TI Developer Conference
March 7-9, 2007 • Dallas, TX

Minds in Motion

Embedded Ogg Vorbis Decoder

An efficient implementation on the TMS320C6416 DSP processor

Santa Clara University

Santa Clara University

Ogg Vorbis

- Vorbis is an open source lossy audio compression codec.
  - Comparable to other formats used to store and play digital music, such as MP3, VQF, AAC, and other digital audio formats
  - Ogg is the general purpose media container format.
  - Founded by Xiph.org foundation.

Vorbis Source Code

- Xiph.org provides two reference decoder source codes;
  - *libvorbis, a floating-point arithmetic decoder implementation*
  - *Tremor, fix-point arithmetic decoder implementation*
    - *Targeted for embedded implementations of the Vorbis audio compression codec.*

Project Outline

- Port Ogg Vorbis *Tremor* reference decoder to TMS320C6416DSK.
- Define the performance critical modules within the design.
- Optimize performance critical modules.
- Examined performance of the optimized Ogg Vorbis decoder.

Tremor Code Branches

Code branch used for this project.

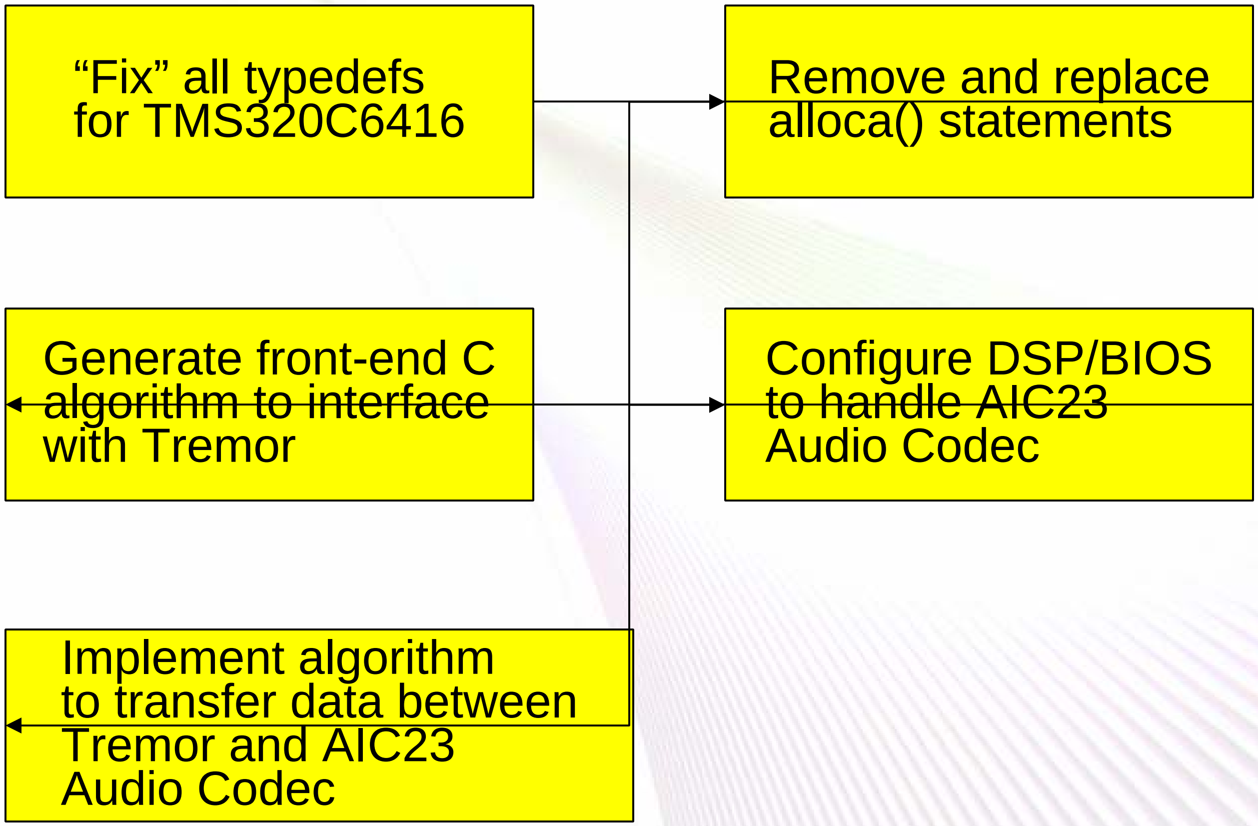
	Default	Low Mem	No Byte *
Processor Requirement	Lower	Higher	Higher
Memory Requirement	Higher	Lower	Lower

* No byte branch is a version of the low-mem branch created for processors whose smallest unit is greater than 8 bits.

Tremor Low-Mem Branch

- Tremor default branch allocates memory dynamically without restriction.
- Tremor low-mem branch improves memory usage with a slight performance penalty although memory is still allocated dynamically without restriction.
- Tremor low-mem branch is better suited for memory restricted embedded environment.

Porting the Source Code



Porting Steps

- Create type definitions to match the intended variable bit width with TMS320C6416 variable bit width.
  - typedef long long ogg_int64_t
  - typedef int ogg_int32_t
  - typedef unsigned int ogg_uint32t
  - typedef short ogg_int16_t

Porting Steps Cont'

- Tremor source code provides a self check for bitwise operations used for decoding.
  - Verify all "assumed" bit width by the source code is consistent with the actual bit width used by TMS320C6416

Porting Steps Cont'

- Fix the known issues with the *Tremor* source code.
  - Add free() statement to appropriate locations to remove memory leaks.
  - Fix all compiler warnings.
- Replace alloca() statement with malloc() and free() statements.

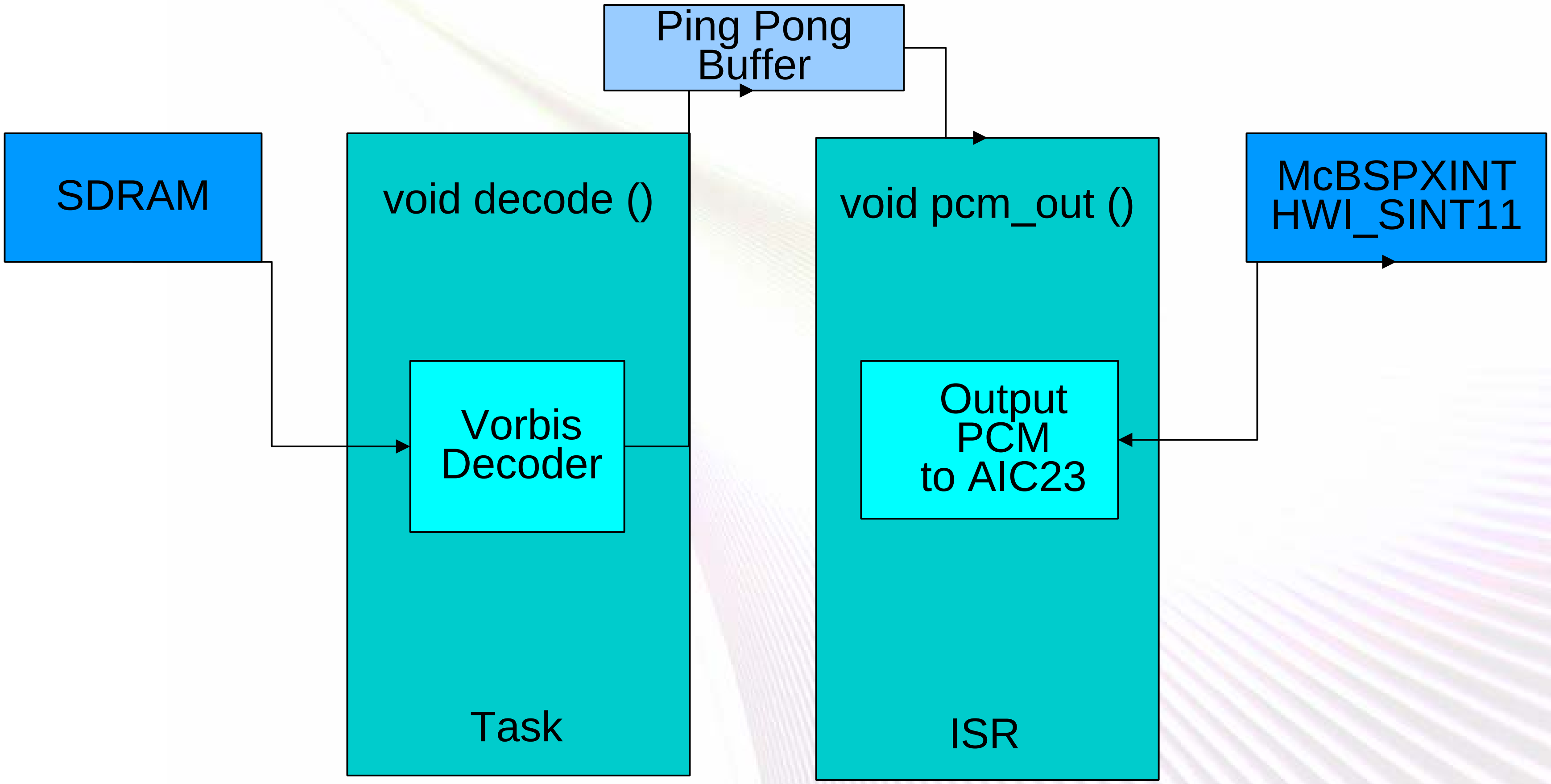
Porting Steps Cont'

- Create a generic file system to store a Vorbis audio file.
  - Read a Vorbis audio file into on-board SDRAM in Main() during DSP/BIOS initialization.
  - Add file system functions to the Tremor low-mem source code to access and read Vorbis the audio file.
- Implement the setup and tear down steps required for Tremor source code.

Porting Steps Cont'

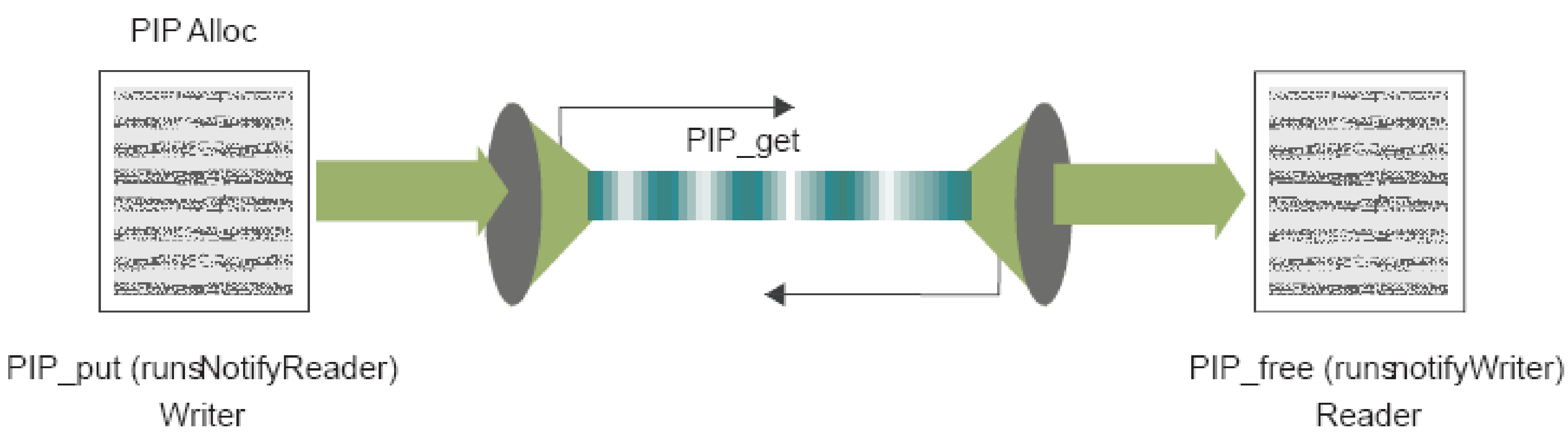
- Implement ping pong buffers to store decoded samples.
- Configure DSP/BIOS to output the decode samples using AIC23 audio codec interrupt (HW_INT11) while performing decoding in the background.

Design Block Diagram



PIP Objects

- PIP Module (Pipe Manager) manages block I/O used for streams of program input and output.
- Data notification functions are used to synchronize data transfers.

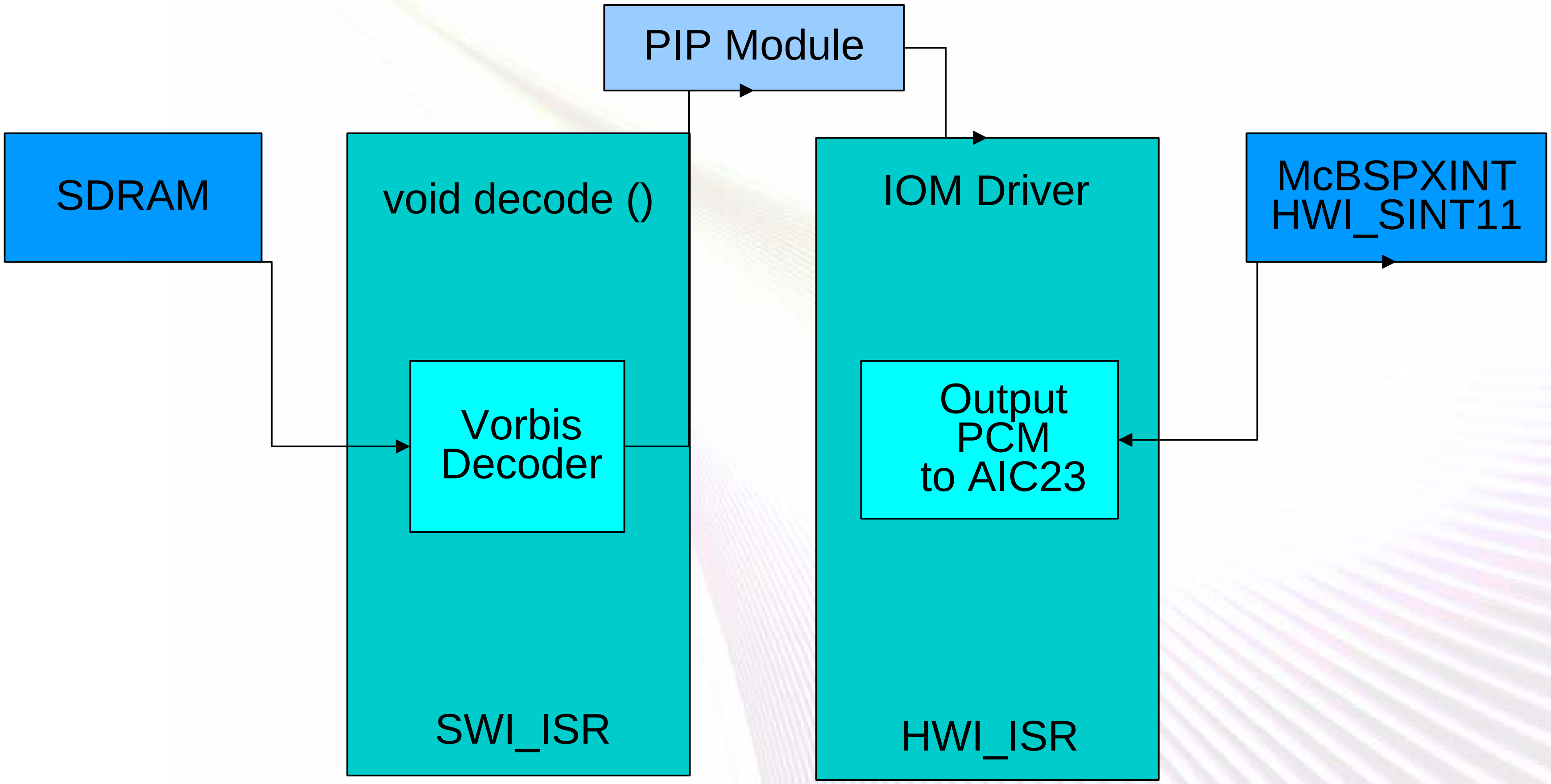


PIP vs Ping Pong Buffer

- Ping Pong requires more user coding
  - Higher chances of mistake
  - May not be optimized
- PIP easier to implement and low overhead
  - PIP module management functions are available. Only requires few function calls to manage buffer.
  - Optimized for DSP/BIOS

PIP Module simplifies input and output data stream implementation.

Design Block Diagram with PIP



Performance Results

Ping Pong Buffer Implementation

	Cycles	Percentage
IMDCT	75823572	44.40%
Residue Upack	51500363	30.00%
AIC23 Codec + Buffer Overhead	738718	0.40%
Other	42687551	25.20%

PIP Module Implementation

	Cycles	Percentage
IMDCT	74523681	44.00%
Residue Upack	50243698	30.00%
AIC23 Codec + PIP Overhead	695231	0.40%
Other	43564251	25.60%

Profiled by decoding ~5 second 128kps 44.1Hz stereo clip

Profiling the Results

- IMDCT and residue unpacking takes up the majority of the available clock cycles.
  - A better IMDCT algorithm can provide speed improvement without writing assembly.
- The difference between ping pong buffer and PIO module is minimal compared to the number of cycles taken by the Vorbis decoder.

DCT and DFT

- MDCT can be rewritten as an odd-time odd-frequency discrete Fourier transform.

MDCT-IV is defined as shown below,

$$\mathrm{DCT}^{\mathrm{IV}}_{[\mathrm{K}]} \; : \; C_k^{IV} = \sum_{r=0}^{K-1} x_r \cos\left(\frac{\pi}{4K}(2k+1)(2r+1)\right)$$

and K-th coefficient number of odd time odd frequency DFT of length N is defined as shown below,

$$\mathrm{O^2DFT}_{[\mathrm{N}]_k}\{\underline{u}\} = \mathcal{U}_k = \sum_{r=0}^{N-1} u_r e^{-j\frac{\pi}{2N}(2k+1)(2r+1)} .$$

Using zero padding,

$$x'_r = \begin{cases} x_r & \text{for } \; 0 \le r \le K-1 \\ 0 & \text{for } \; K \le r \le 2K-1 \end{cases}$$

The relationship between MDCT-IV and DFT is defined as the following,

$$C_k^{IV} \;=\; \Re e\left\{\mathrm{O^2DFT}_{[2\mathrm{K}]_k}\{\underline{x}'\}\right\} , \quad k \in \{0, \ldots, K-1\} .$$

New O² DFT Algorithm

- O^2 DFT can be calculated using one n/4 point FFT with pre-rotation and post-rotation.

Taking advantage of built-in symmetry,

$$\mathcal{P}_k = \mathcal{U}_{2k} + j\mathcal{U}_{N/2+2k} =$$

$$= 2 \sum_{r=0}^{N/4-1} \left(u_{2r} - j u_{N/2+2r}\right) e^{-j\frac{2\pi}{N}(2k+\frac{1}{2})(2r+\frac{1}{2})}$$

$$= 2e^{-j\frac{2\pi}{N}(k+\frac{1}{8})} \underbrace{\sum_{r=0}^{N/4-1} \left\{\left\{u'_r\, e^{-j\frac{2\pi}{N}(r+\frac{1}{8})}\right\} e^{-j\frac{2\pi}{N/4}rk}\right\}}_{\text{N/4 pointFFT}},$$

$$u'_r = \left(u_{2r} - j u_{N/2+2r}\right),$$

$$N = 4i;\ i\ \text{integer};\quad r, k = 0, 1, \ldots, \frac{N}{4} - 1\ .$$

New MDCT Algorithm

- MDCT is a special case of the new O^2 DFT algorithm.

$$u'_r = (x_{2r} + jx_{K-1-2r})$$

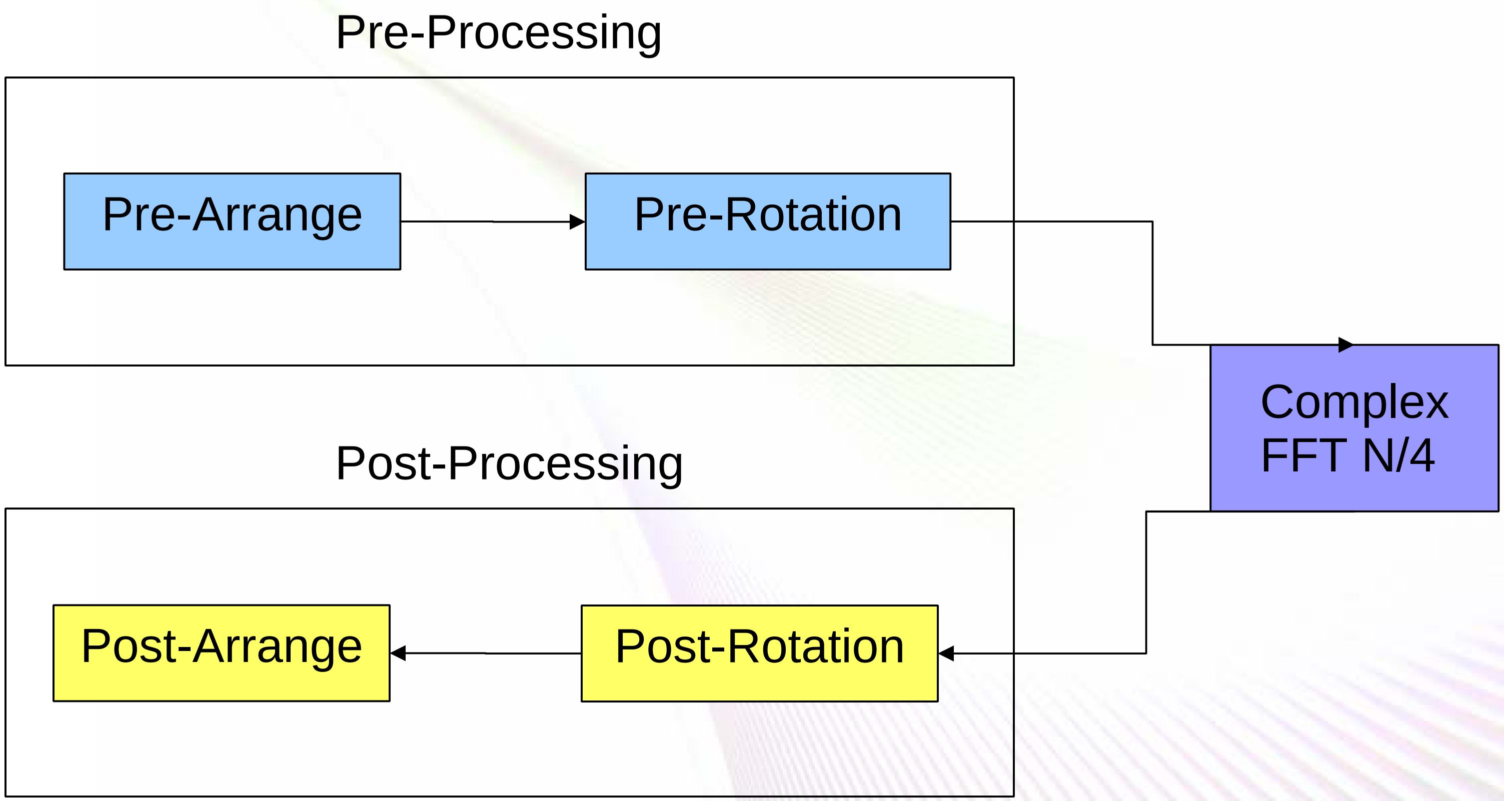
$$C^{IV}_{2k} = \frac{1}{2}\Re e\{\mathcal{P}_k\}$$

- IMDCT is basically a scaled version of MDCT.

Performance Optimization

- New IMDCT implementation
  - The optimized IMDCT algorithm is performed as follows:
    1. Pre-processing
    2. N/4-point complex FFT
    3. Post-processing
  - Uses optimized DSP_fft32x32 function from with built-in bit reversal from C64X TI DSPLIB

Optimization Block Diagram



Pre-Processing

- Pre-Arrange and Pre-Rotation

```
for (i = 0; i < n/4; i++)
   X'[i] = X[2*i] * rotation factor[i]
   X'[i+1] = X[(n/2-1)-2*i] * rotation factor[i+1]
```

Complex FFT N/4

- Utilize the DSP_fft32x32 from the TI DSPLIB
  - Generate twiddle factor with “tw_fft32x32.exe” from DSPLIB
  - Include “dsp_fft32x32.h” header file
  - Call DSP_fft32x32 to perform FFT

Post-Processing

- Post-Arrange and Post-Rotation

```
for (i = 0; i < n/4; i++)
   x[i] = x'[i] * rotation factor[i]
```

Performance Result after Optimization

	MIPS	Percent Improvement
Before IMDCT Optimization	~124	N/A
After IMDCT Optimization	~82	35.00%

Profiled by decoding ~75 second 128kps 44.1Hz stereo clip

Future Improvements

- Remove dynamic memory allocation from the Vorbis Decoder.
- Write pre- and post- processing section from the new IMDCT algorithm in assembly.
- Write the residue decoding portion in assembly.
- Make the code 100% RF3 compliant.

References

1) Erik Montnemery and Joannes Sandvall, *“Ogg/Vorbis in Embedded Systems”,* February, 2004.
2) R. Gluth, *“Regular FFT-Related Transform Kernels for DCT/DST-Based Polyphase Filter Banks”*, In ICASSP, 1991, pp. 2205-8 vol 3.
3) Xiph.org Foundation, *“Vorbis I Specification”*, www.xiph.org, 2004.
4) Texas Instruments, *“TMS320C6000 Programmer's Guide”*, SPRU198, August, 2002.
5) Texas Instruments, *“DSP/BIOS User's Guide”*, SPRU423, April, 2004.
6) Texas Instruments, *“TMS320C64x DSP Library Programmer's Reference”,* SPRU565, April, 2002.

TI | Developer Conference
March 7-9, 2007 • Dallas, TX

Embedded Ogg Vorbis Decoder

An efficient implementation on the TMS320C6416 DSP Processor

Santa Clara University

Santa Clara University

Minds in Motion

Technology for Innovators™

Texas Instruments

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265