensive signal processing, whilst at the same time retaining the power and efficiency of a high-performance RISC microcontroller. The ARM DSP extensions have been implemented in the ARM946E-S, ARM966E-S, ARM926EJ-S, etc. cores. The hardware architecture to support the DSP-enhanced extensions is based on the existing ARM9TDMI RISC core, that is, a five-stage pipeline and Harvard memory architecture. The DSP-enhanced cores are best suited for applications that require a blend of high-quality DSP performance and efficient control implementation. This includes highvolume applications such as in mass storage devices, speech coders, speech recognition and synthesis, networking applications, automotive control solutions, smartphones and communicators, and modems.

3.3 Implementation of in-line assembler

In the case of the released G729.1 coder, because the basic arithmetic and data processing operations have the function form, the execution time is considerably lengthened according to the function call and return. These functions can be changed to in-line C code type within the source code. However, because these function codes require themselves much processing time it is more efficient to translate those functions into the in-line assembly code form for the performance improvement. It is necessary to understand the instructions supported in the target processor to program the assembly in-line code in order to prevent the unnecessary code generation.

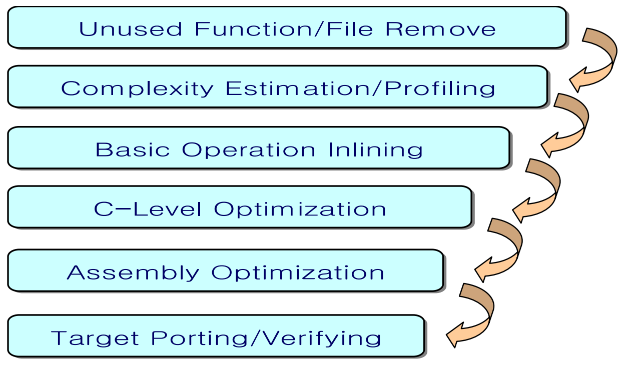
Using the ADS [6] we translated high complexity functions into ARM in-line assembly codes and after verifying these codes with the test vector, translated them into the GCC in-line assembly codes to load them on the target system operating in Linux OS. Since the syntaxes of these two are different completely, we had to rewrite the codes referencing the GCC compiler manual [7]. The table 2 shows the average complexity of the G.729.1 coder simulation results.

Table 2 Assembly Code Functions

Function Name	Assembler Function
Log2	Log2.s
Pow2	Pow2.s
div_s	div_s.s
G729EV_G729_AutocorrLSP	G729EV_G729_AutocorrLSP.s
G729EV_G729_Az_lsp	G729EV_G729_Az_lsp.s
G729EV_G729_ Cheb	G729EV_G729_ Cheb.s
G729EV_G729_ Convolve	G729EV_G729_ Convolve.s
G729EV_G729_ Cor_h_X	G729EV_G729_ Cor_h_X.s
G729EV_G729_ Lag_max	G729EV_G729_ Lag_max.s
G729EV_G729_ Lsp_lsf	G729EV_G729_ Lsp_lsf.s
G729EV_G729_ Lsp_pre_select12	G729EV_G729_ Lsp_pre_select12.s
G729EV_ G729_ Pred_lt_3	G729EV_ G729_Pred_lt_3.s
G729EV_G729_ Residu	G729EV_G729_ Residu.s
G729EV_G729_ Residu2	G729EV_G729_ Residu2.s
G729EV_G729_ Syn_filt	G729EV_G729_ Syn_filt.s
G729EV_G729_ Syn_filt2	G729EV_G729_ Syn_filt2.s
G729EV_ CELP2S_d4i40_17_fast	G729EV_ CELP2S_d4i40_17_fast.s

3.4 Optimization Process of the Coder

In order to use the G.729.1 coder on the ARM9E processor, we had to transform most of the fixed point original ANSI C codes of this coder to assembly codes so as to minimize processing power of the application processor. In the case of the released G729.1 coder, because the basic arithmetic and data processing operations have the function form, the execution time is considerably lengthened according to the function call and return. Before optimization of the codec, it is necessary to classify the coder functions that require much more complexity than others and to analyze the performance finding the part which optimization has to be intensively accomplished. After analyzing the performance of the coder, we changed most of the coder into ARM in-line assembly codes and after verifying these codes with the test vectors, translated them into the GCC in-line assembly codes to load them on the target system. The coder optimization process is shown in Fig. 3.



(Fig. 3) Coder Optimization Process

4. Simulation and Test Results

We executed the encoder and decoder after compiling in ADS and verified the results using the test vectors provided by ITU-T. As to the test vectors, there are 44 total, 14 for encoder and 30 for decoder. The simulation result of the optimized G.729.1 coder matched the test vectors bit exactly. The table 3 shows the average complexity of the G.729.1 coder simulation results. At 32kbit/s, the encoder and decoder part had complexity of 31.2MCPS and 22.8 MCPS respectively and improved about 79.2% than the before optimization.

The method of measurement of the Million Cycles Per Second (MCPS) mainly used as the index of the optimization performance is as follows [8].

Table 3 Simulation Results of the G.729.1 at 32kbit/s

Symbol	Before Optimization	After Optimization	Ratio
Encoder	168.2	31.2	81.5%
Decoder	91.2	22.8	75.0%
Total	259.4MCPS	54.0MCPS	79.2%

1) In ADS, the total cycle number in the statistics is acquired after the encoder or decoder executed in order to find out the number of execution cycle.

2) By using the below formula, MCPS is calculated.

$$MCPS = \frac{Ncycles \bullet Fs}{N_{frame} \bullet M_{samples}} \bullet 10^{-6} \qquad (1)$$

Ncycles : Executed Cycle No., *Fs* : Sampling Frequency, *Nframe* : Frame No., *Msamples* : Samples per Frame

Encoder = {(392825196) / (629 frames)} * (1/20ms)
= 31.2 MCPS
Decoder = {(286993554) / (629 frames)} * (1/20ms)
= 22.8 MCPS

Finally, we tested and verified the program on the target system having the ARM9E processor for the real-time operation. The directory including the encoder and decoder execution files is shared between the target system and host computer through Ethernet for convenient debugging and verification. In the target system, using the time measurement command time, the execution time of the encoder and decoder was measured. Table 4 shows the execution time results of the encoder and the decoder at 32kbit/s.

Table 4 Execution Time Measurements at 32kbit/s

Symbol	Before Optimization	After Optimization	Ratio
Encoder	37.39 ms	6.75 ms	81.9%
Decoder	20.45 ms	4.76 ms	76.7%
Total	57.84ms	11.51ms	80.1%

The optimized coder operated with the call processing program in the real Internet phone. The voice signal was inputted with microphone and it was outputted on the speaker of the other side telephone and the real-time voice call was confirmed. We verified that the mouth to ear delay is below 100ms which is suitable for real-time wideband VoIP communications.

5. Conclusion

In this paper, we implemented the G.729.1 wideband coder in the real-time basis running on ARM9E processor core which has the DSP enhanced extensions to provide signal processing algorithms. With the WMOPS calculation and the profiling of the released G.729.1 original ANSI-C coder, we analyzed the performance of it. And then by using ARM9E assembler around the blocks in which complexity is high, we optimized based upon the analyzed results. The coder optimized on a real-time basis, in 32kbit/s, has the complexity of about 54MCPS. And it takes 11.5ms to operate the coder on the target system. We confirmed the real-time wideband voice communication by applying this coder in the real Internet phone which inputs voice frames and outputs it per 20ms. As a result, this coder can be applied to the wideband voice and audio applications which are various with Internet phone, interactive e-learning system, Internet audio system, digital voice recorder, and etc.

Acknowledgements

This work was supported by the IT R&D program of the MKE and IITA in Korea [2008-S-011-01]; A Study on Acoustic Convergence Codec and its Control Technology for FMC.

References

[1] ITU-T Rec. G.729.1, “G.729 based Embedded Variable bit-rate coder : An 8-32kbits/s scalable wideband coder bitstream interoperable with G.729” May 2006.

[2] ITU-T Rec. G.729, "Coding of speech at 8 kb/s using conjugate-structure algebraic code-excited linear prediction (CS- ACELP)," June 1995.

[3] Hedley Francis, “ARM DSP-Enhanced Extensions”, ARM White Paper, May 2001.

[4] ITU-T Software Tool Library 2005 User’s Manual, August 2005.

[5] GNU gprof, http://www.cs.utah.edu/dept/old/texinfo/as/gprof_toc.html.

[6] ARM Ltd., ARM Developer Suite Version 1.2-CodeWarrior IDE Guide, March 2003.

[7] Richard M. Stallman, GCC Developer Community, “Using the GNU Compiler Collection”, May 2004.

[8] http://www.dsprelated.com/groups/speechcoding/show/940.php.