# Development of an Audio Player as System-on-a-Chip using an Open Source Platform

Pattara Kiatisevi*, Luis Azuara†, Rainer Dorsch‡, Hans-Joachim Wunderlich§
*National Institute of Informatics, Tokyo 101-8430, E-mail: pattara@grad.nii.ac.jp
†intel GmbH, 38122 Braunschweig, Germany, E-mail: luis.l.azuara@intel.com
‡IBM Deutschland Entwicklung GmbH, 71003 Böblingen, Germany, E-mail: dorsch@de.ibm.com
§University of Stuttgart, 70565 Stuttgart, Germany, E-mail: wu@informatik.uni-stuttgart.de

***Abstract*— Open source software are becoming more widely-used, notably in the server and desktop applications. For embedded systems development, usage of open source software can also reduce development and licensing costs. We report on our experience in developing a System-on-a-Chip (SoC) audio player using various open source components in both hardware and software parts as well as in the development process. The Ogg Vorbis audio decoder targeted for limited computing resource and low power consumption devices was developed on the free LEON SoC platform, which features SPARC-V8 architecture compatible processor and AMBA bus. The decoder runs on the open source RTEMS operating system making use of the royalty-free open source Vorbis library. We also aim to illustrate the use of hardware/software co-design techniques. Therefore, in order to speed up the decoding process, after an analysis, a computing-intensive part of the decoding algorithm was selected and designed as an AMBA compatible hardware core. The demonstration prototype was built on the XESS XSV-800 prototyping board using GNU/Linux workstations as development workstations. This project shows that development of SoC using open source platform is viable and might be the preferred choice in the future.**

## I. Introduction

Open source software is more and more prevalent in server and desktop applications. Also for embedded software, it can reduce development costs, royalty costs, and often increase reliability of the systems. Embedded system development, especially in the hardware part, typically requires commercial development tools and hardware platforms, e.g., an ARM system with an AMBA bus or a PowerPC core with a CoreConnect bus. For the commercial platforms typically one time license fees for the development environment and a per produced chip license fee is charged. However, in recent years there are open source hardware platform available, e.g., LEON [1], Opencores [2].

In this paper we describe our experience in developing an open source Ogg Vorbis [3] audio player as a SoC using the LEON open source platform, which is roughly comparable to an ARM9 system. The target systems are low CPU performance embedded devices like PDAs or cell phones. We intended to explore the possibility of using open source components in SoC developments. Another objective is to illustrate the use of hardware/software co-design techniques to improve system performance. After the design process, a demonstration prototype was be developed.

The target audio format of the player is Ogg Vorbis. In 1997, Xiph.Org Foundation developed a new audio compression algorithm called *Ogg Vorbis* as a part of Ogg multimedia framework. Ogg Vorbis is a fully open, non-proprietary, patent-and-royalty-free general purpose compressed audio format for high quality audio (44.1-48.0kHz sampling frequency, 16 bits per sample or more, polyphonic) at fixed and variable bit-rates from 16 to 128 kbps/channel [3]. Ogg Vorbis is categorized in the same class as MPEG-4 Audio (AAC and TwinVQ) and claims to have higher performance than MPEG-1/2 audio layer 3, MPEG-4 audio (TwinVQ), WMA and PAC, which are all non royalty-free.

The SoC platform used in the project is LEON [1]. LEON features a 32-bit SPARC compatible embedded processor, an Advanced Micro-controller Bus Architecture (AMBA) [4], I/O cores, e.g., UART and PCI interfaces. It was developed by the European Space Agency (ESA) and is available freely with full VHDL source code under LGPL (GNU Lesser General Public License). The LEON platform is extensively configurable and may be efficiently implemented on both FPGAs and ASIC technologies. The AMBA [4] enables an easy integration of user-defined cores. The version of LEON at the time of project was Leon-2 1.0.2a.

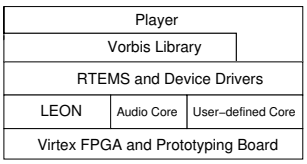


Fig. 1. Overview of the system

For rapid demonstration prototype development, the FPGA-based XESS XSV-800 prototyping board is used. The XSV-800 board is equipped with a Xilinx Virtex XCV800 FPGA with 800,000 gates, 2 banks of 512K x 16-bit SRAM (2 MB in total) and 1 MB of Flash memory. XSV-800 offers a lot of peripheral interfaces, e.g., audio, USB, PS2, and VGA. The audio chip AKM AK4520A can process stereo audio signals with up to 20 bits per sample and a bandwidth of 20 kHz.

## II. The Ogg Vorbis SoC

Components are classified into layers shown in Figure 1. The whole system was implemented on the FPGA and the prototyping board. The LEON SoC platform and additional cores compose the upper hardware layer. The open source RTEMS embedded operating system is used for tasks management and hardware-resource abstraction. RTEMS and device drivers for hardware cores run on top of LEON. The Vorbis player makes use of Vorbis decoding library and runs as a sole process in the system.

In order to enable decoding of Vorbis data on systems with low CPU performance, a part of the decoder was implemented as a user-defined hardware to speed up the computation. Therefore the Vorbis player is modified to decode compressed music data with the help of the user-defined core and delivered music output to the audio core via the audio device driver.

The audio core designed and developed in the *Digital Dictation Machine as System-on-a-Chip* project [5], which acts as an interface to the audio chip, was imported and reused in our system. This audio core and the later developed user-defined core are integrated into the platform via AMBA APB and AHB buses. The overall platform configuration is shown in Figure 2.

## III. SoC Design and Development

This section describes our work methodology and details of the development process. Work in the project was evenly divided into

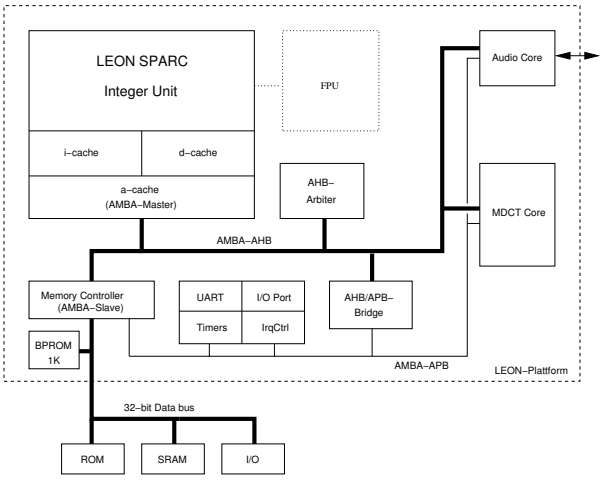


Fig. 2. Platform configuration

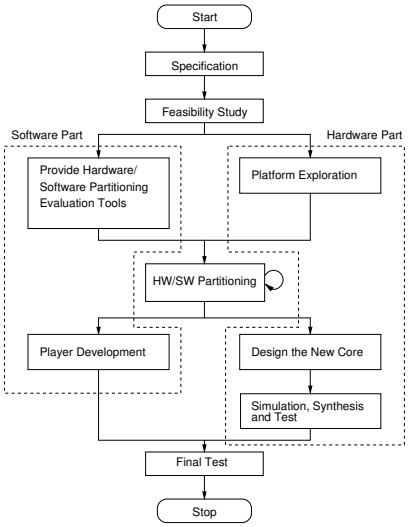


Fig. 3. Work packages diagram

hardware and software part and was done in the collaborative manner but independent enough so that multiple tasks in both parts could be done simultaneously. This is illustrated in Figure 3. After the specification, the feasibility study is done to investigate if the hardware platform, with future reasonable optimizations, was powerful enough to decode Vorbis stream and whether some parts of algorithm could be efficiently implemented in hardware.

Main tasks were then divided into hardware and software parts. In the software part, hardware/software partitioning evaluation tools were prepared. At the same time, in the hardware part's platform exploration phase, the platform is investigated. The capability and limitation of the platform were discovered and the parameters were configured. Once the target platform is set up, hardware/software partitioning can be done. The Vorbis decoding algorithm was studied. Based on the result, several hardware/software partition candidates were proposed. The most suitable ones were chosen with the help of the prepared hardware/software partitioning evaluation tools. The selected hardware part of algorithm was designed, modelled and implemented in hardware part. Concurrently in the software part, the full version of Vorbis player was developed.

In the final test phase, all components were tested together. The final player decoded the Vorbis stream with the help of the user-defined core on the real hardware. Next, important phases are discussed in more details:

- *Feasibility Study:* A simple performance observation of the reference Vorbis decoder library and the reference client player *ogg123* on a PC gave preliminary performance information of Vorbis decoding process. The machine under test was equipped with Intel CPU Pentium-III 600 MHz, 256 MB RAM and Linux kernel version 2.4.18. A test sound file with 44kHz sampling frequency and 16-bit stereo data format was generated using the reference *oggenc* encoder at default quality level (level 3). Based on this resource consumption information on PC and result from Purify [6], the performance requirement of the target system was approximated. Profiling of Vorbis decoder gave useful information for indicating the computing-intensive part of the algorithm, which was later inspected by human to see if it could be developed as hardware.

After an analysis [7], it was estimated that 60-70% speed improvement must be done in order to decode Ogg Vorbis streams at real-time speed on the target hardware. Considering the possibility to implement a part of software as hardware to achieve about 30- 40% performance improvement and to apply possible software optimizations to get another 30-40% improvement, we regarded the project as feasible and proceeded.

- *Platform Exploration:* In this phase the target hardware platform was studied for capabilities and limitations. Several important properties were discovered as follows:

- **Number of gates:** The XCV-800 FPGA used in the prototyping board can hold designs with up to 800,000 gates.
- **LEON timing and the on-board system clock:** LEON can be synthesized for Xilinx XCV-800 speed grade 4 in the range of 25-28 MHz. The on-board clock provides frequencies in sub-multiples of 100 MHz, i.e., 50,33,25,20,10,5 MHz. Hence the maximum possible clock frequency that can be used is 25 MHz.
- **Sampling frequency:** The audio sampling frequency is a sub multiple of the system clock. As described in [5] the sample frequency of the audio core is according to the formula $fs = \frac{clk}{256(2+2\,scalerup)}$ , where *scalerup* can be *0,1*, or *2*. For this reason the standard audio sample frequencies such as 44 kHz and 22 kHz are not possible to achieve. The best approximations are 48kHz and 24kHz.
- **Limited internal RAM on the FPGA:** Since the memory on the XCV-800 has to be shared among different elements of the platform, such as cache and user-defined cores, it is important to allocate this memory wisely.
- **Meiko FPU limitation:** The FPU design from Sun's Meiko SPARC implementation can be used with LEON. It can be run at the maximum frequency of 25 MHz. However, it consumes a large amount of space on the FPGA.

After an analysis, we decided not to use an FPU since it requires a large amount of gates on the FPGA and increases the complexity of the design. Also an integerized version of the Vorbis decoder library (which is also faster than the floating point version) has been developed. Thus the need for an FPU could be safely eliminated. And the clock frequency is set to 25 MHz because of the requirements of timing and system clock mentioned above — 25 MHz is the highest frequency to run LEON on the board.

- *Embedded Software :* Software development in the project was done on Debian GNU/Linux workstations. Main tools used are *TSIM* for system simulation, various utilities from *LEON/ERC32 Cross Compilation System (LECCS)* provided at [1] including *GNU Compiler Collection* for general software development, RTEMS [8] as operating system (to be discussed in the Player Development section), and GNU gprof, and kprof, XEmacs, gdb, and DDD for Ogg Vorbis library study and optimization.

In order to achieve fast software development and verification, TSIM (LEON simulator) was used to simulate the whole system on the software. TSIM is a simulator of SPARC architecture capable of simulating LEON developed by Gaisler Research [1]. It runs on Linux-x86, SPARC, and Windows/Cygwin platforms. Developers could then develop, test and debug their programs on TSIM running on their favorite workstations before testing on the real hardware.

TSIM provides also the capability to plug-in external *user-defined modules* (in C programming language) to the system as if the user-defined hardware core is attached to the LEON platform via AMBA bus. Read and write access to some provided addresses in the memory by the application running on TSIM will be mapped to this user-defined module (which is running on the host machine) as if the application is accessing that corresponding user-defined hardware core. Interrupt from user-define module is also supported. Based on this, our hardware platform was first simulated on TSIM. The simulation version of the audio hardware core is developed as an external I/O module.

A simple player program for testing decoding Ogg Vorbis audio data on TSIM was developed. With all the settings corresponding to those on the real hardware, it decoded a 15-second test Vorbis stream in 32.90 seconds, which is very far from real-time. At this point we need a huge improvement.

As we have no FPU, all floating point calculations have to be simulated and hence very slow. The reference Vorbis library uses a lot of floating point calculations. The first attempt was to investigate if it could use integer calculations instead. At the time of the project, there was no freely available integer version of Vorbis library[1], therefore we studied, optimized and integerized the Vorbis library. The tools used to aid the study of Vorbis code include GNU *gprof* (part of GNU binutils software package) and *kprof* . The integerization was successful. The need to have a real FPU on the system is eliminated.

The resulting integerized version of the Vorbis library gave 40.97% better performance than the original floating-point version. With this integerized version, the simple player decoded 15-second music in 19.42 seconds. Although this is still not real-time, from this figure we have an idea of how much performance gain we need when doing the hardware/software partitioning in the next step. The simulated platform on TSIM set up in this phase will be used further as software/hardware partitioning evaluation tool in the next phase.

- *Hardware/Software Partitioning:* TSIM is the main tool used in this process since we can model any desired user-defined hardware cores as TSIM external I/O modules and test for performance gain before designing the real hardware core. With proper settings of the simulated software to match the real hardware configuration and the profile data obtained in the previous phases, several hardware/software partition candidates were proposed and tested. The most two suitable partitions are:

- **MDCT:** includes the whole inverse MDCT transform function used in the decoding process.
- **Mini-MDCT:** includes major part of the inverse MDCT transform, covering roughly 60-70% of the whole MDCT.

The test was conducted on TSIM to identify suitable partitions. The player decoded the 15-second test Vorbis stream in three configurations: without any hardware module, with Mini-MDCT partition, and with MDCT partition. Table I shows the time needed for the player to decode the data in each configuration.

According to the result above, the MDCT partition was proposed as the preferred solution in order to have the music decoded and played faster than real-time with some safety margin. Mini-MDCT was proposed as the second choice in case that the implementation of the whole MDCT core was not feasible.

[1]Later in September 2002, XIPH released *Tremor*, the decoder library which is integer-only and fully Ogg Vorbis compliant under a free BSD-like license.

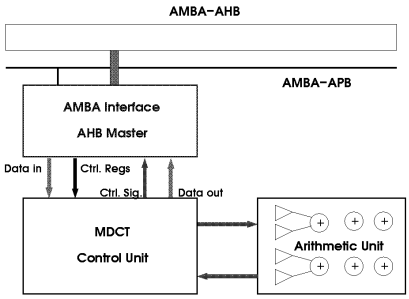


Fig. 4. Mini-MDCT core architecture

- *Designing the New Core:* From the two proposed partitions in the hardware/software partitioning process, the Mini-MDCT partition was chosen because the resource on the prototyping board (notably the internal RAM) was not sufficient to accommodate the larger MDCT core.

MDCT or *Modified Discrete Cosine Transform* is widely used in state of the art audio codecs such as MPEG 1 Layer III, Dolby AC/3, or MPEG AAC and in Ogg Vorbis. It is a linear orthogonal lapped transform, based on the idea of time domain aliasing cancellation (TDAC). The inverse MDCT function in Ogg Vorbis is implemented following the algorithm presented in [9]. For more details information please consult [7].

As shown in Figure 4, the Mini-MDCT core is designed to have the following components:

- **AMBA Interface** provides the connection with AMBA AHB and APB buses. Access to the RAM is done via AHB. Communication with software holding memory mapped registers is done via APB.
- **Arithmetic Unit** is a set of arithmetic elements, 4 (32X32 bits) multipliers and 6 (32 bits) adders, for data processing according to the Ogg-Vorbis MDCT algorithm.
- **Control Unit** commands the activities of the AMBA interface, and uses the arithmetic unit to calculate the MDCT as a finite state machine.

- *Simulation, Synthesis and Test:* The design steps are to simulate the system, fix bugs, and proceed to the synthesis and test on the real hardware. The tools used are MODELSIM for simulation, SYNPLIFY PRO for synthesis, and XILINX ISE 4.1I. In Figure 5, the hardware design flow is shown. The rectangles with round corners represent these tools[2]. Ellipses are other necessary utilities and rectangles are data files in intermediate states. Starting from the top, the source VHDL files of the LEON model, DDM, and MDCT (or Mini-MDCT) core are shown. The left flow is the simulation path, which leads to the Modelsim simulator in order to test the design, the right flow, the synthesis path, produces the output design to be uploaded to the XSV-800 board at the bottom for the implementation on the real hardware.

- *Player Development:* On top of the LEON processor, there are choices of either running the application directly or using an operating system. The test player used in the previous sections was an example of running the application directly without the operating systems (OS). Running the application without the OS (and directly accessing the hardware) is free from the OS overhead. However, with an OS, one can get benefits from the provided services like tasks management, memory management, and abstraction of hardware devices. Also it would be much easier to port the application code to

[2]None of these tools, however, were open source.

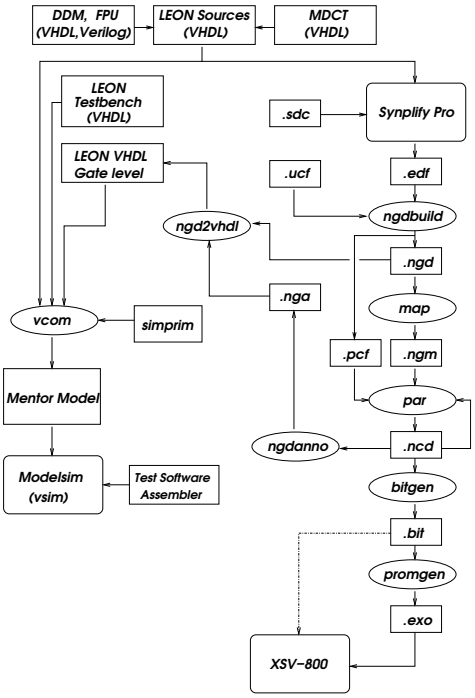


Fig. 5. Hardware design flow

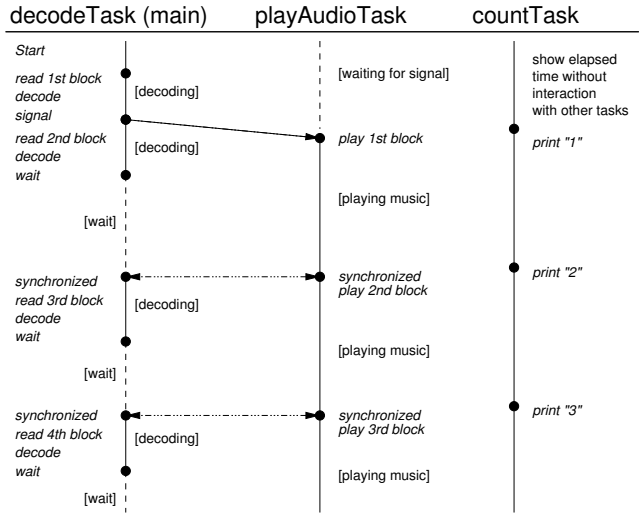


Fig. 6. Tasks in the player

other platforms in the future because it uses the abstraction interface of hardware provided by the OS instead of having direct hardware accesses. For these reasons, it was decided to have an OS for the final version of player. Viable choices at that time were RTEMS [8] and uCLinux [10]. RTEMS was selected because of its features and maturity [7].

The RTEMS operating system offers the benefits of multi-threading (POSIX Thread), task synchronization and hardware abstraction. For hardware abstraction, a device driver for audio device was written to provide access the hardware audio core. The player in the previous section was extended to be the multi-threaded final player with complete Vorbis decoding and sound output functions via the developed audio device driver. Interactions between tasks are shown in Figure 6 [7].

After the user-defined hardware core (Mini-MDCT) had been developed, the final player was extended to utilize the hardware core instead of calculating the whole inverse MDCT function on its own.

## IV. Result

With the developed Mini-MDCT hardware core plugged in to the LEON platform, the final Ogg Vorbis player runs successfully on the real hardware. The time needed for the player to decode and play the music is shown in Table I. The Mini-MDCT core speeds up the decoding process from 21.1 seconds to 17.9 seconds or by a factor of 1.18.

TABLE I
Result running time (in seconds) for the tests on both TSIM and real hardware under various player configurations

| Configuration/Platform | TSIM | XSV-800 |
|---|---|---|
| Player software only | 19.42 | 21.1 |
| Player with Mini-MDCT core | 15.41 | 17.9 |
| Player with (full) MDCT core | 11.70 | - |

Notice that the time needed in the hardware test was higher than the estimated result from the software test since the software simulation did not take the processing time of Mini-MDCT hardware core into account. Also the test had to be conducted at the sampling frequency of 48 kHz instead of 44 kHz (limitation of the hardware), hence more data to decode. A different test was done at 24 kHz sampling frequency and the player could successfully decode Vorbis data and deliver output music in real time.

The physical amount of RAM available on the prototyping board places a restriction on our design. If more internal RAM was available, we could have allow larger design of Mini-MDCT core (or even extend it to MDCT) and could achieve faster speed.

## V. Conclusion

This project shows a successful development of a SoC application based on various open source components including the Ogg-Vorbis reference library, the RTEMS operating system, the LEON SoC platform, and Debian GNU/Linux workstations equipped with various free software. The only commercial component in the process were the hardware simulation and synthesis software because there was no good enough open source choice available. Nonetheless, this project has proved that development of SoC using many open source core components is viable. Lessons learned in this project could be applied to others in the future.

## References

[1] Jiri Gaisler. LEON Web Site. [Online]. Available: http://www.gaisler.com/
[2] The Open Cores. Project Web Site. [Online]. Available: http://www.opencores.com/
[3] XIPH. Ogg Vorbis Web Site. [Online]. Available: http://www.xiph.org/ogg/vorbis/
[4] ARM limited. (1999) AMBA Specification 2.0. [Online]. Available: http://www.arm.com/
[5] Daniel Bretz, "Digitales Diktiergeraet als System-on-a-Chip mit FPGA-Evaluierungsboard," Master's thesis, Institute of Computer Science, University of Stuttgart, Germany, February 2001. [Online]. Available: http://www.ra.informatik.uni-stuttgart.de/Leon/
[6] Rational Inc. Rational Software – Purify. [Online]. Available: http://www.rational.com/products/pqc/index.jsp
[7] Luis Azuara, Pattara Kiatisevi, "Design of an Audio Player as System-on-a-Chip," Master's thesis, Institute of Computer Science, University of Stuttgart, Germany, July 2002. [Online]. Available: http://oggonachip.sf.net/
[8] OAR Corporation. RTEMS Web Site. [Online]. Available: http://www.oarcorp.com/
[9] Kh. Brandenburg, Th. Sporer, B. Edler, "The use of multirate filter banks for coding of high quality digital audio," *6th European Signal Processing Conference (EUSIPCO)*, 1992.
[10] uClinux. uClinux – Embedded Linux/Microcontroller Project. [Online]. Available: http://www.uclinux.org/