Abstract
Opus audio codec is increasingly considered for high-end video conferencing application. SoCs targeting VC products typically have a combination of DSP and ARM cores. With increasing application complexities and features, ARM load available for multimedia processing is minimal. PathPartner’s OPUS solution solves OEM’s main point by making OPUS available on ARM with proxy APIs as low as 2% load [ARM clocked at 1GHz] and actual processing happening on the powerful DSPs. TI’s DM816x is considered for showcasing this solution.

Introduction
Opus is an open source, royalty-free and highly versatile audio codec developed by Internet Engineering Task Force (IETF) as RFC 6716 which incorporated technology from Skype’s SILK codec and Xiph.Org’s CELT codec. Opus is unmatched for interactive speech and music transmission over the Internet, but is also intended for storage and streaming applications.

The DM816x DaVinci™ Video Processors are a highly-integrated, programmable platform that leverages TI's DaVinci™ technology to meet the processing needs of the applications: Video Encode, Decode, Transcode, and Transrate, Video Security, Video Conferencing, Video Infrastructure, Media Server and Digital Signage. Programmability is provided by an ARM® Cortex™-A8 RISC CPU with NEON™ extension, TI C674x VLIW floating-point DSP core and high-definition video and imaging coprocessors.

SOLUTION
Optimization of OPUS Codec on C674x VLIW DSP
The floating point implementation of OPUS codec (from www.opus-codec.org, Version 1.0.2) was optimized for C674x VLIW floating point DSP.
The key steps to achieve the highly optimized implementation on C674x were:

- Clean-up of open source code to make it suitable for embedded applications
- Global C level optimization
- Codec level and function level profiling
- Function level C optimization
- Function level processor-specific optimization
- Code review
- Profiling on target hardware and performance tuning by efficient usage of L1 and L2 memories

**Integration of OPUS Codec to RPE**

Since applications run on the CortexA8 applications processor, there needs to be a mechanism for applications to be able to invoke the codec APIs from CortexA8 and IPC to marshal data and messages between ARM and DSP on which the codec algorithm actually runs. To accomplish this, Remote Processor Engine (RPE) mechanism was used.

The advantages of RPE mechanism:

1. RPE is a very light weight module which helps in maintaining the ARM load requirement to the minimum.
2. Several DSP-based audio codecs had been integrated to EZSDK on DM8168 using RPE.

Since the OPUS Codec is not XDM compliant and RPE has been developed keeping XDM compliant codecs as base, the codec could not be directly integrated with RPE. We had two ways to go about this problem.

1. Developing a XDM wrapper around the OPUS codec to make it XDM compliant.
2. Change the RPE design so as to make the RPE compliant to the OPUS API’s.

We chose the first method as we found it easier to develop and test the XDM compliant OPUS codec at standalone before integrating into the RPE.
Design and Execution

The team decided to use RPE which would take care of the inter-processor communication. As the OPUS codec is now made XDM compliant it could be directly integrated into the RPE. A new module, OPUS WRAPPER had to be developed so that the application making the OPUS CODEC calls is made to interface with the RPE CLIENT. In Figure 1, the application calls OPUS API, the OPUS WRAPPER interprets this call into suitable RPE call. The RPE client present on the ARM side will invoke the RPE server which makes XDM calls. The XDM wrapper is developed in such a way that the XDM calls made by the RPE server could be interpreted to OPUS CODEC calls as it was done by the application present in the ARM. After processing, the data moves back to ARM in the same flow.

In OPUS, the control call (used to view or change the codec parameter values) is different for different parameter compared to control call of the RPE or XDM interfaces, where all the parameter values are set in one structure and one control call is made. Directly interpreting the OPUS control call to RPE control call would take huge computation as each control call from application would have to be passed on to DSP.
and corresponding value has to be brought back. Hence, a separate structure was maintained in the OPUS WRAPPER so that each parameter change from application is stored and just before the RPE process call, the RPE control call is made and values are changed in the codec present in DSP. After the process call, this structure is updated to the values as that of the DSP side values for the application to view.

The OPUS can support multi-stream encoding and decoding with a maximum frame-size of 120ms. The encoder expects the multi-stream input data to be contiguous and gives out the encoded data in packed format. The decoder receives packed data and gives out contiguous multi-stream pcm data. In the standard reference code, the multi-stream and single-stream interface functions are different and are exposed to the application. Whereas in XDM implementation, both the multi-stream and single-stream interface functions are merged and are exposed as a single interface function. Based on the nb_streams parameter from input arguments it will either go into multi-stream or single-stream flow.

The RPE_INIT call, which initializes the RPE environment is developed in such a way that the call should be made only once. But running multiple streams in a single execution call would create a problem. So, few changes in the RPE_INIT call was made so that it could be called multiple times.

As OPUS contains both speech and audio codec which takes very high stack size compared to other audio codecs (almost 4 times). So, necessary care was taken in calculating the stack size and setting the same while integrating the codec into RPE.

**Testing and Validation**

Decoder passed the conformance test suite specified in Opus standard (RFC 6716 of IETF). Both encoder and decoder passed all the unit tests included in the open source Opus codec.

In addition, the OPUS encoder and decoder floating-point implementation was tested on DM8168 using RPE set-up with PathPartner’s own comprehensive test suite.

**Conclusion**

By optimizing Opus codec on TI C674x floating-point DSP, we were able to achieve 5.1 channel real-time encoding and decoding with floating-point code. The light weight RPE framework provides inter-processor communication between DSP and Cortex A8 and exposes OPUS API interface to applications.

**References:**
About PathPartner

PathPartner Technology based out of California, USA and Bangalore, India is a leading provider of products and services for multimedia centric embedded devices. PathPartner has extensive experience in audio & video codecs, video analytics & vision, imaging, multimedia middleware and application development.

PathPartner has an expert management team with rich experience in Technology, Engineering & Business practices. The company has Sales & marketing presence in USA, Europe, Korea Taiwan and India. PathPartner specializes in addressing challenges faced by leading Silicon vendors, OS providers and OEMs in their product development.

http://www.pathpartnertech.com/

sales@pathpartnertech.com