AN2278
Rev. 1, 11/2004

ITU-T G.729B Implementation on the StarCore™ SC140/SC1400 Cores

By Mircea Ionita, Mihai Fecioru, Bogdan Costinescu, Costel Ilas

This application note describes the process of porting and optimizing the ITU-T G.729 Recommendation with Annex B (G729B) on the StarCore™ SC140 core. Annex B proposes a silence compression scheme to reduce the transmission rate during the silence periods. This scheme consists of three algorithms:

- Voice activity detection (VAD)
- Discontinuous transmission (DTX)
- Comfort noise generator (CNG)

In the optimization phase, there are three categories of functions:

- Functions reused from G729 (no modification required)
- Functions that also exist in G729 but require changes
- New functions related to the silence compression scheme

During optimization, most of the functions are optimized in C, and only a few functions are written in assembly language to achieve better speed and size results. The StarCore compiler reduces the effort required in the optimization phase. The tests to verify the bit exactness use all available ITU test vectors, as well as an extended set of internal Freescale test vectors.[1]

The ITU-T G.729B achieved high performance on the SC140 core—36 channels on a 300 MHz SC140 core using 185 KB of memory for all channels.

CONTENTS

1 Introduction

G.729 is an 8kbit/s Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP) speech codec employed in simultaneous voice and data applications [1]. Annex B of the ITU-T G.729 Recommendation proposes a silence compression scheme for terminals used with ITU-T G.729 or ITU-T G.729A. It contains three algorithms for reducing the transmission rate during periods of silence, as follows:

- *Voice activity detection.* Indicates the presence or absence of voice. Detection of voice activates the vocoder to code/decode active voice frames. This algorithm makes a decision on voice activity every 10 ms in correlation with the frame size of the G.729. The output of the VAD module is either a 1 or 0, indicating the presence or absence of voice activity, respectively. The decision is based on a set of difference parameters: the full band energy, the low band energy, the zero-crossing rate and a spectral measure. If the output of a VAD module is 1, the speech codec is invoked to code/decode active voice frames (speech). If the output is 0, the DTX/CNG algorithms generate non-active voice frames (silence). A block diagram of the silence compression speech system is depicted in **Figure 1**.

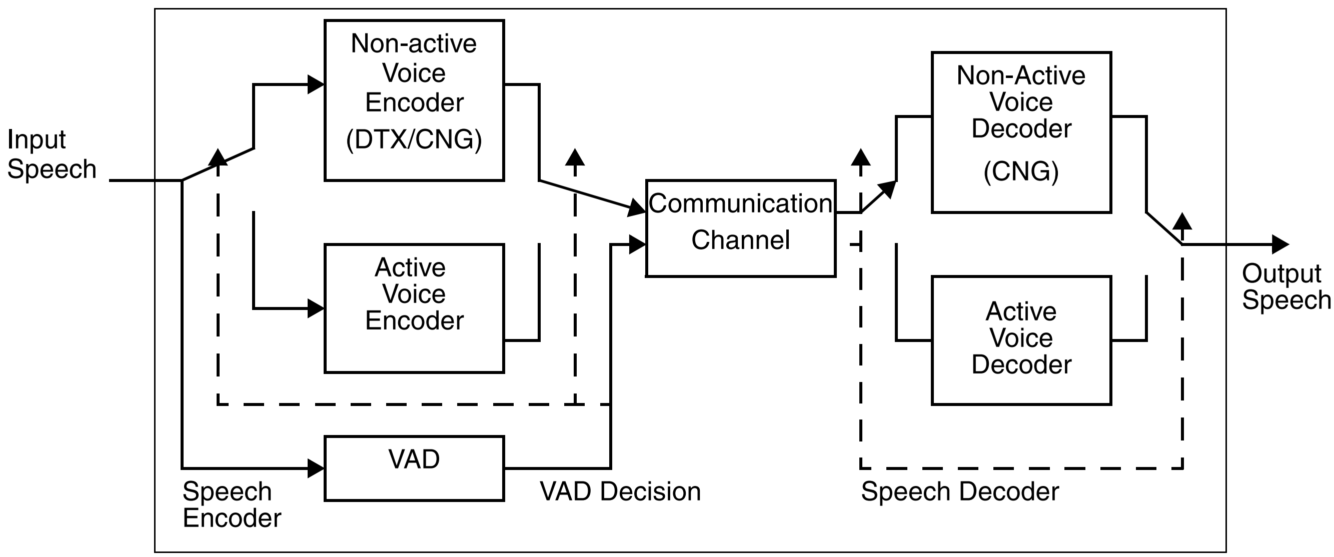

Figure 1. Speech Communication System with VAD

- *Discontinuous transmission*. Receives the active/non-active voice information from the VAD module and sends a set of non-active voice update parameters to the speech decoder by measuring the changes in the non-active voice signal. The update decision is based on the absolute and adaptive thresholds on the frame energy and the spectral distortion measure. If an update is required, the encoder sends information to generate a signal similar to the original non-active voice signal. This information comprises an energy level and a description of the spectral envelope. If no update is required, the decoder generates the signal on the basis of the last received energy level and the spectral shape information of the non-active voice frame. The decision output of the DTX module is as follows:
  — 0 = untransmitted frame

1. The optimization techniques, along with specific code examples, are described in detail in the following Freescale application notes: *ITU-T G.729 Implementation on the StarCore SC140/SC1400 Core* (AN2094) [4] and *ITU-T G.729AB Implementation on the StarCore SC140/SC1400 Cores* (AN2261) [7].

— 1 = active speech frame

— 2 = Silence Insertion Descriptor (SID) frame

- *Comfort noise generator*. The decoder reproduces non-active voice frames based on the decisions of the VAD (output 0) and DTX modules at the encoder. The comfort noise is generated by introducing a pseudo-white excitation signal of a controlled level into interpolated LPC filters, in the same way that the decoder produces active speech by filtering the decoded excitation. The excitation level and LPC filters are obtained from the previous SID information.

2 Implementation Process

Table 1 summarizes the main phases of the implementation process.

Table 1. Implementation Phases

Implementation phase	Description
Porting ITU-G729B to the SC140 core	Adding new functions, channel data, and tables. Providing data type definitions and introducing pragmas and intrinsic functions.
Integrating and reusing the existing C functions	The C functions from G729 (already optimized) identical to their corresponding function in Annex B are integrated and reused.
Integrating and reusing the existing ASM functions	The ASM functions from G729 (already optimized) identical to their corresponding function in Annex B are integrated and reused.
Function-level C optimization	C optimization techniques are applied for the new functions.

2.1 Differences Between ITU-T G.729 and ITU-T G.729B

The differences between Annex B of the ITU-T G.729 Recommendation and the ITU-T G.729 Recommendation consist of added channel data, functions, and tables for implementing Annex B. However, some data and functions must be changed, as summarized in the following sections. Comparing the reference C code from the ITU-T is helpful in porting and integrating Annex B with the ITU-T G.729 optimized implementation.The compiler for implementing and porting ITU-T G.729B is Metrowerks® CodeWarrior® for StarCore, Release 1.5.

Porting Annex B to the SC140 core required the use of the following elements:

- Defined data types (for example, Word16, Word32) to comply with theSC140 architecture.
- Intrinsic functions defined by the compiler instead of their emulated versions.
- Overflow handling intrinsics instead of the Boolean overflow at the base of the emulated functions.

Integrating Annex B with the ITU-T G.729 optimized implementation requires the following steps:

1. Integrate the new functions in the ITU-T G.729 optimized implementation by adding the calls and code of these functions. Split these functions into files, thus using the code structure of the ITU-T G.729 optimized implementation.
2. Add the modified code corresponding to the changed functions as they are implemented in the C reference code, optimizing them in the next project implementation phase.
3. Add/modify the data structures of the channel data with the new global and static variables from each new/changed added function
4. Add the new Annex B-defined tables.

Table 2 summarizes the results of the SC140 porting phase of the ITU-T G.729B. The results are obtained using only C versions of the functions.

Table 2. Performance After Porting to the SC140 Core

Implementation	Speed (MCPS)	Code Size (KB)	Data Size (Bytes)	Compiler Options
C only	24.00	45.7	6580	–O3
C only	18.91	52.3	6580	–Og –O3

2.2 Integrating New Functions

The C reference code of the ITU-T G.729 and ITU-T G.729B recommendations are compared at the function level. **Table 3** summarizes the results of the comparison. This analysis is used for integrating the new functions in the ITU-T G.729 optimized implementation. The functions are integrated in conjunction with the modification of the channel data and data tables. The new functions implement three algorithms of the Annex B (VAD, DTX, and CNG) and the quantization algorithms for computing the SID. According to **Table 3**, besides the new functions, only a few functions differ (so the optimizations applied may differ), and most are identical. The differences consist of new added computations.

Table 3. Differences Between G.729 and G.729B Functions

Encoder Functions		Common Functions		Decoder Functions	
Function Name	Diffs	Function Name	Diffs	Function Name	Diffs
ACELP_Codebook	same	Copy	same	calc_rc0_h	same
Az_lsp	same	Gain_predict	same	calc_st_filt	same
Chebps_10	same	Gain_update	same	compute_ltp_l	same
Chebps_11		Get_lsp_pol	same	Dec_gain	same
Check_Parity_Pitch	same	Int_qlpc	same	Dec_lag3	same
Cor_h	same	Inv_sqrt	same	Decod_ACELP	same
Cor_h_X	same	Log2	same	D_lsp	same
corr_xy2	same	Lsp_Az	same	filt_mu	same
D4i40_17	same	Lsp_expand_1_2	same	filt_plt	same
Enc_lag3	same	Lsp_get_quant	same	Gain_update_erasure	same
Gbk_presel	same	Lsf_lsp2	same	Lsp_iqua_cs	same
G_pitch	same	Lsp_lsf2	same	Lsp_decw_reset	same
Interpol_3	same	Lsf_lsp	same	Post_Process	same
Lag_max	same	Lsp_lsf	same	pst_ltp	same
Lsp_expand_1	same	Lsp_prev_extract	same	scale_st	same
Lsp_expand_2	same	Lsp_prev_update	same	search_del	same
Lsp_get_tdist	same	Lsp_stability	same	select_ltp	same
Lsp_last_select	same	Pow2	same	Dec_cng	new
Lsp_pre_select	same	Pred_lt_3	same	Get_decfreq_prev	new
Lsp_prev_compose	same	Residu	same	sid_lsfq_decode	new
Lsp_qua_cs	same	Set_zero	same	Decod_ld8k	diff
Lsp_select_1	same	Syn_filt	same	Post	diff

Table 3. Differences Between G.729 and G.729B Functions (Continued)

Encoder Functions		Common Functions		Decoder Functions	
Function Name	**Diffs**	**Function Name**	**Diffs**	**Function Name**	**Diffs**
Lsp_select_2	same	Weight_Az	same		
Norm_Corr	same	Calc_exc_rand	new		
Parity_Pitch	same	Copy2x	new		
perc_var	same	Gauss	new		
Pitch_fr3	same	Quant_Energy	new		
Pitch_ol	same	Qua_Sidgain	new		
Pre_Process	same	Sqrt	new		
Qua_gain	same	Random	diff		
Qua_lsp	same				
Relspwed	same				
update_exc_err	same				
Calc_pastfilt	new				
Calc_RCoeff	new				
Calc_sum_acf	new				
Cmp_filt	new				
Cod_cng	new				
Convolve	new				
Get_freq_prev	new				
lsfq_noise	new				
MakeDec	new				
New_ML_search_1	new				
New_ML_search_2	new				
Qnt_e	new				
Update_cng	new				
Update_freq_prev	new				
Update_sumAcf	new				
vad	new				
Coder_ld8k	diff				
Levinson	diff				

2.3 Modifying the Channel Data

The modified channel data consists of global data used by both the encoder and decoder of the speech codec to perform the following tasks:

- Decode the SID information
- Make the DTX decision
- Make the VAD decision
- Generate the comfort noise (CNG algorithm)

The changed functions required additions of data to the global data for Linear Prediction Coding (LPC), Linear Spectral Pair (LSP), and LSP quantization.

2.4 Adding the Annex B-Defined Tables

Some changes were applied to the initial G729 table set in order to integrate the ITU-T Annex B Recommendation:

- Word32 **lag[]**. Two values added, due to G729B implementation
- Word16 **freq_prev_reset[]**. New vector.

2.5 Verifying Bit Exactness Using Test Vectors

ITU provides a set of test vectors to verify the bit exactness of the encoder and decoder in the G.729B vocoder. Other internal Freescale test vectors are also used. In addition, the optimized implementation of ITU-T G.729B maintains the bit exactness of the ITU-T G.729B Recommendation when the VAD is disabled. The ITU test vectors are summarized in the **Table 4**.

Table 4. ITU-T G729B Test Vectors

Encoder Input	Encoder Output/Decoder Input	Encoder Output
Tstseq1.bin Tstseq2.bin Tstseq3.bin Tstseq4.bin	Tstseq1.bit Tstseq2.bit Tstseq3.bit Tstseq4.bit Tstseq5.bit Tstseq6.bit	Tstseq1.out Tstseq2.out Tstseq3.out Tstseq4.out Tstseq5.out Tstseq6.out

3 Results

Table 5 summarizes the performance results of the ITU-T G.729B optimized implementation on the StarCore SC140 core. All C optimized functions are compiled for speed (`-O3` compiler flag), thus obtaining a higher speed with only a small increase in code size. As **Table 5** shows, the values for the C-only implementations differ from the implementations using assembly versions of the functions in terms of code size and speed. The assembly versions of some functions from the ITU-T G.729B implementation provided the basis for performance enhancements.

Table 5. Performance Results of the G729B Optimized Implementation

Implementation	Speed (MCPS)	Code Size (KB)	Data Size (Bytes)	Compiler Options
C only	11.30	53.0	6672	-O3
C and Assembly	8.32	48.7	6576	-O3

4 References

[1] ITU-T Recommendation G.729—*Coding of Speech at 8 kbit/s Using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP*), March 1996.

[2] ITU-T Recommendation G.729/Annex A—*Reduced Complexity 8 kbit/s CS-ACELP Speech Code*c, November 1996.

[3] ITU-T Recommendation G.729/Annex B— *A Silence Compression Scheme for G.729 Optimized for Terminals Conforming to Recommendation V.70*, November 1996.

[4] *TU-T G.729 Implementation on StarCore SC140*, Freescale, AN2094.

[5] *TU-T G.729A Implementation on StarCore SC140*, Freescale, AN2151.

[6] *implementing the Levinson-Durbin Algorithm on the SC140*, Freescale, AN2197.

[7] *ITU-T G.729AB Implementation on StarCore SC140*, Freescale, AN2261.

[8] *Speech coders filters on StarCore SC140,* Freescale, AN2152.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations not listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GMBH
Technical Information Center
Schatzbogen 7
81829 München, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
+800 2666 8080

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN2278
Rev. 1
11/2004