A Low Power Configurable SoC for Simulating Delay-based Audio Effects

Ling Liu[1], Jeremia Bär[2], Felix Friedrich[1], Jürg Gutknecht[1]
Computer Systems Institute
ETH Zürich
Zürich, Switzerland
[1]{*ling.liu, felix.friedrich, gutknecht*}*@inf.ethz.ch,* [2]*baerj@student.ethz.ch*

Shiao-Li Tsao[3]
Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
[3]*sltsao@cs.nctu.edu.tw*

***Abstract*—The rapid growth in the capability of modern FPGA devices allows developers to build a complete system on a single chip. These types of FPGA-based SoC (System-On-a-Chip) can normally achieve reduced system power, cost and size, and at the same time offer users a great deal of flexibility. The development of such SoCs normally starts from using a hardware / software co-design methodology in order to partition system tasks into computation-intensive and flexibility-demanding parts. Then, dedicated hardware and software will be implemented to realize these two parts. This paper presents an example which demonstrates the result of applying the hardware / software co-design methodology, a power efficient and performance reliable system architecture for realizing audio delay effects. Compared to similar implementations, our system architecture can save 40% of dynamic power consumption while offering the same data throughput and user flexibility.**

***Keywords*-low-power SoC, hardware / software co-design, FPGA, delay-based audio effects.**

I. Introduction

In the field of music, delay-based audio effects simulation systems are widely used to create a sense of space for listeners. The algorithms for delay-based audio effects and their realization techniques such as FIR(Finite Impulse Response), IIR(Infinite Impulse Response) filters, delay lines and periodical modulators have been studied extensively in the last fifty years [1], [2], [3], [4], [5]. However, very few study results have been found in the entire system architecture, especially with regard to effects on system power consumption.

For a real-time audio effects simulation system, the performance requirement is that the system can produce outputs at a sampling rate. Apart from the performance requirement, it is normally expected that the system allows users to configure the effect parameters and route the audio stream during the run time. The goal of our system architecture design is to meet these performance and programmability requirements with low power consumption cost. To achieve this goal, hardware / software co-design methodology is used in the system development process. The delay-based audio effects SoC presented here is implemented on an FPGA chip. In the system, dedicated hardware is used to realize the audio effects and route the audio stream. A tiny register machine (TRM) soft-core processor and the software running on top of it are used to control the parameters of the audio effects and the routing matrix. Compared to the existing studies of audio effects SoC implementation [6], [7], the system architecture presented here has the following characteristics.

- Instead of soft-core-centric design, this paper presents a distributed on-chip system. That is, the data buffers required by the audio effects hardware are neither controlled by the soft-core nor accessed via the software running on the soft-core. They are implemented in the dedicated hardware using BRAMs (Block RAMs) on an FPGA chip. This design reduces the possible performance unreliability caused by software and decouples the hardware from software to allow the hardware component to run at a much lower clock rate, in our case, 48KHz.
- Instead of 48MHz or an even higher clock rate, the audio effects hardware in our SoC runs at a sampling rate of 48KHz. Therefore, our system can save 40% dynamic power according to estimation results.
- Instead of software, a switch matrix is implemented to control the routing of the audio stream. Using dedicated hardware to control the routing of the audio stream can avoid unpredictable delay caused by the software and allow hardware to run at a much lower clock rate.
- Instead of a fully fledged commercial soft-core, such as MicroBlazer from Xilinx and NIOS from Altera, a customized soft-core processor TRM is implemented on the FGPA to minimize the resource and area cost. The TRM implementation on a Virtex-5 FPGA only uses 728 LUTs, while most commercial soft-core processors require roughly 2000 LUTs or LEs. Therefore, the use of a customized soft-core processor dramatically reduces the area cost of the entire system and, as a result, also reduces the power consumption of the entire system.

The system architecture is presented in Section II. The dedicated hardware for delay-based audio effects and the routing matrix are introduced in Section III. The parameter control is described in Section IV. Section V presents the system evaluation results. Finally, we conclude this paper in

978-1-4673-2921-7/12/$31.00

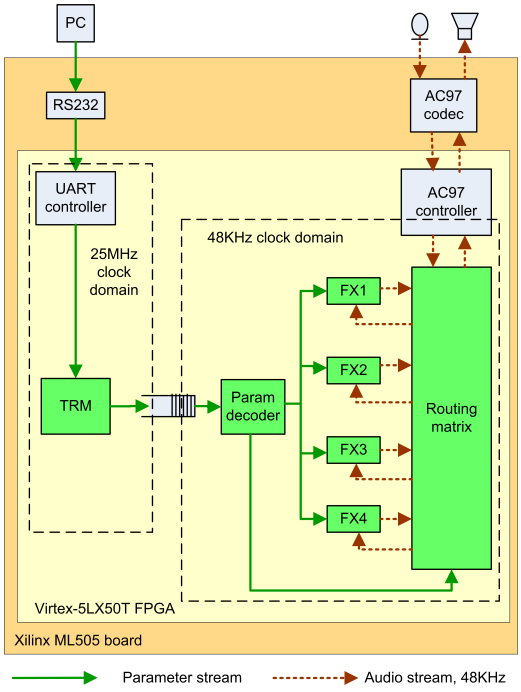

Figure 1. Block diagram of our delay-based audio effects SoC.

Section VI.

II. System Architecture

Figure 1 shows that our real-time delay-based audio effects SoC is implemented on a Virtex-5LX50T chip, which is used on our target platform - a Xilinx ML505 board. The board uses an AC'97 compatible audio codec with 48KHz sampling rate and 20-bit resolutions. Four delay-based audio effects, i.e. chorus, flanger, reverberation and vibrato, are realized in the system. These effects are implemented in the dedicated hardware, and can send and receive a real-time audio-stream to / from a routing matrix to allow different audio effects to be dynamically combined. The audio effects hardware and the routing matrix receive a parameter stream from the FIFO (First In and Fist Out) connected to the TRM soft-core processor. The system architecture shown in Figure 1 can provide reliable performance, i.e. 48K samples / second, as required by real-time audio effects with flexible configurability.

To achieve the reliable performance and system flexibility with low-power consumption, the system implementation is divided into two clock domains: a 25MHz clock domain and a 48KHz sampling clock domain. The 25MHz clock drives the TRM and a UART (Universal Asynchronous Receiver and Transmitter) controller. The 48KHz clock drives a parameter decoder, delay-based audio effects and the routing matrix hardware to allow the system to produce output samples at the sampling rate. An asynchronous FIFO is used to allow communication between these two clock domains.

The reliable performance of the system is ensured by the dedicated hardware used to realize audio effects and route the audio stream. The buffer memory required by the audio effects is implemented using on-chip BRAMs. Therefore, there is no off-chip memory like SDRAM built in the system. In addition, the Xilinx ISE library provides various interfaces of BRAM macros. This allows us to cascade and combine BRAMs to form different sizes of buffer memory for each effect. This buffer memory implementation avoids memory bottleneck and unreliable performance caused by a global buffer memory accessed via the software running on a soft-core processor. Together with the dedicated routing matrix hardware, it finally decouples the software from the hardware and allows us to use different clock domains to achieve low-power consumption with ensured performance.

The system flexibility is achieved by the software running on the soft-core processor TRM. The software receives effect parameters and routing commands via the UART to control the audio effects and the routing matrix during the run time. According to a user's requirements, a set of simple commands or a complicated graphical application can be developed on the user's desktop or tablet PC. The PC-side application can send the effect parameters and routing commands to the target system to allow users to adjust the effects in real time. A wireless-to-UART adapter can also be used to allow users to control the target system in a more convenient way. If another user interface, for example a MIDI controller, is preferred by end users, simply replacing the UART controller with another I/O controller will be sufficient. The performance critical data path will not be affected at all. Therefore, this system architecture not only allows end users to easily control the system, but also allows developers to easily adapt the system to different I/O interfaces.

The interface between the software and hardware is an asynchronous FIFO buffer. The TRM processor accesses this FIFO buffer via memory mapped I/O. The simple interface between the software and the hardware parts reduces the area cost. As a result, it also reduces the system power consumption.

The low-power consumption of the system is mainly achieved by the low clocking rate used in the dedicated hardware. In our system, 74% (2302 LUTS out of a toal 3100 LUTS) of the circuitry is used for the dedicated hardware. Considering that our dedicated hardware is running at 48KHz, the dynamic power consumption of our system is much less than the existing systems [6], [7]. In addition, the power consumption of our system is further reduced by not using off-chip memory, the small footprint of the soft-core processor and the simple interface between the soft-core processor and the dedicated hardware.

III. Dedicated Hardware Design

This section introduces the architecture and the configurable interfaces of the hardware components in the audio stream data path. The hardware components and their organization introduced here take advantage of the DSP support in modern FPGA devices, such as multipliers, fast carry chains and BRAMs, to reduce the resource and power costs of the system.

A. Basic Building Blocks

Delay-based digital audio effects are widely used in recording studios and live musical events to create different interpretations of the sense of space and environment. This section presents the design and implementation of the configurable delay-based audio effects.

1) Delay Line: The delay line is an elementary functional unit which models acoustic propagation delay. It is a fundamental building block of both delay-effects processors and digital-waveguide synthesis models. The function of a delay line is to introduce a time delay, corresponding to M samples between its input and output [8].

Our configurable delay line implementation has an input signal, delaySize, to allow the configuration of the delay buffer size at the run time. The number of cycles for the circuitry to react to the new delay size configuration is |newDelaySize - oldDelaySize|, the difference between the new delay size and the old delay size. During that time, a sample value is interpolated into the audio stream. This delay line implementation can store a maximum of 5K samples. Because the sampling rate of AC'97 codec is 48KHz, the longest delay that can be simulated by the delay line implementation is $(1s/48000) \cdot 5120 = 106.7ms$. According to [9], if the delay is in the range between 10 and 25 ms, a quick repetition called "slapback" or "doubling" can be heard. If the delay is greater than 50ms, an echo will be heard. Therefore, this configurable delay line implementation can simulate both of these effects according to the value of the input signal delaySize.

2) Comb Filter: Comb filters are used to simulate acoustic echo effects. There are two basic comb-filter types: feedforward and feedback comb filters. The feedforward comb filter models a single discrete echo by inserting the delay line into the feedforward path. The feedback comb filter models multiple echoes by inserting the delay line into the feedback loop. Figure 2 shows the schematic of the feedforward and feedback comb filters. The M samples delay in the schematic is implemented by the configurable delay line introduced before. The delay size M and the gain of the comb filters are input signals and therefore can be configured at run time. The multiplication is implemented with DSP slices on the FPGA chip, and uses fixed point numbers.

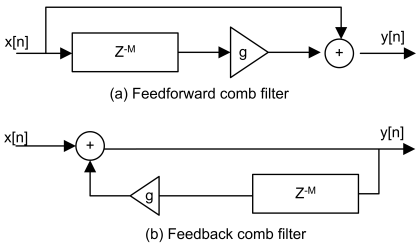

Figure 2. Block diagram of comb filters.

3) All-pass Filter: The feedback comb filter is the main building block for simulating reverberation effects. However, as Schroeder pointed out [10], the amplitude-frequency response of the feedback comb filter has the appearance of a comb with periodic maxima and minima. These periodic maxima and minima impart the undesired "colored" quality to the reverberated sound. To eliminate the undesired "colored" quality and improve the echo density of the reverberation effect, Schroeder and Logan introduced all-pass filters, which have amplitude-frequency 1 for each frequency. In our system, a direct-form-II implementation [11] of an all-pass filter has been implemented.

B. Delay-based Audio Effects

1) Reverberation: Reverberation refers to the prolongation of sound by the environment, which is essentially caused by the reflectivity of surfaces and by the slow speed of sound in air, only about 345m/s at room temperature [12]. The characteristics of natural reverberation include the reverberation time, the dependency of the reverberation time and the frequency, the time gap between the direct sound and the reverberation and, finally, the echo density rate. The reverberation time is the time for a sound to die away to -60dB, and is also called T_{60}. Different audible frequencies have different reverberation times. In a concert hall, for example, a low frequency sound tends to fade away last. The time gap between the direct sound and reverberation gives the listener a different sensation of the space. For example, a delay less than 5ms creates a sensation of a small space, while a delay greater than 50ms gives a distinct echo. When sound radiates from a source, it first reaches the listener along a direct path, then some early reflected and decayed signals reach the listener, and in the end signals reflected several times arrive at the listener and give the sense of very dense echoes. Therefore, the density rate of the echoes in the reverberation should be high enough to emulate this last stage of the sound traveling process. Normally, to obtain a flutter-free reverberation, that is with no distinct audible echoes, there should be approximately 1000 echoes per second. To reflect these reverberation characteristics in a digital reverberator, Schroeder introduced a design that combines comb filters and all-pass filters. Figure 3 shows the

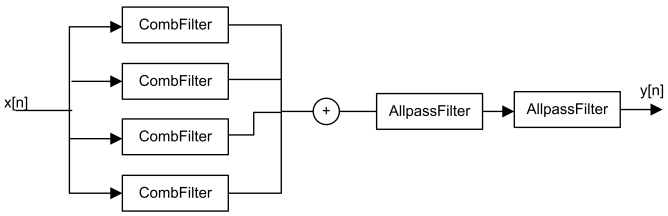

Figure 3. Block diagram of Schroeder reverberator.

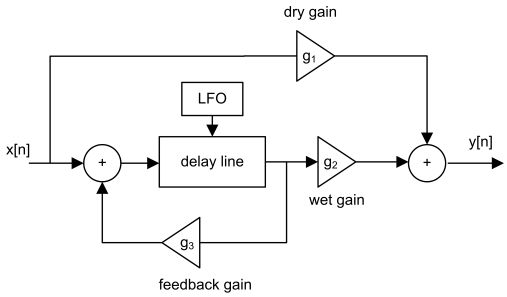

Figure 4. Block diagram of time varing delay effects.

block diagram of the Schroeder reverberator implemented in our system.

The delay line size and the gain of each filter used in Figure 3 can be configured by users at the run time to reflect the reverberation time of the environment.

2) Vibrato, Flanger and Chorus: Vibrato, flanger and chorus effects are all time-varying delay effects designed to thicken the sound via the manipulating of the frequencies of the sound over time. When the delay time between the output signal and the input signal varies periodically, a vibrato effect is generated. To simulate the periodically varying time delay, an LFO (low frequency oscillator) is used with a delay line to change its size periodically. When adding the periodically delayed signal to the original signal, a flanger effect is created. The effect was developed in recording studios in the 1950s by lightly pressing the outer 'flange' of one of two synchronized tape machines alternately. When a number of players perform in unison, the small changes in the amplitudes and timing between each individual result in the chorus effect. Figure 4 gives the block diagram of these three time varying delay effects. Different configurations of the dry gain, wet gain, feedback gain and delay time generate different effects. Here, dry signal means the original signal and wet signal means the delayed signal. Table I shows different configurations used to generate different delay effects.

C. Routing Matrix

The routing matrix used to route audio streams between different effects is implemented using a crossbar interconnect. The 5 inputs of the crossbar are connected to the output audio stream of the AC'97 codec, the output of the chorus effect, the output of the flanger effect, the output of the Schroeder reverberator and the output of the Vibrato effect separately. The 5 outputs of the crossbar are connected to the input of chorus effect, input of flanger effect, input of Schroeder reverberator, input of vibrato effect and the input audio stream to the AC'97 codec.

Table I
TYPICAL PARAMETERS FOR TIME VARYING DELAY EFFECTS.

Effect	g1	g2	g3	LFO Frquency	LFO Amplitude	LFO Offset
Chorus	0-1	0-1	0	0.2-2Hz	0-5ms	15-25ms
Flanger	0-1	0-1	0	0.5-5Hz	0-7.5ms	0-15ms
Vibrato	0	1	0	5-15Hz	0-2.5ms	0-2.5ms

IV. DYNAMIC CONFIGURATION

The dynamic configuration of the system is achieved via the software running on a general purpose soft-core processor, a TRM. This section introduces the TRM architecture and the software design.

A. Tiny Register Machine (TRM)

The TRM was originally designed and implemented on Virtex-5 FPGA by Prof. Niklaus Wirth [13]. The instruction set architecture of TRM is designed to be small but powerful enough to run a program written in a high level programming language, in our case, Oberon[14]. To demonstrate that this design goal is achievable, a multicore processor composed of 12 TRMs and a bus-based interconnect was implemented on a Virtex-5LX50 FPGA [15]. TRM is a Harvard architecture with an 18-bit, 2-address instruction set and 32-bit datapath. The 18-bit instruction encoding allows each memory unit in the Virtex-5 FPGA BRAM slice to store two instructions, Therefore, the scarce memory resource can be fully used. It has 8 working registers and a program counter register (PC). Out of the 8 working registers, register R7 is used in $BL(BranchandLink)$ instruction to store the return address. By default, a TRM processor is configured with 4K instruction memory and 2k data memory. The memory size can be configured to meet the programmers' requirements. The 2-stage pipelined implementation of a TRM runs at 116MHz, and takes 2% LUTs of the Virtex-5XC5VLX50T FPGA. The multiplication in the TRM takes 5 clock cycles.

B. Software Implementation

The software development for a graphically configurable system involved two parts: developing software running on the TRM processor that controls the signal processing engines on the FPGA and developing a graphical software running on a host PC which is connected to the client via some serial interface. The software running on the host PC can be in principle developed using any off-the-shelf tools and languages, such as C#, Java or Matlab. In our

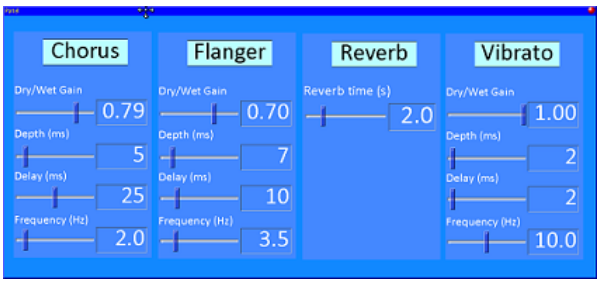

Figure 5. Configuration GUI running on the host PC.

case, Active Oberon is used to develop a graphical user interface (GUI) application. This GUI application allows users to enter configuration parameters and send them to the FPGA via the UART. Figure 5 shows a typical GUI scenario. The software running on the TRM reads the data sent from the host PC, computes the parameters required by the audio effect hardware and writes them into the asynchronous FIFO attached to the TRM and the parameter decoder hardware. The software running on the TRM is also written in Oberon because an Oberon compiler for the TRM had been developed in our group [16]. There is no operating system running on the TRM.

V. System Evaluation

Table II shows the resource usage of the entire system. Table III shows the power consumption results with regards to different system configurations. We consider three different configurations: the proposed 25MHz(TRM)with the 48KHz(Effects) design, and two soft-core-centric approaches that use the same clock for soft-core and audio effects. To the best of our knowledge, the clock used in the existing systems [6], [7] to drive the soft-core and the effects is equal to or more than 48MHz. Therefore, the 25MHz(TRM) with the 25MHz(Effects) and 50MHz(TRM) with the 50MHz(Effects) system configurations listed in Table III are used to estimate the possible power consumption in the similar systems. The power estimation results shown in Table III are generated by XPower[17]. The total power consumption of an FPGA system consists of static power and dynamic power. The static power which can be derived by $V_{DD}I_{leakage}$ is decided by the leakage current ($I_{leakage}$) from the FPGA chipset with V_{DD} supply voltage. As can be seen from Table III, all three configurations consume the same amount of leakage power. The improvement of the proposed solution, which optimizes clock domains for performance-critical hardware components and highly flexible soft-core, comes from the reduction in dynamic power. The dynamic power can be modeled by $ACV_{DD}{}^2 f$, where A, C and f denote the activity factor, the capacitance factor of the FPGA, and the operating clock frequency, respectively. Although the total power consumption of the system is only reduced by 6.5%-13.3% by applying the proposed design

Table II
Resource usage of the system on Virtex-5LX50T chip.

LUTs	BRAMs	DSPs
3100 (10%)	48 (80%)	15 (31%)

idea, we expect that the improvement can increase when more FPGA resources are utilized, because currently only 10% LUTs are used in our system. Reports indicate that dynamic power still dominates the total power consumption of FPGAs and it takes 60%-80% of the total power for the latest 28-nm FPGA [18], [19] assuming the resources are fully used.

Dynamic power includes the dynamic power of soft-core, i.e. TRM, audio effect hardware components, and components such as clock managers and I/O terminations, which are default components in all FPGA designs[20]. Since soft-core and default components are the same for all three cases, the dynamic power reduction comes from reducing the clock rates of audio effect hardware components. Table III reveals that the proposed design concept can reduce dynamic power by 22.9%-39.6% compared with the existing soft-core-centric solutions. After examining the details of the improvement, we find that the power consumption of IOs and PLLs remains unchanged for all three designs. This is because three designs use the same I/Os and the same PLLs which generate the global clocks for the FPGA. DSP is energy efficient and its power consumption is almost negligible in this case study. Clocks and logic reduce dynamic power consumption by 11.5%-27.3% and 26.1%-63.0% compared with soft-core-centric solutions. The improvement is not proportional to the clock frequency since the clock and logic blocks are mainly consumed by the TRM and default components. The dynamic power of signal blocks and block memory (BRAMs) is considerably reduced. The two components are mainly used in audio effect hardware components and can reduce dynamic power significantly by slowing down the clock frequencies from 50MHz and 25MHz to 48KHz. Our experimental measurement of power consumption on the prototype system using a multimeter shows a 25%-30% improvement and matches the estimation results produced by XPower.

VI. Conclusion

The low-power audio effect system presented here gives an example of a heterogeneous architecture for FPGA-based SoCs. The software / hardware partitioning and the user interface of this system show that it can offer the same flexibility as the pure software-based implementation. The clock domain partitioning and the dedicated hardware implementation provide the reliable performance required by a real-time SoC. More importantly, the power consumption estimation results show that this heterogeneous architecture is very power efficient.

Table III
COMPARISONS OF POWER CONSUMPTION FOR DIFFERENT SYSTEM CONFIGURATIONS.

Power consumption item (unit: mW)		25MHz(TRM) 48KHz(Effects)	25MHz(TRM) 25MHz(Effects)	50MHz(TRM) 50MHz(Effects)
Total power		577.78	617.81 (6.5%)*	666.08 (13.3%)*
Leakage power		444.15	444.43	444.77
Dynamic power	Total	133.63	173.38 (22.9%)*	221.31 (39.6%)*
	Clocks	36.04	40.71 (11.5%)*	49.55 (27.3%)*
	Logic	0.17	0.23 (26.1%)*	0.46 (63.0%)*
	Signals	0.62	2.26 (72.6%)*	4.24 (85.4%)*
	IOs	6.91	6.91 (0%)*	6.91 (0%)*
	BRAMs	3.29	36.72 (91.0%)*	73.45 (95.5%)*
	DSPs	0.07	0.07 (0%)*	0.14 (50%)*
	PLLs	86.52	86.48 (0%)*	86.56 (0%)*

*% power saving compared with the proposed 25MHz(TRM), 48KHz(Effects) design.

ACKNOWLEDGMENT

A substantial part of the work is the result of the project 'Supercomputer in the Pocket'. We thank the Microsoft Innovation Cluster for Embedded Software for funding this project. The authors would also like to thank Professor Niklaus Wirth for his work with TRM processor design.

REFERENCES

[1] Z. Smekal, J. Schimmel, and P. Krkavec, "Optimizing digital musical effect implementation for harvard dsp architecture," in *Proceedings of the COST G-6 Conference on Digital Audio Effects*, ser. DAFx-01, Limerick, Ireland, 2001, pp. 33–38.

[2] T. Choi, Y.-C. Park, and D. H. Youn, "Design of time-varying reverberators for low memory applications," *IEICE Transactions*, vol. 91-D, no. 2, pp. 379–382, 2008.

[3] F. P. Ling, F. K. Khuen, and D. Radhakrishnan, "An audio processor card for special sound effects," in *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems*, Lansing MI, 2000, pp. 730–733.

[4] J. Dattorro, "Effect design: Part 1: Reverberator and other filters," *Journal of Audio Engineering Society*, vol. 45, no. 9, pp. 660–684, 1997.

[5] N. Juillerat, S. Schubiger-Banz, and S. M. Arisona, "Low latency audio pitch shifting in the time domain," in *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems*, ser. ICALIP 2008, Shanghai, China, 2008, pp. 29–35.

[6] M. Pfaff, D. Malzner, J. Seifert, J. Traxler, H. Weber, and G. Wiendl, "Implementing digital audio effects using a hardware/software co-deisgn approach," in *Proceedings of the 10th International Conference on Digital Audio Effects*, ser. DAFx-07, Bordeaux, France, 2007, pp. 125–132.

[7] R. Trausmuth, C. Dusek, and Y. Orlarey, "Using faust for fpga programming," in *Proceedings of the 9th International Conference on Digital Audio Effects*, ser. DAFx-06, Montreal, Canada, 2006, pp. 287–290.

[8] J. O. S. III, *Physical Audio Signal Processing*. Stanford University: Julius O. Smith III, W3K Publishing, 2010, available electronically from: https://ccrma.stanford.edu/ jos/pasp/.

[9] U. Zölzer, X. Amatriain, D. Arfib, J. Bonada, G. D. Poli, P. Dutilleux, G. Evangelista, F. Keiler, A. Loscos, D. Rocchesso, M. Sandler, X. Serra, and T. Todoroff, Eds., *DAFX:Digital Audio Effects*, 1st ed. The Atrium, Southern Gate, Chichester West Sussex PO19 8SQ, England: John Wiley Sons, Ltd, 2002.

[10] M. R. Schroeder, "Natural sounding artificial reverberation," *Journal of the Audio Engineering Society*, vol. 10, no. 3, pp. 219–223, 1962.

[11] J. O. S. III, *Introduction to Digital Filters with Audio Applications*. Stanford University: Julius O. Smith III, W3K Publishing, 2007, available electronically from: https://ccrma.stanford.edu/ jos/filters/.

[12] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 20, no. 5, pp. 1421–1448, 2012.

[13] N. Wirth, "The Tiny Register Machine (TRM)," ETH Zürich, Computer Systems Institute, Tech. Rep. 643, 10 2009.

[14] N. Wirth and J. Gutknecht, *Project Oberon : the design of an operating system and compiler*. New York etc.: ACM Press, 1992.

[15] L. Liu, "A 12-core processor implementation on FPGA," ETH Zürich, Computer Systems Institute, Tech. Rep. 646, 10 2009.

[16] F. Friedrich, L. Liu, and J. Gutknecht, "Active cells: A computing model for rapid construction of on-chip multi-core systems," in *ACIS-ICIS*, 2012, pp. 463–469.

[17] Xilinx, "Xpower," *http://www.xilinx.com/products/design_tools/logic_design/verification/xpower.htm*.

[18] J. Hussein, M. Klein, and M. Hart, "Lowering power at 28 nm with xilinx 7 series fpgas," *http://www.xilinx.com/support/documentation/white_papers/wp389_Lowering_Power_at_28nm.pdf*.

[19] Altera, "Reducing power consumption and increasing bandwidth on 28-nm fpgas," *http://www.altera.com/literature/wp/wp-01148-stxv-power-consumption.pdf*.

[20] Xilinx, "Xpower estimator user guide," *http://www.xilinx.com/support/documentation/user_guides/ug440.pdf*.