A Lattice Gas Cellular Automata Simulator on the Cell Broadband Engine^{TM}

Yusuke Arai, Ryo Sawai, Yoshiki Yamaguchi,
Tsutomu Maruyama, Moritoshi Yasunaga

published in

Parallel Computing: Architectures, Algorithms and Applications ,
C. Bischof, M. Bücker, P. Gibbon, G.R. Joubert, T. Lippert, B. Mohr,
F. Peters (Eds.),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **38**, ISBN 978-3-9810843-4-4, pp. 459-466, 2007.
Reprinted in: Advances in Parallel Computing, Volume **15**,
ISSN 0927-5452, ISBN 978-1-58603-796-3 (IOS Press), 2008.

A Lattice Gas Cellular Automata Simulator on the Cell Broadband Engine^{TM}

Yusuke Arai, Ryo Sawai, Yoshiki Yamaguchi
Tsutomu Maruyama, and Moritoshi Yasunaga

Graduate School of Systems and Information Engineering
University of Tsukuba Ten-ou-dai 1-1-1 Tsukuba, Ibaraki, 305-8573, Japan
E-mail: {arai, yoshiki}@islab.cs.tsukuba.ac.jp

The Cell Broadband Engine (CBE) was developed as a high-performance multimedia processing environment. A CBE is a heterogeneous multicore processor that incorporates the PowerPC Processor Element (PPE) and Synergistic Processor Elements (SPEs). Each SPE is a special purpose RISC processor with 128-bit SIMD capability. In this paper, we describe a new computational method for simulating fluid dynamics on a CBE using the FHP-Ⅲ model. The approach is suitable for simulating very complicated shapes. In our implementation, the speedup of the FHP-Ⅲ model is about 3.2 times compared with an Intel Core2 Duo E6600 running at 2.4 GHz when there are about 11 millions lattice sites.

1 Introduction

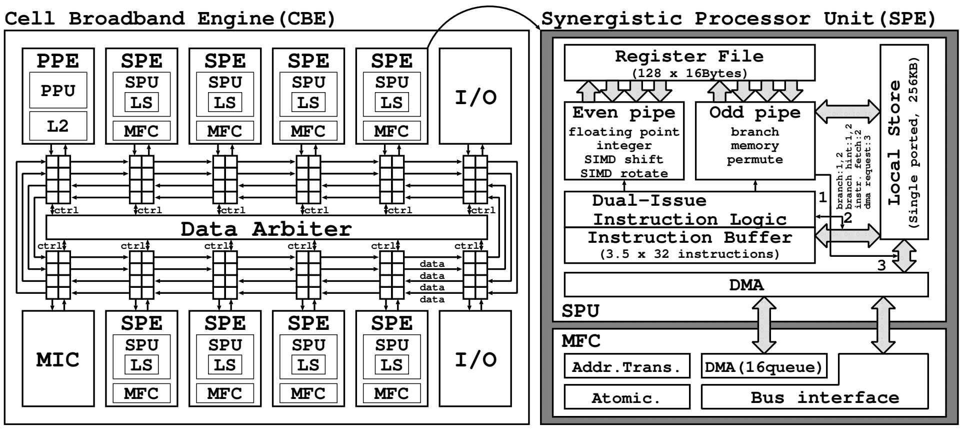
A considerable number of studies have been conducted on the Cellular Automata (CA) that John Von Neumann and Stan Ulam proposed[1]. The CAs have been widely used for modeling different physical systems, such as systems with an emphasis on spin systems and pattern formations in reaction-diffusion systems. Here, the Lattice Gas Automata (LGA) that are a class of the CAs designed for simulating fluid dynamics have been one of the central models.

Within the LGA, a transition function is broken down into two parts: collision stage and propagation stage. In the collision stage, particle collisions are handled by a collision rule, which broadly can be classified into two groups: the Hardy, Pazzis and Pomeau (HPP) model[2], and, the Frisch, Hasslacher, and Pomeau (FHP) model[3,4]. The FHP model is used in this paper. In the propagation stage, the particles move from one lattice site to another adjacent site. Each of the lattice sites can be computed from the values of its own lattice site and adjacent lattice sites. The LGA has computational locality and therefore many approaches with parallel and distributed systems have been proposed[5,6].

The Cell Broadband Engine (CBE) is the multicore processor developed by Sony Computer Entertainment Inc./Sony, TOSHIBA, and IBM[7,8]. As shown in Fig.1, the CBE has nine processor cores, one PowerPC Processor Element (PPE), and eight Synergistic Processor Elements (SPEs) connected by the Element Interconnect Bus (EIB).

The CBE's peek performance is 204.8 GFlops (single precision) or 14.6 GFlops (double precision) running at 3.2 GHz. This is about 11 times better than the performance of an Intel Core2 Duo E6600 processor which achieves 19.2 GFlops (single precision) running at 2.4 GHz.

In this paper, we implement a particle simulation using the FHP-Ⅲ model on a Cell Reference Set (CRS)[10,11]. The Cell Reference Set produced by Toshiba Co. Ltd. includes one CBE, Toshiba's own boards and a cooling system as shown in Figs. 2, 3, and 4.



Cell Broadband Engine (CBE)
PPE
PPU
L2
SPE
SPU
LS
MFC
I/O
Data Arbiter
ctrl
data
MIC
Synergistic Processor Unit (SPE)
Register File (128 x 16Bytes)
Even pipe
floating point integer SIMD shift SIMD rotate
Odd pipe
branch memory permute
Dual-Issue Instruction Logic
Instruction Buffer (3.5 x 32 instructions)
Local Store (Single ported, 256KB)
branch:1,2 branch hint:1,2 instr. fetch:2 dma request:3
DMA
SPU
MFC
Addr.Trans.
DMA(16queue)
Atomic.
Bus interface

Figure 1. Hardware Block Diagram of the Cell Broadband Engine (CBE)

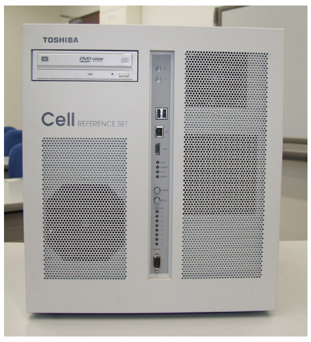


Figure 2. Cell Reference Set (CRS)

Figure 3. The inside of CRS

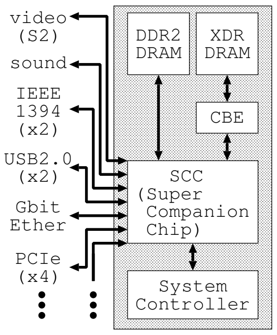


Figure 4. CRS Block Diagram

This paper organized into four sections. In Section 2, we describe the architecture of the CBE. Section 3 reports on the implementation of the FHP-Ⅲ LGA model on the Cell Reference Set with one CBE. Section 4 summarizes our findings.

2 Cell Broadband Engine (CBE)

The overview of the CBE is shown in Fig.1. The CBE is composed of one PowerPC Processor Element (PPE), eight Synergistic Processor Elements (SPEs), one Memory Interface Controller (MIC), I/O, and an Element Interconnect Bus (EIB) as shown in Fig.1. The PPE is a 64 bit Power PC architecture which can run a 64 and 32 bit operating system and applications, and manages the SPEs. The one PPE and eight SPEs are connected with the EIB.

Each SPE has a Synergistic Processor Unit (SPU) and a memory flow controller (MFC). It does not have branch prediction hardware but it allows for branch hint instructions and has a high-performance floating point unit[12, 13]. Therefore, the computational performance of one SPU (3.2 GHz) is 25.6 GFlops (single precision) and 1.83 GFlops (double precision), and high performance has been reported in scientific applications[9]. Each SPU has a 128 bit SIMD processing unit, 128 128 bit registers, and a 256 KB local storage. SPE programs are allocated in the local storage. Direct Memory Access (DMA) is used for the data transfer among the local storage, other SPEs, a PPE, and external memories through the EIB.

3 Implementation of Lattice Gas Automata

3.1 FHP Model

The FHP model is designed for fluid analysis[3,4]. The lattice structure is shown in Fig. 5. One lattice site is connected to six neighbour sites and includes up to seven particles. The particles are divided into two groups according to particle velocity. Six particles and one particle have unit velocity and no velocity, respectively (Fig. 5). The unit velocity, $\mathbf{c}^i$, is obtained by the following equation, where i (=1∼6) is a moving direction.

$$\mathbf{c}^i = \left(cos\frac{(5-i)\pi}{3}, sin\frac{(5-i)\pi}{3} \right) \tag{3.1}$$

Suppose that $\mathbf{x}_k$=(x_k,y_k) is the k-th lattice coordinate, a state on the k-th lattice site at time t ($\mathbf{n}_k(n_k^0, n_k^1, .., n_k^6)$) is described by

$$n_k^i(\mathbf{x}_k + \mathbf{c}_k^i, t+1) = n_k^i(\mathbf{x}_k, t) + \Delta_k^i \left[\mathbf{n}_k(\mathbf{x}_k, t)\right] \tag{3.2}$$

where the function, Δ_k^i, is a variation of $n_k^i(\mathbf{x}_k, t)$ by the collision.

In this paper, we consider an implementation called the FHP-Ⅲ with all 76 collision rules shown in Fig. 6. Each group lists a collection of states that can be in existence before and after a collision with successor states chosen randomly among the alternatives.

3.2 Region Splitting Method on CBE

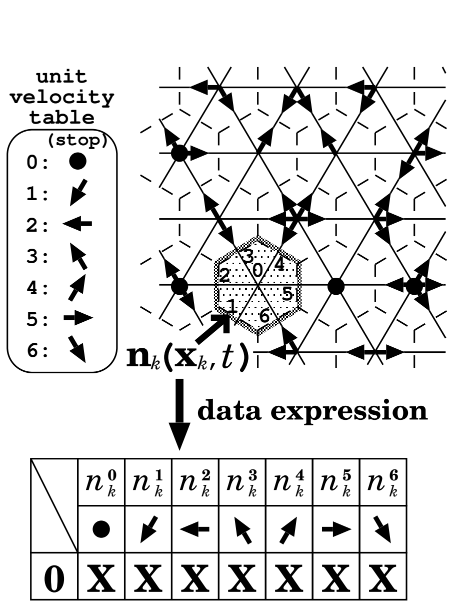
In this paper, we simulated the FHP-Ⅲ LGA model on a 3392×3392 two-dimensional space with 11 million lattice sites. The size of the LGA is too large for the local storage of SPEs in the CBE because the total of a local storage is only 1,792 KB, sufficient only for up to 1354×1354 lattice sites.

We consider the following issues:

(A) computational procedure for a single SPE,

(B) data organization for the FHP-Ⅲ, and

(C) parallel processing arrangement for a whole simulation space.



	n_k^0	n_k^1	n_k^2	n_k^3	n_k^4	n_k^5	n_k^6
	●	↙	←	↖	↗	→	↘
0	**X**	**X**	**X**	**X**	**X**	**X**	**X**

Figure 5. FHP lattice structure

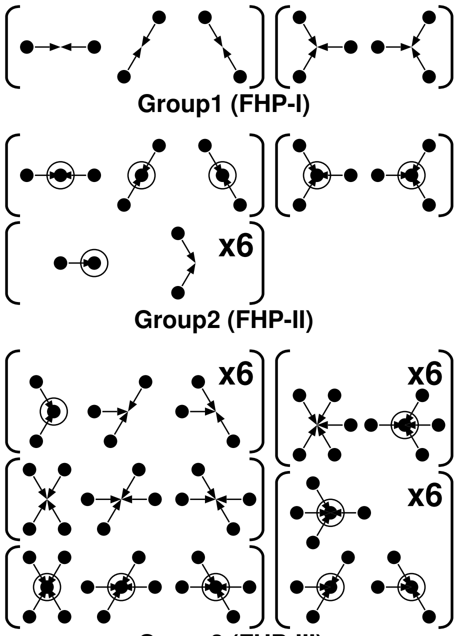


Figure 6. FHP collision rules

Concerning (A), the LGA model is very simple and its computation requires only particle migrations and collisions. It alternates between migration stages and collision stages until the whole simulation finishes. In this procedure, pseudo random numbers are frequently generated and the computational effort is not negligible. We have adopted the combined Tausworthe method[14] and therefore can also reduce the program size on each local storage.

Concerning (B), at each lattice site, a collision among particles happens based on Fig. 6. We can assign a single bit to indicate whether there is a particle or not on each site which is depicted as a shaded hexagon in Fig. 5, and one site can be stored in one byte as shown by the bottom of Fig. 5. Consequently, one byte is a computing unit in our target application. In our implementation, as shown by Fig. 7, we put together 4 procedures of 76 collusion rules in Fig. 6. This enables us to use no branch instruction in collision computation and, using SIMD instructions[15], we achieve high performance with the CBE.

The size of our simulation space can be estimated at about 11 MB ($= 1\,Byte \times 3392 \times 3392$) which is too large for the local storage (256 KB) in a SPE. Hence, concerning (C), we must divide the simulation space to suit a CBE, and face the bottleneck between each SPE and external XDR DRAM (Fig. 4)[16]. DMA data transfers are required every some dozens of microseconds and they cause serious performance loss. For this reason, we consider the region splitting method shown in Fig. 8.

In Fig. 8, suppose that the whole simulation space is 14×14 lattice sites, and 6×6 lattice sites can be stored in a local storage in each SPE. At step 0, the top-left region **A** is loaded to a local storage. After loading **A**, the SPE starts its iteration for the region. This

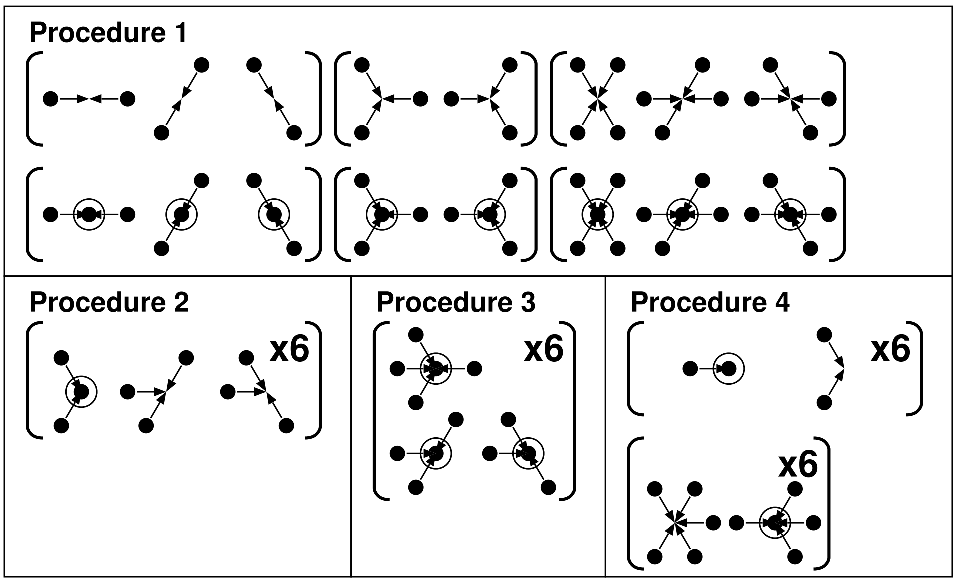


Figure 7. Collision rule integration for the redaction of computational procedure

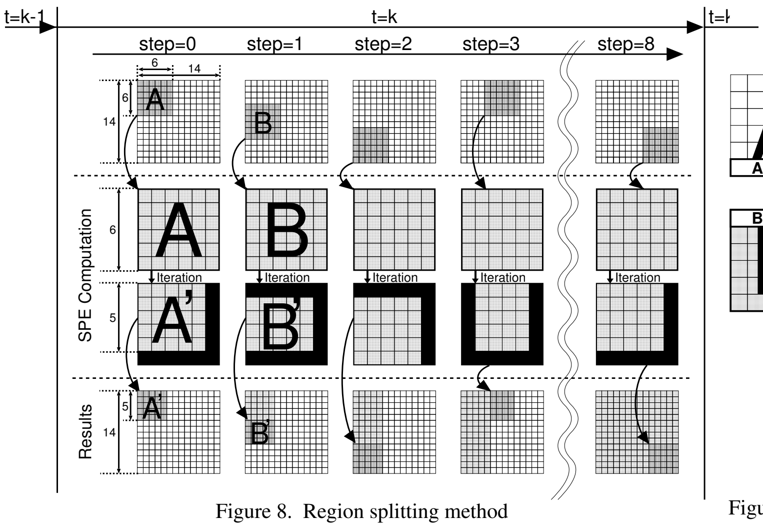


Figure 8. Region splitting method

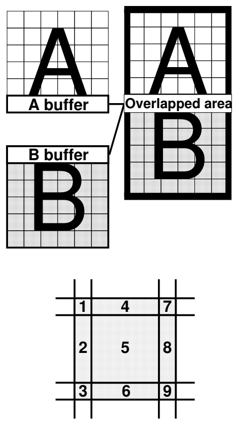


Figure 9. Overlapped area

block approach decreases the number of data transfers.

The LGA is one of stochastic fluid models and we must ensure the consistency of overlapping areas. Focusing on **A** and **B** in Fig. 8, **A buffer** (**A**$_{buf}$) and **B buffer** (**B**$_{buf}$) in Fig. 9 are the same region. Hence, we must use the same pseudo random number generators (RNGs) and their seeds for the computation because the result of **A**$_{buf}$ is consistent with the result of **B**$_{buf}$. Here, we prepare 3 RNGs, **A**$_{RNG}$, **B**$_{RNG}$ and **C**$_{RNG}$ for these

Figure 10. Speedup gain

	Core2 Duo	CBE
clock (GHz)	2.4	3.2
# of core	2	7
L2 Cache	4 MB	7×256 KB
time(msec)	75.7	23.4
speedup	1	3.24

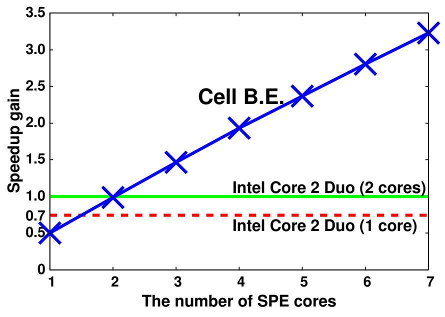


Figure 11. The relationship of speedup gains and no. of SPE cores

computation, which are used for the independent region of **A** and **B**, and the overlapped region, respectively. Therefore, $\mathbf{A}_{RNG}$ and $\mathbf{C}_{RNG}$ are used for SPE1, and $\mathbf{B}_{RNG}$ and $\mathbf{C}_{RNG}$ are used for SPE2 when **A** and **B** are computed by SPE1 and SPE2, respectively. As shown in the bottom of Fig. 9, in our current implementation, one region has normally 9 RNGs to maintain the consistency of surrounding overlapped area.

3.3 Evaluation of Results and Discussion

The computing time and speedup gains are shown in Table 10 using kernel 2.6.17-1 and Intel compiler 9.1 (option: -O3 -xT -parallel -static -openmp). According to Table 10, we achieve about 3.2 times speedup with the CBE compared to an Intel Core2 Duo E6600 running at 2.4 GHz. When we compare one SPE in the CBE to a single core in an Intel Core2 Duo E6600, an Intel Core2 Duo E6600 is about 1.5 faster than one SPE. But the performance is the reflection of cache size; an Intel Core2 Duo E6600 has 4 MB L2 Cache but the SPE has only 256 KB.

In our experimental result, a speedup ratio was calculated by dividing a computational time required by an Intel Core2 Duo E6600 using both cores by a computational time for a simulation on the CBE. A solid line in Fig. 11 shows the base line (=1.0) and a dashed line illustrates the speedup gain (= 0.74) by dividing dual cores' computational time by a single core's computational time. Figure 11 also illustrates the relationship between the speedup ratio and the number of SPE cores used in a simulation. The speedup ratio vs. the number of SPE cores is almost linear and is approximated by the following equation.

$$(Speedup\, ratio) \;=\; 0.45 \times \;(number\ of\ SPE\ cores) \;+ 0.08 \qquad (3.3)$$

As can be expected from Eq. 3.3, the CBE can keep high scalability. One of reasons is that the SPEs are connected by the EIB bus and the result can be read/written from/to the other SPEs. The other is that memory bandwidth of the CBE is larger than an Intel Core2 Duo E6600.

4 Conclusion

In this paper, we reported on the implementation of the LGA on the Cell Reference Set. The speedup ratio of a CBE running at 3.2 GHz, compared with an Intel Core2 Duo E6600 running at 2.4 GHz, was about 3.2 times. Thus, the CBE can achieve sufficient speedup even though the implementation of the LGA does not require floating point computation which is SPE's strong point. In addition, experimental results lead us to the conclusion that the CBE has high scalability in our target application.

Future tasks are to decrease the use of a local storage and the penalty of the instruction for branch on condition, and extract the part that can be parallelized dynamically and executed with SIMD operations. So, implementing the programs with SPEs divided into two groups, and utilizing the PPE not used at the time, we plan to categorize SPEs into the data control elements which extract the computable part and data operating elements which compute the data generated by the data control elements, examine the performance of CBE as one chip.

Acknowledgements

We would like to thank here TOSHIBA CORPORATION and TOSHIBA Semiconductor Company for their cooperation and support of this work. We are also indebted to Dr. Taiji, Deputy Project Director, Computational and Experimental Systems Biology Group in RIKEN GSC.

References

1. J. von Neumann, *The Theory of Self-Reproducing Automata*, A. W. Burks (ed), Univ. of Illinois Press, Urbana and London, (1966).
2. J. Hardy, et al., *Molecular dynamics of a classical lattice gas: transport properties and time correlation functions*, Phys. Rev. A, **13**, 1949–1961, (1976).
3. U. Frisch, B. Hasslacher and Y. Pomeau, *Lattice gas automata for the Navier-Stokes equation*, Phys. Rev. Lett., **56**, 1505–1508, (1986).
4. U. Frish, et al., *Lattice gas hydrodynamics in two and three dimensions*, Complex Systems, **1**, 649–707, (1987).
5. T. Toffoli and N. Margolus, *Cellular Automaton Machines – New Environment for Modeling*, (MIT Press, 1987).
6. G. S. Alamsi and A. Gottlieb, *Highly Parallel Computing*, (Benjamin-Cummings, 1994).
7. D. Pham, et al, *The design and implementation of a first-generation CELL processor*, in: IEEE International Solid-State Circuits Symposium, pp. 184–186, (2005).
8. J. Kahle, et al., *Introduction to the Cell multiprocessor*, IBM Journal of Research and Development, vol. **49**, No.4/5, pp. 589–604, (2005).
9. S. Williams, et.al., *The potential of the Cell processor for scientific computing*, in: ACM International Conference on Computing Frontiers, May (2006).
10. J. Amemiya, et. al., *Cell configuration of Cell reference set software*, Toshiba Review, **61**, 37–41, (2006).
11. S. Osawa, et. al., *Cell software development environment*, Toshiba Review, **61**, 47–51, (2006).
12. Y. Kurosawa, et. al., *Cell Broadband Engine Next-Generation Processor*, Toshiba Review, **61**, 9–15, (2006).

13. Sony, *Synergistic Processor Unit (SPU) Instruction Set Architecture*, Ver. 1.1, Jan. (2006).
14. M. Barel, *Fast hardware random number generator for the Tausworthe sequence*, in: 16th Annual Simulation Symposium, pp. 121–135, (1983).
15. Y. Arai, et al., *An approach for large-scale fluid simulation with a multi-core processor*, in: Symposium on Advanced Computing Systems and Infrastructures, Vol. **2007**, pp. 159–160, (2007).
16. R. Sawai, et al., *Relationship between SPEs and EIB on Cell Broadband Engine*, IEICE Technical Report, Vol. **106**, pp. 87–92, (2006).