Software techniques for low power embedded audio

Shyam Sadasivan, ARM
Yohann Le Guen, Ittiam Systems
Agenda items

- The embedded audio market
  - DSPs or MCUs?

- Features available for audio functionality in MCUs
  - Higher performance and functionality within ultra low power and cost

- The basic of audio technology
  - Requirements of a basic audio playback algorithm

- Optimizing audio codecs on microcontrollers
  - Case study of MP3 and WMA decode on Cortex-M3
  - Audio optimization techniques targeting the Cortex-M4 processor
Today’s embedded audio

Markets and products

Requirements

High-fidelity content
Surround sound enhancements
Heavy duty post-processing

Rich user experience
Multiple standard compatibility
Portability and battery life

Basic playback
Ultra-low cost
Very long battery life
What this means to embedded design

- Audio and speech signal processing is now done almost exclusively in the digital domain

- DSP (Digital Signal Processing) is heavily leaning on MAC (Multiply Accumulate) operations inside large loops
  - Time-domain processing (filter algorithms)
  - Or Frequency-domain processing (FFT algorithms)
- Yet typical audio algorithms have a lot of control code as well
- A blend of traditional 32-bit RISC processing and signal processing features makes a lot of sense
Digital signal controllers

MCU
Ease of use
C Programming
Interrupt handling
Ultra low power

Digital Signal Controller

DSP
Harvard architecture
Single cycle MAC
Floating Point
Barrel shifter
Efficient digital signal controller CPU

- **Cortex-M4 processor**
  - Thumb-2 Technology
  - DSP and SIMD instructions
  - Single cycle MAC (Up to 32 x 32 + 64 -> 64)
  - Optional decoupled single precision FPU
  - Integrated configurable NVIC
  - Compatible with Cortex-M3

- **Microarchitecture**
  - 3-stage pipeline with branch speculation
  - 3x AHB-Lite Bus Interfaces

- **Configurable for ultra low power**
  - Deep Sleep Mode, Wakeup Interrupt Controller
  - Power down features for Floating Point Unit

- **Flexible configurations for wider applicability**
  - Configurable Interrupt Controller (1-240 Interrupts and Priorities)
  - Optional Memory Protection Unit
  - Optional Debug & Trace
Single cycle MAC essential for audio

<table>
<thead>
<tr>
<th>OPERATION</th>
<th>INSTRUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$16 \times 16 = 32$</td>
<td>SMULBB, SMULBT, SMULTB, SMULTT</td>
</tr>
<tr>
<td>$16 \times 16 + 32 = 32$</td>
<td>SMLABB, SMLABT, SMLATB, SMLATT</td>
</tr>
<tr>
<td>$16 \times 16 + 64 = 64$</td>
<td>SMLALBB, SMLALBT, SMLALTB, SMLALTT</td>
</tr>
<tr>
<td>$16 \times 32 = 32$</td>
<td>SMULWB, SMULWT</td>
</tr>
<tr>
<td>$(16 \times 32) + 32 = 32$</td>
<td>SMLAWB, SMLAWT</td>
</tr>
<tr>
<td>$(16 \times 16)$ $(16 \times 16) = 32$</td>
<td>SMUAD, SMUADX, SMUSD, SMUSDX</td>
</tr>
<tr>
<td>$(16 \times 16)$ $(16 \times 16) + 32 = 32$</td>
<td>SMLAD, SMLADX, SMLSD, SMLSDX</td>
</tr>
<tr>
<td>$(16 \times 16)$ $(16 \times 16) + 64 = 64$</td>
<td>SMLALD, SMLALDX, SMLSLD, SMLSLDX</td>
</tr>
<tr>
<td>$32 \times 32 = 32$</td>
<td>MUL</td>
</tr>
<tr>
<td>$32 \ (32 \times 32) = 32$</td>
<td>MLA, MLS</td>
</tr>
<tr>
<td>$32 \times 32 = 64$</td>
<td>SMULL, UMULL</td>
</tr>
<tr>
<td>$(32 \times 32) + 64 = 64$</td>
<td>SMLAL, UMLAL</td>
</tr>
<tr>
<td>$(32 \times 32) + 32 + 32 = 64$</td>
<td>UMAAL</td>
</tr>
<tr>
<td>$32 \ (32 \times 32) = 32 \ (upper)$</td>
<td>SMMLA, SMMLAR, SMMLS, SMMLSR</td>
</tr>
<tr>
<td>$(32 \times 32) = 32 \ (upper)$</td>
<td>SMMUL, SMMULR</td>
</tr>
</tbody>
</table>

All the above operations are single cycle on the Cortex-M4 processor
Embedded audio possibilities

- Wide coverage of audio codecs
  - Portable Audio – MP3, AAC, HE-AAC, WMA, Vorbis
  - Performance optimizations possible for range of processors
  - High code density to optimize memory utilization
  - Can be optimized for MHz, Memory and Power utilization

- Audio effects and post processing
  - Special processing functions to improve the listening experience
  - Equalizer / Bass & Treble / Loudness
  - Stereo Widening / Virtual Surround / Reverb
  - Resampler / DRC / TSM / PSM

- All this can now be done on general purpose devices!
Low Power Audio Processing on Cortex-M Processors

Ittiam Systems Private Limited
Functional Blocks of Audio decoder

MP3 Input → Bit Stream Demux → Entropy Decode → IMDCT → Overlap & Add → Synthesis Filter Bank → PCM Output

Block Diagram of MP3 Decoder

- Bit Stream Decode – Performs MP3 bit stream demux
- Entropy Decode – Huffman Decode and Inverse Quantization
- IMDCT – Inverse of MDCT applied
- Overlap & Add – Performs Windowing and overlap & add
- Synthesis Filter Bank – Reconstruction of time domain samples from filter bank domain data

Audio processing blocks grouped as different categories

- MAC/Compute Intensive
- Control Code
- Mix of Compute & Control Code
Functional Blocks of Audio Codec

• **MAC/Compute Intensive Modules**
  - Primarily involves number crunching – MUL, MAC, SHIFT, NORM
    - Example: Filter, Overlap and Add, Windowing routines
  - Support for multiplicands with mixed bit widths
  - SIMD and Parallelism helps to improve performance
  - Saturation arithmetic is needed

• **Compute + Control Code Modules**
  - Compute intensive code mixed with control code
    - Example: FFT, MDCT, IMLT routines
  - Single cycle MAC instructions improve performance
  - Support for packed data processing
  - Instructions for – Bit reverse, loop control are of help

• **Control Code Modules**
  - Code flow is data dependent
    - Example: Huffman Decoding, Run Length Decoding
  - Bit manipulation instructions, Unsigned operations are useful
  - Better Code density improves the overall performance as well
• **32 Bit Processing**
  – Essential for meeting Standard compliance criteria
  – Precision of intermediate result higher than 32 bit

• **DSP instructions with SIMD capability**
  – For efficient implementation of signal processing routines
  – SIMD exploits nature of compute intensive routines

• **Support for operands with mixed bit widths**
  – Filter stability, output precision determines bit width
  – Need for operating on data of different bit widths

• **Support for Saturation Arithmetic**
  – Saturation arithmetic is required for 16bit PCM output
• **Support for packed data processing**
  – Audio routines involves operating on packed data like window, filter coefficients and twiddles

• **Support for unsigned operations**

• **Support for bit manipulation instructions**
  – Efficient bit stream processing routines require unsigned and bit manipulation operations
Optimizing Audio on Cortex-M3

- **Cortex-M3: Advantages for Audio Processing**
  - Single Cycle Multiply, Saturate, Bit manipulation instructions
    - *MUL, SSAT, REV*
  - High precision Multiply instructions - SMULL, SMLAL
  - Hardware Divide instructions - SDIV, UDIV
  - Code density advantage due to Thumb2

<table>
<thead>
<tr>
<th>Audio Codec</th>
<th>Code Size kB</th>
<th>Average MHz</th>
<th>Peak MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP3 Decoder (Stereo 48kHz 320 kbps)</td>
<td>20.8</td>
<td>19.73</td>
<td>19.93</td>
</tr>
<tr>
<td>WMA Decoder (Stereo 48kHz 320 kbps)</td>
<td>23.7</td>
<td>10.83</td>
<td>16.14</td>
</tr>
</tbody>
</table>

Cortex-M3 Numbers obtained on FPGA implementing zero wait state memory configuration
Optimizing with Cortex-M4 features

• **Salient Features**
  – DSP instructions with SIMD – SMUAD, SMLSLDX
  – Packed data processing – PKHTB, SHASX
  – Support for Saturated & Rounding Arithmetic
  – Support for Unsigned operations
  – Division instructions – SDIV, UDIV
  – Reverse and Saturate instructions

• **Incremental features over Cortex-M3**
  – Single cycle MAC instructions (64bit result)
  – Mixed bit width arithmetic – SMULWB, SMLAWT
  – Packed data operations – SMULBB, SMULTB
  – SIMD Operations – SMLAD, SMLSDX
Example - Window Overlap and Add

\[ x_{32}[i] = c_{32}[i] * w_{16}[i] + p_{32}[i] * v_{16}[i]; \]

\[ w_{16}, v_{16} - 16 \text{ bit window coefficients}; \quad 0 < i < \text{frame length} \]

\[ c_{32}, p_{32} - 32 \text{ bit input data from current and previous frame} \]

Cortex-M3 Code Segment:

WIN_OLA_LOOP:
LDR R3,[R0],#4 ; (2) Load input \( c_{32} \)
LDR R5,[R2],#4 ; (2) Load window \( w_{16}, v_{16} \)
LDR R4,[R1],#4 ; (2) Load input \( p_{32} \)
LSL R9, R5 ; (1) Extract \( v_{16} \)
AND R5, R5,#CNST ; (1) Extract \( w_{16} \)
SMULL R6, R7,R3,R5 ; (3-7) \( x_{32} = w_{16}[i] * c_{32}[i] \)
SMLAL R6, R7,R4,R9 ; (3-7) \( x_{32} += v_{16}[i] * p_{32}[i] \)
STR R7,[R10],#4 ; (1) Store 32bit output
SUBS R8, R8, #1 ; (1) loop count
BNE WIN_OLA_LOOP ; (2)

Note:
1. #CNST = #FFFF0000
2. * - SMULL, SMLAL cycles depend on data.
   (Early Terminate feature)

Cortex-M4 Code Segment:

WIN_OLA_LOOP:
LDR R3,[R0],#4 ; (1) Load input \( c_{32} \)
LDR R5,[R2],#4 ; (1) Load window \( w_{16}, v_{16} \)
LDR R4,[R1],#4 ; (2) Load input \( p_{32} \)
SMULWB R6, R3, R5 ; (1) \( t = c_{32}[i] * w_{32}[i] \)
SMLAWT R6, R6, R4,R5 ; (1) \( x = p_{32}[i] * v_{32}[i] + t \)
STR R6,[R7],#4 ; (1) Store 32bit output
SUBS R8, R8, #1 ; (1) loop count
BNE WIN_OLA_LOOP ; (2)

Processor | Kernel cycles | Total Cycles | Instructions
--- | --- | --- | ---
Cortex-M3 | 8-16* | 18-26 | 10
Cortex-M4 | 2 | 10 | 8
Advantage | 4x-8x | 1.8x-2.6x | ~1.25x
Example - FIR Filter

\[ y_{32}[n] = \sum_{i=0}^{N} x_{16}[n-i] \cdot c_{16}[i] \]

- \( y_{32} \) - filter output (32 bit); \( N \) - filter order
- \( x_{16} \) - 16 bit input data; \( c_{16} \) - 16 bit filter coeff

**Cortex-M3 Code Segment:**

FIR_LOOP:

- LDR R2,[R0],#4 ; (2) Load input \( x_{16} \)
- LDR R3,[R1],#4 ; (2) Load coeff \( c_{16} \)
- SXTH R4, R2 ; (1) Extract \( x_{16}[n-i] \)
- ASR R2, R2,#16 ; (1) Extract \( x_{16}[n-i-1] \)
- SXTH R5, R3 ; (1) Extract \( c_{16}[i] \)
- ASR R3, R3,#16 ; (1) Extract \( c_{16}[i+1] \)
- MLA R6, R4, R5 ; (2) \( y_{32} += x_{16}[n-i]\cdot c_{16}[i] \)
- MLA R6, R2, R3 ; (2) \( y_{32} += x_{16}[n-i-1]\cdot c_{16}[i+1] \)
- SUBS R7, R7, #2 ; (1) loop count -= 2
- BNE FIR_LOOP ; (2)

Note:
1. In these examples, FIR_LOOP is unrolled by 2
2. This example assumes number of taps is even.

**Cortex-M4 Code Segment:**

FIR_LOOP:

- LDR R2,[R0],#4 ; (1) Load input \( x_{16}[n-i], x_{16}[n-i-1] \)
- LDR R3,[R1],#4 ; (2) Load coeff \( c_{16}[i], c_{16}[i+1] \)
- SUBS R5, R5, #2 ; (1) loop count -= 2
- SMLAD R4, R2, R3 ; (1) \( y_{32} += x_{16}[n-i,n-i-1]\cdot c_{16}[i,i+1] \)
- BNE FIR_LOOP ; (2)

<table>
<thead>
<tr>
<th>Processor</th>
<th>Kernel cycles</th>
<th>Total Cycles</th>
<th>Number of Instructions</th>
<th>Register usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cortex-M3</td>
<td>8</td>
<td>15</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>Cortex-M4</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Advantage</td>
<td>8x</td>
<td>~2.2x</td>
<td>2x</td>
<td>1.4x</td>
</tr>
</tbody>
</table>
Energy efficient audio on Cortex-M4

- **Audio module improvements using Cortex-M4**
  - Core loop cycle advantage of 4x to 8x
    - *Mixed bit width arithmetic (DSP instruction)*
    - *Packed processing with SIMD*
  - Registers usage is comparatively less
  - Overall cycle advantage of approximately 2x
  - Lesser code size further improves performance

- **Cortex-M4 energy efficiency gain**
  - Chip can be clocked at less than half the MHz
  - Leads to longer battery life and higher energy efficiency
## Optimized Audio on Cortex-M4

<table>
<thead>
<tr>
<th>Audio Codec</th>
<th>Performance</th>
<th>Cortex-M3 MHz</th>
<th>M3 code running on Cortex-M4 MHz*</th>
<th>Estimated MHz for « M4 optimized » code*</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MP3 Decoder</strong> <em>(Stereo 48kHz 320 kbps)</em></td>
<td><em>Peak MHz</em></td>
<td>19.93</td>
<td>18</td>
<td>~9.5</td>
</tr>
<tr>
<td></td>
<td><em>Average MHz</em></td>
<td>19.73</td>
<td>17.83</td>
<td></td>
</tr>
<tr>
<td><strong>WMA Decoder</strong> <em>(Stereo 48kHz 320 kbps)</em></td>
<td><em>Peak MHz</em></td>
<td>16.14</td>
<td>14.8</td>
<td>~10</td>
</tr>
<tr>
<td></td>
<td><em>Average MHz</em></td>
<td>10.83</td>
<td>9.9</td>
<td></td>
</tr>
</tbody>
</table>

*Cortex-M3 numbers obtained on FPGA implementing zero wait state memory configuration*

* - All Cortex-M4 numbers are best estimates*
Summary and Conclusion

• **Cortex-M3 addresses low power audio markets**
  – Cortex-M3 MP3 and WMA decode in less than 20MHz

• **Cortex-M4 enables even longer battery life**
  – DSP instructions with SIMD capability
  – Instructions for mixed bit width arithmetic
  – Instructions for Packed processing and Saturated Arithmetic
  – Cortex-M4 MP3 and WMA decode in less than 10MHz

• **Low power audio is no longer for DSPs alone!**