Software Configurable Processor Array For Decoding Multiple Ogg Vorbis Audio

Nikhil H, Harishiva B, Niranjan U C
Manipal Dot Net Pvt. Ltd
Manipal 576 104, India
nikhil.hegde@manipal.net, harishiva.b@manipal.net, niranjan@manipal.net

Abstract— This paper illustrates the use of the Software Configurable Processor Array (SCPA) of Stretch Inc., for accelerating the compute-intensive application of multiple audio decoding. It discusses the implementation of four Ogg Vorbis decoders on the SCPA. Vorbis is an open-source, royalty-free audio codec and the Ogg is a free container format for holding the encoded data. The SCPA is a processor array system, built by inter-connecting the software configurable processors (SCP) of Stretch Inc. The unique feature of the SCP is that the compute intensive part of the application code can be run on the on-chip hardware known as ISEF (Instruction Set Extension Fabric). The computational cost analysis of the Ogg Vorbis decoder identified the inverse modified discrete cosine transform (IMDCT) as the most computationally expensive part. We implemented the IMDCT on the ISEF, thereby speeding up the decoding process. The SCPA run time environment is utilized to simultaneously decode four encoded bit streams on the four processors. The PCIe interface on the SCPA is used to connect to a host personal computer (PC) for the transfer of encoded and decoded audio data.

Index Terms—Software Configurable Processor; Instruction Set Extension Fabric; Ogg Vorbis Tremor Decoder; Modified Discrete Cosine Transform;

I. INTRODUCTION

Vorbis is the open-source audio codec by xiph.org and Ogg is the container format to hold the encoded data [1]. Ogg Vorbis is a fully open, royalty-free, patent-free audio compression format for high quality audio (sampling frequency 44.1 to 48.0KHz, 16 bits/sample, polyphonic) at bit rates ranging from 16 to 128Kbps/channel.

Vorbis is a frequency domain audio compression technique based on the modified discrete cosine transform (MDCT). In the encoder, the audio data is segmented into blocks of appropriate size in accordance with the psycho-acoustical details of the data. The MDCT is applied to the segmented data and transformed to frequency domain. The transformed audio is modeled as dot product of a baseline spectrum known as floor and finer spectrum known as residue. The audio data thus represented by the floor and residue, is encoded using entropy coding and vector quantization (VQ). The decoding process runs the same sequence of operations in reverse order.

The open source nature of the Ogg Vorbis codec prompted many decoder implementations in the past, such as the one on C-based hardware [2] and System-on-a-Chip [3]. In this paper we report the implementation on an array of software configurable processor. We show that the compute-intensive part of the decoder can be run on the on-chip hardware to accelerate its execution. The multiple processors in the array are exploited to decode multiple music files simultaneously.

II. S6000 ARCHITECTURE

The software configurable processor (SCP) of Stretch Inc., has been shown to have high performance/price and low silicon area/performance ratios, making it a desirable processor for compute-intensive applications [4]. The SCP has a core made up of Tensilica LX, which is VLIW RISC architecture. The architecture of the S6000 family of SCP is shown in Fig.1. The heart of the processor is specialized to do multiple arithmetic and logic operations in parallel and is called the Instruction Specific Extension Fabric (ISEF). It is a reconfigurable piece of hardware embedded in the processor chip and is made up of 64 multiplication units each capable of 8x16bit multiplication, 4096 arithmetic units and 64KB of embedded RAM. An application with high computational load can offload those computations on to the ISEF to accelerate their execution. Typical examples of intensive computational algorithm are motion estimation in video coding, transform and filter operations in audio coding. In a typical application development flow, the user can identify the computationally intensive parts of the program by profiling the application code. The Stretch design tools enable the user to port these high-processor-cycle-consuming parts of the application code onto the ISEF, by writing them as Stretch extension functions in C/C++ language. The use of high-level language to program algorithms running on ISEF shortens the design time. The Stretch extension functions thus coded are seen by the SCP as extension instructions (EI) and are interleaved into the regular instruction schedule of the application. The operands are passed to the ISEF through the wide register file or ISEF RAM. The execution of the EI in the ISEF with its inherent parallel processing feature accelerates the running of the application. The relative ease, with which compute-intensive code can be converted into custom-made processor instruction running on the on-chip hardware, makes the SCP well suited for processing of audio, video and multimedia data.

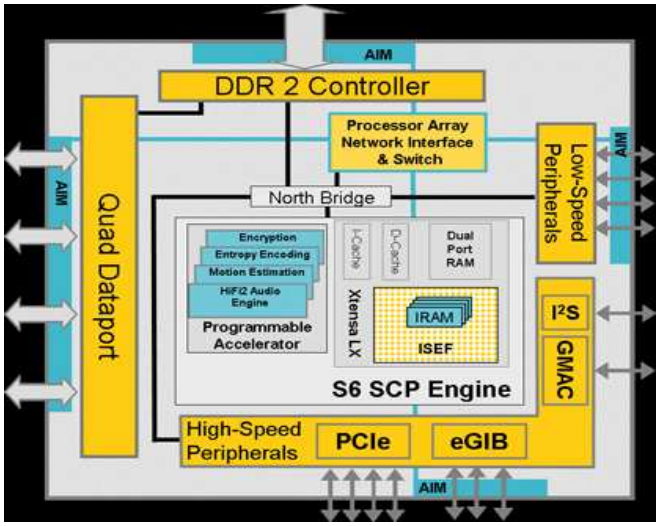

Fig. 1 The S6000 Family Architecture

In addition, each S6000 member includes external memory support with a DDR2-667 SDRAM controller with 16- or 32-bit interface and an enhanced generic interface bus (GIB) for FLASH and other memory mapped peripherals. On-chip memory sub-systems include instruction and data cache, as well as a 64-kbyte block of SRAM. The 40 DMA controllers facilitate moving data on and off the devices with minimal processor interaction. The S6100 family member includes a four-lane PCIe interface. Other integrated peripherals include triple speed Ethernet MAC, two multichannel Inter–IC Sound (I2S) interfaces, two-wire interface (TWI), serial peripheral interface (SPI), two UARTs, and general purpose I/O (GPIO).

III. SOFTWARE CONFIGURABLE PROCESSOR ARRAY (SCPA)

The SCPA system comprises four interconnected Software Configurable Processors as shown in Fig. 2. The SCP has on-chip array interface module ports for connecting to other SCPs. These ports do not need any additional glue logic and transfer data at the rate of 1.2GB/sec in one direction. The network interface and switch are built in the processor for fast routing of data. Each PE has a local DDR memory of size 128MB and DMA allows speedy data transfer between local memories of different PEs. The system is housed on a Board of compact form-factor and can be inserted into the PCIe slot of a computer.

All Stretch devices residing in a Processor Array network share the same memory map and can collaborate on tasks. The topology of the network for a particular application is described within the development tools suite by means of a simple text file. The described network of devices is programmed as if it were a single compute resource. A single executable is then developed for the network and, at boot time, the master allocates tasks within the network members according to the distribution assigned by the designer.

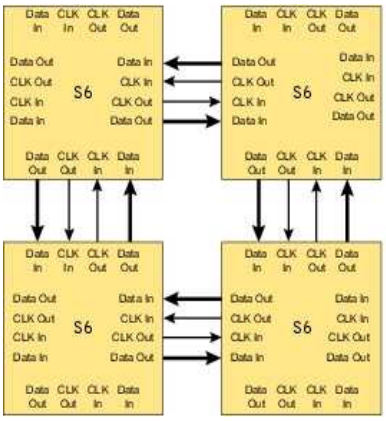

Fig .2 Device Connections in SCPA

Processor Array also provides an emulation mode on the host platform. An application using SCPA can be built as a native executable for the host platform, using the emulation libraries. This aids development and debugging.

The software tools developed by Stretch Inc., allow easy implementation of a parallel-computable project on the SCPA. All the tools are accessible from the Stretch Integrated Development Environment and a multi-processor application can be built, debugged and profiled conveniently.

The SCPA run-time environment is structured for co-operative multi-processing, where multiple tasks can be run on multiple PEs simultaneously. The run-time environment has a rich set of Application Programming Interfaces (API) for task creation and management; inter-processor and PCIe communication; and processor-memory management.

Processor Array API does not support preemptive tasks and context switching is deterministically controlled by the tasks themselves. A simple round robin, single-priority scheduler switches tasks in and out of execution, and tasks may block on several conditions without wasting CPU time.

SCPA supports an efficient block transfer mechanism through what are known as channels and a flexible small message-passing mechanism through messages. Messages can be used to send parameters no more than 16 bytes among processors, while data of larger size are communicated between processors using channels. A channel needs to be set up before transmitting any data on it.

IV. Ogg Vorbis Audio Decoder

We implemented the fixed-point Ogg Vorbis decoder titled Tremor on the SCPA. Tremor is available in the libvorbis library from the xiph.org, which is ported to the SCPA, so that future changes can be easily incorporated. The Tremor decoder is mainly made up of two functions: codec set-up and teardown. The first function sets up the decode engine by constructing the information, comments and VQ table from the first three headers of the encoded bitstream. The second function reads the encoded audio packets recursively and extracts the floor and residue values. The inverse modified discrete transform (IMDCT) of the dot product of the floor and residue values reconstruct the PCM audio samples. In the SCPA port, the reading of the encoded data is changed from file I/O to memory I/O. The malloc functions in the original code were modified to allocate memory on each PE in the SCPA system. As Stretch SCP is a little endian processor, the application code was made to run in that mode.

The IMDCT is the most processor cycle consuming part of the Tremor decoder [2, 3]. A fast algorithm for computing IMDCT is proposed in [5], by decomposing it as a number of butterfly computations. We wrote an ISEF function to do a butterfly computation, each butterfly accomplishing four multiplications of four 32-bit arguments. These ISEF butterfly implementations were plugged into the original IMDCT code. The operands for the butterflies were packed into the wide registers and passed to the ISEF. It was seen that the ISEF pipeline consumes 20 cycles before the butterfly product is made available. As this latency is unacceptable, different butterflies working on independent sets of data were identified. Such butterfly computations were scheduled on the ISEF continuously and this is known as loop unrolling. After appropriate loop unrolling and handling of the data through wide register file, the ISEF-based IMDCT showed 34% reduction in compute cycles, when run on the SCPA Board. This result is comparable to those implementations reported in [2, 3]. The ISEF implemented IMDCT used 768 arithmetic units out of 4096 and 4096 multiplication units out of 8192.

The SCPA system we used is made up of four Software Configurable Processors (Processing Elements, PE), with interconnections between them. The SCPA uses the PCIe interface to communicate with the external world. A host PC connected to the other end of the PCIe sends four Vorbis-encoded music data to the SCPA Board. The Tremor decoder engine is programmed as a task to read and decode the encoded data. The SCPA run-time environment instantiates the decoder engine task on all the four PEs, each PE decoding one of the four music data. The decoding task runs simultaneously on all the PEs, as the encoded data are mutually independent. Initially, multiple DMA operations on the PCIe bus move the encoded data from the host PC to individual PEs. After all the PEs receives data, the decoding begins. Once the decoding is complete, multiple DMA transfers move decoded data from the PEs to the host PC. The block diagram of the proposed decoder system and the flowchart of the program execution are shown in figures 3 and 4.

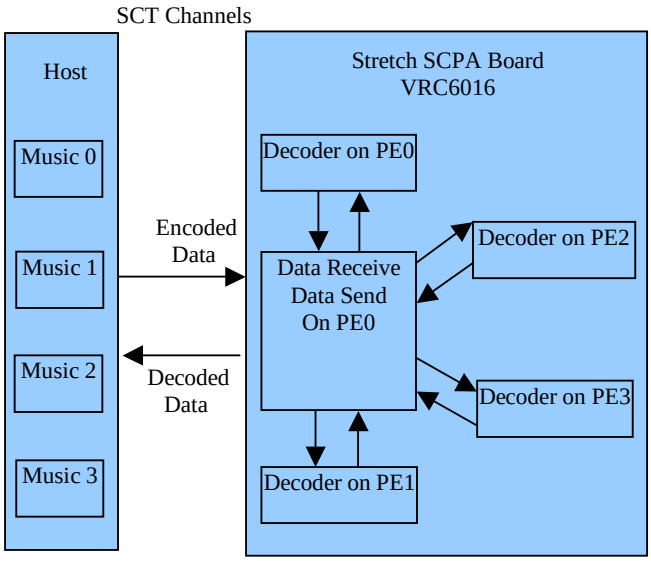

Fig. 3 Decoding of Ogg Vorbis Data on SCPA

V. RESULTS

Four Ogg Vorbis encoded stereo music files were decoded, each PE decoding one file. The music files were 4 to 5 minutes in duration, of average size 4.2MB and their bit rate ranged between 115Kbps and 128Kbps. The decoded data size ranged between 40MB and 53MB. The decoded music files were played back in the PCM format and were found to match exactly with the original Ogg Vorbis encoded music.

The proposed decoder system decoded a 302 seconds long music file (44.1 KHz, 115Kbps) in 119 seconds, while the non-ISEF based decoder system took 149 seconds. The music was stereo with two channels and each decoded PCM sample per channel was 16 bits wide. The decoding time can be further reduced by storing the encoded and decoded audio data in the dataram (dual port RAM) of each PE, instead of the DDR memory. This would reduce the processor cycles needed to read and store the data, thereby improving the performance of the decoder.

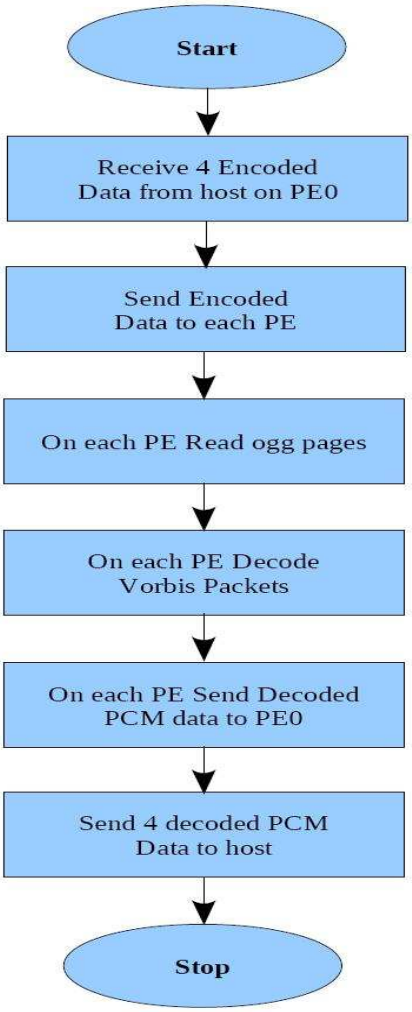

Fig. 4 Ogg Vorbis Decoder Flow in SCPA

VI. CONCLUSION

The details of implementation of four Ogg Vorbis decoders on the SCPA platform, for decoding four music files were presented. The original decoder was profiled to identify compute-intensive parts. The highest compute-intensive part, namely the IMDCT was programmed to run on the ISEF. The operands for the ISEF were transferred through wide registers. The proposed IMDCT computation accelerated the decoding by 34%. Storing the data in the on-chip RAM can further enhance the acceleration factor. It was shown that the proposed system can simultaneously decode multiple streams of Ogg Vorbis encoded data in real time.

VII. ACKNOWLEDGMENT

We are grateful to Dr. Narasimha B. Bhat for providing us with opportunity to carry out this work at Manipal Dot Net Pvt. Ltd. We also thank Stretch Inc., CA, USA.

VIII. REFERENCES

[1]. xiph.org Foundation, 'The Ogg Vorbis CODEC project', http://www.xiph.org/ogg/vorbis

[2]. S Maeta, A Kosaka, A Yamata, T Onoye, T Chiba and I Shirakawa, C-based Hardware Design of IMDCT Accelerator for Ogg Vorbis Decoder', Proc. XII European Signal Processing Conference, 2004, pp 1361-1364.

[3]. P Kiatisevi, L Azuara, R Dorsch, H Wunderlich, 'Development of an Audio Player as System-on-a-Chip using an Open Source Platform', Proc. IEEE ISCAS, 2005, pp 2935-2938.

[4]. Stretch Inc, s6ArchitectureOverview.pdf, http://www.stretchinc.com

[5]. T Sporer, K Brandenburg and B Edler, 'The use of Multirate Filter Banks for Coding of high quality Digital Audio'. Proc. 6th European Signal Processing Conference, 1992, vol. 1, pp 211-214.