IMPLEMENTATION AUDIO COMPRESSION TECHNIQUES USING EMBEDDED SYSTEM

Mr. A. V. Warhade*

Prof A. D. Bijwe**

Prof. A. P. Deshpande**

Abstract

In this Paper study of various architecture for audio codec in an embedded system to demonstrate its effectiveness this is used to create completely self-contained digital audio encoder, based around a high-quality. General purpose audio compression algorithm The resulting audio encoder is able to operate 33% faster than the original Software only algorithm & hardware is able to achieve compression of 8KHz, mono, audio data in real-time.

Dynamic Range Compressors are a complex type of audio effect. The number of choices to be made during the design phase is vast. However, even using a compressor has its challenges, since the influences of the different parameters on the signal are not always obvious –especially to beginners. While properly setting up a compressor takes a lot of experience, a badly configured compressor can introduce a number of unpleasant artifacts' to the sound.

* M-Tech IVsem, PIET, Nagpur

** Assitt. Prof, PIET Nagpur

Introduction –

Needs of audio compression:

Audio compression (data), a type of lossy compression in which the amount of data in a recorded waveform is reduced for transmission with some loss of quality Dynamic range compression, also called audio level compression, in which the dynamic range, the difference between loud and quiet, of an audio waveform is reduced. Dynamic range compression, also called DRC (often seen in DVD and car CD player settings) or simply compression reduces the volume of loud sounds or amplifies quiet sounds by narrowing or "compressing" an audio signal's dynamic range

Goals Of audio Compression:

Reduced bandwidth or storage make decoded signal as close as possible lowest implementation complexity reasonable arithmetic requirement**.** Application to as many signal type as possible robust, scalable extensible an `audio CODEC' is a system for the encoding, and Decoding of audio data for use in digital systems The term `Lossy' refers to the fact that once audio data have gone through this process and been reconstructed, some information will be lost and the resulting signal will not be identical to that sampled. The key to successful CODEC is being able to identify where the redundant information in the signal is and being able to remove it, while at the same

Time minimizing the perceived impact on the listener of the reconstructed signal.

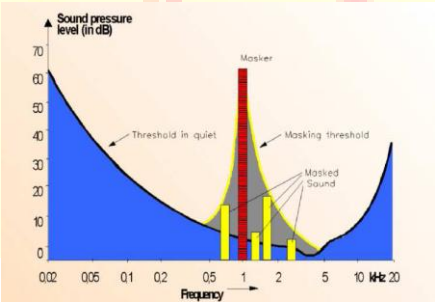


Fig.a) shows the threshold of energy required in the 20Hz-20 KHz range for an average human to perceive sound in quiet, measured in deci-Bells (dBs).

Through use of embedded audio compression:

MP3 players now sporting flash memory cards capable of storing whole albums, it seems logical that the next step should be not only playback, but also the capture of digital audio with these devices. To capture and store audio samples directly into a digital format, any time or place, is often a highly desirable goal. By keeping a digital version of captured audio it is possible to backup, distribute and replay a sample an unlimited number of times without having to worry about any kind of degradation, a problem that traditional analogue storage methods are often proneto. Unfortunately in its raw digitally captured format, high quality audio can be rather unwieldy in terms of its data storage requirements. So to compensate for this, raw audio data are often stored in an `encoded' form which greatly reduces the storage requirement, possibly at the expense of losing some parts of the original signal which are redundant to human listeners.

Encoding digital audio data into a compressed format does however come at a price in the form of a high processing requirement. This comes into direct conflicts with the goal of being able to capture an audio sample at any time and place, where in many cases computing power and resources maybe limited. One solution to this problem is to use a dedicated embedded audio encoding system that has been optimized for the capture and storage of digital audio. The absolute threshold of hearing makes use of the fact that humans do not perceive all frequencies of sound equally.

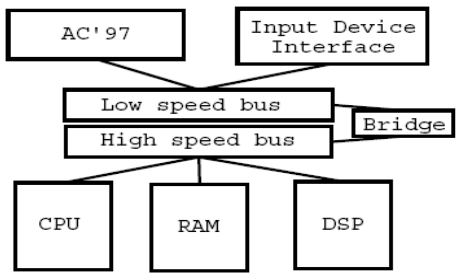


Fig.b Shows system design

Audio Functionality plays a critical role in embedded media processing. While audio takes less processing time than video. Then data is presented to the processor from variety of audio convertor. Sound is longitude displacement wave that propagates through air& sounds will be measured in dBSPL One possible approach for this project could have been to create an entirely new general purpose sequential processor from scratch. The advantage of this route would be full control over every aspect of the design, potentially allowing for maximal efficiency.

There were many distinguishing features amongst all of these, but in essence (apart from the PPC core in the Virtex-II Pro) they all sport a 32bit RISC

Choosing the right processor is not the end of story because the total quality of an audio system is dictated by the level of the lowest achieving component Besides the processor a complete system includes analog component like microphone & speaker as well as the converters to translate signal between the analog & digital domains Some of the major distinguishing points of the pre-made SoC's

Processor for audio compression Various DSP processors are available for signal processing but I have to consider dynamic range compression one processor choose for this project are BLACKFIN BF537

BLACKFIN DSP is the architectural base for a whole new family of DSPs from ADI (AnalogDevices,Inc). It is built upon the *Micro Signal Architecture (MSA)* core developed through the Joint Development with Intel Corporation. BLACKFIN DSPs incorporate the industry's highest performance 16-bit DSP architecture. It has Dynamic Power Management capabilities which deliver the lowest power consumption. BLACKFIN DSPs are optimized for processing data, communications and video streams for penetration into new market spaces. High Performance for real Time video signal processing easily programmed to support complex, new standards.

Handles the DSP and Control code with equal efficiency. Maximizes work and minimizes energy per cycle

BLACKFIN Processors Embed MCU Features

Arbitrary bit and bit-field manipulation, insertion and extraction Integer operations on 8/16/32-bit data-types Memory protection and separate user and supervisor stack pointers Scratch SRAM for context switching Population and leading digit counting Byte addressing DAGs Compact Code Density

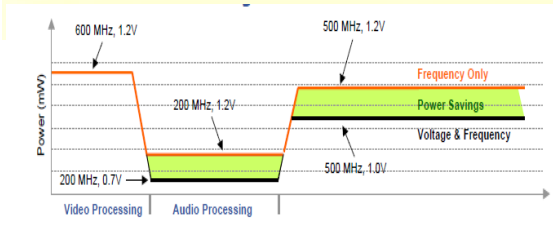
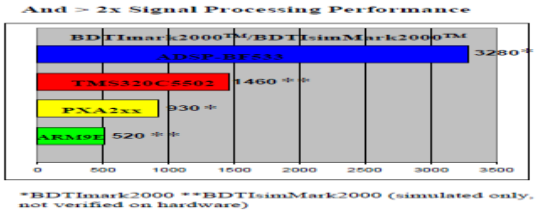


Fig.e Enhance dynamic power management increases battery life

BLACKFIN Application & architecture, performance :

PDA Internet audio , Digital Still Camera ,Video Camera Digital Printing ,Audio ,Modem ,Mobile phone



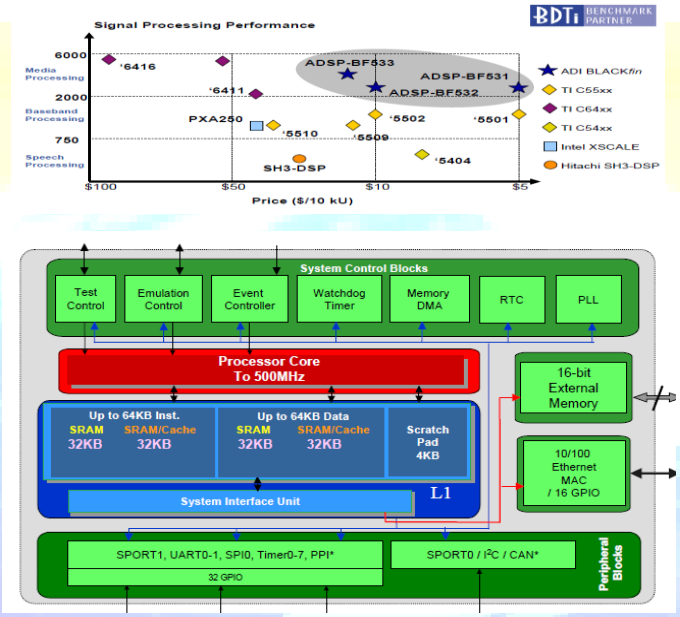


Fig.f. BLACKFIN Application & architecture, performance

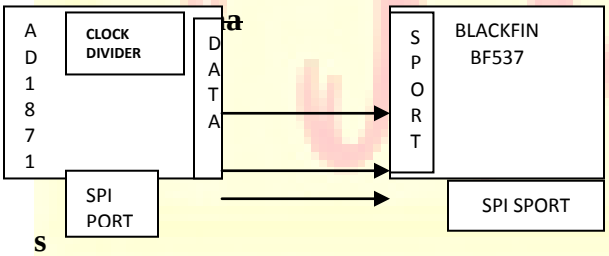


Fig.g ADC & BLACKFIN Bf 537 Processor Connection Diagram

Lab view 8.5:

Lab VIEW is a highly productive development environment that hardware integration to rapidly design and deploys measurement and control systems. The virtual interface (VI) consists of two main components: the front panel (user interface) and the block diagram)(Source

code).Graphical development environments, such as National Instruments Lab VIEW, are effective means to fast prototype and deploy developments from individual algorithms to full system-level designs onto embedded processors. The LABVIEW Embedded Module for Analog Devices BLACKFIN Processors allows Programs written in the LABVIEW to be executed and debugged on BLACKFIN processors .The software was jointly developed by Analog Devices and National Instruments, to take

Advantage of the NI LABVIEW Embedded technology and the convergence of capabilities Of the BLACKFIN EZ-KIT, for control and signal processing applications.

Then build option Select Source Files from the Category list and select your top level VI in the source files list. Target configuration The Target Configuration dialog box is separated into three configuration tabs, Target Settings, Debug Options and Hardware **Non-instrumented Debugging** Debug via JTAG/EZ-KIT USB is the same type of debugging available in the Visual DSP++ environment, but with the added benefit of LABVIEW'S interactive debugging features **Instrumented Debugging** Instrumented debugging allows the user to fully interact with a running embedded application without disrupting the processor during execution.

Use the Signal Generation VIs to generate one dimensional array with specific waveform patterns. The Signal Generation VIs generate digital patterns & waveforms Chirp Signal Use the Signal Generation VIs to generate one dimensional array with specific waveforms.

A chirp is a signal in which the frequency increases ('up-chirp') or decreases ('down-chirp') with time. In some sources, the term chirp is used interchangeably with sweep signal. A chirp signal can be generated with analog circuitry via a VCO, and a linearly or exponentially ramping control voltage. It can also be generated digitally by a DSP an DAC, using a (DDS) and by varying the step in the numerically controlled oscillator.

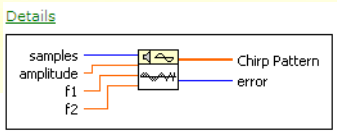


Fig.h shows chirp pattern diagram in lab view

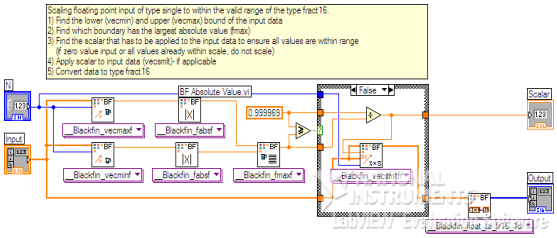


Fig. I shows inner view of BLCKFIN processor

Scaling floating point input of type single to within the valid range of the type fract 16 find the upper vector & lower vector of the input data. Find which boundary has the largest absolute value. Find the scalar of the applied data, apply scalar to input data convert data type to Fract 16.

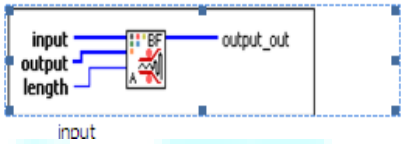


Fig.j shows A Law compression

Input is the input array to compress. Output is the same as output out to avoid reallocating memory in every loop iteration. Basically, output is a placeholder in memory for the output. Output must have the same type and number of elements as input.

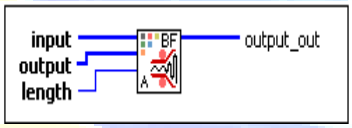


Fig.k shows A Law Expansion

Output out is the A-law compressed value of input. Input is the input array to expand. Output is the same as output out to avoid reallocating memory in every loop iteration. Basically, output is a placeholder in memory for the output. Output must have the same type and number of elements as input. Length indicates the number of elements to expand. Output out is the A-law expanded value of input.

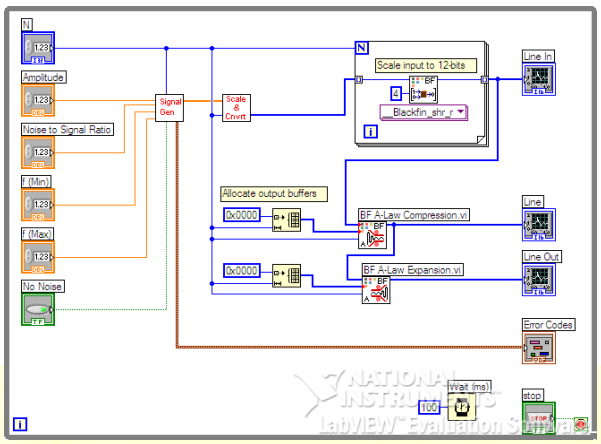


Fig.l Shows final VI For Compression using BLACKFIN

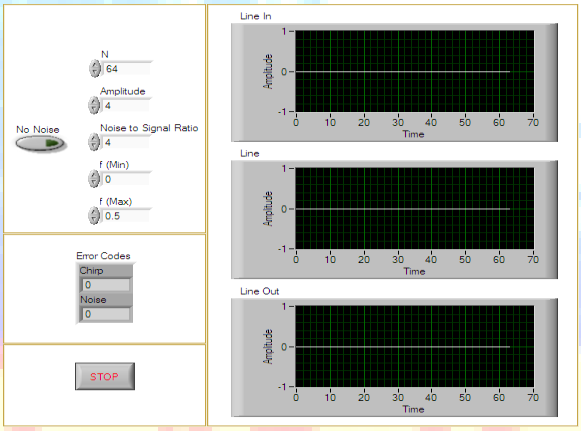
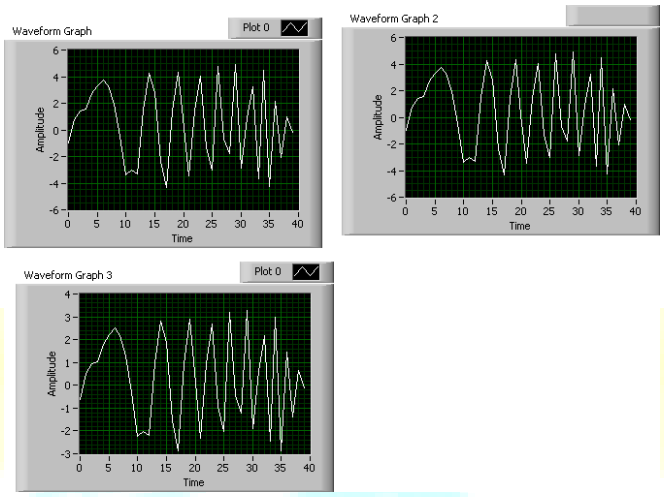
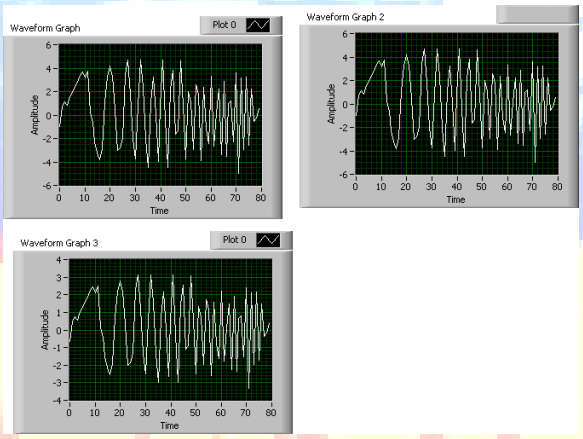
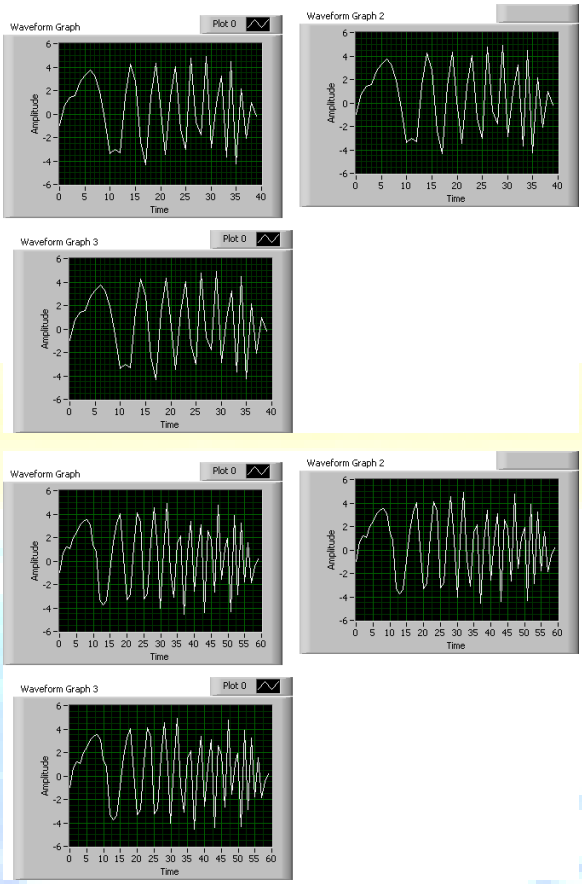


Fig m. shows Front Panel VI for Compression





Compression output for amplitude 4V & N=40&60



Expansion output for amplitude 4V & N=40&60

Conclusion:

The exhaustive literature survey was carried out on the said topic; the literature survey revealed that most of the work was carried out considering LAB VIEW 8.5 model for BLACKFIN

References:

- B. Bailey, G. Martin, and A. Piziali, *ESL Design and Verification: A Prescription for Electronic System Level Methodology*. Elsevier, 2007.
- Abdallah, A. Gamatié, R. Ben-Atitallah, and J.-L. Dekeyser, "Correct and Energy-Efficient design of a Multimedia Application on SoCs,"INRIA, Report 7715, 2011, Gamatié, S. L. Beux, E. Piel, A. Etien, R. Ben-Atitallah, P. Marquet,and J.-L. Dekeyser,
- "A model driven design framework for high performance embedded systems," INRIA, Research Report 6614, 2008,http://hal.inria.fr/inria-00311115/en.A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic, and R. de Simone, "The synchronous languages twelve years later," *Proc.of IEEE*, vol. 91, no. 1, pp. 64–83, Jan. 2003.
- Embedded Signal Processing with the Micro Signal Architecture By Dr. Woon-Seng S. Gan, Dr. Sen M. Kuo© 2006 John Wiley and Sons, Inc.
- T. Nishitani, "Trend and perspective on domain specific programmable chips," in *Proc. IEEE Workshop Signal Processing Systems (SiPS'97), Nov. 1997, pp. 1–8.*J.
- Goto, K. Ando, T. Inoue, M. Yamashima, H. Yamada, K. Aono, M. Toyokura, A. Ohtani, H. Kodama, and K.Okamoto, "A video digital signal processor with a vector pipeline architecture," *IEEE J. Solid-State Circuits, vol. 27, no.*12, pp. 1886–1893, 1992.
- P. Foley, "The Impact media processor redefines the multimedia," in *Proceedings of Compcon. New York: IEEE Computer* Science Press, 1996, pp. 311–318,