AN FPGA-BASED ACCELERATOR FOR SOUND FIELD RENDERING

Yiyu Tan [*]
RIKEN Center for Computational Science
Kobe, Hyogo, Japan
tan.yiyu@riken.jp

Toshiyuki Imamura
RIKEN Center for Computational Science
Kobe, Hyogo, Japan
imamura.toshiyuki@riken.jp

ABSTRACT

Finite difference time domain (FDTD) schemes are widely applied to analyse sound propagation, but are computation-intensive and memory-intensive. Current sound field rendering systems with FDTD schemes are mainly based on software simulations on personal computers (PCs) or general-purpose graphic processing units (GPGPUs). In this research, an accelerator is designed and implemented using the field programmable gate array (FPGA) for sound field rendering. Unlike software simulations on PCs and GPGPUs, the FPGA-based sound field rendering system directly implements wave equations by reconfigurable hardware. Furthermore, a sliding window-based data buffering system is adopted to alleviate external memory bandwidth bottlenecks. Compared to the software simulation carried out on a PC with 128 GB DDR4 RAMs and an Intel i7-7820X processor running at 3.6 GHz, the proposed FPGA-based accelerator takes half of the rendering time and doubles the computation throughput even if the clock frequency of the FPGA system is about 267 MHz.

1. INTRODUCTION

Sound field rendering models sound propagation in spatial and time domains, and is fundamental to numerous scientific and engineering applications, which vary widely from interactive applications, such as computer games and virtual reality, to offline applications like architectural design and noise control. Generally, the sound field rendering algorithms are categorized into geometric methods and wave-based methods. The geometric methods make the assumption that surface primitives are much larger than the wavelength of sound. As a result, the low-frequency diffraction effects of sound wave are lost while the computation capability is reduced significantly. Nowadays, the geometric methods are widely applied in interactive applications because of easy implementation and low computation demand. In contrast, the wave-based methods, such as finite difference time domain (FDTD) methods, boundary element and finite element methods, directly solve the acoustic wave equations in either time domain or frequency domain using numerical methods, and they provide highly accurate modelling of all aspects of sound propagation, including full wave diffraction.

Among the wave-based methods, the FDTD method, which numerically solves the wave equation by using a finite number of grids in a discretized space at discrete time steps, has become an essential approach in room acoustic simulation owing to its ease of implementation and parallelization since it was introduced to analyse acoustical behaviour by O. Chiba et al., and D. Botteldooren et al. [1–3]. However, the FDTD method suffers from the numerical dispersion, which is an inherent problem constraining the valid usable bandwidth.

To reduce numerical dispersion in the FDTD method, many works have been done in 3-D scheme. L. Savioja et al., G. R. Campos et al., and D. Murphy et al. proposed alternative digital waveguide mesh topologies [4–6]; K. Kowalczyk and M. Walstijn developed the explicit second-order accurate schemes, including the 27-point compact explicit FDTD scheme [7]. J. Mourik and D. Murphy investigated two-step high-order explicit "large-star" schemes [8]. B. Hamilton and S. Bilbao introduced the fourth-order accurate explicit and implicit FDTD schemes for 2-D and 3-D wave equations [9][10], respectively. They also developed a set of two-step explicit FDTD schemes with high-order accuracy in both spatial and time domains for 3-D room acoustics [11].

On the other hand, numerical dispersion is still challenge in sound field rendering with the FDTD method, and conventional approach to alleviate the effects of numerical dispersion through spatial grid oversampling incurs significant computational cost. This results in the FDTD method scales poorly with the volume of sound spaces and the analysed maximum frequency. Generally, the computing capability of solving wave equations in the FDTD method is increased as the fourth power of frequency [12] and proportionally with the volume of sound spaces. Given the auditory range of humans (20 Hz–20 kHz), simulating sound wave propagation in a space like a concert hall or a cathedral for the maximum simulation frequency of 20 kHz requires petaflops of computing power and terabytes of memory. Only large computer cluster or supercomputer can satisfy such requirements in current computer systems, but they are prohibitive expensive.

In recent years, GPGPUs and FPGAs were applied to accelerate computation in sound field rendering because of their coarse-grain parallelism of thousands of arithmetic units [13-22]. Currently, an FPGA chip has more hardware resources owing to the development of fabrication technology, including thousands of hardened floating-point arithmetic units, large on-chip block memories, millions of reconfigurable logic blocks. Unlike software simulations on PCs and GPGPUs, FPGA-based sound rendering systems directly implement sound wave equations by configurable logic blocks and hardened arithmetic units inside an FPGA. The system data-path and input/output (I/O) interfaces can be customized in accordance to applications, and thousands of arithmetic units are coordinated to work in parallel to improve computation performance. The incident signals and the rendering results can be directly put in and out through the customized I/O interfaces. From the point of view of real-time processing, FPGA provides a promising solution to real-time sound rendering applications. In our previous work, a FPGA-based accelerator was developed for real-time sound field rendering [17-23]. Although the accelerator outperformed PC-based simulations significantly in

[*] This work was supported by the I-O DATA foundation and JSPS KAKENHI Grant Number JP19K12092.

rendering speed, the rendered sound space is small (32 × 32 × 16 grids) because only small on-chip block memories were applied. In this paper, a FPGA-based accelerator for sound rendering is designed by using high-level synthesis approach, and large external memory is applied to extend the rendering sound volume. The main contributions of this work are shown as follows.

(1) Uniform computing units in rendering algorithm to simplify system design. The explicit FDTD rendering algorithm and its uniform computing format are derived, which makes it easy to design the computing unit in hardware system.
(2) Sliding window-based data buffering to reduce the data access overhead and the requirement of memory bandwidth.
(3) Design and implementation of an FPGA-based accelerator for sound field rendering by using high-level synthesis, including design flow, system architecture, and system implementation.
(4) Evaluation and analysis of system performance based on the prototype machine. The rendering time and computation throughput of the FPGA-based prototype machine are evaluated and compared with those of the software simulation carried out on a PC with 128 GB DDR4 RAMs and an Intel i7-7820X processor running at 3.6 GHz.

The rest of this paper is organized as follows. The rendering algorithm is introduced in Section 2, including the updated equations for general grids and grids on the reflective boundary. In Section 3, the system design and implementation by using the FPGA board DE5a-NET are described, as well as the system architecture and the functions of main components. System performance of the FPGA-based prototype machine is presented in Section 4, followed by the conclusions drawn in Section 5.

2. RENDERING ALGORITHM

The wave equations for 3-D room acoustic simulation is expressed as:

$$(\frac{\partial^2}{\partial t^2} - c^2\nabla^2)p(\mathrm{x},t) = 0 \tag{1}$$

Here, $p(\mathrm{x},t)$ is the sound pressure at time t and position x, $\mathrm{x} = (x,y,z) \in \mathcal{R}^3$ is the spatial position with coordinates being (x,y,z) in a 3-D space, c is the propagation speed of sound in air, the operator $\frac{\partial^2}{\partial t^2}$ denotes the second partial derivative with respect to time, the operator ∇^2 stands for the spatial 3-D Laplacian operator, and $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$. Then, the wave equation (1) can be described by the time domain formulation shown in Equation (2)

$$\frac{\partial^2 p(\mathrm{x},t)}{\partial t^2} = c^2(\frac{\partial^2 p(\mathrm{x},t)}{\partial x^2} + \frac{\partial^2 p(\mathrm{x},t)}{\partial y^2} + \frac{\partial^2 p(\mathrm{x},t)}{\partial z^2}) \tag{2}$$

In order to discretize Equation (2) at time and spatial domains, using $P^n(i,j,k) \cong p(i\Delta x, j\Delta y, k\Delta z, nT)$ as an approximation to $p(\mathrm{x},t)$ at time $t = nT$ and position $\mathrm{x} = (i,j,k)$, where T is the time step, n is the number of time steps, and $\Delta x, \Delta y, \Delta z$ are the Cartesian grid spacing in *x*, *y*, and *z* axes, respectively. Then the temporal and spatial difference operators can be defined as

$$\frac{\partial^2 p(\mathrm{x},t)}{\partial t^2} = \frac{P^{n+1}(i,j,k) - 2P^n(i,j,k) + P^{n-1}(i,j,k)}{T^2}$$

$$\frac{\partial^2 p(\mathrm{x},t)}{\partial x^2} = \frac{P^n(i+1,j,k) - 2P^n(i,j,k) + P^n(i-1,j,k)}{\Delta x^2} \tag{3}$$

$$\frac{\partial^2 p(\mathrm{x},t)}{\partial y^2} = \frac{P^n(i,j+1,k) - 2P^n(i,j,k) + P^n(i,j-1,k)}{\Delta y^2}$$

$$\frac{\partial^2 p(\mathrm{x},t)}{\partial z^2} = \frac{P^n(i,j,k+1) - 2P^n(i,j,k) + P^n(i,j,k-1)}{\Delta z^2}$$

In a cubical grid, letting $\Delta x = \Delta y = \Delta z = \Delta l$ and inserting Equation (3) in Equation (2), Equation (2) is discretized and Equation (4) is yielded.

$$\begin{aligned} P^{n+1}(i,j,k) &= \chi^2[P^n(i+1,j,k) + P^n(i-1,j,k) \\ &+ P^n(i,j+1,k) + P^n(i,j-1,k) + P^n(i,j,k+1) \\ &+ P^n(i,j,k-1)] + (2-6\chi^2)P^n(i,j,k) - P^{n-1}(i,j,k) \end{aligned} \tag{4}$$

where $\chi = \frac{cT}{\Delta l}$ is the Courant number, and cannot be larger than $\frac{1}{\sqrt{3}}$ because of numerical stability in a 3-D sound space. From Equation (4), to compute the sound pressure of a grid needs three multiplications, six additions, and one subtraction. In order to reduce the multiplication operations, which need more clock cycles and hardware resources, χ is assumed to be $\frac{1}{2}$, and Equation (4) is then rewritten as [18][19][23]

$$\begin{aligned} P^{n+1}(i,j,k) &= \frac{1}{4}[P^n(i+1,j,k) + P^n(i-1,j,k) \\ &+ P^n(i,j+1,k) + P^n(i,j-1,k) + P^n(i,j,k+1) \\ &+ P^n(i,j,k-1) + 2P^n(i,j,k)] - P^{n-1}(i,j,k) \end{aligned} \tag{5}$$

In Equation (5), two multiplication operations can be replaced by right and left shift operations, which are easily implemented by hardware and consume less clock cycles.

2.1. Reflective boundary

In realistic room acoustics, the boundary conditions should be considered to model the reflection and absorption from walls. In this study, the reflective boundary is concerned. A reflective boundary can be modelled as a locally reacting surface by assuming that wave does not propagate along with the boundary surface, and the acoustical behavior is only affected by the sound pressure and particle velocity perpendicular to the boundary surface. If a sound wave travels in a positive axis (x,y,z) direction, the boundary impedance Z is denoted by the sound pressure and the particle vibration through Equation (6) [24].

$$Z = \frac{P}{U} \tag{6}$$

Here, U is the particle velocity component perpendicular to the boundary, and P is the sound pressure. For a boundary perpendicular to an axis, the momentum conservation equation of wave propagation is

$$\nabla P + \rho\frac{\partial U}{\partial t} = 0 \tag{7}$$

where ρ is the air density. Differentiating both sides of Equation (6) with respect to t and inserting Equation (7), the boundary conditions are obtained in terms of sound pressure [19][23][25].

$$\frac{\partial P}{\partial t} = -c\xi \nabla P \tag{8}$$

where $\xi = \frac{Z}{\rho c}$ is the normalized boundary impedance. For a cubical sound space, boundary grids are classified into interior grids of a boundary, edges, and corners according to their positions. Different formulas are applied to update their sound pressures. For example, for the interior grids of right boundary, wave travels along the positive x axis direction, and Equation (9) is derived by discretizing Equation (8).

$$\frac{P^{n+1}(i,j,k)-P^{n-1}(i,j,k)}{2T} = -c\xi \frac{P^n(i+1,j,k)-P^n(i-1,j,k)}{2\Delta x} \tag{9}$$

Rearranging the terms in Equation (9) and introducing the parameter χ, Equation (10) is obtained to represent a virtual grid outside the sound space.

$$P^n(i+1,j,k) = P^n(i-1,j,k) + \frac{1}{\chi\xi}[P^{n-1}(i,j,k) - P^{n+1}(i,j,k)] \tag{10}$$

Substituting $P^n(i+1,j,k)$ in Equation (4) with Equation (10), then

$$P^{n+1}(i,j,k) = \Big[\chi^2\big(2P^n(i-1,j,k) + P^n(i,j-1,k) + P^n(i,j+1,k) + P^n(i,j,k-1) + P^n(i,j,k+1)\big) + (2-6\chi^2)P^n(i,j,k) + \left(\frac{\chi}{\xi}-1\right)P^{n-1}(i,j,k)\Big] / \left(\frac{\chi}{\xi}+1\right) \tag{11}$$

By introducing the reflection factor R as $\frac{(\xi-1)}{(\xi+1)}$ and χ being $\frac{1}{2}$, Equation (11) is changed to Equation (12) [19][23], which is applied to update the sound pressure of the interior grids of right boundary.

$$P^{n+1}(i,j,k) = \frac{1+R}{2(3+R)}[2P^n(i-1,j,k) + P^n(i,j-1,k) + P^n(i,j+1,k) + P^n(i,j,k-1) + P^n(i,j,k+1) + 2P^n(i,j,k)] - \frac{3R+1}{3+R}P^{n-1}(i,j,k) \tag{12}$$

Equation (12) consists of the sum of the sound pressures of a grid and its neighbor grids at the time step n, and its sound pressure at the time step $n-1$. Compared with Equation (5), except for the multiplicands, Equation (12) only replaces the sound pressure of the virtual grid $P^n(i+1,j,k)$ by the sound pressure of the neighbor grid $P^n(i-1,j,k)$ in the summation. Moreover, for the interior grids of other boundaries, the multiplicands are same except for the sound pressure of the substituted virtual grid in the summation. For example, the updated equation for an interior grid of left boundary is obtained by substituting the sound pressure of the virtual grid $P^n(i-1,j,k)$ with the sound pressure of the direct neighbor grid $P^n(i+1,j,k)$ in Equation (12). Hence, the summation is changed as $2\,P^n(i+1,j,k) + P^n(i,j+1,k) + P^n(i,j-1,k) + P^n(i,j,k+1) + P^n(i,j,k-1) + 2\,P^n(i,j,k)$. The similar derivation procedure can be applied to edges and corners by using different boundary conditions.

Equations (5) and (12) show that to compute sound pressure of a grid needs the sound pressures of its own and neighbors at previous time steps. For different types of grids, the updated equations have similar formats except for the multiplicands for the summation and $P^{n-1}(i,j,k)$, respectively. From Equations (5) and (12), a uniform updated Equation (13) can be derived.

$$P^{(n+1)}(i,j,k) = D1 * [P^n(i-1,j,k) + P^n(i+1,j,k) + P^n(i,j-1,k) + P^n(i,j+1,k) + P^n(i,j,k-1) + P^n(i,j,k+1) + 2P^n(i,j,k)] - D2 * P^{(n-1)}(i,j,k) \tag{13}$$

As shown in Table 1, the parameters D1, D2, and items in the summation in Equation (13) are associated with the position of a grid. For grids on boundaries, the sound pressures of the virtual grids are replaced by those of the related direct neighbor grids.

Table 1: *Parameters*

Grid position	D1	D2
General	$\frac{1}{4}$	1
Interior	$\frac{R+1}{2\,(R+3)}$	$\frac{3R+1}{R+3}$
Edge	$\frac{R+1}{8}$	R
Corner	$\frac{R+1}{2\,(5-R)}$	$\frac{5R-1}{5-R}$

3. SYSTEM DESIGN AND IMPLEMENTATION

3.1. Design flow

The accelerator is designed using OpenCL, which is a programming language for high-level synthesis of FPGA. As shown in Figure 1, the OpenCL design flow consists of the host and kernel, and the related codes are compiled separately. The accelerator is designed as kernels using OpenCL, which are then compiled to an intermediate representation (LLVM IR), optimized, and converted to the Verilog files by the Intel FPGA SDK for OpenCL. The EDA tool Quartus Prime Pro is called to perform synthesis, placement and routing to generate the FPGA bitstream, which is finally downloaded in the FPGA and executed. The host is developed using C or C++ programming language. It initializes the kernels, maintains the computation flow, and charges data exchange between the host machine and FPGA board. The system drivers and controllers for I/O, such as PCIe bus and DDR memory controllers, are generated and integrated in the system by the Intel FPGA SDK for OpenCL automatically. Therefore, user mainly focuses on designing the kernels. The system design becomes much easier, and the development period is shortened significantly.

3.2. System design

Sound field rendering is memory-intensive. It is impossible to store all data in the on-chip memories inside FPGA as the space volume is increased even if the size of the on-chip memories inside current FPGAs has been increased significantly. Instead, the external large DDR memory is adopted in this research. To improve system performance, the overhead of data access to external memory should be shortened. In the system, a sliding window-based data buffering system is introduced to speed up data access between the rendering engine and the on-board external memory.

In the sliding window-based data buffering, the blocking technique is applied to reduce the buffer size and memory bandwidth demand, in which a large sound space with M × N × K grids is divided into sub-cubes with each having $N_x \times N_y \times N_z$ grids. The sub-cubes are read into the system along the *Z* direction, and computations are carried out. Inside a sub-cube, sound pressures of grids on two consecutive *x-y* planes are kept by buffers. Therefore, the buffer size is reduced from M × N to $N_x \times N_y$. The data buffering system is implemented by using the high-speed and high-bandwidth on-chip block RAMs inside FPGA. However, the problem of the blocking technique is that the halo exists between two sub-cubes, which will incur additional computations.

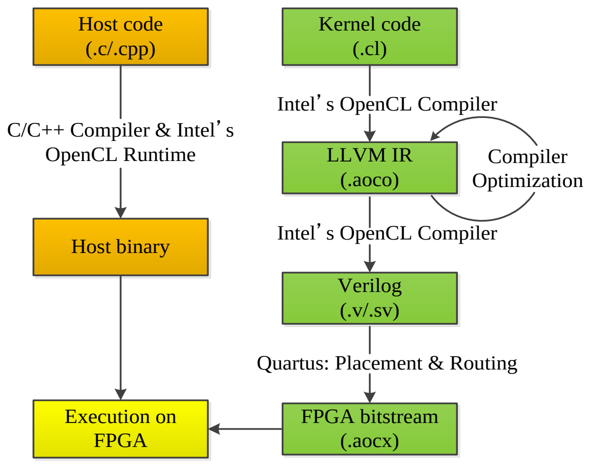

Figure 1: *Design flow*

The system diagram is illustrated in Figure 2, which consists of system controller, three buffers (shift_register_p1, shift_register_p2, and shift_register_posi), computing units, and output controller. The incident data and position flags of grids are firstly written into the on-board DDR memory from the host machine before computation is started. The functions of each module are described as follows.

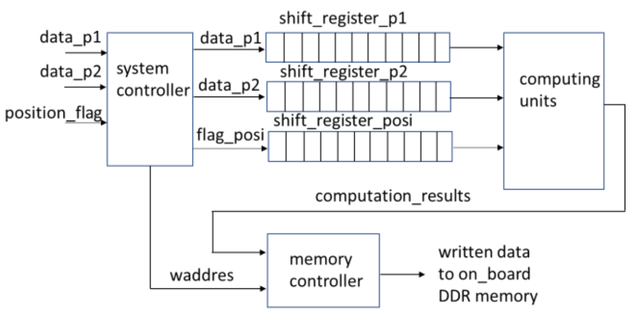

Figure 2: *System diagram*

- system controller. The system controller maintains computation flow and generates control signals according to the computing flow. It reads data and position flags from the on-board DDR memory, and writes them to the buffers shift_register_p1, shift_register_p2, and shift_register_posi, respectively. After computation is completed, it also generates the writing addresses of the grids to store the computation results back to the DDR memory. At each time step, sound pressure of the observation grid is stored in the on-board DDR memory. It is not written back to the host machine until computations at all time steps are finished.
- shift_register_p1, shift_register_p2, and shift_register_posi. The shift_register_p1, shift_register_p2, and shift_register_posi are three buffers to store the data involved in computations and their position flags, respectively. Before computation is started, data in the two continuous *x-y* planes of the sub-cube are streamed into the buffers. The data at the time step *n-1* are stored in the buffer shift_register_p1 while the data at the time step n-2 are kept by the buffer shift_register_p2. And the corresponding position flags of grids are streamed into the shift_register_posi. If the sub-cube contains $N_x \times N_y \times N_z$ grids and *i* grids are computed concurrently, the depth of the shift_register_p1, shift_register_p2, and shift_register_posi is $N_x * N_y + i$. Along with computation, the three buffers are shifted right by *i* data, and another *i* new data and their position flags are streamed into the buffers at each clock cycle. Such procedure is repeated until sound pressures of all grids inside a sub-cube are computed, and then computation is moved to the next sub-cube. The three buffers are implemented by the high-width and high-speed block memory inside FPGA. In the current design, the sub-cube has 256 × 256 × 256 grids and *i* is 16.
- computing units. The computing units is the arithmetic unit to calculate sound pressures of *i* grids concurrently according to the input sound pressures at previous time steps and location indicators (position_flag). The location indicator is used to select the multiplicands D1 and D2 in Equation (13). As shown in Figure 3, a uniform computing unit is designed based on Equation (13), which consists of a 7-input adder, a subtractor, two multipliers, and four multiplexers [19][23]. In Figure 3, the multipliers are used for boundary grids while they are replaced by the right and left shifters for general grids. Two multiplexers are applied to select the multiplicands D1 and D2 in accordance to the location indicator of a grid. At each clock cycle, sound pressures of *16* grids are computed in parallel.
- memory controller. The memory controller stores the computation results to the external on-board DDR memory.

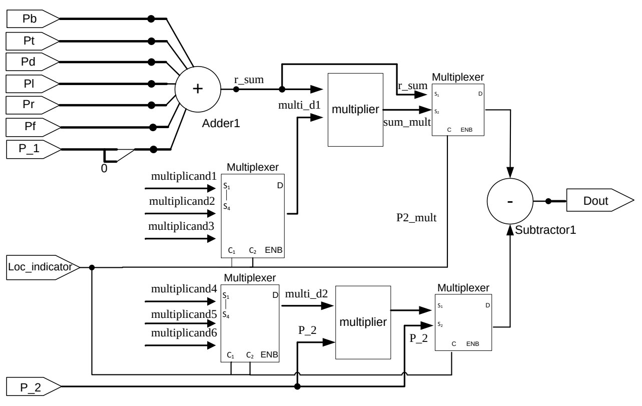

Figure 3: *Computing unit*

3.3. System implementation

The accelerator is implemented by using the FPGA board DE5a-NET from the Terasic Company [26], which contains an Intel Arria 10 FPGA and 8 GB on-board DDR memory arranged in two independent channels. As shown in Figure 4, the incident data and position information of all grids are firstly written into the on-board memory from the host machine through the PCIe bus. The Rendering Engine reads an incident datum from the memory, calculates the sound pressures of all grids, and stores the computation results back to the DDR memory. Then another incident datum is read into, and computations are repeated. This procedure is iterated until all incident data are read into the rendering engine, and sound pressures of all grids are obtained. Finally, the sound pressure at the observation point will be written back to the host machine. The two independent DDR memory will be updated in turn.

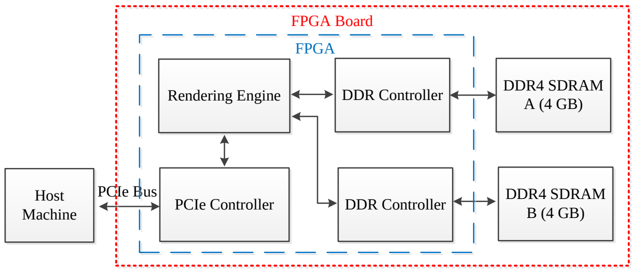

Figure 4: *System diagram*

When the accelerator is implemented by the FPGA board DE5a-net, the hardware resource utilization is shown in Table 2. From Table 2, the current design consumes less half of the available hardware resources inside the FPGA chip Arria 10, and the system performance can be improved further by using more computation kernels to work in parallel.

Table 2: *Hardware resource utilization*

Logic utilization	DSP blocks	RAM blocks	Clock frequency
70701 (17%)	152 (10%)	891 (33%)	267 MHz

4. PERFORMANCE EVALUATION

To estimate the performance of the proposed accelerator, the rendering time in the sound spaces with grids being 128 ×128 × 128, 256 × 256 × 256, 510 × 510 × 510, and 764 × 764 × 764, is measured. The reflection coefficient of boundaries is 0.95, and the time steps are 1000. As a comparison, the same system is developed using C++ programming language, parallelized using OpenMP, and executed on a PC with 128 GB DDR4 memory and an Intel i7-7820X processor, which has eight cores running at 3.6 GHz. The reference C++ codes are compiled by the gcc compiler with the soption -O3 and -fopenmp to use all eight cores in the PC. The simulation and execution environment are shown in Table 3. As shown in Table 3, the memory size of the FPGA-based system, including the external and on-chip memories, is much smaller than that of the PC in software simulation, and the clock frequency of FPGA is about 267 MHz while the PC runs at 3.6 GHz.

4.1. Rendering time

Figure 5 shows the rendering time taken by the software simulations on the PC and the FPGA-based system in the case of different sound space volumes. In Figure 5, the sub-cube is with 128 × 128 × 128 grids in the case of sound space volume being 128 × 128 × 128 grids while it is with 256 × 256 × 256 grids in other cases. The number of grids computed in parallel is 16. As shown in Figure 5, the rendering time taken by the FPGA-based accelerator is almost half of that consumed by the software simulations on PC. In addition, due to the effect of the existing halo, the simulated area becomes a little smaller.

Table 3: *Technology specification*

	FPGA	Software simulation
Model	Arria 10 GX 10AX115N2F45E1SG	i7-7820X
Cores	1518 DSP blocks	8 cores
Clock frequency	About 267 MHz	3.6 GHz
On-chip memory	6.625 MB block RAMs	L1 cache: 256 KB L2 cache: 8 MB L3 cache: 11 MB
External memory	8 GB	128 GB
OS	CentOS 7.0	CentOS 7.0
Programming language	OpenCL	C
Compiler	Intel FPGA SDK for OpenCL 17.1	gcc 4.8.5
Fabrication	20 nm	14 nm

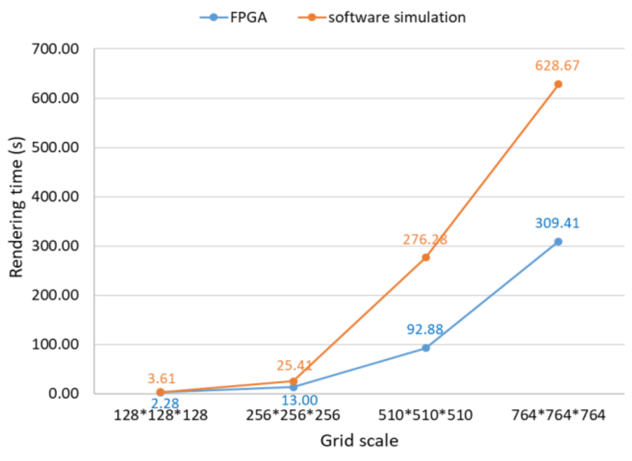

Figure 5: *Rendering time*

4.2. Computation throughput

The computation throughput denotes the updated speed of grids at each time step, and is calculated by using the following formula.

$$D_{throughput} = \frac{N_{grid}}{t_{per\_timestep}} \tag{14}$$

where N_{grid} is the number of grids in a sound space, and $t_{per\_timestep}$ is the rendering time at each time step. Figure 6 presents the computation throughput in the case of different grid scales in the FPGA-based accelerator and the software simulations on the PC. Figure 6 indicates that the proposed accelerator almost doubles the computation throughput of the software simulations, especially in the case of large sound spaces.

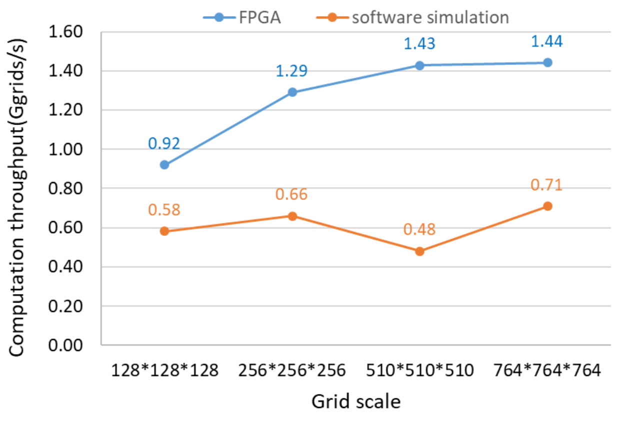

Figure 6: *Computation throughput*

5. CONCLUSIONS

Sound field rendering is computation-intensive and memory-intensive. FPGAs provide an alternative solution to sound field rendering, especially for real-time applications because the I/O interfaces are easily tailored according to applications. In this study, a FPGA-based accelerator for sound field rendering is developed using high-level synthesis in FPGA, in which the sliding window-based data buffering scheme is applied to reduce the demand of memory bandwidth. Although the FPGA-based accelerator runs at 1/13 (0.267/3.6) of clock frequency of the PC in software simulations, and the memory size of the FPGA board is about 1/16 (8/128) of that on the PC, the FPGA-based accelerator doubles the performance of the software simulations carried out on the PC. However, Figure 5 indicates that the rendering time at a time step is still long in the accelerator, which results in low sampling rate at the output rendered results. Hence, the current design is not suitable for real-time applications. From Table 2, we can find that the hardware resource utilization is low, and more computing units may be involved in calculation. Then, more grids may be computed concurrently to shorten the computation time at a time step. The related system is under development.

6. ACKNOWLEDGMENTS

Thanks to the Intel for the donation of the FPGA board DE5a-NET and the related software tools through University Program. This work was supported by the I-O DATA Foundation and JSPS KAKENHI Grant Number JP19K12092.

7. REFERENCES

[1] D. Botteldooren, “Acoustical Finite-difference Time-domain Simulation in a Quasi-Cartesian Grid,” *J. Acoust. Soc. Am.*, vol. 95, pp. 2313-2319, 1994.

[2] D. Botteldooren, “Finite-difference Time-domain Simulation of Low-frequency Room Acoustic Problems,” *J. Acoust. Soc. Am.*, vol. 98, pp. 3302-3308, 1995.

[3] O. Chiba, T. Kashiwa, H. Shimoda, S. Kagami, I. Fukai, “Analysis of Sound Fields in Three Dimensional Space by the Time-dependent Finite-difference Method based on the Leap Frog Algorithm,” *J. Acoust. Soc. Jpn.*, vol. 49, pp. 551-562, 1993.

[4] L. Savioja, and V. Valimaki, “Interpolated Rectangular 3-D Digital Waveguide Mesh Algorithms with Frequency Warping,” *IEEE Trans. Speech Audio Process.*, vol. 11, pp. 783-790, 2003.

[5] G. R. Campos, and D. M. Howard, “On the Computational Efficiency of Different Waveguide Mesh Topologies for Room Acoustic Simulation,” *IEEE Trans. Speech Audio Process.*, vol. 13, pp. 1063-1072, 2005.

[6] D. Murphy, A. Kelloniemi, J. Mullen, and S. Shelley, “Acoustic Modeling Using the Digital Waveguide Mesh,” *IEEE Signal Process. Mag.*, vol. 24, pp. 55-66, 2007.

[7] K. Kowalczyk and M. Walstijn, “Room Acoustics Simulation Using 3-D Compact Explicit FDTD Schemes,” *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, pp. 34-46, 2011.

[8] J. Mourik, and D. Murphy, “Explicit Higher-order FDTD Schemes for 3D Room Acoustic Simulation,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 22, pp. 2003-2011, 2014.

[9] B. Hamilton, and S. Bilbao, “Fourth-order and optimised finite difference schemes for the 2-D wave equation,” In *Proc. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, Sept. 2013, pp. 2-6.

[10] B. Hamilton, S. Bilbao, and C. J. Webb, “Revisiting implicit finite difference schemes for 3D room acoustics simulations on GPU,” In *Proc. Digital Audio Effects (DAFx-14)*, Erlangen, Germany, Sept. 2014, pp. 41-48.

[11] B. Hamilton, and S. Bilbao, “FDTD Methods for 3-D Room Acoustics Simulation with High-order Accuracy in Space and Time,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 25, pp. 2112-2124, 2017.

[12] V. Valimaki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, “Fifty Years of Artificial Reverberation,” *IEEE Trans. Audio Speech Lang. Process.*, vol. 20, no. 5, pp. 1421-1448, 2012.

[13] T. Ishii, T. Tsuchiya, and K. Okubo, “Three-dimensional Sound field Analysis Using Compact Explicit Finite Difference Time Domain Method with Graphics Processing Unit Cluster System,” *Jpn. J. Appl. Phys.*, vol. 52, pp. 07HC11, 2013.

[14] T. Tsuchiya, “Three-dimensional Sound Field Rendering with Digital Boundary Condition Using Graphics Processing Unit,” *Jpn. J. Appl. Phys.*, vol. 49, pp. 07HC10, 2010.

[15] C. Spa, A. Rey, and E. Hernandez, “A GPU Implementation of an Explicit Compact FDTD Algorithm with a Digital Impedance Filter for Room Acoustics Applications,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 8, pp. 1368–1380, 2015.

[16] M. Tanaka, T. Tsuchiya, and K. Okubo, “Two-dimensional Numerical Analysis of Nonlinear Sound Wave Propagation using Constrained Interpolation Profile Method Including

Nonlinear Effect in Advection Equation," *Jpn. J. Appl. Phys.*, vol. 50, pp. 07HE17, 2011.

[17] Y. Y. Tan, Y. Inoguchi, E. Sugawara, M. Otani, Y. Iwaya, Y. Sato, H. Matsuoka, and T. Tsuchiya, "A Real-time Sound Field Renderer Based on Digital Huygens' Model," *J. Sound Vib.*, vol. 330, pp. 4302–4312, 2011.

[18] Y. Y. Tan, Y. Inoguchi, Y. Sato, M. Otani, Y. Iwaya, H. Matsuoka, and T. Tsuchiya, "A Hardware-oriented Finite-difference Time-domain Algorithm for Sound Field Rendering," *Jpn. J. Appl. Phys.*, vol. 52, pp. 07HC03, 2013.

[19] Y. Y. Tan, Y. Inoguchi, Y. Sato, M. Otani, Y. Iwaya, H. Matsuoka, and T. Tsuchiya, "A Real-time Sound Rendering System Based on the Finite-difference Time-domain Algorithm," *Jpn. J. Appl. Phys.*, vol. 53, pp. 07KC14, 2014.

[20] Y. Y. Tan, Y. Inoguchi, Y. Sato, M. Otani, Y. Iwaya, H. Matsuoka, and T. Tsuchiya, "Analysis of sound field distribution for room acoustics: from the point of view of hardware implementation," In *Proc. Digital Audio Effects (DAFx-12)*, York, UK, Sept. 2012, pp. 93-96.

[21] Y. Y. Tan, Y. Inoguchi, Y. Sato, M. Otani, Y. Iwaya, H. Matsuoka, and T. Tsuchiya, "A FPGA implementation of the two-dimensional digital Huygens' model," In Proc. Field Program. Technol. (FPT 2010), Beijing, China, Dec. 2010, pp. 304-307.

[22] Y. Inoguchi, Y. Y. Tan, Y. Sato, M. Otani, Y. Iwaya, H. Matsuoka, and T. Tsuchiya, "DHM and FDTD based hardware sound field simulation acceleration," In *Proc. Digital Audio Effects (DAFx-11)*, Paris, France, Sept. 2011, pp. 69-72.

[23] Y.Y. Tan, Y. Inoguchi, M. Otani, Y. Iwaya, and T. Tsuchiya, "A Real-Time Sound Field Rendering Processor", *Applied Sciences*, vol. 8, no. 35, 2018.

[24] H. Kuttruff, *Room Acoustics*, Taylor & Francis: New York, NY, USA, 2009.

[25] K. Kowalczyk, and M. Walstijn, "Formulation of Locally Reacting Surfaces in FDTD/K-DWM Modelling of Acoustic Spaces," *Acta Acust. United Acust.*, vol. 94, pp. 891-906, 2008.

[26] https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=231&No=970, Accessed June 18, 2019.