Audio Decoding on C54X

Chuck Lueck

Alec Robinson

Jon Rowlands

Presentation Overview

- Platform Description
  - C54X DSP
- Algorithm Overview
  - MPEG-1 Layer 3 (MP3)
  - MPEG-2 Advanced Audio Coding (AAC)
- Implementation Details
  - What it takes to port to the C54X

Platform Description

- Development for C54X 16-bit fixed-point DSP
- Initial target was C5410 and goal is a C5409
- C5410 has 64 k RAM and no ROM
- C5409 has 32 k of RAM and 16 k ROM
- C5409 is lower cost and power
- Used Spectrum Digital C54X EVM for non-real time development, profiling, and compliance testing.
- Real-time development on TI Internet Audio C5410 EVM.

Algorithms Overview

- MPEG-1 Layer 3 (MP3)
- MPEG-2 Advanced Audio Coding (AAC)
- Supports stereo, 8-128 kbits/s, 8-48 kHz
- MP3 has recently gained popularity for downloading audio content via the internet.
- AAC has been adopted as the high-quality audio codec in MPEG-4, and for use in the future Japanese HDTV standard.

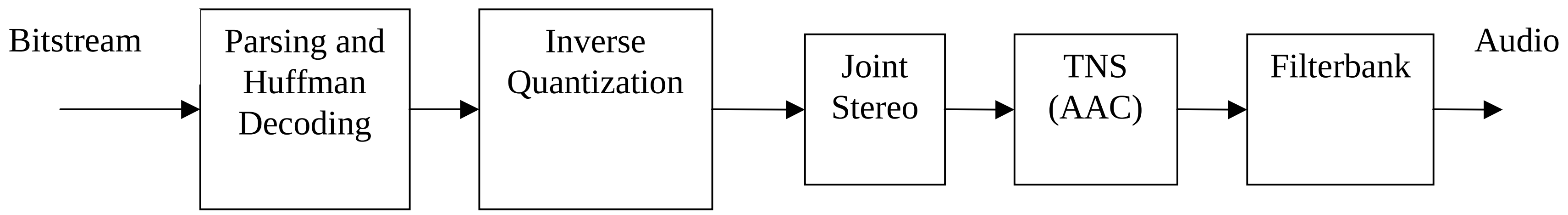
Implementation Details

- Started from a floating-point C reference running on the PC
- Compiled this code for the C54X and got in running on a Spectrum Digital EVM.
- Worked out dynamic range issues on PC
- Floating-point to fixed-point C implemented initially on the PC
- Used a C library of fixed-point arithmetic routines
- Got the fixed-point C code running on the C54X EVM.
- Profiled fixed-point C and optimized slow sections in Assembly.

Implementation Details

- Real-time development was done on TI Portable Audio EVM
- Portable Audio EVM provides support for:
  - 5 Band Graphic EQ
  - Sample Rate Converter
  - Volume
  - Decryption
  - Compact Flash

MP3/AAC Block Diagram



Parser

- Both MP3 and ACC require significant Huffman decoding.
- Computationally intensive tasks are assembly coded
- Audible fast-forward/reverse implemented in the parser
- Non-trivial because FF/RW requires jumping forward or backward into a bitstream with variable length codes.
- Our Huffman decoder in AAC uses a fast prefix counter implementation.
- MP3 code books are structured differently and require a different algorithm

Inverse Quantizer

- Both AAC and MP3 use a non-linear exponential function
- Table lookups handle the most common values
- Others are computed with a more accurate assembly coded algorithm
- This gives us a trade off between MIPS and memory: a bigger table requires fewer MIPS but more memory

Inverse Filter bank

- MP3 and AAC use different filter banks
- MP3 uses a hybrid; IMDCT followed by Polyphase subband.
- AAC uses one large IMDCT
- Both have "window switching" capabilities to control quantization noise spread in the time domain
- MP3 has 576 spectral lines and AAC 1024
- Our IMDCT implementation minimizes "state buffer" requirements.
- Much of the FB is double precision (32-bit) to keep PCM output precision high.

Temporal Noise Shaping

- Unique to AAC
- Requires an FIR filter in the encoder and a fixed-point all-pole filter in the decoder
- Improves the quality of coding signals with highly varying time-domain envelopes, such as speech, with a transform-based coder

Conclusion

- We currently have both MP3 and AAC running on the C5410
- Both algorithms will be on the C5409 soon.
- Questions?