Optimal Implementation of MPEG4 HEAAC v2 Decoder on C64x+ DSPs

Tejaswi N & Ameya Potadar
Ittiam Systems Pvt. Ltd.
tejaswi.n@ittiam.com
ameya.potadar@ittiam.com
Agenda

- Overview of HEAAC v2 technology
- Generic optimization approach
- Optimization details of HEAAC v2 decoder blocks
- Conclusions
Agenda

- Overview of HEAAC v2 technology
- Generic optimization approach
- Optimization details of HEAAC v2 decoder blocks
- Conclusions
HEAAC v2

- AAC - a psycho acoustic based transform codec
- Transparent quality at 128 kbps, good quality at 96 kbps for stereo 44.1kHz audio streams
- Combination of SBR and AAC called HEAAC v1, here high frequency part of the spectrum is replicated from low-band
- Good quality at 48 to 64 kbps, used widely in DMB applications
- Combination of AAC, SBR and PS called HEAAC v2, here stereo channel redundancy is exploited
- Good quality at 32 kbps, used in mobile streaming
HEAAC v2

**ENCODER**

- Stereo PCM data
- Parametric Stereo Coding
- Mono data
- High Frequency Spectral Band Replication Coding
- Mono Low Band data
- Core AAC Encoder
- AAC Stream
- Bit stream Muxer
- HEAAC v2 Stream
- Parameters for high frequency generation
- Parameters for stereo regeneration

**DECODER**

- HEAAC v2 Bit stream
- Demuxer
- AAC Stream
- Core AAC Decoder
- Mono Low Band data
- High Frequency Spectral Band Replication
- Mono data
- Parameters for high frequency generation
- Parameters for stereo regeneration
- Parametric Stereo Decoding
- Stereo PCM data
- Bit stream

HEAAC v2

- **AAC Decoder**
  - Entropy decoding, Inverse quantize, Apply encoder tools, IMDCT

- **SBR Decoder**
  - Analysis to QMF (Quadrature Mirror Filter) bank domain, HF (High Frequency) generation, HF adjustment, HF addition, Synthesis to time domain

- **PS Decoder**
  - Analysis to higher resolution QMF banks, De-correlation, Apply rotation, Hybrid synthesis
Agenda

- Overview of HEAAC v2 technology
- Generic optimization approach
- Optimization details of HEAAC v2 decoder blocks
- Conclusions
Codec Development

1. Algo / Matlab / Simulation
2. Floating Point C Model
3. Optimization and Fixed Point C Conversion
4. Fixed Point C Model

Generic Optimization
Codec Development

1. Platform Specific C Optimization
2. Adding Intrinsic's to Basicops
3. Profiling to Identify Key functions
4. Using C Compiler Directives
5. Linear Assembly Coding
6. Hand Assembly Coding
Optimization Phases

- **Compile on CCS the fixed point CMODEL**
  - Build options used
    - -O3
    - No Debug
    - -mt
    - Speed Most Critical

- **Add intrinsic to Basic Operations**
  - Commonly used intrinsic
    - _sshl, _smpy, _sshvl, _sshvr, _min2, _max2, _sadd, _ssub, _norm, _abs, _mpylir, _mpyhir, _mpyluhs
Optimization Phases

- **Profile breakup to find high complexity modules**
  - Classify modules to different types
    - Only C optimizations
    - Standard Library available (DSPLIB)
    - Linear assembly coding required
    - Hand assembly coding required
Optimization Phases

- **Standard Library usage for available functions**
  - E.g. FFT

- **C level optimizations**
  - Change C code to suit c64x instructions (shifts, mults)
  - Removing indirection from loops
  - “restrict” keyword usage
  - Specify double word alignment
  - #pragma MUST_ITERATE and UNROLL
Optimization Phases

◆ **Linear assembly coding**
  - Coded if addition of intrinsic has not lead to best kernels or if SIMD instructions can be used
  - Compiler feedback used to reduce bottlenecks

◆ **Hand assembly coding**
  - Should be used as a last resort
  - Coded only if nested loops or control code present
  - Prologue and Epilogue merge of outer loop
Agenda

- Overview of HEAAC v2 technology
- Generic optimization approach
- Optimization details of HEAAC v2 decoder blocks
- Conclusions
AAC Decoder Blocks

- **Entropy Decoding**
  - Huffman decoding, Loop with control code

- **Inverse quantization**
  - Power (x, 4/3), Loop with lookup table and mults
  - Apply scale factors x * power(2, sf/4), Loop with shifts and mults

- **Encoder optional tools**
  - IS/MS: stereo redundancy, Loop with add, sub and mults
  - TNS: better transient coding, Nested loop of IIR filtering
  - PNS: Noise generation, Loop with mults

- **IMDCT and OLA**
  - Transformation to time domain (PreTwiddle+FFT+PostTwiddle), Loops with mults, shifts and adds
# AAC Decoder Blocks

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
<th>Approach</th>
<th>Initial MCPS</th>
<th>Final MCPS</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy Decoding of spectral and scale factor data</td>
<td>Control code</td>
<td>Intrinsic C and Compiler Directives</td>
<td>3.0</td>
<td>0.7</td>
<td>4.3x</td>
</tr>
<tr>
<td>Inverse Quantize and Apply Scale Factors</td>
<td>Loops with mac and shifts</td>
<td>Intrinsic C and Compiler Directives</td>
<td>1.5</td>
<td>0.4</td>
<td>3.8x</td>
</tr>
<tr>
<td>IS, MS, PNS</td>
<td>Loops with add, mults, norms</td>
<td>Intrinsic C and Compiler Directives</td>
<td>0.5</td>
<td>0.1</td>
<td>5.0x</td>
</tr>
<tr>
<td>TNS</td>
<td>Nested loop with mac and shifts</td>
<td>Hand Assembly Coded</td>
<td>1.5</td>
<td>0.4</td>
<td>3.8x</td>
</tr>
<tr>
<td>FFT</td>
<td>Loops with add, mults and shifts</td>
<td>Standard library</td>
<td>3.0</td>
<td>0.4</td>
<td>7.5x</td>
</tr>
<tr>
<td>Pre/Post twiddle for IMDCT and OLA</td>
<td>Loops with mac and shifts</td>
<td>Intrinsic C and Compiler Directives</td>
<td>1.0</td>
<td>0.3</td>
<td>3.3x</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>10.5</td>
<td>2.3</td>
<td>4.6x</td>
</tr>
</tbody>
</table>
SBR Decoder Blocks

- **Analysis Filter bank**
  - Window and Add, Loop with 16x16 macs
  - DCT as pre/post twiddle and FFT, Loops with mults, shifts and adds

- **HF Generation**
  - Auto correlation and LPC calculation, Loop with macs and control code
  - LPC filtering, Loop with macs

- **HF adjustment**
  - Calculation of signal and noise/tone gains, control code
  - Adjust time slot, Loop with mult and adds

- **Synthesis Filter bank**
  - DCT as pre/post twiddle and FFT, Loops with mults, shifts and adds
  - Window and Add, Loop with 16x16 macs
## SBR Decoder Blocks

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
<th>Approach</th>
<th>Initial MCPS</th>
<th>Final MCPS</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis and Synthesis Filter bank DCT twiddling and rescaling</td>
<td>Loops with add, mults and shifts</td>
<td>Intrinsic C and Compiler Directives</td>
<td>8.0</td>
<td>1.9</td>
<td>4.2x</td>
</tr>
<tr>
<td>Analysis and Synthesis window and add</td>
<td>Loops with 16x16 mac</td>
<td>Linear Assembly Coded</td>
<td>2.0</td>
<td>0.6</td>
<td>3.3x</td>
</tr>
<tr>
<td>High frequency generation</td>
<td>Loops with add, mults and shifts</td>
<td>Intrinsic C and Compiler Directives</td>
<td>3.0</td>
<td>0.5</td>
<td>6.0x</td>
</tr>
<tr>
<td>High frequency adjustment</td>
<td>Loops with add, mults, shifts and control code</td>
<td>Intrinsic C and Compiler Directives</td>
<td>7.0</td>
<td>2.1</td>
<td>3.3x</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>20</strong></td>
<td><strong>5.1</strong></td>
<td><strong>3.9x</strong></td>
</tr>
</tbody>
</table>
PS Decoder Blocks

- **Hybrid Analysis**
  - Further analysis of lower bands for better frequency resolution, Loops with macs

- **De-correlation**
  - Generation of orthogonal vector
  - All pass filter with constant phase delays for each sub band, Loops with 16x16 macs

- **Apply Rotation**
  - Using the ICC and IID parameters to generate the rotation matrix, Loops with mults, table lookups and adds
  - Apply rotation matrix on the orthogonal vectors to get back the original intensity and time difference, Loops with mults, adds and shifts

- **Hybrid Synthesis**
  - Synthesis of lower bands, Loop with adds
## PS Decoder Blocks

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
<th>Approach</th>
<th>Initial MCPS</th>
<th>Final MCPS</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid Analysis</td>
<td>Loops with add and mults</td>
<td>Intrinsic C and Compiler Directives</td>
<td>2.0</td>
<td>0.5</td>
<td>4.0x</td>
</tr>
<tr>
<td>De correlation</td>
<td>Loops with 16x16 mac</td>
<td>Linear Assembly Coded</td>
<td>5.3</td>
<td>1.0</td>
<td>5.3x</td>
</tr>
<tr>
<td>Apply Rotation</td>
<td>Loops with add, mults and shifts</td>
<td>Linear Assembly Coded</td>
<td>1.0</td>
<td>0.3</td>
<td>3.3x</td>
</tr>
<tr>
<td>Hybrid Synthesis</td>
<td>Loops with adds</td>
<td>Intrinsic C and Compiler Directives</td>
<td>0.3</td>
<td>0.1</td>
<td>3.0x</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>8.6</strong></td>
<td><strong>1.9</strong></td>
<td><strong>4.5x</strong></td>
</tr>
</tbody>
</table>
Agenda

- Overview of HEAAC v2 technology
- Generic optimization approach
- Optimization details of HEAAC v2 decoder blocks
- Conclusions
MCPS vs. Effort

Our approach
Hand assembly

Effort (in weeks)

MCPS
## MCPS vs. Effort

### Our approach

<table>
<thead>
<tr>
<th>Stages</th>
<th>MCPS</th>
<th>Effort (in weeks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross compile on CCS</td>
<td>51.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Intrinsic for Basicops</td>
<td>39.1</td>
<td>1.0</td>
</tr>
<tr>
<td>Standard Library</td>
<td>36.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Compiler Directives - AAC part</td>
<td>32.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Hand Asm - AAC part</td>
<td>30.9</td>
<td>1.0</td>
</tr>
<tr>
<td>Compiler Directives - SBR part</td>
<td>17.4</td>
<td>2.5</td>
</tr>
<tr>
<td>Linear Asm - SBR part</td>
<td>16.0</td>
<td>0.5</td>
</tr>
<tr>
<td>Compiler Directives - PS part</td>
<td>14.3</td>
<td>1.5</td>
</tr>
<tr>
<td>Linear Asm - PS part</td>
<td>9.3</td>
<td>1.5</td>
</tr>
</tbody>
</table>

### Hand assembly optimization

<table>
<thead>
<tr>
<th>Stages</th>
<th>MCPS</th>
<th>Effort (in weeks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross compile on CCS</td>
<td>51.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Intrinsic for Basicops</td>
<td>39.1</td>
<td>1.0</td>
</tr>
<tr>
<td>Hand Asm - AAC Part</td>
<td>30.6</td>
<td>4.0</td>
</tr>
<tr>
<td>Hand Asm - SBR Part</td>
<td>15.5</td>
<td>13.0</td>
</tr>
<tr>
<td>Hand Asm - PS Part</td>
<td>8.6</td>
<td>4.0</td>
</tr>
</tbody>
</table>
## MCPS

<table>
<thead>
<tr>
<th>Description of stream</th>
<th>MCPS (Peak)</th>
<th>MCPS (Average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAC only</td>
<td>5.0</td>
<td>4.1</td>
</tr>
<tr>
<td>AAC+SBR</td>
<td>8.9</td>
<td>6.1</td>
</tr>
<tr>
<td>AAC+SBR+PS</td>
<td>9.3</td>
<td>8.2</td>
</tr>
</tbody>
</table>
Conclusions

- **Best numbers achieved with C level optimizations itself**
  - Minimal effort (reduced time to market)
  - Better code maintenance

- **Hand assembly coding if significant reduction expected**
References

- TMS320C6000 Optimizing Compiler User’s Guide (spru187k)
- TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide (spru732b)
Optimal Implementation of MPEG4 HEAAC v2 decoder on C64x+ DSPs

Tejaswi N & Ameya Potadar
Ittiam Systems Pvt. Ltd.
tejaswi.n@ittiam.com
ameya.potadar@ittiam.com
Backup Slides
SBR
SBR
SBR