Hindawi Publishing Corporation
Journal of Computational Engineering
Volume 2014, Article ID 634269, 8 pages
http://dx.doi.org/10.1155/2014/634269

Research Article

FDTD Acceleration for Cylindrical Resonator Design Based on the Hybrid of Single and Double Precision Floating-Point Computation

Hasitha Muthumala Waidyasooriya, Masanori Hariyama, Yasuhiro Takei, and Michitaka Kameyama

Graduate School of Information Sciences, Tohoku University, Aoba 6-6-05, Aramaki, Aoba, Sendai, Miyagi 980-8579, Japan

Correspondence should be addressed to Hasitha Muthumala Waidyasooriya; hasitha@ecei.tohoku.ac.jp

Received 25 June 2014; Accepted 18 November 2014; Published 4 December 2014

Academic Editor: Fu-Yun Zhao

Acceleration of FDTD (finite-difference time-domain) is very important for the fields such as computational electromagnetic simulation. We consider the FDTD simulation model of cylindrical resonator design that requires double precision floating-point and cannot be done using single precision. Conventional FDTD acceleration methods have a common problem of memory-bandwidth limitation due to the large amount of parallel data access. To overcome this problem, we propose a hybrid of single and double precision floating-point computation method that reduces the data-transfer amount. We analyze the characteristics of the FDTD simulation to find out when we can use single precision instead of double precision. According to the experimental results, we achieved over 15 times of speed-up compared to the CPU single-core implementation and over 1.52 times of speed-up compared to the conventional GPU-based implementation.

1. Introduction

Computational electromagnetic simulation shows a rapid development recently due to the introduction of processors that have parallel processing capability such as multicore CPUs and GPUs (graphic processing units). The FDTD (finite-difference time-domain) algorithm [1, 2] is one of the most popular methods of computational electromagnetic simulations due to its simplicity and very high computational efficiency. It has been widely used in many applications such as coil modeling [3] and resonance characteristics analysis of a cylindrical cavity [4, 5]. Many of these applications require double precision floating-point computation to satisfy the stability condition [6].

The FDTD simulation requires a large amount of data. When more processor cores are used in parallel, more data transfers occur between the memory and the processor cores. Therefore, the memory-bandwidth limitation is a major problem in the FDTD simulation using computers. To overcome this problem, we have to reduce the data transfer amount, so that we can use more cores in parallel. To do this, we propose a hybrid precision computation method that uses both single and double precision. Single precision data use 4 bytes compared to 8 bytes used in double precision data. Therefore, using single precision reduces the data amount and increases the processing speed. However, using single precision could bring inaccurate results. In some cases, the FDTD simulation does not converge when it is executed for a large number of iterations.

In this paper, we consider the FDTD simulation of a cylindrical resonator [5]. It is one of the most fundamental types of resonant cavities and has been used to construct, for example, wavelength filters for microwaves [7, 8]. It also has a number of important applications in the lightwave field, such as couplers and laser cavities [9–11]. In such applications, the quality factor (Q) of a cylindrical cavity, which is a basic characteristic, depends on the performance of the cavity walls. The fundamental characteristics, such as resonance wavelength, Q factor, and modal fields, are calculated by numerical simulation. The same simulation is executed many

times by changing the parameters such as the radius of the cavity and the thickness and the depth of the cavity wall. Therefore, the processing time of the simulation is very large.

This paper proposes a hybrid of single and double precision computation method to reduce the processing time of the FDTD simulation of a cylindrical resonator. The proposed method can be used in both multicore CPUs and GPU accelerators. We analyze the characteristics of the application to find out where we can use single precision instead of double precision. According to the experimental results, we achieved 1.41 and 5.19 times of speed-up, respectively, for CPU and GPU implementations, compared to the conventional double precision implementation using 12 threads and 6 CPU cores. Compared to the conventional GPU implementation, the proposed hybrid precision method using GPU has over 1.52 times of speed-up. Using the proposed method, we can extract more performance from the same hardware without any additional overhead.

2. The FDTD Simulation of Cylindrical Resonator

A cylindrical cavity surrounded by a curved resonant grating wall is proposed in [5]. Its resonance characteristics are described using the FDTD simulation. In this section, we briefly explain how the FDTD simulation is performed for this application. Figure 1 shows a cross section of the cylindrical cavity which is similar to a ring. The cavity wall has a curved resonant grating. PBC and RBC stand for periodic and radiation boundary conditions, respectively. The depth and the base thickness of the grating are $2a$ and d_0, respectively. The pitch of the grating is Λ. When the free space wavelength is λ_0, $\lambda_0 \approx 2\Lambda$. The FDTD simulation is performed by simplifying the structure as a two-dimensional (2-D) one. Figure 2 shows the 2-D grid for the simulation. The polar coordinates of the computation area in Figure 1 are transformed to a 2-D grid of orthogonal coordinates. Note that the boundary conditions are calculated separately and a small portion of the area inside the ring is calculated using 1-D FDTD. The rest of the area is calculated using 2-D FDTD simulation. According to the experimental results, this simulation does not converge when using only single precision floating-point computation.

The flow-chart of the FDTD simulation is shown in Figure 3. The total number of iterations and the iteration number are given by I_{tot} and n, respectively. The electric and magnetic fields are computed one after the other. In between these computations, the boundary conditions are applied. To design a cylindrical resonator for a particular resonance wavelength, the FTDT simulation is executed many times by changing the parameters such as the radius of the cavity and the thickness and the depth of the grated wall. This requires a lot of processing time.

There are many recent works such as [12–16] that use GPUs to accelerate FDTD. GPU acceleration of 2-D and 3-D FDTD for electromagnetic field analysis is proposed in [12, 13], respectively. Multi-GPUs are used to accelerate the FDTD in [14, 15, 17]. Although [15] gives good results for

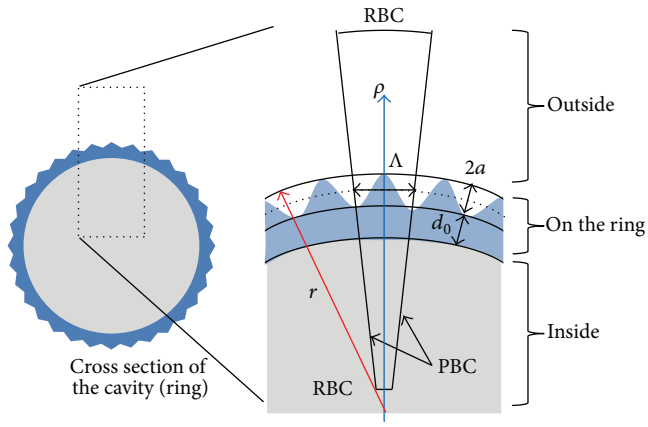


Figure 1: Cross section of a cylindrical cavity.

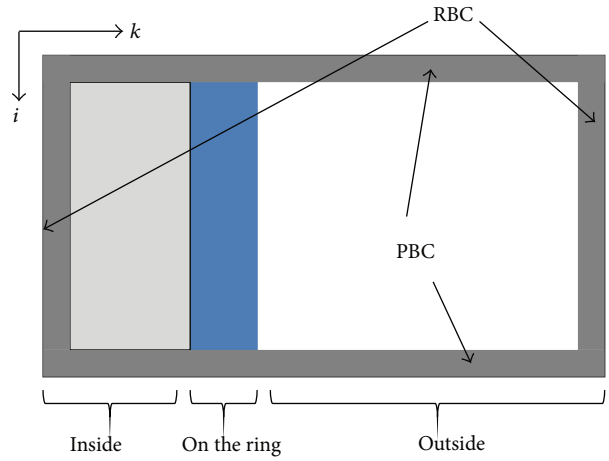


Figure 2: Computation grid for the FDTD simulation.

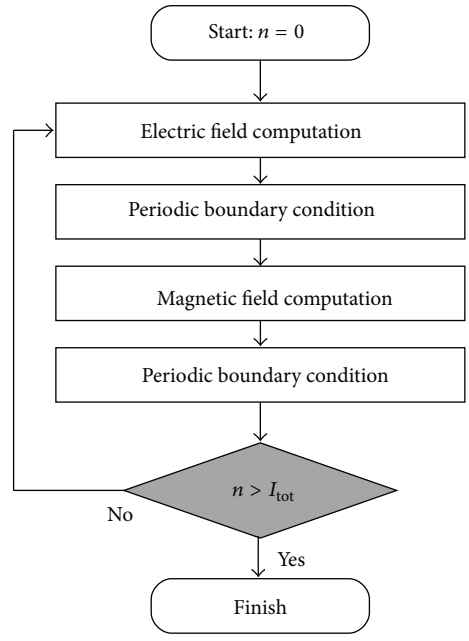


Figure 3: Flow chart of the FDTD computation.

few GPUs, the communication overhead between GPUs via host memory could be a serious bottleneck for a system with a large number of GPUs. Periodical data exchange between GPU and CPU is proposed in [16] to overcome the GPU memory limitation. However, it comes with a 10% performance loss. Moreover, recent GPUs have GPU-to-GPU data transfer capability so that multi-GPU method can be better than this approach. The speed-up of these methods comes with an additional hardware cost.

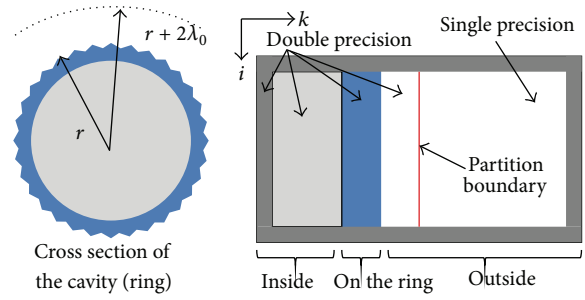
In this paper, our approach is different from the previous works. We focus on increasing the throughput of the GPUs without adding extra hardware. We propose a method to reduce the processing time of the FDTD simulation by using a hybrid of single and double precision floating-point computation. We evaluate using both multicore CPU and GPU to show the effectiveness of our method. Moreover, the proposed method can be used together with most of the previous works to increase the speed-up further.

3. Hybrid of Single and Double Precision Floating-Point Computation

3.1. Hybrid Precision Computation. Since the FDTD simulation usually requires a large amount of data, its processing speed is decided by the memory-bandwidth. Our idea in this paper is to reduce the data amount by using more single precision floating-point computations instead of double precision. Double precision requires 64-bit data while the single precision requires only 32-bit data. Therefore, the data amount can be reduced by half if only the single precision is used. However, some areas on the computation domain have big fluctuations of the electromagnetic field so that the single precision cannot be used. Therefore, we use single precision computation for the area where the fluctuation of the field is very low. We use double precision on the rest of the area. We set a partition boundary on the computation domain that separates the area of single precision computation and the double precision computation. The optimal position of the partition boundary in hybrid precision computation is the grid coordinates that separate the single and double precision computation area in such a way that the processing time is minimized, under which the condition of simulation converges. Note that we assume the simulation converges when using double precision computations. The amount of fluctuations of the electromagnetic field depends on the simulation model and it could only be found by doing the simulation. Therefore, the optimal partition boundary is very hard to determine theoretically.

Due to this problem, we propose a practical approach to partition the computation domain to achieve a sufficient processing time reduction. This practical approach depends on two factors. The first one is that the electromagnetic field reduces exponentially with the distance from its source. It is proportional to $e^{-\alpha\rho}$, where ρ is the distance from the center of the ring (the source). The term α depends on the grating thickness Λ of the cylindrical cavity (refer to [5]). The second one is that the partition boundary is placed outside the ring (or cavity wall). Since the resonator is designed to preserve the field of a certain wavelength, a strong field could exist in the cavity and computing it using single precision may not be possible. Considering these two factors, we conduct experiments and observed the electromagnetic field at different distances from the center. According to these experimental results, we found that the field measured $r + 4\Lambda$ away from the center is very weak and has very small fluctuations. Since $\lambda_0 \approx 2\Lambda$, the partition boundary is given by (1), where r is the radius of the ring as shown in Figure 4(a):

$$\text{Partition boundary} = r + 2\lambda_0. \quad (1)$$



(a) Partition boundary on the ring

(b) Partition boundary on the grid

Figure 4: Partitioning of the computation domain.

The partition boundary on the grid is shown by Figure 4(b). It is calculated by converting the polar coordinates of the partition boundary to the orthogonal coordinates of the grid.

We know that the grid-points away from the partition boundary have small fluctuations so that the computations can be done by single precision floating-point computation. However, we do not know the degree of the fluctuations of the electromagnetic field on or near the cavity wall. It depends on the parameters of the simulation model and we have to do the simulation first to find it out. Therefore, similar to the conventional method, we use double precision floating-point for the computation for the grid-points on or near the cavity wall and the boundaries.

Figure 5 shows the flow-chart of the proposed hybrid precision floating-point computation. The total number of iterations and the iteration number are given by I_{tot} and n, respectively. A part of the area is computed using double precision while the rest is computed using single precision as shown in Figure 4. In the GPU implementation, the initial and output data transfers are done by the CPU and all the other computations are done by the GPU. The single and double precision areas are identified by the thread-ID [18]. Moreover, this hybrid precision computation can also be used in multicore CPUs. In the CPU implementation, single and double precision computation areas are identified by the grid coordinates.

4. Evaluation

For the evaluation, we use an "Intel Xeon ES-2620" CPU and "Nvidia Tesla C2075" GPU [19]. CPU has the hyperthreading

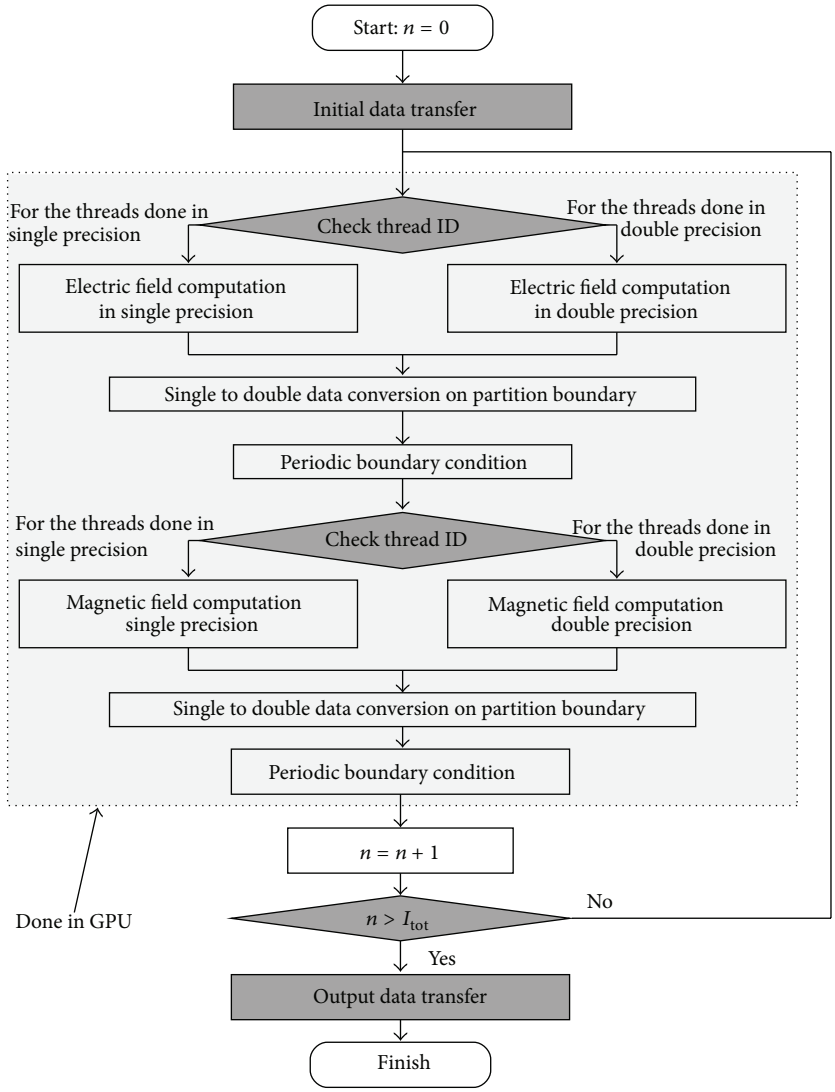


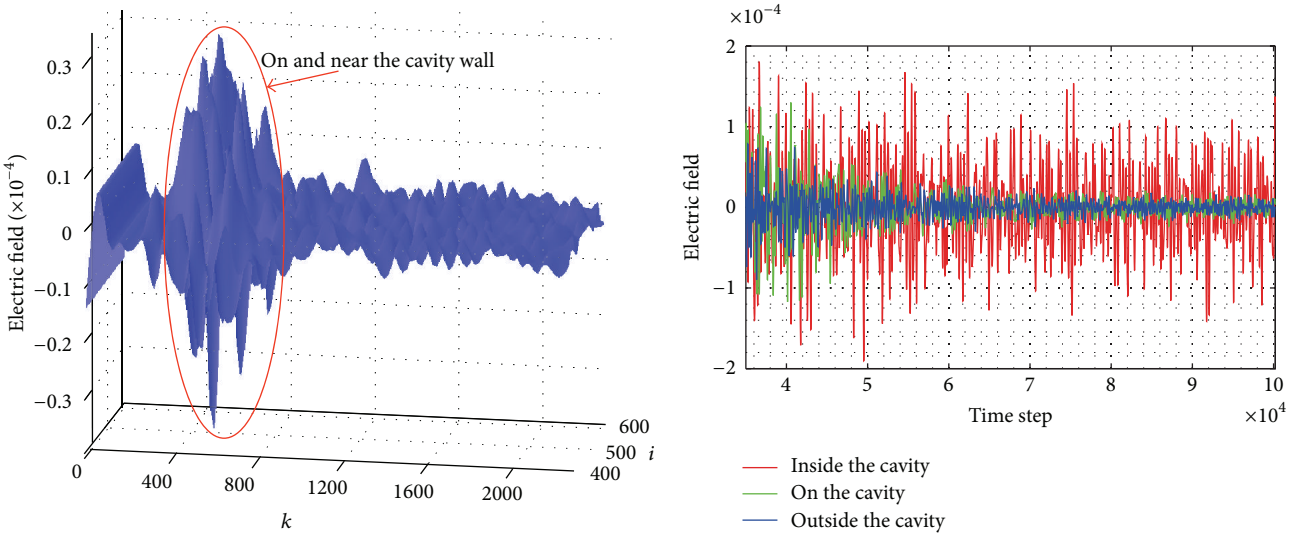
Figure 5: Hybrid single/double precision computation on GPU.

technology with 6 physical cores. The detailed specifications are given in Table 1. We use gcc compiler and CUDA 4.2 [18] for the compilation. GPU is programmed by CUDA which is basically the "C language" with GPU extensions. A detailed description of CUDA architecture and its programming environment are given in [18, 20, 21]. The specifications of the GPU and CPU used for the evaluation are given in Table 1.

Figure 6 shows the electromagnetic field fluctuations against time (iteration) and area. Figure 6(a) shows the electric field at the distance from the center of the ring after 100,000 iterations. The field on and near the cavity wall is strong while the field outside the cavity is weak. We observed similar characteristics from the magnetic field data analysis as well. Figure 6(b) shows the fluctuations of the electric field against the number of iterations. We plot the electric field values of three points: one point inside the ring, one on the ring, and the other outside the ring. The electric field inside the ring remains quite strong while the field outside the ring gets weaker with the time. According to the observations in Figure 6, the electromagnetic field values on and near the cavity walls show large fluctuations, so that high-precision computation is required. Other areas show small fluctuations, so that low-precision computation can be applied. In the proposed hybrid precision computation method, we use single precision for the most of the area

Table 1: Specifications of the evaluation environment.

	CPU	GPU
Type	Xeon ES-2620	Tesla C2075
Frequency	2.0 GHz	1.15 GHz
Number of cores	6	448
Maximum power	95 W	225 W
Global memory (ECC)	8 GB (on motherboard)	6 GB (on card)
Memory bandwidth	42.6 GB/s	144 GB/s



(a) Variation of electric field with area

(b) Variation of electric field with time

Figure 6: Variation of electric field.

Table 2: Estimated memory bandwidth versus number of threads.

Precision	Estimated memory bandwidth (GB/s) Number of threads					
	1	2	3	4	6	12
Double	13.63	25.99	35.00	39.56	39.60	41.71
Hybrid	9.63	18.86	26.66	33.91	36.63	39.24

outside the cavity wall. The rest of the computation area and the boundaries are done in double precision. Note that some of the computation area inside the cavity is done by 1-D FDTD, so that the processing time required is very small. Since the boundary area is very small compared to the total grid size, the time required to process the boundary data is also very small.

Figure 7 shows the processing time of multicore CPU implementation using conventional double precision and the proposed hybrid precision methods. The parallel thread execution is done using OpenMP with C language. Compiler is gcc. According to the results of the double precision computation, we can see a significant processing time reduction from 1 to 3 threads. However, from 4 threads, the processing time does not change much. The results for the hybrid computation are similar to those of the double precision computation. However, the gap between the curves of double precision and hybrid precision widens after 3 threads. To explain this, we estimated the memory bandwidth of each implementation.

Table 2 shows the estimated memory bandwidth in each implementation. It is estimated by calculating the data capacity and dividing it with the processing time. The estimated data capacity that needs to be transferred in a single iteration is about 490 MB. Since there are 100211 iterations, total of over 49 GB of data are transferred. Since the data transfer amount is so large and the cache memory of the CPU is only

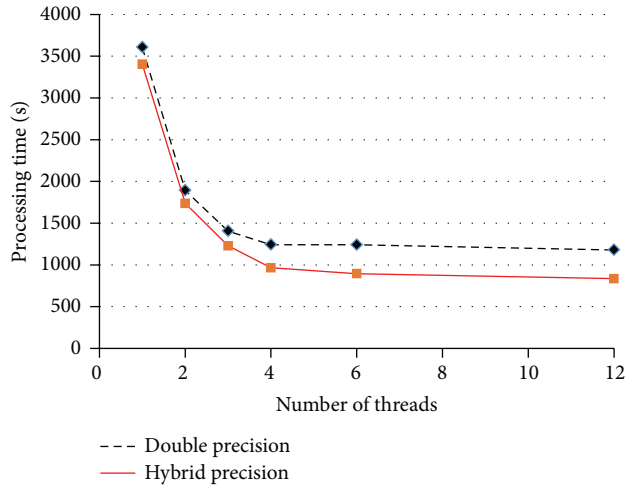


Figure 7: Precision versus processing time of FDTD on CPU.

15 MB, we assume that the impact of the cache memory is the same for all single core and multicore implementations. Since we are only comparing the performance of different implementations, we did not include the impact of cache memory for the memory bandwidth estimation. According to the results in Table 2, memory bandwidths of double and hybrid precision computations near their peaks after 3 and 6 parallel threads. After 3 threads, the hybrid precision that has a smaller data capacity than the double precision gives better results. Therefore, the proposed method gives better results for multicore processors with a lot of parallel processing. Note that, in 1 to 2 thread implementations, the memory bandwidth is not a bottleneck so that both double and hybrid precision take similar amount of execution time (although hybrid precision is slightly better).

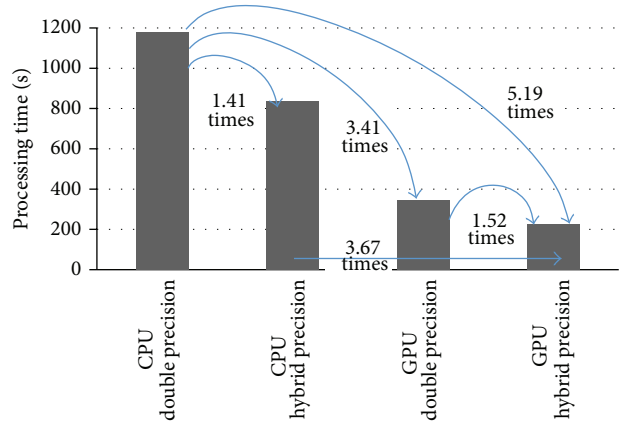


Figure 8: Processing time comparison.

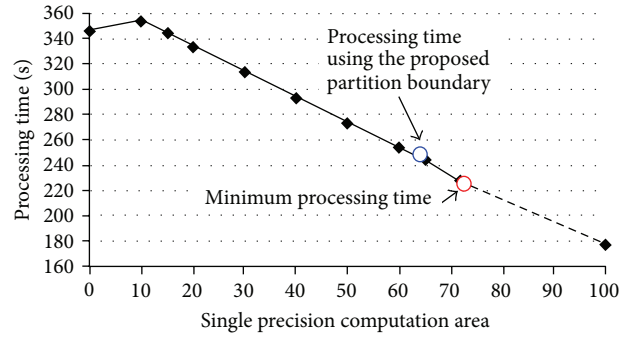


Figure 9: Processing time versus single precision computation area.

Figure 8 shows the comparison between double and hybrid precision computation on CPU and GPU. The CPU implementation is done with 12 threads on 6 cores, using Intel hyperthreading technology. According to the results, GPU implementation with hybrid precision is 5.19 times and 1.52 times faster than the conventional double precision implementations on the CPU and GPU, respectively. Hybrid precision computation on the GPU is 3.67 times faster than that on the CPU. Moreover, hybrid precision computation on the CPU is 1.41 times faster than the conventional double precision implementation on the CPU. The GPU speed-up factors compared to the CPU in double and hybrid precision computations are 3.41 and 3.67, respectively. Interestingly, these figures are very close to the memory bandwidth ratio in Table 1 between GPU and CPU which is 3.38. Therefore, we can assume that the processing times of multicore-CPU and GPU implementations are almost decided by the memory bandwidth. Since single precision floating-point data used in hybrid precision computation need only 4 bytes compared to the 8 bytes in double precision, more data can be transferred with the same bandwidth. As a result, the processing time is reduced.

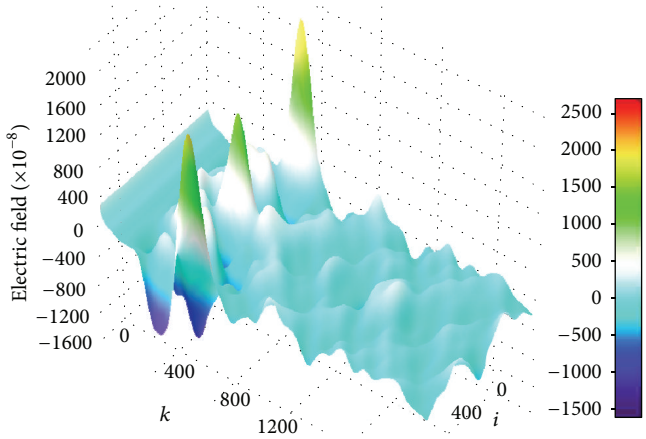
Figure 9 shows the processing time reduction against the single precision computation area for a simulation model. According to these results, the processing time decreases when the percentage of the single precision computation increases. That is, if we can push the partition boundary towards the center of the ring, we can reduce the processing time further. In this example, we push the partition boundary by increasing the single precision area to find the smallest processing time. According to the results, the minimum processing time is observed when 71% of the area is computed using single precision, while the speed-up is 1.52 times. The proposed partition boundary defined in (1) allows only 64% of the computation to be done by single precision and the speed-up is 1.41 times. Since the speed-up of the proposed method is very similar to the best speed-up, we can say that the proposed partition boundary is very effective. Moreover, if the same simulation is executed numerous times by slightly changing the parameters, it would be useful to move the partition boundary towards the center for aggressive processing time reduction, even if there is a risk of a divergence. If the simulation does not converge, we can always move back the partition boundary away from the wall.

As shown in Figure 9, more than 10% of the computation area must be done in single precision to reduce the processing time. Otherwise, the processing time increases due to the single-double precision conversion overhead on the partition boundary. Since the electric (or the magnetic) field is calculated by its near-by magnetic (or electric) field data, both single and double precision data are required to process the data on the partition boundary. Therefore, if the single precision area is too small, the proposed method cannot be applied.

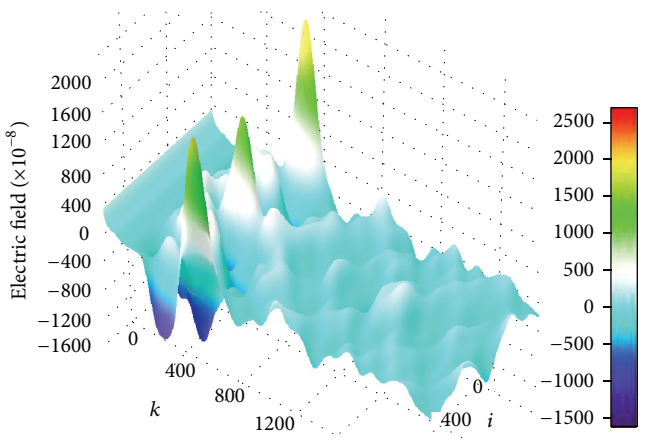
We compare the FDTD simulation results of double and hybrid precision computations in Figure 10 of the revised paper. The electric fields calculated using double precision and hybrid precision computations are shown in Figures 10(a) and 10(b), respectively. The two electric fields are almost identical where a strong field can be seen on the ring area. We observed that the magnetic fields obtained by double and hybrid precision computations are also very similar. Figure 10(c) shows the absolute difference between the double and hybrid precision computations. The absolute difference is smaller than 1.0×10^{-9} and is very small compared to the electric field values shown in Figures 10(a) and 10(b). Considering the almost identical electric field distribution and very small computation difference, we conclude that the hybrid precision computation is accurate enough to be used in FDTD simulations of a cylindrical resonator.

5. Conclusion

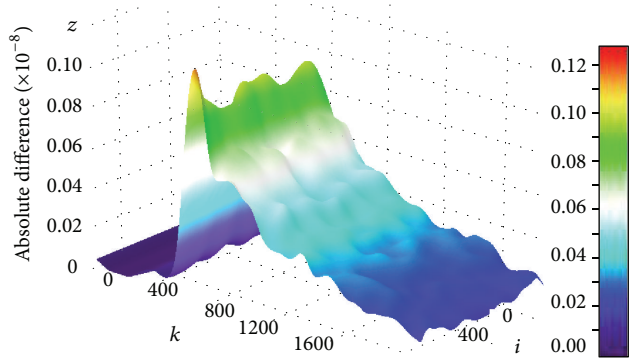
In this paper, we proposed an FDTD computation acceleration method that uses hybrid of single/double precision floating-point to extract the maximum performance from GPU accelerators and multicore CPUs. In multicore parallel processing, the speed of the FDTD computation depends on the memory access speed and the memory bandwidth. Since the amount of data required for double precision floating-point is two times larger than that required for the single precision floating-point, we can increase the processing speed by doing more computation in single precision. Otherwise, we can reduce the cost by using cheaper medium-range GPUs (or CPUs) to extract the same performance of the expensive



(a) Electric field obtained by double precision computation



(b) Electric field obtained by hybrid precision computation



(c) Absolute difference of the double and hybrid computations

FIGURE 10: Comparison of the simulation results using double and hybrid precision computations.

high-end GPUs (or CPUs). According to the results, we achieved over 15 times of speed-up compared to the single-core CPU implementation and over 1.52 times of speed-up compared to the conventional GPU acceleration.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is supported by MEXT KAKENHI Grant no. 24300013.

References

[1] H. S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, vol. 14, no. 3, pp. 302–307, 1966.

[2] A. Taflove and S. C. Hagness, *Advances in Computational Electrodynamics: The Finite-Difference Time-Domain Method*, Artech House, ', 3rd edition, 2005.

[3] T. S. Ibrahim, R. Lee, B. A. Baertlein, Y. Yu, and P.-M. L. Robitaille, "Computational analysis of the high pass birdcage resonator: finite difference time domain simulations for high-field MRI," *Magnetic Resonance Imaging*, vol. 18, no. 7, pp. 835–843, 2000.

[4] E. Kuramochi, H. Taniyama, T. Tanabe, K. Kawasaki, Y.-G. Roh, and M. Notomi, "Ultrahigh-Q one-dimensional photonic crystal nanocavities with modulated mode-gap barriers on SiO2 claddings and on air claddings," *Optics Express*, vol. 18, no. 15, pp. 15859–15869, 2010.

[5] Y. Ohtera, S. Iijima, and H. Yamada, "Cylindrical resonator utilizing a curved resonant grating as a cavity wall," *Micromachines*, vol. 3, no. 1, pp. 101–113, 2012.

[6] A. Taflove and M. E. Brodwin, "Numerical solution of steady-state electromagnetic scattering problems using the time-dependent maxwell's equations ," *IEEE Transactions on Microwave Theory and Techniques*, vol. 23, no. 8, pp. 623–630, 1975.

[7] R. E. Collin, "Electromagnetic resonators," in *Foundations for Microwave Engineering*, pp. 496–517, Wiley, Hoboken, NJ, USA, 2nd edition, 2001.

[8] S. B. Cohn, "Microwave bandpass filters containing high-Q dielectric resonators," *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-16, no. 4, pp. 218–227, 1968.

[9] N. G. Alexopoulos and R. K. Stephen, "Coupled power theorem and orthogonality relations for optical disk waveguides," *Journal*

of the Optical Society of America, vol. 67, no. 12, pp. 1634–1638, 1977.

[10] K. J. Vahala, "Optical microcavities," *Nature*, vol. 424, no. 6950, pp. 839–846, 2003.

[11] V. S. Ilchenko and A. B. Matsko, "Optical resonators with whispering-gallery modes—part II: applications," *IEEE Journal on Selected Topics in Quantum Electronics*, vol. 12, no. 1, pp. 15–32, 2006.

[12] B. Zhang, Z.-H. Xue, W. Ren, W.-M. Li, and X.-Q. Sheng, "Accelerating FDTD algorithm using GPU computing," in *Proceedings of the 2nd IEEE International Conference on Microwave Technology and Computational Electromagnetics (ICMTCE '11)*, pp. 410–413, May 2011.

[13] T. Nagaoka and S. Watanabe, "A GPU-based calculation using the three-dimensional FDTD method for electromagnetic field analysis," in *Proceedings of the 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC '10)*, pp. 327–330, Buenos Aires, Argentina, September 2010.

[14] P. Micikevicius, "3D finite difference computation on GPUs using CUDA," in *Proceedings of the 2nd Workshop on General Purpose Processing on Graphics Processing Units*, pp. 79–84, March 2009.

[15] T. Nagaoka and S. Watanabe, "Multi-GPU accelerated three-dimensional FDTD method for electromagnetic simulation," in *Proceedings of the 33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS '11)*, pp. 401–404, Boston, Mass, USA, September 2011.

[16] L. Mattes and S. Kofuji, "Overcoming the GPU memory limitation on FDTD through the use of overlapping subgrids," in *Proceedings of the International Conference on Microwave and Millimeter Wave Technology (ICMMT '10)*, pp. 1536–1539, May 2010.

[17] N. Kawada, K. Okubo, and N. Tagawa, "Multi-GPU numerical simulation of electromagnetic field with high-speed visualization using CUDA and OpenGL," *IEICE Transactions on Communications*, vol. 95, no. 2, pp. 375–380, 2012.

[18] "NVIDIA CUDA C Programming Guide Ver.4.1," 2011.

[19] http://www.nvidia.com/docs/IO/43395/NV-DS-Tesla-C2075.pdf.

[20] Y. Inoue, T. Sekine, and H. Asai, "Fast transient simulation of plane distribution network by using GPGPU-LIM," *IEICE Transactions on Electronics*, vol. 93, no. 11, pp. 406–413, 2010 (Japanese).

[21] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with CUDA," *Queue—GPU Computing*, vol. 6, no. 2, pp. 40–53, 2008.