Reconfigurable Accelerator for WFS-Based 3D-Audio

Dimitris Theodoropoulos
D.Theodoropoulos@tudelft.nl

Georgi Kuzmanov
G.K.Kuzmanov@tudelft.nl

Georgi Gaydadjiev
g.n.gaydadjiev@tudelft.nl

Computer Engineering Laboratory
EEMCS, TU Delft
P.O. Box 5031, 2600 GA Delft, The Netherlands

Abstract

In this paper, we propose a reconfigurable and scalable hardware accelerator for 3D-audio systems based on the Wave Field Synthesis technology. Previous related work reveals that WFS sound systems are based on using standard PCs. However, two major obstacles are the relative low number of real-time sound sources that can be processed and the high power consumption. The proposed accelerator alleviates these limitations by its performance and energy efficient design. We propose a scalable organization comprising multiple rendering units (RUs), each of them independently processing audio samples. The processing is done in an environment of continuously varying number of sources and speakers. We provide a comprehensive study on the design trade-offs with respect to this multiplicity of sources and speakers. A hardware prototype of our proposal was implemented on a Virtex4FX60 FPGA operating at 200 MHz. A single RU can achieve up to 7x WFS processing speedup compared to a software implementation running on a Pentium D at 3.4 GHz, while consuming, according to Xilinx XPower, approximately 3 W of power only.

1. Introduction

Creation of an accurate aural environment has been studied for many decades. The first stereophonic transmission was done by Clement Ader at the Paris Opera stage in 1881, while the first documented research on directional sound reproduction was done at AT & T Bell Labs in 1934 [1]. During 1938 and 1940, the Walt Disney studio designed the Fantasound stereophonic sound technology, the first one that introduces surround speakers, with audio channels derived from Left, Center and Right. An improved technology was designed in 1976 by Dolby Laboratories that introduced the quadraphonic surround sound system. It was called Dolby Stereo (or Dolby Analog) and consisted of four separate channels (left, center, right and mono surround) [2]. In 1994 the International Telecommunication Union (ITU) specified the speaker layout and channel configuration for stereophonic sound systems [3].

Today, there are many multichannel audio technologies requiring various speakers setups. However, sound reproduction techniques can be split into 3 fundamentally different categories: 1) stereophony; 2) generation of the signals that reach the ears (binaural signals); and 3) synthesis of the wavefronts emitting from sound sources.

In this paper, we focus on the third category and, more precisely, to the Wave Field Synthesis (WFS) technology [4]. Furthermore, we propose a reconfigurable hardware accelerator that efficiently accelerates its most computationally intensive part. The idea stems from the fact that all previous audio systems that utilize WFS technology, are based on standard PCs. Such an approach introduces processing bottlenecks which limits the number of sound sources that can be rendered in real-time. Power consumption is also increased because in most cases more than one PCs are required to drive a large number of speakers.

In contrast, the proposed reconfigurable accelerator offers:

- An efficient processing scheme with a performance of 531 clock cycles at 200 MHz per 1024 audio samples;
- Low resources utilization which leads to a low power consumption;
- The option to configure more rendering units (RUs) and process samples concurrently.

A prototype with a single RU was built based on the proposed accelerator and mapped on a Virtex4 FX60 FPGA with the following characteristics:

- Rendering up to 64 real-time sound sources when driving 104 speakers, while commercial products based on a single PC can support up to 64 sources rendered through 32 speakers [5], [6], [7];
- 3032 Virtex4 slices;
- 7x speedup compared to a Pentium D running at 3.4 GHz;
- 218 MHz maximum operating frequency after design is placed and routed;

- Estimated total power consumption of 3 W per RU.

The remainder of this paper is organized as follows: Section 2 presents a brief analysis of the previously proposed audio technologies, discusses all required arithmetic operations in order to render sound sources based on the WFS technology and describes some audio systems that utilize it. In Section 3, we describe and analyze the proposed accelerator, while Section 4 reports performance results and compares our work to other audio systems. Finally, in Section 5, we conclude the discussion.

2. Background And Related Work

In this section we present an overview of the previously proposed audio technologies. We also provide a theoretical background on the WFS technology and discuss various audio systems based on it.

Audio technologies: Stereophony is the oldest and most widely used audio technology. The majority of home theater and cinema sound systems are nowadays based on the ITU 5.1 standard. This is mainly caused by the fact that such systems are easy to be installed due to their rather small number of speakers. However, the ITU 5.1 standard requires a specific speaker configuration in the azimuthal plane, which unfortunately cannot be satisfied in most cases. Furthermore, various tests have shown that sound perception on the sides and behind the listener is poor, due to the large distance between the speakers. Another important drawback of stereophony is that phantom sources cannot be rendered between the speakers and the listener [8] [2].

Binaural synthesis (or binaural recording) refers to a specific method used for audio recording that implies putting two microphones facing away from each other at a distance equal to human ears (approximately 18 cm). However, with this microphone topology recorded signals do not take into account how head, torso, shoulders and outer ear pinna would affect frequency adjustments of a sound while it arrives at ears. The influence of the aforementioned parts of the human body on the frequency spectrum can be modeled by special filter functions, so called Head Related Transfer Functions (HRTF) [9]. Binaural systems can deliver a high quality of sound perception and localization. However, they require that the listener wears headphones or, when sound is rendered through speakers, additional crosstalk cancelation filters [2].

Finally, as we mentioned, an additional way of delivering a natural sound environment is audio technologies that can synthesize wavefronts of a virtual source. The most important benefit of these technologies is that they do not constrain the listening area to a small surface, as it happens with stereophonic systems and binaural setups without headphones. On the contrary, a natural sound environment is provided in the entire room, where every listener experiences an outstanding sound perception and localization. However, their main drawback is that they require large amount of data to be processed and many speakers to be driven.

The two technologies that try to synthesize wavefronts are Ambisonics and Wave Field Synthesis (WFS). Ambisonics was proposed from Oxford Mathematical Institute in 1970 [10]. Researchers focused on a new audio system that could recreate the original acoustic environment as convincingly as possible. In order to achieve this, they developed a recording technique that utilizes a special surround microphone, called the Soundfield microphone. Ambisonics sound systems can utilize an arbitrary number of loudspeakers that do not have to be placed rigidly.

WFS was proposed by Berkhout [4]. It is essentially based on Huygens' principle stating that a wavefront can be considered as a secondary source distribution. In the audio domain, Huygens' principle is applied by stating that a primary source wave front can be created by secondary audio sources (plane of speakers) that emit secondary wavefronts, the superposition of which creates the original one. However, some limitations arise in real world systems. For example, in practise a plane of speakers is not feasible, so a linear speaker array is used, which unavoidably introduces a finite distance between the speakers. This fact introduces artifacts such as spatial aliasing, truncation effects, amplitude and spectral errors of the emitted wavefront [11].

Theoretical background: Figure 1 illustrates an example of a linear array speaker setup. Each speaker has its own unique coordinates (xs_i, ys_i) inside the listening area. In order to drive each one of them so as the rendered sound source location is at A(x1, y1), the following operations are required to calculate the so called *Rayleigh 2.5D operator* [12]: filtering of the audio signals with a 3 dB/octave correction filter [13] and calculation of the delayed sample and its gain, according to each speaker distance from the virtual source. To render a source behind the speaker array, the inner product z between its distance from each speaker $\overrightarrow{d_1}$ and each speaker normal vector $\overrightarrow{n}$ must be calculated. Then the amplitude decay *AD* is given by the following formula [12]:

$$AD = \sqrt{\frac{Dz}{(Dz + z) * |\overrightarrow{d}|}} * cos(\theta) \quad (1)$$

where Dz is called *reference distance*, the distance where the Rayleigh 2.5D operator can give sources with correct amplitude, $|\overrightarrow{d}| = |\overrightarrow{d_1}|$, and $cos(\theta)$ is the cosine of angle θ between the vectors $\overrightarrow{d_1}$ and $\overrightarrow{n}$, as shown in Figure 1.

In order to render a moving source from a point A to a point B behind the speaker array, a linearly interpolated trajectory is calculated [12]: Distance $|\overrightarrow{d_2}| - |\overrightarrow{d_1}|$ is divided by the samples buffer size *bs*, in order to calculate how the source advances with every sample or, in other words, the distance between 2 consecutive audio samples, defined as *unit distance* (UD):

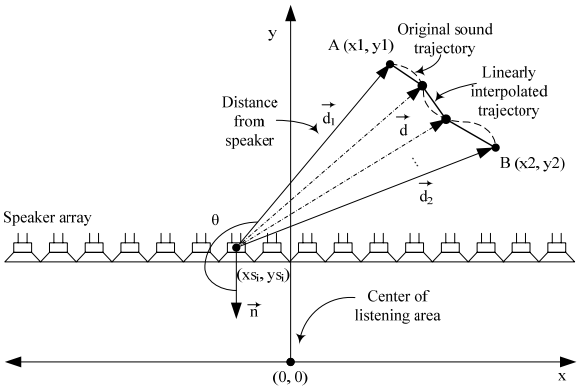

Figure 1. Speaker array setup

$$UD = \frac{(|\overrightarrow{d_2}| - |\overrightarrow{d_1}|)}{bs} \quad (2)$$

Based on the distance *UD*, the source distance $|\overrightarrow{d}|$ from speaker *i* with coordinates (xs_i, ys_i) is updated for every sample by the formula:

$$|\overrightarrow{d}| = |\overrightarrow{d}| + UD \quad (3)$$

According to the current distance $|\overrightarrow{d}|$ from speaker *i*, an output sample is selected based on the formula:

$$delayedsample = -(l + (df * |\overrightarrow{d}|)) + (s++) \quad (4)$$

where $df = f_s/\upsilon_s$ is the distance factor (f_s is the sampling rate, υ_s is the sound speed), *s* is the current output audio sample and l is an artificial latency. Finally, the *delayed sample* is multiplied by the amplitude decay *AD* and the system master volume. The result is stored to an output samples buffer. Further details can be found in [14], [15], [13] and [12]. In Section 3 we explain how these formulas were mapped into our hardware design.

Related work: A sound system that was built in IRT in Munich and called the Binaural Sky [16], actually combines both binaural and Wave Field Synthesis technologies. The Binaural Sky concept is based on the avoidance of Cross Talk Cancelation (CTS) filters real time calculation, while the listener head is rotated. Instead of using two speakers the authors utilize a circular speaker array that synthesizes focused sound sources around the listener. The system uses a head tracking device and, instead of real time CTS filter calculation, it adjusts the speaker driving functions such as delay times and attenuations. The speaker array consists of 22 broadband speakers and a single low frequency driver. All real time processing is done on a Linux PC with a 22 channel sound card. Input sound signals are fed to a software module based on the BruteFIR software convolution engine. Its output is a binaural signal which goes directly to a second software module in order to be convolved with precalculated filters and then drive the speaker array.

In [17], the authors apply WFS technology to a multi tiled hardware architecture called "Scalable Software Hardware computing Architecture for Embedded Systems" (SHAPES). Each of these tiles consists of a Distributed Network Processor for inter-tile communication, a RISC processor and one mAgicV VLIW floating point processor. According to the paper, a WFS system capable of supporting 32 sound sources while driving up to 128 speakers, would require 64 such tiles.

Two companies, SonicEmotion [5] and Iosono [6], produce audio systems based on WFS technology. SonicEmotion rendering unit is based on Intel Core2Duo processor and consumes an average power of 360 W. It supports rendering up to 64 real-time sound sources, while driving a 24 speaker array. Iosono rendering unit is also based on a standard PC approach and supports up to 64 real-time sources while driving 32 speakers. In both cases, when more speakers are required, additional rendering units have to be cascaded.

The authors of [18] describe a real-time immersive audio system that exploits WFS technology. The system performs sound recording from a remote location A, transmits it to another one B, and renders it through a speaker array utilizing the WFS technology. In order to preserve the original sound exact coordinates, a tracking device is employed. A beamformer also records the sound source, but without the acoustic properties of the recording location A. Thus, a dry source signal with its coordinates is transmitted to B. The WFS rendering unit receives this information along with the acoustic properties of the reproduction room B. The result is the same sound source being rendered exactly at the same position under B acoustic properties. The complete system consists of 4 PCs, out of which one used for the WFS rendering.

In [19], the authors propose an immersive audio environment for desktop applications. Their system also utilizes the WFS technology. Small speakers are placed around the computer display, which allows the listener to move freely inside the listening area. Again, the system is based on a standard 2 GHz PC.

3. Proposed Design

This section describes our complete Fabric Co-processor Module (FCM)[1] that accelerates the WFS algorithm considered. We start by analyzing our design specifications and continue with an extensive hardware analysis.

Results Accuracy: Our goal is a design capable of supporting sound sources rendered in a listening area that spans from 1 m in front of the speaker array (focused sources) up to 16 m behind the speaker array (normal sources). The reason why we limit rendering area to the above mentioned dimensions, is

1. We follow Xilinx terminology.

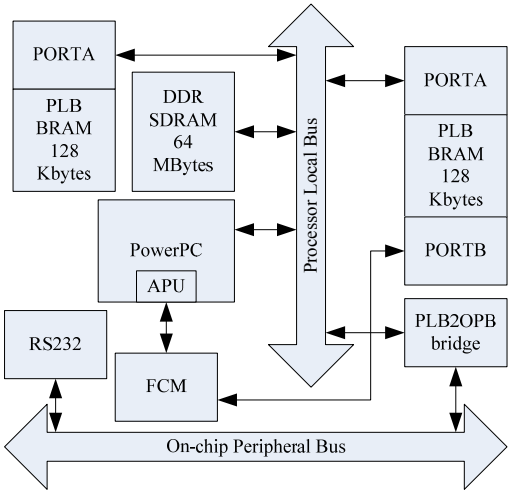

Figure 2. Complete Design Infrastructure

because inside this area the Rayleigh 2.5D operator can provide sources rendering with acceptable amplitude errors [20].

Utilizing a floating point format (e.g. IEEE 754) for our calculations would result in a complex hardware design with unnecessary high accuracy. For this reason we wrote a software program that simulates a hypothetical speaker array setup consisting of 50 speakers with a distance of 15 cm between each other. We placed sound sources in front and behind it with 5 cm steps and analyzed all internal calculations with respect to the needed calculations accuracy. Previous experiences with WFS audio systems, suggest that sources with at least 0.5 m/sec velocity should be identified as moving (slower sources are rendered as still ones). Results suggested that if our system supported fixed point operations with 5 integer bits and 17 decimal bits, it could identify moving sources (spanning in the previously described listening area) with the aforementioned velocity.

Complete Infrastructure: Figure 2 illustrates the complete infrastructure of the system we consider for our design. The PowerPC utilizes a 128-Kbytes instruction memory connected to the Processor Local Bus (PLB) through its PORTA. A second memory of 128 Kbytes is used by the PowerPC for temporal storage of on-chip data through its PORTA. For this reason the latter is connected to the PLB, while PORTB is connected directly to the FCM. This shared memory implementation allows the FCM to access memory more efficiently compared to accessing it through the PLB. A 64-Mbytes DDR SDRAM is used to store audio samples, which can be accessed from PowerPC again through the PLB. The FCM is connected directly to the PowerPC through its Auxilary Procesor Unit (APU) interface [21]. In our case, we configured it to decode one User Defined Instruction (UDI) that would start the FCM. In order also to monitor the correct functionality of our system, we connected the FPGA board through an RS232 module to a standard PC.

Audio Hardware Accelerator: In each loop, the PowerPC fetches 1024 16-bit audio samples from SDRAM and stores them to an on-chip BRAM. When samples storing is done, the FCM execution is initiated via our customized UDI, as shown in the following pseudocode snippet:

```
For all audio samples in SDRAM
{
    copy 1024 samples from SDRAM to BRAM;
    UDI (source Header, samples Address);
    copy samples from BRAM to SDRAM;
}
```

Figure 3 presents the FCM organization; it consists of a primary controller, a 64-tap FIR filter and a RU. The latter integrates a speaker coordinates buffer and two modules called *Preprocessor* and *WFS engine*. The speaker coordinates buffer is used to store all speakers coordinates inside the listening area. The Preprocessor is responsible for calculating the unit distance, amplitude decay and distance from each speaker at a specific time. The WFS engine selects all appropriate filtered audio samples, with respect to the Preprocessor results.

Figure 4 shows the FCM functionality as a flowchart. The FCM receives two parameters after UDI decode; a sound source header, i.e. its coordinates inside the listening area, and a pointer to audio samples array previously stored into BRAM. The FCM controller starts reading audio data from BRAM and forwards them to the FIR filter. All filtered samples are stored in a 1024x16 samples buffer that resides inside the WFS engine. Once samples filtering is done, the FCM forwards the sound coordinates along with current speaker coordinates to the Preprocessor and starts its execution. When the Preprocessor finishes, it acknowledges the FCM controller, which then starts the WFS engine. The *i* variable refers to the speaker whose data are being processed. The FCM controller pipelines internally the Preprocessor and the WFS Engine execution. As soon as the first speaker coordinates are processed from the Preprocessor, the latter forwards the results to the WFS Engine, but also starts directly processing the second speaker coordinates. The Preprocessor always finishes before the WFS Engine does. Such an execution overlap between the Preprocessor and the WFS Engine, essentially "hides" the execution time of the former. The WFS Engine processes two samples per cycle that are stored back to BRAM. The same process is repeated until all audio samples for all speakers have been calculated. Once the FCM has finished, all processed samples are written back to the SDRAM and 1024 new audio samples are fetched from SDRAM to BRAM for processing. We should note that, since there are many data transfers between the SDRAM and the BRAM, a Direct Memory Access (DMA) controller can be employed to improve the data-transfer rate.

Preprocessor: In the previous section, we mentioned that the unit distance *UD* (eq. (2)), amplitude decay *AD* (eq. (1)) and distance $|\overrightarrow{d}|$ (eq. (3)) from all speakers are calculated in

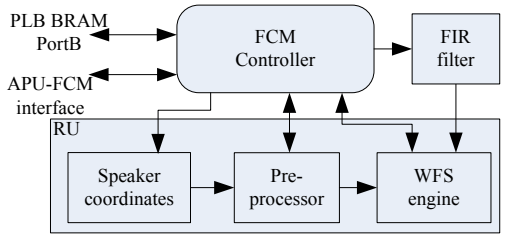

Figure 3. FCM organization

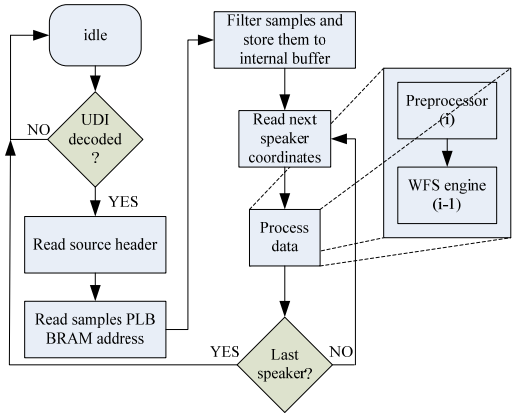

Figure 4. Flowchart that shows FCM functionality

the WFS algorithm. The Preprocessor is designed to calculate all these operations. A more detailed operation analysis suggests that a total of 9 additions/subtractions, 9 multiplications, 3 square root operations and 2 divisions are required per speaker.

Figure 5 illustrates the Preprocessor organization. Targeting a minimalistic design, we decided to utilize only 1 adder/subtractor, 1 multiplier, 1 square root unit and 1 fractional divider. Furthermore, as mentioned before, the Preprocessor always finishes execution before the WFS Engine does. Thus, spending additional resources to accelerate its execution, would eventually make the Preprocessor just being idle for a longer time.

Current speaker coordinates along with source header are stored into local registers. Since there is direct data dependency among many of these operations, the Preprocessor controller issues them serially to the corresponding functional unit. Results are stored again to local registers and reused for further calculations. The Preprocessor requires 142 clock cycles at 200 MHz to complete data processing and the final results are forwarded to the WFS Engine.

WFS Engine: The WFS engine is the core computational part of the design, sketched in Figure 6. As stated above, once the Preprocessor is done, it acknowledges the primary FCM controller. The latter starts the WFS Engine, which reads from the Preprocessor local registers the unit distance, amplitude decay and distance with respect to the current speaker. These data are forwarded to 2 Sample Selection Cores (SSC), SSC_1 and SSC_2, which select the appropriate filtered sound samples from samples buffer (eq. (4)). Each SSC consists of 1 multi-

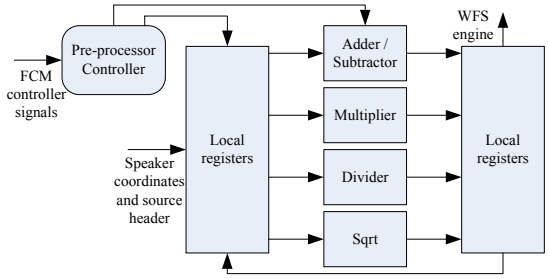

Figure 5. Preprocessor organization

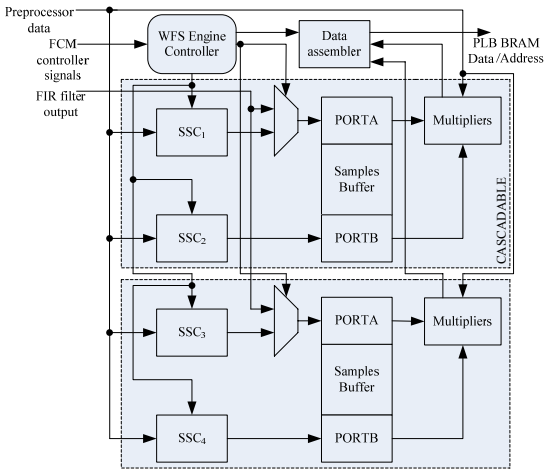

Figure 6. WFS Engine organization

plier, 1 subtractor, 2 accumulators and 1 adder, as illustrated in Figure 7.

Selected samples from SSC_1 and SSC_2 according to equation (4), are multiplied by the system master volume level and amplitude decay and forwarded to the Data Assembler. The latter generates a 64-bit word consisting of four 16-bit audio samples that are written back to on-chip BRAM through its PortB.

The WFS Engine repeats the above process for 1024 samples, processing 2 samples per clock cycle, thus a total of 512 cycles. Also there are 11 more cycles spent on communication among internal modules, which results in a total of 523 required cycles at 200 MHz for all samples.

The number of used SSCs was based on the tradeoff between performance and available resources. RU performance versus the SSCs number for processing 1024 samples, is calculated according to the following formula:

$$cc = 11 + 8 + \frac{buffersize}{SSC} \quad (5)$$

where 11 cycles are the aforementioned communication overhead among the WFS Engine internal modules, and 8 cycles are required for communication among the WFS Engine, the Preprocessor and the FCM primary controller. Formula (5) gives a performance of 1043, 531 and 275 clock cycles for 1, 2 and 4 SSCs respectively. Utilizing more SSCs would cause a BRAM write-back bottleneck, since its width is 64 bits.

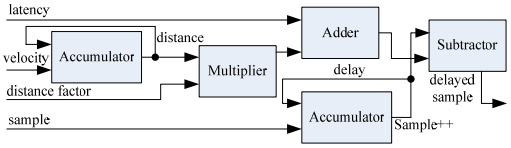

Figure 7. SSC organization

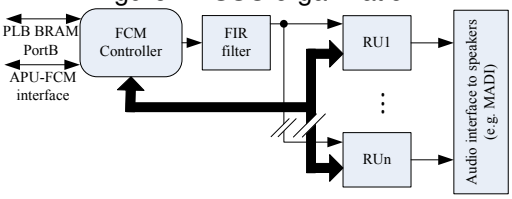

Figure 8. FCM with more RUs working concurrently

An approach of 2 and 4 SSCs would increase the RU performance 1043/531=1.96x and 1043/275=3.79x respectively comparing to a single SSC approach, however, it would require 2x and 4x resources. Based on this analysis, we decided to utilize 2 SSCs which offer a good tradeoff between performance increase and occupied resources. However, we should note that more than 4 SSCs could be cascaded (along with half samples buffers) when data are forwarded to multiple multichannel audio interfaces, as illustrated from the shaded part in Figure 6.

Design Scalability: Specific attention was paid on designing a compact, yet efficient, while also scalable hardware organization. Figure 8 shows how more RUs can be connected when a larger FPGA is available. The FIR filter is a common structure for all RUs. All filtered audio data are broadcasted to every RU and stored inside a local samples buffer. All RUs can work in parallel and forward their results to an interface capable of carrying multiple channels of digital audio, such as the Multichannel Audio Digital Interface (MADI) [22].

For parallel data processing, the speaker coordinates have to be distributed among the RUs local buffers. As an example, if we assume a speaker setup with 32 speakers, we can utilize 4 RUs, where RU0 will process speakers 1 to 8, RU1 speakers 9 to 16, RU2 speakers 17 to 24 and RU3 speakers 25 to 32.

4. Experimental Results

To build a complete system prototype, we used a Xilinx ML410 board with a V4FX60 FPGA on it, which integrates two PowerPC processors. Our WFS accelerator was designed in VHDL and synthesized using the Xilinx Integrated Synthesis Environment (ISE) 9.1.03 and the Xilinx Synthesis Tool (XST).

Hardware complexity: Table 1 displays the FPGA resource utilization with one RU integrated in the FCM. We analyzed how many slices were distributed on each submodule of the system and concluded that the FIR filter consumes approximately 57% of their total number when utilizing one RU. The reason for that is because we implemented the filter utilizing the Xilinx IP core Distributed Arithmetic (DA) approach [23]

Table 1. Embedded system resource utilization

Maximum frequency (MHz)	218
Total Power Consumption (W)	3
XtremeDSP Slices	14
RU Slices	3032
FIR Filter Slices	7152
Peripheral Slices	2205
Total Slices	12389

Table 2. Slices versus XtremeDSP slices proportion

FPGA	Available Slices	Slices / XtremeDSP	RUs Fit
V4FX40	9267	193	3
V4FX60	15923	124	5
V4FX100	32819	205	10
V4FX140	53811	280	13

and not the conventional multiply-accumulate (MAC) one [24]. The main advantage of DA over MAC is that the number of required cycles to produce a result does not depend on the filter length but on the filter input and coefficients width [23]. In contrast, a single cycle output MAC approach of a 64-tap FIR filter would require 64 XtremeDSP slices [24] for up to 18x18 bits data sizes. Such an approach would make our design prohibitive even for large FPGAs that do not have many XtremeDSP slices. Since the size of data that will be filtered is only 16-bit, the DA approach is more suitable. However, one DA drawback is that a single cycle output FIR implementation will utilize an increased number of FPGA slices, since it is always mapped to logic and not to XtremeDSP slices.

We explored the relation between the number of conventional slices and XtremeDSP slices that our design must satisfy, in order to efficiently utilize FPGA resources. In Table 2, we subtracted from each FPGA those slices spent on the FIR filter and peripherals, such as PLB, OPB and RS232 module, i.e. 9357 slices. "Slices/XtremeDSP" column shows a good approximation of what the proportion ($\frac{\#Slices}{\#XtremeDSP}$) in our design between slices and XtremeDSP should be, in order to utilize each FPGA in the most efficient way. We used this analysis as our guideline during the RU design, in order to fit as much RUs as possible in large FPGAs. The rightmost column of Table 2 shows the number of RUs that eventually can fit in each FPGA.

System Verification: We rendered various moving sound sources located inside the hypothetical listening area mentioned in Section 3. Under the same speaker setup, we also run a software version of the WFS rendering function and rendered sources following the same trajectories. Selected delayed audio samples (eq. 4)) from the software version and the WFS Engine coincided, while amplitude decay (eq. 1)) was precise up to the third decimal digit. As an example, Figure 9 illustrates the comparison of the calculated amplitude decay between the Preprocessor and the software implementation, when a source moved from A(1.45m, 3.50m) to B(1.30m, 3.75m). As we can see, hardware hardware results follow the software ones with very high precision (3 decimal digits).

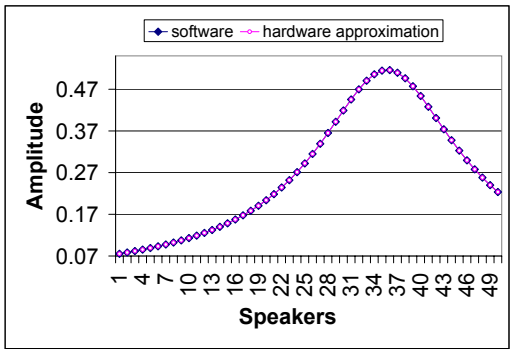

Figure 9. Amplitude comparison between SW and HW

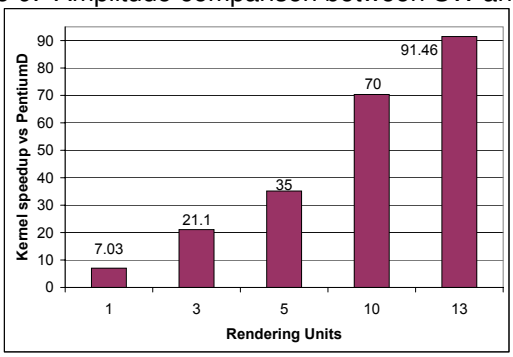

Figure 10. Speedup over WFS software implementation

Performance: In order to calculate the overall performance benefits, we first run the WFS rendering function on a Pentium D 940 at 3.4 GHz with Linux Fedora. We used gprof to measure runtime and the result was 1010 μsecs. Comparison between the software and hardware versions is depicted in Figure 10. A single RU implementation achieves a 7x speedup compared to the software version running on Pentium D. In the same figure, we also provide the potential speedup that can be achieved by placing more RUs, running in parallel.

Finally, we compared our design against the products of SonicEmotion and Iosono, and against the current WFS audio system developed by the Laboratory of Acoustical Imaging and Sound Control of TU Delft [7], [15], [20]. Comparison results are in Figure 11. As we mentioned in Section 2, the SonicEmotion and Iosono WFS rendering units can render up to 64 real-time sources driving 24 and 32 speakers respectively[1]. If additional speakers are required, more rendering units need to be cascaded. In contrast a single RU implementation on a medium size FPGA such as the V4FX40 can render up to 64 real-time sources when driving 104 speakers. Of course we should note that our design does not support all functionalities that professional audio equipment does. In Figure 12, we show the number of real-time rendered sources when multiple RUs are utilized in a single FPGA. As we can see, cascaded RUs

1. SonicEmotion rendering-unit data were confirmed by personal communication with SonicEmotion at info@sonicemotion.com. In order to derive a performance estimation of Iosono's rendering unit, we considered the link http://www.idmt.fraunhofer.de/eng/about_us/facts_figures.htm from the official Fraunhofer web site, stating that 6 Iosono PCs are used to drive 192 speakers.

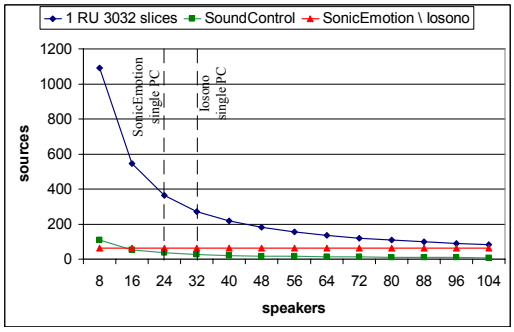

Figure 11. Number of real-time rendered sound sources according to speaker setup

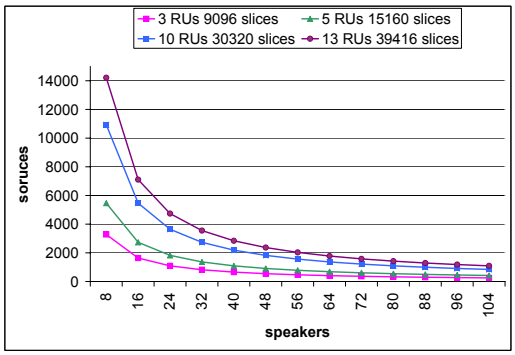

Figure 12. Estimated number of real-time rendered sound sources when multiple RUs are used

can support rendering many hundreds of sources in real-time, even when driving 104 speakers.

Energy efficiency: Another benefit of our FPGA design is that it requires significanlty less power than all other presented systems based on high-end CPUs. We used Xilinx XPower to analyze the complete system power consumption, which reported a total of 3 W. In contrast high-end CPUs, when not in idle mode, normally require tens of Watts, which in essence is an order or two of magnitude difference in favor of our design.

5. Conclusions

In this paper, we proposed a design that accelerates the most computationally intensive part of the WFS algorithm used in contemporary 3D-Audio systems. Previous approaches are based on standard PCs, which still cannot satisfy the computational demands for high number of sources and speakers and cannot meet critical power dissipation constraints. Our observations indicate that commercial, single PC, approaches can offer up to 64 real-time sources when driving no more than 32 speakers, while consuming tens of Watts of power. Furthermore, when more speakers are required, then additional rendering units need to be cascaded, which increases even more the cost and power expenses in traditional PC based systems. In contrast, our reconfigurable design can alleviate the processing bottlenecks. It requires reasonably few resources

and its scalability allows more RUs to process audio samples concurrently. A single RU approach supports up to 64 real-time sources when driving 104 speakers, which is more efficient than traditional PC systems. Meanwhile our single RU design occupies 12389 Xilinx Virtex 4 slices in total, it achieves a 7x speedup compared to Pentium D at 3.4GHz, while consuming only a small fraction of the power, consumed by a general purpose processor.

Acknowledgment

This work was partially sponsored by hArtes, a project (IST-035143) of the Sixth Framework Programme of the European Community under the thematic area "Embedded Systems"; and the Dutch Technology Foundation STW, applied science division of NWO and the Technology Program of the Dutch Ministry of Economic Affairs (project DCS.7533).

The authors would like to explicitly thank Lars Hörchens and Jasper van Dorp Schuitman from the Laboratory of Acoustical Imaging and Sound Control of TU Delft for their valuable contribution to accomplish this work.

References

[1] H. Fletcher, "Auditory perspectiveBasic requirements," in *Electrical Engineering*, vol. 53, 1934, pp. 12–17.

[2] C. Kyriakakis, "Fundamental and Technological Limitations of Immersive Audio Systems," in *Proceedings of the IEEE*, vol. 86, May 1998, pp. 941–951.

[3] T. Holman, *5.1 Surround Sound Up and Running*. Focal Press, December 1999.

[4] A. Berkhout, D. de Vries, and P. Vogel, "Acoustic Control by Wave Field Synthesis," in *Journal of the Acoustical Society of America*, vol. 93, May 1993, pp. 2764–2778.

[5] SonicEmotion Company, "http://www.sonicemotion.com."

[6] Iosono Company, "http://www.iosono-sound.com."

[7] J. van Dorp Schuitman, L. Hörchens, and D. de Vries, "The MAP-based wave field synthesis system at TU Delft (NL)," in *1st DEGA symposium on wave field synthesis*, September 2007.

[8] E. Armelloni, P. Martignon, and A. Farina, "Comparison Between Different Surround Reproduction Systems: ITU 5.1 vs PanAmbio 4.1," in *118th Convention of Audio Engineering Society*, May 2005.

[9] A. Mouchtaris, P. Reveliotis, and C. Kyriakakis, "Inverse of Filter Design for Immersive Audio Rendering Over Loudspeakers," in *IEEE Transactions on Multimedia*, vol. 2, June 2000, pp. 77–87.

[10] M. A. Gerzon, "Periphony: With-Height Sound Reproduction," in *Journal of the Audio Engineering Society*, vol. 21, 1973, pp. 2–10.

[11] J. Daniel, R. Nicol, and S. Moreau, "Further Investigations of High Order Ambisonics and Wave Field Synthesis for Holophonic Sound Imaging," in *114th Convention of Audio Engineering Society*, March 2003, pp. 58–70.

[12] J. van Dorp Schuitman, "The Rayleigh 2.5D Operator Explained," Laboratory of Acoustical Imaging and Sound Control, TU Delft, The Netherlands, Tech. Rep., June 2007.

[13] P. Vogel, "Application of Wave Field Synthesis in Room Acoustics," Ph.D. dissertation, TU Delft, The Netherlands, 1993.

[14] M. Boone, E. Verheijen, and P. van Tol, "Spatial Sound Field Reproduction by Wave Field Synthesis," in *Journal of the Audio Engineering Society*, vol. 43, December 1995, pp. 1003–1012.

[15] W. P. J. D. Bruijn, "Application of Wave Field Synthesis in Videoconferencing," Ph.D. dissertation, TU Delft, The Netherlands, October 2004.

[16] D. Menzel, H. Wittek, G. Theile, and H. Fast, "The Binaural Sky: A Virtual Headphone for Binaural Room Synthesis," in *International Tonmeister Symposium*, October 2005.

[17] T. Sporer, M. Beckinger, A. Franck, I. Bacivarov, W. Haid, K. Huang, L. Thiele, P. S. Paoloucci, P. Bazzana, P. Vicini, J. Ceng, S. Kraemer, and R. Leupers, "SHAPES - a Scalable Parallel HW/SW Architecture Applied to Wave Field Synthesis," in *International Conference of Audio Engineering Society*, September 2007, pp. 175–187.

[18] H. Teutsch, S. Spors, W. Herbordt, W. Kellermann, and R. Rabenstein, "An Integrated Real-Time System For Immersive Audio Applications," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, October 2003, pp. 67–70.

[19] R. H. Alois Sontacchi, Michael Strauß, "Audio Interface for Immersive 3D-Audio Desktop Applications," in *International Symposium on Virtual Environments, Human-Computer Interfaces, and Measurement Systems*, July 2003, pp. 179–182.

[20] E. Hulsebos, "Auralization using Wave Field Synthesis," Ph.D. dissertation, TU Delft, The Netherlands, October 2004.

[21] Xilinx Inc, "PowerPC 405 Processor Block Reference Guide," July 2005.

[22] A. E. Society, "AES10-2003: AES Recommended Practice for Digital Audio Engineering – Serial Multichannel Audio Digital Interface (MADI)," in *Rev 2003*, May 2003.

[23] Xilinx Inc., "Distributed Arithmetic FIR Filter v9.0," April 2005.

[24] ——, "MAC FIR v5.1," April 2005.