DESIGN OF FPGA-BASED HIGH-ORDER FDTD METHOD FOR ROOM ACOUSTICS

Yiyu Tan
Department of Systems Innovation Engineering
Iwate University
Morioka, Japan
tanyiyu@iwate-u.ac.jp

Guanghui Liu
Inflammatory Bowel and Immunobiology Research Institute
Cedars-Sinai Medical Center
Los Angeles, US
guanghui.liu@cshs.org

Xin Lu
Department of Systems Innovation Engineering
Iwate University
Morioka, Japan
luxin@iwate-u.ac.jp

Peng Chen, Yusuke Tanimu
Digital Architecture Research Center
National Institute of Advanced Industrial Science and Technology
Tokyo, Japan
{chin.hou,yusuke.tanimura}@aist.go.jp

ABSTRACT

Sound field rendering with finite difference time domain (FDTD) method is computation-intensive and memory-intensive. This research investigates an FPGA-based acceleration system for sound field rendering with the high-order FDTD method, in which spatial and temporal blockings are applied to alleviate external memory bandwidth bottleneck and reuse data, respectively. After implemented by using the FPGA card DE10-Pro, the FPGA-based sound field rendering systems outperform the software simulations conducted on a desktop machine with 512 GB DRAMs and a Xeon Gold 6212U processor (24 cores) running at 2.4 GHz by 11 times, 13 times, and 18 times in computing performance in the case of the 2nd-order, 4th-order, and 6th-order FDTD schemes, respectively, even though the FPGA-based sound field rendering systems run at much lower clock frequency and have much smaller on-chip and external memory.

1. INTRODUCTION

Room acoustic simulation exhibit numerical methods to model sound propagation phenomena in spatial and time domain, and are applied widely in many engineering and scientific applications, such as sound source localization [1-3], virtual reality [4-5], artificial reverberation [6], boundary impedance estimation [7], and so on. Many analysis algorithms have been proposed for sound field rendering in room acoustics, in particular, FDTD method, which has already become one of essential methods in room acoustics since it was introduced to analyse acoustical behaviour by O. Chiba et al., D. Botteldooren et al., and L. Savioja et al. [8-11]. FDTD method solves wave equation with a finite number of stencil points in a discretized sound space using numerical method, and provides much higher accuracy over other methods like geometric methods. The inherent problem of FDTD method is dispersion error, and oversampling in spatial grids is usually required to suppress the numerical dispersion. As a result, computation and memory demand are increased significantly. Although many works were done at algorithmic level to solve this problem, such as digital waveguide mesh topologies [12-15], explicit second-order accurate schemes [16], high-order explicit "large-star" schemes [17], and two-step explicit FDTD schemes with high-order accuracy [18-20], these approaches still suffer from high computational cost. In general, to solve wave equations using FDTD method, computing capability is increased as the fourth power of frequency and is proportional with the volume of a sound space [6], and the size of the required memory is third power of frequency. Given the auditory range of humans (20 Hz-20 kHz), analyzing sound wave propagation in a space corresponding to a concert hall or a cathedral (e.g. volume of 10000-15000 m^3) for the maximum simulation frequency of 20 kHz requires petaflops of computing capability and terabytes of memory. This requires computing systems to have huge computational capability and large memory bandwidth.

In recent years, graphic processing units (GPUs) and field programmable gate arrays (FPGAs) have been applied to speed up computation in sound field rendering because of their much higher parallel computational capability over traditional general-purpose processors [21-36]. In particular, latest FPGAs contain thousands of hardened floating-point arithmetic units, several Megabytes of on-chip block memories to cache data, and millions of reconfigurable logic blocks. These on-chip hardware resources may be applied to directly implement sound wave equations to accelerate computation in contrast with software simulations in GPUs and general-purpose processors. Furthermore, system data paths can be customized in accordance with the data flow of a sound field rendering system to improve computing performance. On the other hand, the high-order FDTD method provides more accurate approximation on the second-order partial derivative and reduces dispersion. In this research, an FPGA-based accelerator is developed to speed up computation in sound field rendering with the high-order FDTD method. The main contributions of this work are summarized as follows.

(1) A high-order FDTD method. The related formula is derived, including approximation of the second partial derivative using Lagrange polynomial interpolation, the updated equation of 4th-order and 6th-order FDTD schemes.
(2) Design and implementation of an FPGA-based sound field rendering system with the high-order FDTD method. Spatial and temporal blockings are adopted to reduce memory bandwidth requirement and reuse data.
(3) Performance evaluation and analysis based on the prototype machine. The proposed rendering system is designed using OpenCL and implemented using the FPGA card DE10-Pro.

* This work was supported by the JSPS KAKENHI Grant Number JP22K12123

Its performance is evaluated through analyzing sound propagation in a three-dimensional shoebox with dimensions being 16m×8m×8m, incidence being an impulse, and sampling rate of sound being 44.1 kHz. Compared with the software simulations performed on a desktop machine with 512 GB DDR4 RAMs and an Intel's Xeon Gold 6212U processor running at 2.4 GHz, the proposed rendering systems speed up computation by 11 times, 13 times, and 18 times in the 2nd-order, 4th-order, and 6th-order FDTD schemes, respectively.

The rest of this paper is organized as follows. The high-order FDTD schemes are introduced in Section 2. In Section 3, system design is described, including spatial blocking, temporal blocking, and system architecture. System performance of the FPGA-based prototype machine is presented in Section 4, followed by the conclusions drawn in Section 5.

2. HIGH-ORDER FDTD SCHEME

A high-order approximation in FDTD method gives more accurate approximation, reduces dispersion, and increases valid bandwidth [37]. In general, Lagrange interpolation [38] and Taylor series expansion [39] are applied for such approximation. In this research, the Lagrange polynomial method is used to approximate the second-order partial derivative in spatial domain.

2.1. Approximation of second-order partial derivative

In a 4th-order scheme, the Lagrange polynomial is assumed as equation (1) and pass through five adjacent points $(0, f_0),(\Delta, f_1),(2\Delta, f_2),(3\Delta, f_3),(4\Delta, f_4)$ along *x* axis. The Δ is the unit of *x* axis.

$$f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \tag{1}$$

Then we have

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ \Delta^4 & \Delta^3 & \Delta^2 & \Delta & 1 \\ 16\Delta^4 & 8\Delta^3 & 4\Delta^2 & 2\Delta & 1 \\ 81\Delta^4 & 27\Delta^3 & 9\Delta^2 & 3\Delta & 1 \\ 256\Delta^4 & 64\Delta^3 & 16\Delta^2 & 4\Delta & 1 \end{bmatrix} \begin{bmatrix} a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{2}$$

Equation (2) can be solved through matrix inversion, and the parameters a_0, a_1, a_2, a_3, a_4 are obtained as shown in Equation (3) [17][40].

$$\begin{cases} a_0 = f_0 \\ a_1 = \dfrac{-25f_0 + 48f_1 - 36f_2 + 16f_3 - 3f_4}{12\Delta} \\ a_2 = \dfrac{35f_0 - 104f_1 + 114f_2 - 56f_3 + 11f_4}{24\Delta^2} \\ a_3 = \dfrac{-5f_0 + 18f_1 - 24f_2 + 14f_3 - 3f_4}{12\Delta^3} \\ a_4 = \dfrac{f_0 - 4f_1 + 6f_2 - 4f_3 + f_4}{24\Delta^4} \end{cases} \tag{3}$$

Then the second derivative of $f(x)$ equals to Equation (4). In order to get a centered difference approximation, the middle point $(2\Delta, f_2)$ of the five adjacent points are chosen to approximate the second derivative, which is shown in equation (5).

$$f''(x) = \frac{f_0 - 4f_1 + 6f_2 - 4f_3 + f_4}{2\Delta^4}x^2 + \frac{-5f_0 + 18f_1 - 24f_2 + 14f_3 - 3f_4}{2\Delta^3}x + \frac{35f_0 - 104f_1 + 114f_2 - 56f_3 + 11f_4}{12\Delta^2} \tag{4}$$

$$f''(2\Delta) = \frac{-f_0 + 16f_1 - 30f_2 + 16f_3 - f_4}{12\Delta^2} \tag{5}$$

Thus, the approximated parameters for the second derivative is $(-\frac{1}{12}, \frac{4}{3}, -\frac{5}{2}, \frac{4}{3}, -\frac{1}{12})$, and $(f_0, f_1, f_2, f_3, f_4)$ corresponds to the values of the points $(i-2, j, k), (i-1, j, k), (i, j, k), (i+1, j, k), (i+2, j, k)$ along *x* axis in a three dimensional Cartesian space, respectively. The similar derivation can be conducted for the 6th-order approximation, in which $f(x)$ is assumed to be $f(x) = a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ and seven adjacent points are required to solve the equation.

2.2. High-order FDTD scheme

Sound wave propagation in a cubic space is governed by the equation.

$$\frac{\partial^2 P}{\partial t^2} = c^2\left(\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} + \frac{\partial^2 P}{\partial z^2}\right) \tag{6}$$

where *P* denotes sound pressure, *c* is the speed in air, *t* is time, *x*, *y* and *z* are Cartesian coordinates in a three-dimensional space. To solve Equation (6), high-order approximation, such as equation (5) for the 4th-order approximation, is applied to approximate the second-order partial derivative instead of the second-order center difference method. In general, the high-order approximation in time domain increases memory requirement because more data at previous time steps are involved in computation while the high-order approximation in spatial domain introduces additional computations due to more neighbor grids are needed to update value of a grid. In order not to increase memory requirement but just increase computations of updating sound pressure of a grid, the second-order approximation in time domain and high-order approximation in spatial domain are applied on Equation (6), which are shown as follows.

$$\begin{aligned} \frac{\partial^2 P}{\partial t^2} &= \frac{P^{n-1}_{i,j,k} - 2P^n_{i,j,k} + P^{n+1}_{i,j,k}}{\Delta t^2} \\ \frac{\partial^2 P}{\partial x^2} &= \frac{-\frac{1}{12}(P^n_{i-2,j,k} + P^n_{i+2,j,k}) + \frac{4}{3}(P^n_{i-1,j,k} + P^n_{i+1,j,k}) - \frac{5}{2}P^n_{i,j,k}}{\Delta x^2} \\ \frac{\partial^2 P}{\partial y^2} &= \frac{-\frac{1}{12}(P^n_{i,j-2,k} + P^n_{i,j+2,k}) + \frac{4}{3}(P^n_{i,j-1,k} + P^n_{i,j+1,k}) - \frac{5}{2}P^n_{i,j,k}}{\Delta y^2} \\ \frac{\partial^2 P}{\partial z^2} &= \frac{-\frac{1}{12}(P^n_{i,j,k-2} + P^n_{i,j,k+2}) + \frac{4}{3}(P^n_{i,j,k-1} + P^n_{i,j,k+1}) - \frac{5}{2}P^n_{i,j,k}}{\Delta z^2} \end{aligned} \tag{7}$$

Letting $\Delta x = \Delta y = \Delta z = \Delta l$ and inserting Equation (7) into Equation (6), the updated equation for the 4th-order scheme is obtained and shown in Equation (8) [40], in which $\chi = c\Delta t/\Delta l$ is

the Courant number. A similar derivation can be conducted on the 6th-order scheme and Equation (9) is yielded to update sound pressure of a grid.

$$\begin{aligned} P^{n+1}_{i,j,k} = \chi^2[-\frac{1}{12}(&P^n_{i-2,j,k} + P^n_{i+2,j,k} + P^n_{i,j-2,k} + P^n_{i,j+2,k} \\ &+ P^n_{i,j,k-2} + P^n_{i,j,k+2}\ ) + \frac{4}{3}\ (P^n_{i-1,j,k} + P^n_{i+1,j,k} + P^n_{i,j-1,k} \\ &+ P^n_{i,j+1,k} + P^n_{i,j,k-1} + P^n_{i,j,k+1})] + (2 - \frac{15}{2}\chi^2)P^n_{i,j,k} - P^{n-1}_{i,j,k} \end{aligned} \tag{8}$$

$$\begin{aligned} P^{n+1}_{i,j,k} = \chi^2[\frac{1}{90}(&P^n_{i-3,j,k} + P^n_{i+3,j,k} + P^n_{i,j-3,k} + P^n_{i,j+3,k} \\ &+ P^n_{i,j,k-3} + P^n_{i,j,k+3}\ ) - \frac{3}{20}(P^n_{i-2,j,k} + P^n_{i+2,j,k} + P^n_{i,j-2,k} \\ &+ P^n_{i,j+2,k} + P^n_{i,j,k-2} + P^n_{i,j,k+2}\ ) + \frac{3}{2}(P^n_{i-1,j,k} + P^n_{i+1,j,k} \\ &+ P^n_{i,j-1,k} + P^n_{i,j+1,k} + P^n_{i,j,k-1} + P^n_{i,j,k+1})]\ -\ P^{n-1}_{i,j,k} \\ &+ (2 - \frac{49}{6}\chi^2)P^n_{i,j,k} \end{aligned} \tag{9}$$

Equations (8) and (9) show that sound pressures of the neighbor grids along axes at previous time steps are needed to update sound pressure of a grid. The neighboring grids are in six axial directions. Two and three neighbor grids are in each direction in the 4th-order and 6th-order schemes, respectively. Computing sound pressure of a grid requires 11 additions, 2 subtractions, 3 multiplications, and 14 memory accesses in the 4th-order FDTD scheme while it needs 17 additions, 2 subtractions, 4 multiplications, and 20 memory accesses in the 6th-order FDTD scheme. In addition, since the high-order and 2nd-order approximation are applied on the space domain and time domain, respectively, the proposed FDTD scheme has high-order accuracy in space domain while it remains 2nd accuracy in time domain. For example, the 4th-order FDTD scheme provides 4th-order accuracy in space and 2nd-order accuracy in time.

Stability condition and dispersion are important in the FDTD method. J. Mourik discussed the stability condition and dispersion of high-order FDTD method and claimed that the 4th-order scheme was the best in terms of valid bandwidth up to 16th-order scheme [17][41]. The valid bandwidth of the 4th-order scheme was about 1.5 times and 1.1 times of those of the 2nd-order and 6th-order schemes, respectively, and the valid bandwidth dropped a little bit along with every increase of the order after the 4th-order. The stability condition for the 4th-order and 6th-order FDTD schemes are $\chi \le 0.5$ and $\chi \le \sqrt{15/68}$, respectively. In addition, high-order FDTD boundary conditions were also investigated by J. Mourik [41]. To simplify system design and evaluation, boundary conditions are not discussed in this paper.

3. SYSTEM DESIGN

From Equations (8) and (9), sound pressures of grids at previous two continuous time steps (time steps n and n-1) are required to compute sound pressures of grids at time step n+1, and huge amounts of data are read from and written back to memory as the grid dimensions are increased. Therefore, it is impossible to store all data in the on-chip block RAMs of FPGA, which are about several Megabytes in size, to reduce data access overhead in the case of large sound spaces even though the size of on-chip block memories inside current FPGAs has been increased significantly. Instead, external on-board DDR4 DRAMs on the FPGA card, which are several Gigabytes in size are needed to store data during computing. Another challenge is how to reuse data and reduce memory bandwidth requirement. In this research, spatial blocking is introduced to reduce the required memory bandwidth between the computing engine and on-board memory, and temporal blocking is employed to reuse data and reduce data accesses to external memory.

3.1. Spatial Blocking

Spatial blocking is applied to reduce the required on-chip memory, and it is employed in many deep-pipeline implementations of stencil computation on FPGA [42-43]. As shown in Fig. 1(a), a large sound space with $Nx \times Ny \times Nz$ grids is decomposed into small spatial blocks and each spatial block has $Cx \times Cy \times Nz$ grids. A small spatial block is further partitioned into *x-y* planes along the *z* dimension (Fig. 1(b)). Computations are performed plane by plane in a spatial block while they are carried out along the *x* dimension in a plane. Equations (8) and (9) indicate that data values of three adjacent planes are required to calculate new results. In the current design, shift registers are introduced as on-chip buffers to stream in data. As illustrated in Fig. 1, *n* values of the plane *i+1*, all values of the planes *i-1* and *i* are firstly streamed into a shift register from external memory to compute sound pressures of grids on the plane *i*, the computing unit then fetches data from the shift register and computes sound pressures of *n* grids on the plane *i* concurrently. Then, the shift register is shifted right by *n* data, and another *n* new data are written into the head of the shift register while *n* old data are evicted from the tail at each clock cycle. When computations in a plane are completed, data in a new plane are streamed in the shift register and computation is moved to the next plane. This procedure is repeated until sound pressures of all grids in a spatial block are computed, and then computation is switched to the next spatial block. The shift-register-based buffer can be efficiently implemented by the on-chip block RAMs inside an FPGA.

Using shift register minimizes the size of on-chip buffer by only storing sound pressures of the needed grids in a spatial block. Furthermore, current FPGAs provide abundance of block RAMs, therefore, much larger on-chip buffers can be implemented to store sound pressures of grids in a large spatial block to speed up data access. In addition, to parallelize computation spatially and improve utilization efficiency of the external on-board memory bandwidth, data are coalesced, and computations are vectorized to calculate *n* grids concurrently through loop unrolling in each spatial block. If the dimension of a spatial block is $Cx \times Cy$ and *n* grids are computed in parallel, the depth of the shift register is calculated through Equation (10).

$$depth = 2 \times rad \times Cx \times Cy + n \tag{10}$$

where *rad* is the stencil radius and it is 1, 2, and 3 for the 2nd-order, 4th-order, and 6th-order FDTD schemes, respectively. In contrast, the depth of the shift register is $2 \times rad \times N_x \times N_y + n$ if the spatial blocking is not applied. During implementation on an FPGA, parts of the shift register will be replicated to support parallel accesses because of the limited number of ports in each block RAM unit. Such replication will require more block RAMs inside an FPGA.

Computing sound pressures of grids on boundary planes (front, real, right, and left) of a spatial block needs data from its neighbor spatial blocks. But these data are not read into the shift

register during the computations of current spatial block. To avoid data exchange between adjacent spatial blocks, overlapped blocking is applied and such grids on boundary planes of a spatial block are treated as internal grids of the related neighbor spatial blocks and computed later. The size of the overlapped parts of neighbor spatial blocks are linearized to the stencil radius *rad* and the dimension of a spatial block (*Cx* × *Cy*).

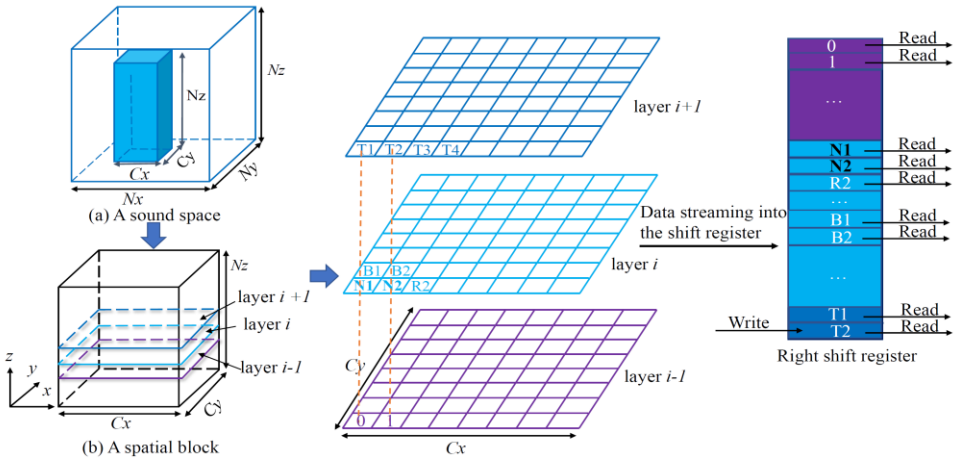

Figure 1: *Spatial blocking*

3.2. Temporal Blocking

Temporal blocking allows system to continuously compute sound pressures of grids of a spatial block at different time steps. Hence, data access to external memory is reduced. To implement temporal blocking, a computing kernel consisting of several replicated processing elements (PEs) is designed, and each PE computes sound pressures of grids in the same spatial block at different time steps. As shown in Fig. 2, several PEs are cascaded to compute sound pressures of grids in a same spatial block at continuous time steps. For example, PE_0 calculates sound pressures of grids at time step *n*. The computed results are sent to PE_1 and then PE_1 computes sound pressures of grids in the same spatial block at time step *n+1*. Such computation procedure is repeated until the final PE computes sound pressures of grids at time step *n+k-1*. Thus, access to external memory is reduced, and computation is sped up because sound pressures of a spatial block at several time steps are computed concurrently. Since computation of a given PE starts only after the outputs of the previous PE are available, computation in a PE is always behind its previous PE.

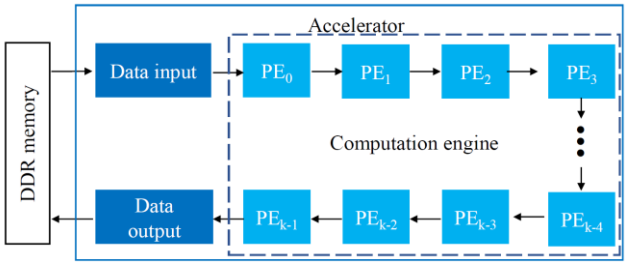

Figure 2: *System diagram*

3.3. System Design

The system diagram of the FPGA-based sound field rendering system is presented in Fig. 2, which consists of the Data input module, Computation engine, and Data output module. The Data input module streams data of a spatial block from the external DDR DRAMs on the FPGA card plane by plane, and feeds data to the computation engine. The computation engine consists of 16 PEs. Each PE computes sound pressures of grids in a spatial block at a time step, and all PEs are applied to compute sound pressures of grids in the same spatial block at continuous 16 time steps. The Data output module writes the computation results back to the external memory.

A PE computes sound pressure of a grid according to its position, incidence, and sound pressures of its neighbor grids at previous time steps. The computed results are sent to the neighbor PE except for the final PE, in which they are written back to the external memory through the Data output module. As shown in Fig. 3, a PE includes system controller, four buffers (shift_register_p1, shift_register_p2, shift_register_posi, and shift_register_incidence), and computing units. The functions of each module are described as follows.

- System controller. Each grid has an associated position flag, which will be applied to choose the updated equation in the computing unit. The system controller reads position flag and data values at previous time steps from the Data input module or a neighbor PE according to the computation flow, and writes them into the related shift registers like shift_register_p1, shift_register_p2, shift_register_posi, respectively. Then the computing unit computes sound pressures and sends the computation results to the neighbor PE except for the final PE, in which the computed results are written back to the external memory directly through the Data output module.
- Shift_register_p1, shift_register_p2, shift_register_posi, and shift_register_incidence. To compute sound pressures of grids at time step *n*, data values at time step *n-1* and *n-2* are streamed in the shift_register_p1 and shift_register_p2, respectively. In a PE, the input data data_p1 is directly passed to the next neighbor PE as the data values at time step *n-2* while the computed results are output to the next neighbor PE as the data values at time step *n-1*. The data values are exchanged through high bandwidth channels between neighbor PEs. The

position flags of grids are kept in the shift_register_posi. Since all PEs compute sound pressures of grids in a same spatial block, the position flag of a grid is same in all PEs, and a PE just passes the position_flag to its next neighbor PE. The incident data are stored in the shift_register_incidence.

- Computing unit. The computing unit fetches data from four buffers and compute sound pressures. It is designed based on the sound field rendering algorithm, namely Equations (8) and (9) for the 4th-order and 6th-order FDTD schemes.

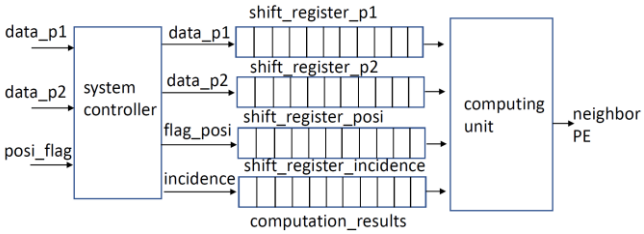

Figure 3: *PE structure*

4. PERFORMANCE EVALUATION

The proposed sound field rendering systems based on the 2nd-order, 4th-order, and 6th-order FDTD schemes were designed using the OpenCL programming language and implemented using the FPGA card DE10-Pro from Terasic Company [44]. The FPGA card contained a Stratix 10 SX FPGA (1SX280HU2F50E1VG) and 8 GB on-board external DDR4 DRAMs. To verify and estimate the performance of the developed sound field rendering systems, sound propagation in a three-dimensional shoebox with dimension being 16m×8m×8m was analyzed. The incidence was an impulse, and the number of the computed time steps was 32. As a comparison, relative counterpart systems were developed using the C++ programming language, and executed on a desktop machine with 512 GB DDR4 DRAMs and an Intel Xeon Gold 6212U processor (24 cores) running at 2.4 GHz. The OpenCL codes were compiled using the Intel FPGA SDK for OpenCL 19.1 while the reference C++ codes were compiled using the GNU compiler (version: 4.8.5) with the option -O3 and -fopenmp to use all 24 processor cores. During analysis, the sound speed was 340 m/s, sampling rate is 44.1 kHz, the Courant number χ was $\sqrt{3}/3$, $1/2$, $\sqrt{15/68}$ in the 2nd-order, 4th-order, 6th-order FDTD schemes, respectively, and all boundaries were clamped to 0, i.e. phase-reversing fully reflective boundaries. Data were single-precision floating point in both the FPGA-based rendering systems and software simulations. The development environment in the FPGA-based sound field rendering system and software simulation is shown in Table 1. As presented in Table 1, the memory size of external and on-chip memories in the FPGA-based system is much smaller than that of the desktop machine in the software simulation, and the FPGA system runs at much lower clock frequency over the desktop machine.

4.1. Hardware resource utilization

Table 2 presents the hardware resource utilization of the FPGA-based sound field rendering systems with the 2nd-order, 4th-order, and 6th-order FDTD schemes when the size of a spatial block is 128 × 128, the number of PEs is 16 in the computation engine, and the number of grids computed concurrently is 16. Equations (8) and (9) indicate that as the order of the FDTD scheme is increased, the number of operations are increased, more data are streamed in the shift registers, more DSP blocks, which are utilized to implement multipliers, are involved in computation, and more RAM blocks are required to implement the shift registers to store data during computing. From Equation (10), the utilized RAM blocks are significantly affected by the size of a spatial block. If the size of a spatial block is changed from 128 × 128 to 256 × 256 in the 2nd-order FDTD scheme, the number of utilized RAM blocks will be increased from 1785 to 5129. In addition, since the control of shifting and reading out data from the shift registers at a clock cycle is complicated in the sound field rendering system with the higher-order FDTD scheme, the system data path becomes more complex, and the clock frequency is decreased.

Table 1: *Development environment*

	FPGA	software simulation
computing unit	Stratix 10 SX	Intel Xeon Gold 6212U
# of cores	5760 DSP blocks	24 cores
frequency	about 350 MHz	2.4 GHz
on-chip memory	28.6 MB block RAMs	L1 cache: 1.5 MB
		L2 cache: 24 MB
		L3 cache: 35.75 MB
external memory	8 GB DDR4-2400	512 GB DDR4-2933
operating system	CentOS 7.2	CentOS 7.2
programming language	OpenCL	C++
compiler	Intel FPGA SDK for OpenCL 19.1	GNU compiler (version: 4.8.5)
fabrication	14 nm	14 nm

Table 2: *Hardware resource utilization*

orders	logic utilization	DSP blocks	RAM blocks	clock frequency (MHz)
2nd	269,159 (29%)	342 (6%)	1,785 (15%)	357
4th	293,001 (31%)	630 (11%)	3,764 (32%)	355
6th	335,237 (36%)	918 (16%)	4,309 (37%)	337

From Table 2, the hardware resources are not utilized efficiently in the current design. The logic blocks, DSP blocks, and RAM blocks are used by 29%, 6%, and 15% of the relative valid resources inside the FPGA, respectively. Thus, the size of the spatial block and the number of grids computed in parallel can be further increased in the current design. On the other hand, as the size of the spatial block and the number of grids computed in parallel are increased, the data path in the hardware system may become complicated, and clock frequency may be decreased, which will result in the degradation of computing performance. Therefore,

the size of the spatial block and the number of grids computed concurrently cannot be increased unlimitedly.

4.2. Computation time

When the size of the spatial block is 128 × 128, the number of PEs is 16, and the number of grids computed concurrently is 16, Table 3 presents the average rendering time at each time step in the FPGA-based sound field rendering systems and software simulations in the case of the FDTD schemes with different orders. Although the desktop machine in the software simulations runs at much higher clock frequency and has much larger external and on-chip memories than the FPGA-based sound field rendering systems, the FPGA-based sound field rendering systems speed up computation by 11 times, 13 times, and 18 times in the 2nd-order, 4th-order, and 6th-order FDTD schemes, respectively, over software simulations performed on the desktop machine. In the FPGA-based rendering system, sound pressures of grids at time steps *n* and *n-1* are stored into two independent DDR4 DRAMs, and they are fetched through two independent channels and streamed into the on-chip shift registers inside FPGA. The overhead to access data from the shift registers is usually one clock cycle. In contrast, all sound pressures are stored in external memory in the software simulations, and external memory is accessed frequently to fetch or write back data during computation. The data access is constraint by the memory bandwidth and the access overhead is very large. Although on-chip caches inside the processor may reduce the overhead of accessing data, their benefits to the computing performance improvement are limited as the grid dimension is increased. Moreover, data are reused through temporal blocking in the FPGA-based system, and sound pressures of a spatial block at 16 continuous time steps are computed in parallel. This further reduces data access to the external memory. All these lead to the performance improvement of computation in the FPGA-based sound field rendering system.

Table 3: *Rendering time per time step (s)*

orders	FPGA	software simulation
2nd	0.0486	0.5363
4th	0.0333	0.4458
6th	0.0238	0.4437

In the current performance evaluation, the Courant number is $\sqrt{3}/3$, $1/2$, $\sqrt{15/68}$ in the 2nd-order, 4th-order, 6th-order FDTD schemes, respectively. As the order of the FDTD scheme is increased, although the clock frequency of the FPGA-based sound field rendering system decreased a little bit, the computing time at each time step is decreased significantly because the grid dimension becomes smaller and the number of grids is reduced. But it is worth noting that different Courant numbers will impact upon the valid bandwidth of the outputs in each FDTD scheme.

4.3. Computational Throughput

The computational throughput stands for the number of grids updated per second at each time step and is calculated by using the following formula.

$$SP_{updated} = \frac{N_{grid}}{t_{time\_step}} \tag{11}$$

where N_{grid} is the number of grids, and t_{time_step} is the average computing time at a time step. Table 4 shows the computational throughput in the FPGA-based sound field rendering systems and software simulations in the case of the FDTD schemes with different orders. As shown in Table 4, the FPGA-based system updates grids at much higher speed over the software simulations because it achieves much better computing performance at each time step. On the other hand, as the order of the FDTD scheme is increased from the 2nd to 6th, the computational throughput is improved about 9.5%.

Table 4: *Computational throughput (Ggrids/s)*

orders	FPGA	software simulation
2nd	8.8457	0.8015
4th	8.3604	0.6235
6th	9.6882	0.5207

4.4. Discussion

In the current evaluation, sound propagation in a simple three-dimensional shoebox was analyzed using the developed FPGA-based sound field rendering system with the high-order FDTD method. For a sound space with complex geometries, decomposition methods to discretize a sound space into a grid mesh are required. In the hardware system, the data flow to stream data from the external on-board memory will be changed, and the system data path may become complicated. Furthermore, all boundaries were clamped to 0 in current evaluations. If complex boundary conditions are adopted, the updated equations for the grids on the boundaries are needed to be derived from the high-order FDTD method, and the computing unit inside a PE will be changed in the FPGA-based sound field rendering systems because the updated equation is different in accordance with the position of a grid.

On the other hand, the computing pattern in sound field rendering with FDTD methods is stencil computation in principle, in which the bottleneck of computing performance is memory bandwidth. In the current design, the spatial blocking is applied to alleviate the required memory bandwidth, and the temporal blocking is adopted to reuse data and reduce memory access to external memory. Although FPGA provides on-chip block memories with large bandwidth, the size of on-chip block memories is limited, such as several Mega bytes. And the FPGA card DE10-Pro provides large size on-board external memory, which is 8GB DDR4-2400 DRAMs. In the FPGA-based acceleration system, computation is sped up through customization of data path according to the data flow during computing and parallelism of PEs. In contrast, current GPUs provide several Giga bytes high speed and high memory bandwidth (HBM) memories, and data access overhead will be reduced significantly. Moreover, development of an FPGA-based system needs much hardware knowledge even though high level synthesis is widely applied in recent years. Development of a GPU-based system is relatively easier than that of FPGA-based system. All these results in that GPUs are more popular in computing than FPGAs. At next step, a sound field rendering system will be developed using GPU and compared with the

proposed FPGA-based sound field rendering system to validate which platform is better for sound field rendering.

5. CONCLUSIONS

High-order FDTD method provides more accurate approximation and smaller dispersion. The sound field rendering with FDTD method is computationally intensive and memory intensive. In this research, an FPGA-based sound field rendering system based on the high-order FDTD method is developed to speed up computation. The spatial blocking is applied to reduce the size of the required on-chip buffer and memory bandwidth, and the temporal blocking is adopted to reuse data and compute sound pressures of grids in the same spatial block at 16 continuous time steps in parallel. In the sound field rendering system with the 2nd-order, 4th-order, and 6th-order FDTD schemes, the FPGA-based system achieves much higher performance in computing and computational throughput over the software simulations carried out in a desktop machine even though the FPGA-based rendering systems run at much lower clock frequency and has smaller on-chip and external on-board memories. The evaluation results demonstrate that FPGAs are promising for sound field rendering. In future work, the decomposition methods to discretize a sound space with complex geometries into a grid mesh and the high-order FDTD schemes with complicated boundary conditions will be studied, and a real-time sound field rendering system based on the proposed architecture and high-order FDTD methods with complicated boundary conditions will be investigated, in which input incidence, computation, and computed results are all handled at real time. As a comparison, a counterpart system based on GPUs will be developed to compared with the FPGA-based sound field rendering system and explore the suitable platform for sound field rendering.

6. ACKNOWLEDGMENTS

Thanks for Intel's donation of the FPGA card DE10-Pro and EDA tools through its University Program. This work was supported by the JSPS KAKENHI Grant Number JP22K12123.

7. REFERENCES

[1] S. Kitic, N. Bertin, and R. Gribonval, "Hearing behind walls: localizing sources in the room next door with cosparsity," in Proc. *IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 3087 - 3091.

[2] N. Antonello, T. Waterschoot, M. Moonen, and P. Naylor, "Source localization and signal reconstruction in a reverberant field using the FDTD method," in *Proc. Eur. Signal Process. Conf.*, 2014, pp. 301 - 305.

[3] N. Bertin, S. Kiti, and R. Gribonval, "Joint estimation of sound source location and boundary impedance with physics-driven cosparse regularization", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 6340 - 6344.

[4] R. Mehra and D. Manocha, "Wave-based sound propagation for VR applications", in *Proc. IEEE Virtual Real.*, Minnesota, USA, 2014.

[5] N. Raghuvanshi, R. Narain, and M. C. Lin, "Efficient and Accurate Sound Propagation Using Adaptive Rectangular Decomposition," *IEEE Trans. Visualization and Computer Graphics*, vo. 15, no. 5, pp. 789 - 801, 2009.

[6] 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.

[7] N. Antonello, T. Waterschoot, M. Moonen, and P. Naylor, "Identification of surface acoustic impedances in a reverberant room using the FDTD method," in Proc. *IEEE Int. Workshop Acoust. Signal Enhancement*, pp. 114 - 118, 2014.

[8] Botteldooren, "Acoustical Finite-Difference Time-Domain Simulation in a quasi-cartesian node," *J. Acoust. Soc. Am.*, vol. 95, pp. 2313 - 2319, 1994.

[9] D. Botteldooren, "Finite-Difference Time-Domain Simulation of Low-Frequency Room Acoustic Problems," *J. Acoust. Soc. Am.*, vol. 98, pp. 3302 - 3308, 1995.

[10] O. Chiba, T. Kashiwa, H. Shimoda, S. Kagami, and 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.

[11] L. Savioja, T. Rinne, and T. Takala, "Simulation of room acoustics with a 3-D finite difference mesh," in *Proc. Int. Computer Music Conf. (ICMC)*, Aarhus, Denmark, Sept. 1994, pp. 463-466.

[12] J. O. Smith III, "Physical Modeling Using Digital Waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74 - 91, 1992.

[13] 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.

[14] G. R. Campos and D. M. Howard, "On the Computational Efficiency of Different Waveguide Mesh Topologies for Room Acoustic Simulation," *IEEE Trans. Audio Speech Lang. Process.*, vol. 13, no. 5, pp. 1063 - 1072, 2005.

[15] D. Murphy, A. Kelloniemi, J. Mullen, and S. Shelley, "Acoustic Modeling Using the Digital Waveguide Mesh," *IEEE Signal Process Magazine*, vol. 24, no. 2, pp. 55 - 66, 2007.

[16] K. Kowalczyk and M. Walstijn, "Room Acoustics Simulation Using 3-D Compact Explicit FDTD Schemes," *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, no. 1, pp. 34 - 46, 2011.

[17] J. Mourik and D. Murphy, "Explicit Higher-Order FDTD Schemes for 3D Room Acoustic Simulation," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 22, no. 12, pp. 2003 - 2011, 2014.

[18] 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.

[19] 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.

[20] 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.

[21] T. Ishii, T. Tsuchiya, 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.

[22] T. Tsuchiya and N. Maruta, "Three-Dimensional Compact Explicit Finite Difference Time Domain Scheme with Density Variation," *Jpn J. Appl. Phys.* vol. 57, pp. 07LC01, 2018.

[23] 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.

[24] 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.

[25] L. Savioja, "Real-time 3D finite-difference time-domain simulation of low-and mid-frequency room acoustics," in *Proc. Digital Audio Effects (DAFx'10)*, Graz, Austria, Sept. 2010.

[26] A. Southern, D. Murphy, G. Campos, and P. Dias, "Finite difference room acoustic modelling on a general purpose graphics processing unit," in *Proc. 128th Audio Engineering Society Convention*, London, UK, May 2010, pp. 1393 - 1403.

[27] L. Savioja, D. Manocha, and M. Lin, "Use of GPUs in room acoustic modeling and auralization," in *Proc. Int. Symposium on Room Acoustics (ISRA)*, Melbourne, Australia, August 2010.

[28] C. Webb and S. Bilbao, "Virtual room acoustics: a comparison of techniques for computing 3D-FDTD schemes using CUDA," in *Proc. 130th Audio Engineering Society Convention*, London, UK, May 2011, pp. 1163–1169.

[29] C. Webb and S. Bilbao, "Computing room acoustics with CUDA-3D FDTD schemes with boundary losses and viscosity," in *Proc. Int. Conf. on Acoustics Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 317–320.

[30] 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.

[31] 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.

[32] 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.

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

[34] Y. Y. Tan and T. Imamura, "A FPGA-based accelerator for sound field rendering", Proc. Digital Audio Effect, 2019.

[35] J. Saarelma, J. Califa, and R. Mehra, "Challenges of distributed real-time finite-difference time-domain room acoustic simulation for auralization," in *Proc. AES Int. Conf. Spatial Reproduction -Aesthetics and Science*, Tokyo, Japan, July 2018, PP-1.

[36] Y. Y. Tan and T. Imamura, "An FPGA-based sound field rendering system", in *Proc. IEEE Cluster*, online, 2020.

[37] M. Ciment and S. Leventhal, "Higher Order Compact Implicit Schemes for the Wave Equation," *Math. Comput.*, vol. 29, no. 132, pp. 985-994, 1975.

[38] P. Deuflhard and A. Hohmann, *Numerical Analysis in Modern Scientific Computing*, Springer-Verlag, New York, 2nd edition, 2003.

[39] S. Sakamoto, "Phase-Error Analysis of High-Order Finite Difference Time Domain Scheme and its Influence on Calculation Results of Impulse Response in Closed Sound Field," *Acoust. Sci. Technol.*, vol. 28, no. 5, pp. 295-309, 2007.

[40] Y. Y. Tan and Toshiyuki Imamura, "Design and implementation of high-order FDTD method for room acoustics," in *Proc. 41th USE*, Tokyo, Japan, 2020.

[41] J. Mourik, *Higher-order Finite Difference Time Domain Algorithms for Room Acoustic Modelling*, PhD Thesis, University of York, 2016.

[42] H. Zohouri, A. Podobas, and S. Matsuoka, "Combined spatial and temporal blocking for high-performance stencil computation on FPGAs using OpenCL", in *Proc. FPGA*, 2018.

[43] Yiyu Tan, Toshiyuki Imamura, Masaaki Kondo, "FPGA-based Acceleration of FDTD Sound Field Rendering", *J. Audio Eng. Soc.*, vol. 69, no. 7/8, pp. 542–556, 2021.

[44] https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=13&No=1144&PartNo=1