Optimizing Speech and Audio Codecs on C55x and C64x DSPs

Claude Gravel
VP Engineering

VoiceAge®
The World's Premier Supplier of Speech and Audio Codecs
Contents

- Introduction
- Speech and Audio Models
- AMR-WB+ Hybrid Audio and Speech Coding Model
- AMR-WB+ Performance Results
- Implementation Optimization on TI Platforms
- Conclusions + Q&A
**VoiceAge Corporation – Who are we?**

**Business**: Low bit rate audio compression technologies research, IPR licensing and optimized implementations development

**Headquarters**: Montreal, Canada

**Technologies**
- AMR: 3GPP narrowband voice codec for GSM and WCDMA
- AMR-WB: 3GPP, ITU-T wideband voice codec
- VMR-WB: 3GPP2 wideband voice codec
- AMR-WB+: 3GPP audio codec

**Achievements**: Won every international audio compression standard for which VoiceAge competed in the last 10 years at 3GPP, 3GPP2, ITU, ETSI, TIA

**Implementations**: World-class optimized implementations and proprietary solutions on multiple O/S and processors/platforms (including TI & ARM based systems)

**Deployment**: More than 1.7B mobile phones and over 500M PCs currently use VoiceAge’s technologies
International Standards Using ACELP®

[Diagram showing various international standards organizations and technologies connected to ACELP®]

Established standard
Standardization in progress

Technology for Innovators®

Texas Instruments
Contents

• Introduction
• Speech and Audio Models
  • AMR-WB+ Hybrid Audio and Speech Coding Model
  • AMR-WB+ Performance Results
• Implementation Optimization on TI Platforms
• Conclusions + Q&A
Speech Signal

– Basically, same synthesis model for everyone
– So, speech has a “universal” structure or signature

Voiced fricative
• quasi periodic + noise
• lower energy

Purely Voiced
• quasi periodic
• high energy
• more energy in low frequency
• strongly correlated

Unvoiced
• non periodic
• low energy
• more energy in high frequency
• uncorrelated

Transient
• variable energy
• fast spectral evolution

1.25 sec
Speech Synthesis Model

1. Air from lungs
2. Vocal chords (periodicity)
3. Vocal tract (mouth + lips)

1. Innovative excitation
2. Long-term Prediction
3. Short-term Prediction

Synthesized speech

Minds in Motion

Technology for Innovators™
Audio Signal

– No underlying synthesis model
– What you call music, I may call noise (and vice versa)
– Speech coders not well adapted to music

1.25 sec

180 ms

Typical audio segment
• More stationary
• More complex (structure less visible than speech)

There are typically fewer “pauses” in music than speech
• VAD/DTX don’t work well

Both lower and higher frequency content
• More “presence” (LF)
• More “crispness” (HF)
• Larger bandwidth to encode
Contents

• Introduction
• Speech and Audio Models
• AMR-WB+ Hybrid Audio and Speech Coding Model
• AMR-WB+ Performance Results
• Implementation Optimization on TI Platforms
• Conclusions + Q&A
AMR-WB+ Hybrid Approach

Transform Audio Coders

Speech Coders

Bandwidth (kHz)
Content broadness

Bit Rate (kbps)
AMR-WB+ Technical Highlights

- Backward compatibility with AMR-WB, the mandatory wideband codec in 3GPP Release 5 specification
- Hybrid ACELP/TCX coding model
  - ACELP (Algebraic CELP)
  - TCX (Transform Coded eXcitation) with Algebraic VQ (AVQ)
- Adaptive window length with superframes of 80 ms
- Closed-loop/open-loop excitation mode decision
- Bandwidth extension using low bit rate
- Parametric stereo coding (HeHvS)
- Efficient error concealment against packet losses
- Use of over-/underclocking concepts and AVQ features
  - Codec flexibility/tunability
  - Bit rate scalability (6 to 48 kbps)
  - Bandwidth scalability (6.4 to 19.2 kHz)
AMR-WB+ Decoder Architecture

DEMUX

HF decoder

HF decoder

ACELP/TCX decoder

stereo decoder

Stereo M,S to L,R

Stereo image signal

mixed signal

left HF param

mono/right HF param

mono/mixed LF param

stereo param

left signal

right signal

HF signals folded 0..ISF/4 Hz

LF signals 0..ISF/4 Hz

post-processing and filtering
Sampling and Bandwidth

- Input signal downsampled to internal sampling frequency (ISF) [14.4 – 38.4 kHz] (nominal 25.6 kHz)
- Codec operates on 2048-sample superframes – [160 – 53.33 ms] (nominal 80 ms)
- The superframe is split into two bands of 1024 samples – Low band: 0 – ISF/4
  – High band: ISF/4 – ISF/2
Sampling and Bandwidth

- ISF can be set to limit the encoded signal when the bit rate is reduced.
- ISF can be chosen to exceed the signal bandwidth to maximize the fully encoded band with the core codec.
Encoder Mode of Operation

- ACELP (256 samples)
- TCX (256+32 samples)
- ACELP (256 samples)
- TCX (256+32 samples)
- ACELP (256 samples)
- TCX (256+32 samples)
- TCX (512+64 samples)
- TCX (1024+128 samples)

Time:
- 256 samples
- 512 samples
- 1024 samples
- 32 samples
- 64 samples
- 128 samples
Contents

• Introduction

• Speech and Audio Models

• AMR-WB+ Hybrid Audio and Speech Coding Model

• AMR-WB+ Performance Results

• Implementation Optimization on TI Platforms

• Conclusions + Q&A
Results from 3GPP Characterization

AMR-WB+ provides a 25%-40% BW efficiency improvement for low bit rate quality audio delivery.

Source: 3GPP TR 26.936
3rd Generation Partnership Project; Technical Specification Group SA WG4;
Performance Characterization of 3GPP Audio Codecs (Release 6)
NPR Subjective Testing Results

From: www.npr.org/euonline/pub/low_bit_rate_coder_report.pdf
3GPP Complexity Comparison

Stereo Decoder

Mono Decoder

Source: 3GPP Standardization Specifications
Based on fixed-point implementations
Contents

• Introduction
• Speech and Audio Models
• AMR-WB+ Hybrid Audio and Speech Coding Model
• AMR-WB+ Performance Results
• Implementation Optimization on TI Platforms
• Conclusions + Q&A
Optimization Challenges

• Codec standards reference code is based on 16-bit operations
• High complexity
• Extensive data usage (quantization tables and filter coefficients)
• Multiple TI platforms
Optimization Overview

Phases

• Conversion from float or basic-ops to “C” fixed point using intrinsic functions
• C code optimization
• Assembly optimization
• Further algorithm optimization (not necessarily preserving bit-exactness)
“Basic operators” are a generic set of functions simulating DSP instructions (saturation, normalization, multiply&accumulate, …)

– Replace basic operators by intrinsic arithmetic, +,-,*,/ and shift instructions. Keep saturation.

– For “C” code an “efficient” conversion, recognizing areas where overflow testing can be discarded or moved outside loops

```c
for (i = 0; i < lg; i++) {
    L_tmp = 0;
    for (j = 0; j < L_FIR; j++)
        L_tmp = L_mac(L_tmp, x[i + j], fir_6k_7k[j]);
    signal[i] = round(L_tmp);
}
```

```c
Word32 L_mac (Word32 L_var3, Word16 var1, Word16 var2){
    Word32 L_var_out;
    Word32 L_product;

    L_product = L_mult (var1, var2);
    L_var_out = L_add (L_var3, L_product);
    return (L_var_out);
}
```

```c
Word32 L_mult (Word16 var1, Word16 var2){
    Word32 L_var_out;

    L_var_out = (Word32) var1 *(Word32) var2;
    if (L_var_out != (Word32) 0x40000000L) {
        L_var_out *= 2;
    } else {
        Overflow = 1;
        L_var_out = MAX_32;
    }

    return (L_var_out);
}
```
C Code Optimization

• Structure C code as if writing in assembler (helps compiler do a better job)
  – Look at the disassembly code generated by the compiler (hint)

• Use “restrict” key word to avoid memory dependencies

• Use unsigned int for loop counters

• Use short data for multiplication inputs whenever possible

• Loop unrolling
C Code Optimization

• Give more information about loops (min, max and multiple of loop counts) to the compiler

• Rearrange the order of code (especially in loop) and introduce more variables to reduce data dependencies and encourage parallelism

• Use 32-bit access to operate on 2X16-bit data (alignment needed)

• Modify the code to use dual multiplication on C55x (axb; axc)

• Modify the code to use dual multiplication on C64x (2X16 bits multiplication)
Assembly Optimization

More effective use of registers, pipeline, parallelism, etc.

• Use additional intrinsic functions (besides the intrinsic / basic-op functions)

• Use pure assembly for complex code

• Decide on the amount of assembly that you want (time vs MIPS)
Optimization Summary

• The overall goals of optimization are to reduce the amount of load & store from memory and make efficient use of data while in registers

• Going from 16 to 32 bits
  – Eases the process
  – Helps reduce overflow and saturation checking
  – Reduces the complexity of those operations

• Compromising bit exactness to reduce complexity
## Results

<table>
<thead>
<tr>
<th>Platform</th>
<th>Basic-op (MHz)</th>
<th>Compiler Optimization (MHz)</th>
<th>Intrinsic Functions (MHz)</th>
<th>C Code (MHz)</th>
<th>Assembler (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C55X</strong></td>
<td>3212 MHz</td>
<td>1105 MHz</td>
<td>61 MHz</td>
<td>44 MHz</td>
<td>24 MHz</td>
</tr>
<tr>
<td></td>
<td>10 person-days</td>
<td>1 days</td>
<td>10 days</td>
<td>45 days</td>
<td>85 days</td>
</tr>
<tr>
<td><strong>C64X</strong></td>
<td>2104 MHz</td>
<td>643 MHz</td>
<td>58 MHz</td>
<td>23 MHz</td>
<td>16 MHz</td>
</tr>
<tr>
<td></td>
<td>15 person-days</td>
<td>1 days</td>
<td>15 days</td>
<td>60 days</td>
<td>85 days</td>
</tr>
</tbody>
</table>

*AMR-WB+ decoder complexity (MHz)*
## Results

### AMR-WB complexity (MHz)

<table>
<thead>
<tr>
<th>Platform</th>
<th>basic-op</th>
<th>compiler optimization</th>
<th>intrinsic functions</th>
<th>C code</th>
<th>assembler</th>
</tr>
</thead>
<tbody>
<tr>
<td>C55X</td>
<td>2408 MHz</td>
<td>945 MHz</td>
<td>64 MHz</td>
<td>36 MHz</td>
<td>29 MHz</td>
</tr>
<tr>
<td></td>
<td>10 person-days</td>
<td>1 days</td>
<td>10 days</td>
<td>20 days</td>
<td>70 days</td>
</tr>
<tr>
<td>C64X</td>
<td>1839 MHz</td>
<td>680 MHz</td>
<td>62 MHz</td>
<td>32 MHz</td>
<td>24 MHz</td>
</tr>
<tr>
<td></td>
<td>15 person-days</td>
<td>1 days</td>
<td>15 days</td>
<td>35 days</td>
<td>50 days</td>
</tr>
</tbody>
</table>

**Notes:**
- AMR-WB complexity (MHz)
- Assembler, C code, intrinsic functions, basic-op, compiler optimization, basic-opplatform, functions, compiler, optimization, basic-opplatform.
Contents

• Introduction
• Speech and Audio Models
• AMR-WB+ Hybrid Audio and Speech Coding Model
• AMR-WB+ Performance Results
• Implementation Optimization on TI Platforms
• Conclusions + Q&A
Tools

• DSK for C55 and C64
  – Efficient and inexpensive
  – Code composer debugger and simulator
  – Real-time debugging and testing
  – C std lib library (fread/fwrite) eases the testing

• Code Composer (build tools)
  – Efficient compiler
Conclusions

• Intrinsic functions give you the opportunity to come up with an implementation in a timely manner
• But to get powerful optimized implementations, you need to use assembly language
  – Quite useful on C55
  – Difficult on C64 but compiler already doing a good job
• Can skip C code optimization to jump to assembly directly
• Gain of 2X for C55 and 3X for C64
Thank You!
Download the AMR-WB+ Demo Executable

Go to:
www.voiceage.com/amrwbpplus.php

Download the AMR-WB+ demo executable

Generate encoded/decoded WAV files at different bit rates
Optimizing Speech and Audio Codecs on C55x and C64x DSPs

claude.gravel@voiceage.com