A System-On-Chip Platform for HRTF-Based Realtime Spatial Audio Rendering

Wolfgang Fohl, Jürgen Reichardt, Jan Kuhr
HAW Hamburg
University of Applied Sciences
Hamburg, Germany
Email: fohl@informatik.haw-hamburg.de, juergen.reichardt@haw-hamburg.de, jankuhr@hartschall.de

***Abstract*—A system-on-chip platform for realtime rendering of spatial audio signals is presented. The system is based on a Xilinx Virtex-5 ML507 FPGA platform. On the chip an embedded μBlaze microprocessor core and FIR filters are configured. Filtering is carried out in the FPGA hardware for performance reasons whereas the signal management is performed on the embedded processor. The azimuth and elevation angles of a virtual audio source relative to the listener's head can be modified in real time. The system is equipped with a compass sensor to track the head orientation. This data is used to transform the room related coordinates of the virtual audio source to the head related coordinates of the listener, so that a fixed position of the virtual sound source relative to the room can be attained regardless of the listener's head rotation. Head related transfer functions (HRTF) were sampled in steps of 30° for azimuth and elevation. Interpolation for intermediate angles is done by either interpolating between the coefficients of the measured HRTFs at the four adjacent angles (azimuth and elevation), or by feeding the audio signal through the corresponding four filters, and mixing the outputs together. In the latter case the required four filter processes per output stereo channel do not result in longer computing time because of the true parallel operation of the FPGA system. The system output is identical to the output of a corresponding Matlab prototype.**

***Keywords*-Mixed-reality audio; realtime HRTF interpolation; system-on-chip;**

I. Introduction

Spatial sound rendering is important for audio playback, and for creating realistic virtual environments for simulations and games. For headphone-playback devices, techniques based on head related transfer functions (HRTF) are widely used, not only in virtual reality applications, but also for stereo enhancements in home audio systems and mobile audio players [1]–[4]. There are already consumer products available, that use HRTF-based audio spatialisation with head tracking [5], [6].

For a realistic spatial impression of a virtual sound source, the perceived source location must stay fixed relative to the room when the listener's head is turned, so a headphone-based system will have to perform a coordinate transformation between head-related and room-related coordinates. A quick update of head position data is necessary to prevent a perceivable delay between head movement and HRTF adjustment.

The system described here is aimed at mixed-reality audio applications, which require a mobile device with realtime behaviour. For such applications, systems-on-chip consisting of a field programmable gate array (FPGA) with an embedded microprocessor on it are versatile and flexible platforms. The application of the time-variant HRTFs to the audio signal is done on the FPGA hardware. The filters are configured in the VHDL language (VHDL stands for Very high speed integrated circuit Hardware Description Language [7]). The tasks of signal routing and signal management are performed by the C program running on the embedded μBlaze-Processor.

In the next sections an overview of related work is given, and the fundamentals of audio spatialisation with HRTFs are outlined. Then the design of our system is described with emphasis on the partitioning of the application to hardware and software, and on the interface between the embedded μBlaze processor and the surrounding FPGA chip. Results are presented and the paper finishes with the discussion of results, summary, and outlook to future work.

II. Theoretical Background

A. Related Work

Since its beginnings in the mid-1970's, dummy head stereophony has found continuous research and development interest [8]. With increasing computing power of audio workstations it has become feasible to perform realtime rendering of a virtual audio environment [1]. The problem of proper out-of-head localisation has been addressed by many authors. It turns out, that a HRTF-based solution combined with a room reverberation model yields the best results [9]. HRTF rendering algorithms will always have to interpolate between the stored filter coefficients for measured angles. In a recent investigation the threshold of spatial resolution in a virtual acoustic environment has been investigated [3]. The reported result is, that the auditory localization has a resolution of 4° to 18°, depending of the source direction. Interpolation with minimum phase + allpass [2] In PC-based realtime systems an effective way of designing HRTF filters is to perform a minimum phase plus allpass decomposition, where the minimum phase part models the frequency response of the HRTF, and the allpass part, which is usually replaced by a delay, models the phase response [2]. Crossfading algorithms for HRTF filter interpolations are described in [2] and [3]. FPGA systems turn out to be suitable platforms for mobile audio processing [10], [11].

B. Basic Concepts

1) Spatial Audio Rendering: The human auditory system uses (at least) three binaural properties of a sound signal to determine the direction of the source: Inter-aural intensity differences (IID), inter-aural time differences (ITD), and the angular variation of the spectral properties of the sound. Concerning only the inter-aural time and intensity differences leaves an ambiguity, the "cone of confusion": All source locations on this cone yield the same ITD and IID. This ambiguity is partly removed by the angle-dependent spectral properties of the perceived sound, which result from the transmission properties of the signal path from the source to the eardrums. These three properties are completely represented by

 ISBN: 978-1-61208-110-6

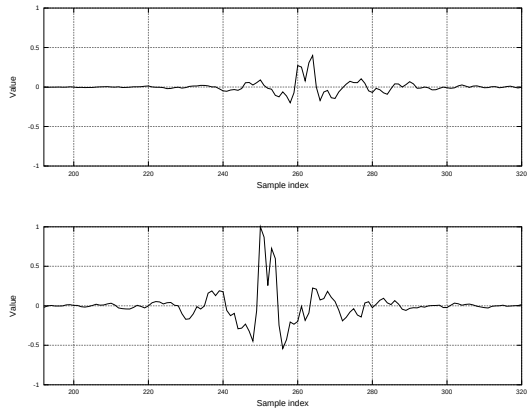

Figure 1. HRTF impulse responses for 0° elevation and 30° azimuth angle. Top: left ear, bottom: right ear

the head related transfer functions (HRTF) for given azimuth and elevation angles.

Figure 1 shows the impulse response of the HRTF at 0° elevation and 30° azimuth angle, where the time and level differences can clearly be seen. As the right ear is closer to the source, the absolute values of the right impulse response samples are larger. The sound reaches the right ear earlier which causes the shift of the maximum of the right channel to shorter delays.

The spatial rendering is done by filtering the source sound with the HRTFs of the corresponding angle and playing the resulting stereo signal back by a set of headphones.

In dynamic listening situations, listeners resolve the ambiguity of the cone of confusion by slightly turning their heads: the resulting changes in ITD, IID and sound spectrum allow a proper localization of the source.

2) FPGA System-On-Chip: The low-level FPGA architecture consists of a pool of logic blocks for combinational and registered logic, RAM-memory and DSP slices. DSP slices consist of a MAC block (*multiply-accumulate*) and registers of appropriate width to perform the multiply-accumulate operations in digital signal processing. The logic cells and DSP slices are interconnected by a user programmable switch matrix. By programming this switch matrix the user defined functionality of the system is obtained.

To handle the complexity of larger systems, the design tools for the FPGA system support a block structured approach by defining *intellectual property blocks* (IP cores), that implement special functions like FIR filters or even microprocessors (in our case the emulation of a μBlaze processor, see section III-C). Once these IP cores have been developed or purchased, the high-level design task is to properly interconnect these cores and to supply the necessary glue logic.

With the availability of a microprocessor core on the FPGA chip, it is possible to design a complete software / hardware system, where the software part is written in C and executed on the processor core, and the hardware part is specified in the VHDL language and is performing time-critical and hardware-related tasks. Systems with this architecture are called *system-on-chip (SoC)*.

C. HRTF coefficients

In preliminary measurements, HRTFs were measured with a dummy head measuring system and an audio spectrum analyser. Attenuation and phase differences were measured for 500 logarithmically spaced sine wave frequencies. The results are in good conjunction with other HRTF measurements [12], [13].

From the frequency response data the FIR filters modelling the angle dependent sound properties were designed using standard frequency-domain design techniques, with the special feature of considering the measured phase by adding it to the linear base phase. This directly introduces the interaural time delays to the FIR coefficients as can be seen in Figure 1, which is actually a plot of the FIR coefficient values over coefficient index.

D. HRTF interpolation techniques

Two approaches of interpolating HRTFs for angles between the sampled positions are considered: the first approach is the *interpolation of the filter coefficients*, the second approach is the *crossfading (mixing) of appropriate filter outputs*. In the stationary case (no variation of source angle) these two approaches are equivalent. In the next two paragraphs the two techniques are explained for the case of constant elevation. If also the elevation angle is to be interpolated, the interpolation of four filters has to be performed (see section III-D5).

1) Coefficient Interpolation: The coefficient interpolation for an angle φ that lies in the interval $[\varphi_k, \varphi_{k+1}]$ is done by linear interpolation of each of the FIR parameters b_i. The implementation of Equation 1 requires $2L$ additions and L multiplications per filter, where L is the FIR filter length.

$$b_i(\varphi) = b_i(\varphi_k) + \frac{\varphi - \varphi_k}{\varphi_{k+1} - \varphi_k} \cdot \Big(b_i(\varphi_{k+1}) - b_i(\varphi_k)\Big) \qquad i \in \{0 \dots L-1\} \tag{1}$$

It should be noted that this approach is not applicable to IIR filters.

2) Crossfade Interpolation: The crossfading approach according to Equation 2 obtains the output signal y_φ by mixing the filter outputs of the two filters corresponding to the interval limits φ_k and φ_{k+1}. The relative contribution of the two outputs is controlled by the parameter m.

$$y_\varphi = (1-m) \cdot y_{\varphi_k} + m \cdot y_{\varphi_{k+1}} \quad \text{with} \quad m = \frac{\varphi - \varphi_k}{\varphi_{k+1} - \varphi_k} \tag{2}$$

This interpolation is also suited for IIR filters. It requires only three multiplications and three additions *per audio sample* at the extra expense of running the audio material through two filters simultaneously.

A key feature of FPGA systems is the ability of *true simultaneous* execution of the filtering: The parallel operation of multiple filters does *not* require more *processing time*, instead it requires more FPGA *resources*, in particular, it requires at least *one DSP slice per filter*. The maximum filter length per DSP slice is given by the ratio of system clock frequency and audio sampling rate [11], [14]. The device presented here works with 125 MHz processor clock frequency and 44.1 kHz audio sampling rate, allowing a maximum filter length of $\frac{125 \cdot 10^6\,\mathrm{Hz}}{44.1 \cdot 10^3\,\mathrm{Hz}} \approx 2800$.

 ISBN: 978-1-61208-110-6

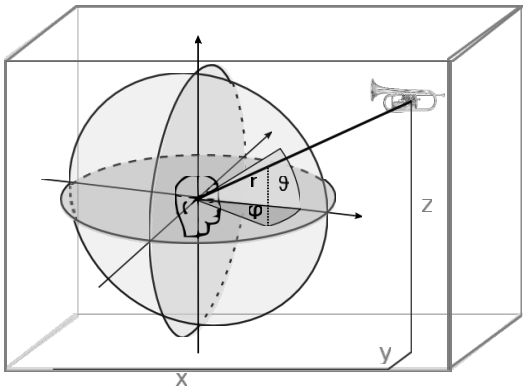

Figure 2. Head Related Coordinates r, ϑ, ϕ and Room Related Coordinates x, y, z

E. Head tracking

For a realistic spatial impression of headphone-based 3D-audio the transformation from head-related to room-related coordinates according to Figure 2 is necessary. For a virtual audio source that is supposed to remain fixed in the room, the orientation of the listener's has to be continuously measured and a corresponding correction of the apparent source direction has to be applied.

III. System Design and Implementation

A. Requirements

The requirements listed here are a consequence of the intended use of the system as a mobile device for spatial audio rendering in mixed-reality environments.

- Low power consumption, small and light system.
- Audio signals in CD quality: 44.1 kHz sampling rate, 16 Bit word length, 2 channels.
- Multiple parallel FIR filters with $\geq$ 512 coefficients.
- Tracking of the head azimuth and pitch (forward) angle. For future developments also the roll (sideways) angle and the acceleration data for three axes must be measured.
- Sufficient memory to hold the filter coefficient sets.
- Audio latency $\leq$ 10 ms to avoid perceivable delay.
- Architecture must be extensible to multiple independently moving virtual audio objects.

B. System Components

Our system is based on a Xilinx Virtex-5 ML507 FPGA evaluation board. In addition to the FPGA chip this board provides a large number of resources and interfaces, the most important ones being an AC97 audio interface, a RS 232 serial interface, a Compact Flash (CF) memory interface, for which a file system driver is provided, and a DDR2 RAM interface. In addition there is a DIP-switch interface for simple user interaction (e.g., switching of operating modes).

The azimuth and elevation angles of the listener's head are provided by a compass sensor (Ocean Server OS5000 [15]), that is mounted on the headphone clip and is connected to the system via the RS 232 link.

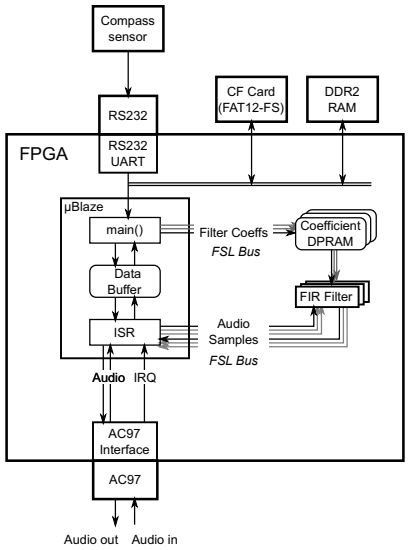

Figure 3. FPGA Components and Interfaces

C. Hardware Architecture

On the FPGA chip is the embedded μBlaze processor IP core for signal and data management, the FIR filter blocks, and the block interconnection logic. The μBlaze is a 32-bit big-endian RISC processor with a library to access the FPGA chip hardware and a runtime environment for a C `main()` routine. The processor was configured without a floating-point coprocessor to save FPGA resources for the HRTF filters. The FPGA chip is configured with 64 kB on-chip RAM for the μBlaze processor, Dual-ported RAM blocks for the FIR filter coefficients, 256 MB of external DDR2 memory and a 512 MB CF card with a FAT12 filesystem, which can be accessed by the standard C file I/O routines. Figure 3 gives an overview of the relevant system components and interfaces.

External devices like the AC97 and the RS 232 can be accessed by the μBlaze program with library functions provided by Xilinx. The interconnection with the on-chip FIR blocks is established via the *Fast Simplex Link* (FSL) bus. The FSL bus is an unidirectional bus which also performs the synchronisation of sender and receiver to the system clock. Three FSL bus instances per filter were implemented for parameter transfer, audio input, and audio output.

Incoming audio samples generate an interrupt which will be served by the interrupt service routine (ISR) on the μBlaze.

The FIR filters for the HRTF filtering are implemented in a direct form I (DF1) structure as a sequential processing pipeline utilising only one DSP slice per filter block [14].

The active FIR filter coefficients are stored in a dual-ported RAM (DPRAM), so the coefficient update and the filtering may be executed asynchronously.

D. Software Architecture

1) Operating Modes: Two basic modes were implemented: a *realtime mode*, and an *offline mode*. In realtime mode, spatial audio rendering is triggered by the interrupts of the AC97 interface. Each incoming sample raises an interrupt, the ISR takes the input sample, transfers it to the filters, receives the filtered audio sample, performs

 ISBN: 978-1-61208-110-6

the mixing if required, and sends it to the AC97 interface for playback.

In *offline* mode audio data is read from wav-files stored on the CF memory card. Audio samples are processed in the same way as in realtime mode, but in offline mode the whole program is executed at maximum speed in the cyclical `main()` program, and the output is stored in a wav-file on the CF card for further evaluation.

In realtime mode the timing and latency of the two interpolation techniques are investigated; in offline mode the correctness of implementation is checked by comparing the output wav-files with the results of corresponding MATLAB computations.

For both modes either of the two interpolation techniques, *coefficient interpolation* or *crossfading* may be selected as interpolation mode. In coefficient interpolation mode, each data update from the compass sensor triggers the calculation of a new interpolated coefficient set for the filters for left and right audio channel according to Equation 1. The coefficient sets are then transferred via the FSL bus to the coefficient DPRAM on the hardware. In crossfading interpolation mode according to Equation 2, each update of the compass sensor data triggers the computation of a new mixing factor, which is written to the global data space where the ISR can access it.

2) Filter Implementation: The FIR filter coefficients of the HRTFs were calculated in MATLAB from measurement data. The normalisation of the filter coefficients was done in an empirical way, starting with $L1-$ normalized coefficients. These coefficients lead to very low output amplitudes and thus a poor S/N ratio. For typical input signals a normalisation factor was determined experimentally, that led to no audible overflow.

Data has been converted to 16-bit Q15 integer format using the MATLAB fixed-point toolbox, and stored in binary format on the CF card. Intermediate results in the filter block are stored in 32 bit wide registers.

Filter coefficients are loaded from CF memory by the `main()` routine of the μBlaze C program, and are transferred to the hardware filters via the FSL bus connections of the filters.

3) Head Tracking: The azimuth and elevation (pitch) angles of the compass sensor are read in the `main()` routine, and are used for calculating the audio source angles in head-related coordinates. The compass sensor provides a third angle, giving the sideways bend of the head (*roll* angle). This angle is neglected because performing the necessary trigonometrical computations in integer arithmetic on the microprocessor would be too time-consuming.

4) Signal Routing: All audio signals are processed by the μBlaze ISR and transferred to the filters via the FSL bus. To minimize overhead, the 16 bit samples for left and right channel are combined to a 32 bit word and transferred as one item to the filters. The filter connection logic then extracts the two samples from the transferred word. Figure 4 shows the interconnection between the μBlaze processor and the hardware filters.

Filter coefficients are transferred from processor to hardware from within the processor's `main()` function using the same technique of transferring two 16 bit coefficients at once.

5) Implementation of Crossfading: Figure 5 shows the principle of crossfading interpolation for azimuth and elevation. The mono input signal is fed to four stereo FIR filter pairs, for the top left, top right, bottom left, and bottom right position of the interpolation interval. From the compass sensor data the azimuth and elevation

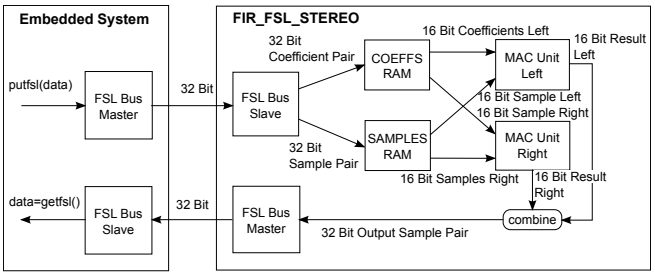

Figure 4. Data Flow for the Filtering Process

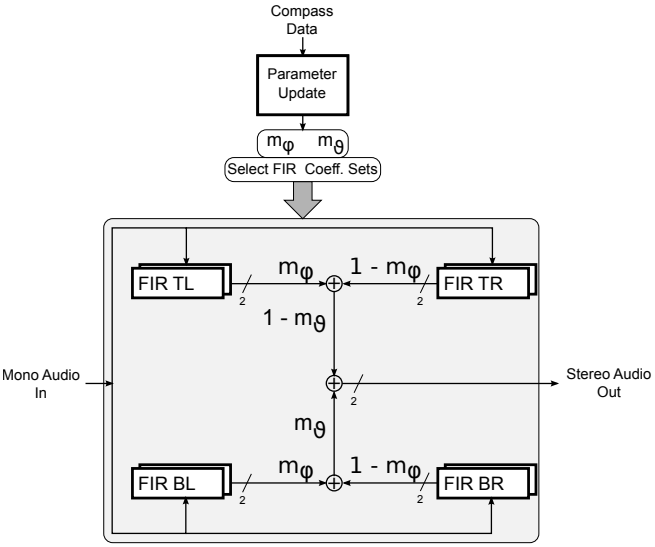

Figure 5. Signal Crossfading Interpolation between Top Left (TL), Top Right (TR), Bottom Left (BL), and Bottom Right (BR) Filter Outputs

mixing parameters m_φ and m_ϑ are computed, and the filter outputs are superposed according to the two mixing parameters. When the azimuth and elevation data from the compass data indicate that the current interpolation interval has been left, the FIR coefficient sets are reloaded according to the new interval.

IV. Results and Discussion

A. Verification of the Static Filtering Algorithms

The filtering algorithms for both interpolation techniques have also been implemented in Matlab using the fixed-point toolbox, and the results have been compared with the wav-files that are produced by the FPGA system in offline mode. Test cases were filtering at the measured HRTF angles and at different constant interpolated azimuth and elevation positions.

In these measurements the outputs of the FPGA system and the Matlab implementation were bit-wise identical.

Both interpolation algorithms produced identical output signals.

B. Signal Processing Latencies

To assess and optimize system performance, and to examine, if the data transfer times will limit the maximum number of audio objects (i.e., independent filter processes), detailed timing measurements have been carried out.

 ISBN: 978-1-61208-110-6

Table of System Latencies:

t_{AC97}: AC 97 audio subsystem (Note 1)	1 ms
t_{FSL}: FSL transfer (round-trip) of one 32 bit word	300 ns
t_{FIR}: FIR processing L=512	4.2 μs
t_{Parm}: Parameter transfer for 512 32-bit parameter pairs	120 μs
t_{gd}: Filter group delay	$\leq$ 7 ms
t_{CS}: Compass sensor sampling time	25 ms
t_{CT}: Compass sensor data transfer (Note 2)	2.4 ... 22 ms
t_{AL}: System audio latency for N audio objects (Note 3)	
$t_{AL} = t_{AC97} + N \cdot t_{FSL} + t_{FIR} + t_{gd}$	8 ms
t_{HLI}: Head tracking latency	
$t_{HLI} = t_{AL} + t_{CS} + t_{CT} + t_{Parm}$	35 ... 55 ms

Notes:

1) This value has been measured in a previous work [11]. It is the time for transferring a stereo audio sample from the AC97 to the FPGA, decode it in hardware, re-code and pass it back to the AC97 output *without* routing the audio signal through the μBlaze processor.
2) The compass data is transferred in ASCII. Small values have less digits and thus need less transfer time then large values.
3) Delay between audio input and audio output.

The table shows that the filtering of one audio sample takes much longer time than the FSL bus transfer, so the ISR can be substantially sped up by replacing the standard *blocking* data transfer routines `putfsl()` and `getfsl()` by their *nonblocking* counterparts `nputfsl()` and `ngetfsl()`. In the case of blocking transfer as shown in Figure 6, the `getfsl()` routine has to wait until the filtering process has finished, whereas in the nonblocking case as shown in Figure 7 the `ngetfsl()` routine returns immediately. In the latter case the ISR gets the result of the *previous* filtering process, adding an extra latency of 1 audio sampling time to the system, which is negligible compared to the group delay of the filter.

The limiting factor for the number of audio objects is the fact, that each audio object will require one FSL bus transfer that is executed *sequentially* in the C program of the μBlaze processor. An upper limit can be estimated by the requirement, that all the FSL bus transfers must be completed within one audio sampling period t_S:

$$N \cdot t_{FSL} < t_S \tag{3}$$

For $t_S = 1/44100\,\mathrm{s}$, the upper limit for N is 75 audio objects.

C. Filtering With Compensation of Head Movement

At the moment of writing, only qualitative listening tests have been performed. The compensation of the head movement drastically increases the spatial impression of the rendered audio material. No front-back ambiguity was noticed. A much better externalization of the sound was perceived, even in the case of source locations directly in front of the listener, where externalization is known to be most difficult to obtain [9].

The latency of approximately 40 ms for the compensation of the head movements by evaluating the compass data is perceivable only at fast and abrupt head movements, where it causes a slight irritation. For head movements at moderate speed no latencies are perceivable. This is due to the limited angle resolution (4° to 18°) of the human auditory system [3].

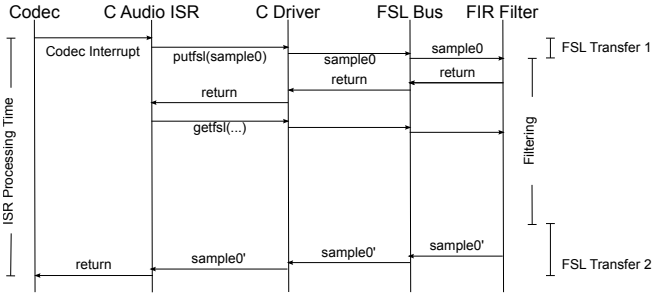

Figure 6. Blocking Filtering Process Sequence Diagram

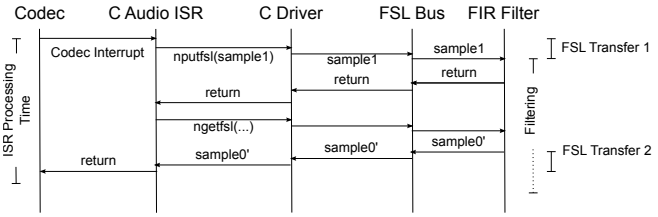

Figure 7. Non-Blocking Filtering Process Sequence Diagram

1) Coefficient Interpolation: The coefficient interpolation algorithm according to Equation 1 leads to artifacts at fast head movements or fast moving sources. This is due to the fact, that the coefficient modification is asynchronous with the filtering, so for fast moving sources the coefficient sets may be inconsistent during the parameter transfer. As shown in section IV-B, this transfer takes $120\,\mu s$, which is approximately 5 audio sampling times, so 5 audio samples will be filtered with inconsistent filter coefficients, which will cause the audible artifacts.

2) Crossfading: With the crossfading interpolation algorithm according to Equation 2 and Figure 5, no artifacts were audible in our listening tests, as long as the source azimuth and elevation angles remain in the same interpolation interval. In the moment, where the interval boundaries are crossed, there is the risk of artifacts which arise from the same reason as in the parameter interpolation case: The parameter update of the four involved filter pairs takes longer than one audio sample, so the filter coefficients are inconsistent during this update.

V. Conclusion

A. Summary

The system-on-chip platform presented in this paper turned out to be well suited as a mobile component of a mixed-reality audio system. The maximum number of virtual audio sources is limited by the 126 DSP slices on the FPGA chip. Two slices are needed for the headphone compensation, so 124 slices remain for the HRTF filters. One audio object requires 8 DSP slices (4 stereo filter pairs at the borders of the interpolation interval), so $\lfloor 124/8 \rfloor = 15$

independent objects could be rendered with the current system design.

Compared to the upper limit of 75 audio objects from the consideration of the bus transfer times in section IV-B, it turns out that the number of available DSP slices is actually the limiting factor for the number of audio objects.

The preferred HRTF interpolation technique is the crossfading of filter outputs. Here the only problem to overcome is the disturbance that occurs when the limits of the interpolation interval are left. In the next section a possible solution to this problem will be outlined.

The coefficient interpolation technique is not suited for this system platform, because one parameter update takes about 5 audio sampling times. During this time the filtering occurs with inconsistent coefficient sets leading to audible artifacts in the output signal.

B. Outlook

Systematic listening tests will have to be conducted to investigate whether the interpolation introduces a perceivable degradation of audio quality. Furthermore, tests will have to be carried out to determine the source localization accuracy with and without head movement compensation.

One issue with the current implementation of the crossfade interpolation is the disturbance caused by reloading the filter coefficients, when the source angles cross the boundaries of the interpolation intervals. This problem can be overcome by introducing additional "standby" filters that provide the output signals of the adjacent angle intervals.

The current implementation uses the audio input of the AC97 subsystem as audio source. To render multiple audio objects, there will be needed multiple source audio streams. These audio streams will have to be transferred to the system via the network or the USB interfaces of the Virtex board.

With the HRTF filtering only the *direction* of a virtual audio source can be rendered. To additonally reproduce the *position* of a virtual source, the influence of the *source distance* on the perceived sound must be modeled and applied to the output signal. Currently we are investigating these effects with the aim of creating a sufficiently simple distance model that can be executed in real-time on the Virtex system-on-chip platform.

REFERENCES

[1] J. W. Scarpaci and H. S. Colburn, "A System for Real-Time Virtual Auditory Space," in *Proceedings of ICAD 05-Eleventh Meeting of the International Conference on Auditory Display, Limerick, Ireland*, vol. 9, 2005, pp. 6 – 9. [Online]. Available: http://www.dell.org

[2] B. Carty and V. Lazzarini, "Binaural HRTF Based Spatialisation: New Approaches and Implementation," in *Proc. Of the 12th Int. Conference on Digital Audio Effects (DAFx-09), Como, Italy*, 2009.

[3] A. Lindau and S. Weinzierl, "On the Spatial Resolution of Virtual Acoustic Environments for Head Movements in Horizontal, Vertical and Lateral Direction," in *Proc. Of the EAA Symposium on Auralization, Espoo, Finland*, vol. 17, 2009, pp. 15 – 17.

[4] A. Lindau, "The Perception of System Latency in Dynamic Binaural Synthesis," in *NAG/DAGA 2009 - Rotterdam*, 2009, pp. 120 – 180.

[5] Beyerdynamic, "Headzone System," Accessed 08/05/2010, available at http://europe.beyerdynamic.com/shop/hah/headphones-and-headsets/at-home/headphones-amps/headzone-home-hz.html.

[6] S. R. LLC, "Realiser A8," Accessed 08/05/2010, available at http://www.smyth-research.com/products.html.

[7] V. Analysis and S. Group, "Behavioural languages–part 1: Vhdl language reference manual," IEC Standard 61691-1-1: 2004, 2004.

[8] J. Blauert, *Spatial Hearing The Psychophysics of Human Sound Localization*. MIT Press, 1983, vol. 9.

[9] T. Liitola, "Headphone Sound Externalization," Master's thesis, Helsinki University of Technology, Department of Electrical and Communications, 2006.

[10] S. Kurotaki, N. Suzuki, K. Nakadai, H. G. Okuno, and H. Amano, "Implementation of Active Direction-Pass Filter on Dynamically Reconfigur able Processor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 8912 – 8913.

[11] W. Fohl, J. Matthies, and B. Schwarz, "A FPGA-based Adaptive Noise Cancelling System," in *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09), Como, Italy*, 2009.

[12] B. Gardner and K. Martin, "HRTF Measurements of a KEMAR Dummy-Head Microphone," MIT Media Lab, Cambridge, MA 02139, MIT Media Lab Perceptual Computing Technical Report No.280, 1994.

[13] O. Warusfel, "LISTEN HRTF Database," Accessed 08/05/2010, available at http://recherche.ircam.fr/equipes/salles/listen/index.html.

[14] J. Reichardt and B. Schwarz, *VHDL-Synthese Entwurf digitaler Schaltungen und Systeme*, 5th ed. München: Oldenbourg Wissenschaftsverlag, 2009.

[15] O. S. T. Inc., "Digital Compass Users Guide, OS5000 Series," Accessed 08/05/2010, available at http://www.ocean-server.com/download/OS5000_Compass_Manual.pdf.

 ISBN: 978-1-61208-110-6