A comparative study on ASIC, FPGAs, GPUs and general purpose processors in the O(N^2) gravitational N-body simulation

Tsuyoshi Hamada*, Khaled Benkrid†, Keigo Nitadori‡ and Makoto Taiji‡
* *Nagasaki Advanced Computing Center, Nagasaki University,*
1-14, Bunkyo-machi, Nagasaki, 852–8521 Japan
Email: hamada@cis.nagasaki-u.ac.jp
† *School of Engineering, The University of Edinburgh*
Email: K.Benkrid@ed,ac.uk
‡ *RIKEN – The Institute of Physical and Chemical Research,*
61-1 Ono, Tsurumi, Yokohama, Kanagawa 230-0046, Japan
Email: {keigo,taiji}@riken.jp

***Abstract*—In this paper, we describe the implementation of gravitational force calculation for N-body simulations in the context of Astrophysics. It will describe high performance implementations on general purpose processors, GPUs, and FPGAs, and compare them using a number of criteria including speed performance, power efficiency and cost of development. These results show that, for gravitational force calculation and many-body simulations in general, GPUs are very competitive in terms of performance and performance per dollar figures, whereas FPGAs are competitive in terms of performance per Watt figures.**

***Keywords*-N-body simulation; GPGPU; High Performance Computing**

I. Introduction

Many-body simulations have widespread use in scientific and engineering applications. Among such applications we can name the problem of investigating the formation and evolution of astronomical systems e.g. planetary systems, globular clusters, galaxies, and clusters of galaxies [1], [2], molecular dynamics simulation [3]–[6], fluid mechanics [7]–[11], acoustics and electromagnetic simulations [12]. Central to many-body simulations is the calculation of interaction forces between all bodies or particles in the system. In the context of astrophysical simulations, we numerically evaluate interactions between the particles e.g. planets, and advance the particles according to Newton' s equation of motion. In this case, and since the gravity is a long-range interaction, the calculation cost is $\mathcal{O}(N^2)$ per time-step, where N is the number of particles in the system. This complexity can be reduced $\mathcal{O}(N \log N)$, by using some approximation algorithms, such as the Barnes-Hut tree-code [13], albeit with a large scaling coefficient. Nonetheless, the size of the many-body simulation (i.e. N) is always limited by the available computational resources, and with increasing need for larger system simulations arises the need for ever more high performance and efficient computational platforms for many-body simulations [6], [14]–[22].

Computational solutions to this problem vary from special purpose ASIC-based platforms [1], [2] to off-the-shelf programmable platforms [18]. The Grape ("GRAvity piPE") project [1], [2] for instance built ASIC-based supercomputer solutions for gravitational force calculations where the calculation of pairwise interactions was implemented in an ASIC chip in the form of a fully pipelined hardwired processor dedicated to gravitational force calculation. The total number of floating-point operations for each pairwise interaction force calculation in the case of gravitational forces is 20, which are pipelined in hardware. Moreover, since each particle interacts with all other particles in the system, instruction-level as well as data-level parallelism can be exploited. Furthermore, data cashing techniques can be used to reduce the required memory bandwidth.

However, ASIC solutions suffer from their lack of flexibility as they can't be reprogrammed to implement new algorithms. Moreover, the cost of building new ASIC chips with state-of-the-art fabrication technologies is becoming increasingly high, especially for relatively moderate volumes. Hence, programmable solutions are set to become more attractive. General purpose processor (GPP)-based solutions have the advantage of being off-the-shelf and less expensive. However, higher power and area consumption and lower speed performance, compared to dedicated hardware solutions, are possible disadvantages.

More recently, new emerging technologies that can address some of the above disadvantages of both ASICs and GPPs have been proposed for many-body simulations [17], [19], [23]. Among these we name Field Programmable Gate Array (FPGA)-based solutions [18], [24], [25] and Graphics Processing Unit (GPU)-based platforms [26]–[28]. In this paper, we will assess the advantages and disadvantages of these technologies in the context of many-body simulations, by presenting comparative implementation results of gravitational force calculations on FPGAs, GPUs, ASICs and GPPs.

978-0-7695-3714-6/09 $25.00 © 2009 IEEE
DOI 10.1109/AHS.2009.55

The remainder of this paper is organized as follows. Section 2 will present background information on gravitational force calculation and many-body simulation. The following section will present our hardware implementation of a generic force calculation pipeline. After that, we will present our hardware implementation results on FPGA, and compare them with other implementations on various GPU boards and GPPs as well as ASIC implementations. These results will then be discussed before conclusions are drawn.

II. Background

As mentioned in the introduction section, the basic operation in many-body simulations consists in calculating the interaction force between pairs of particles. The accumulation of the forces acting on a single particle by all other particles in the system dictates its behaviour i.e. position, velocity and acceleration in the next time step. In the case of gravitational forces, the accumulated force is given by:

$$\boldsymbol{f}_i = m_i \sum_j \frac{m_j \boldsymbol{r}_{ij}}{(\boldsymbol{r}_{ij}^2 + \epsilon^2)^{3/2}}, \quad (1)$$

where $\boldsymbol{r}_i$ and m_i are the position and mass of particle i. $\boldsymbol{r}_{ij} = \boldsymbol{r}_j - \boldsymbol{r}_i$, and ϵ is a softening parameter.

Other interaction forces are described in a more generic way as:

$$\boldsymbol{f}_i = \sum_j \boldsymbol{G}(\boldsymbol{r}_i, \boldsymbol{r}_j), \quad (2)$$

where $\boldsymbol{G}$ is a user-defined function. In the case of gravitational forces:

$$\boldsymbol{G}_i = \frac{m_j \boldsymbol{r}_{ij}}{(\boldsymbol{r}_{ij}^2 + \epsilon^2)^{3/2}}, \quad (3)$$

where $\boldsymbol{r}_{ij} = \boldsymbol{r}_j - \boldsymbol{r}_i$.

It is worth mentioning at this stage that unlike ASICs, FPGAs allow us to plug-in any function $\boldsymbol{G}$ through reconfiguration, hence resulting in a more generic implementation. FPGAs can also allow for various arithmetic types and precision levels to be used and experimented with on the same hardware platform.

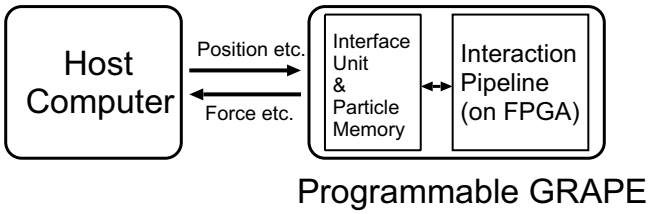

Figure 1. Basic structure of a hardware-accelerated many-body simulation system

Figure 1 shows a basic structure of a hardware accelerated solution for many-body simulations. It consists of a host computer and an acceleration coprocessor for force calculation. An interface unit controls communication between the programmable host computer and the acceleration hardware. The latter could consist of a number of FPGA chips on which the interaction pipelines are implemented. The host computer performs all other calculations e.g. position and velocity update. In addition to the FPGA hardware, a particle memory local to the coprocessor caches all particles involved in a particular round of interaction force calculation with a particular particle i. The particle information stored in memory includes particle position, velocity and mass in the case of gravitational force calculation.

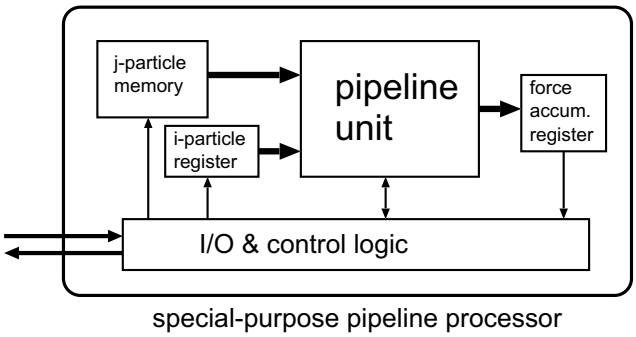

Figure 2. Basic structure of a the hardware accelerator

Figure 2 gives a block architecture of a hardware accelerator instance e.g. implemented on a single FPGA chip.

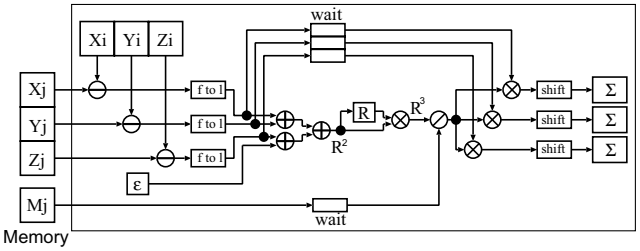

Figure 3. Block diagram of the pipeline for gravitational force

Figure 3 shows a fully pipelined hardware architecture for gravitational force calculation [15], [18], [29]. Here, the position data for both particle i and particle j are in fixed-point format, while m_j is in logarithmic format. The position vector $\boldsymbol{r}_i$ is expressed in its three Cartesian components x_i, y_i, z_i (the same applies to position vector $\boldsymbol{r}_j$). After subtraction, $x_j - x_i$, the results are converted to logarithmic format, and all subsequent calculations are performed in logarithmic format too.

The following sections will give implementation results of the above gravitational force calculator on various FPGA chips, as well as comparative implementation results on GPU accelerators and GPP-based solutions.

III. FPGA Implementation

We have implemented the force calculation pipeline described in Figure3, on a three FPGA platforms: the Virtex2Pro-based for Bioler-3 system 4 [20]–[22], [30],

Figure 4. Photograph of Bioler-3.

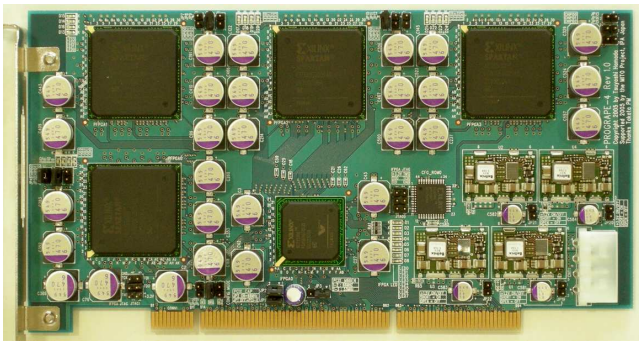

Figure 5. Photograph of PROGRAPE-4.

the Virtex-Pro-based Cray XD1 system [30], [31] and the Spartan3-based PROGRAPE-3 system5. Figure 6 and 7 and 8 give the hardware architectures of the Boiler-3, Cray XD1 and PROGRAPE-3 systems respectively.

Table I
MODEL

Model	Position	Internal(exponent, mantissa)	Accumulation
G3	20bit fixed	14(7,5)bit log	56bit fixed
G5	32bit fixed	17(7,8)bit log	64bit fixed
G5+	32bit fixed	20(7,11)bit log	64bit fixed

Table I gives different accuracy levels implemented on FPGAs. Here, G3 refers to the accuracy used in the ASIC-based GRAPE-3 system, G5 to the accuracy used in the ASIC-based GRAPE-5 system, while G5+ refers to a more accurate G5 version.

Table II presents area and speed implementation results of these pipeline variations on two FPGA chips, namely Virtex2Pro-5(XC2VP70-5) chip and Spartan3-5 (XC3S5000-5) for different number of pipeline stages. At this stage, it is worth mentioning that increased pipelining in FPGAs comes at little logic (LUT) overheads at the expense of an increased use of flip-flops. FPGAs are however rich in flip-flops and their use comes at no extra slices cost (if the slice LUTs are used for the combinatorial path) in contrast to ASICs where flip-flops are very costly.

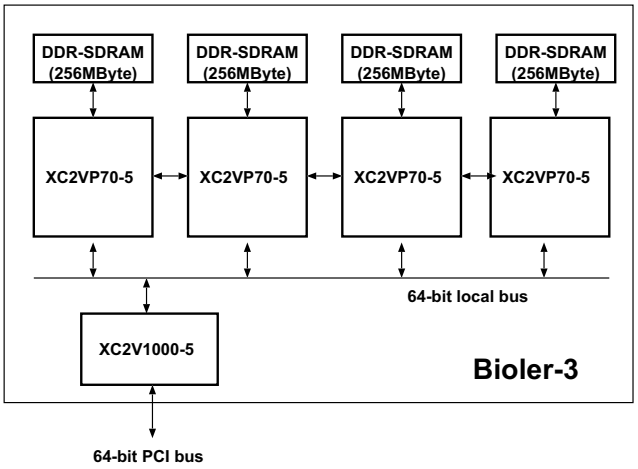

Figure 6. Bioler-3 Hardware Architecture.

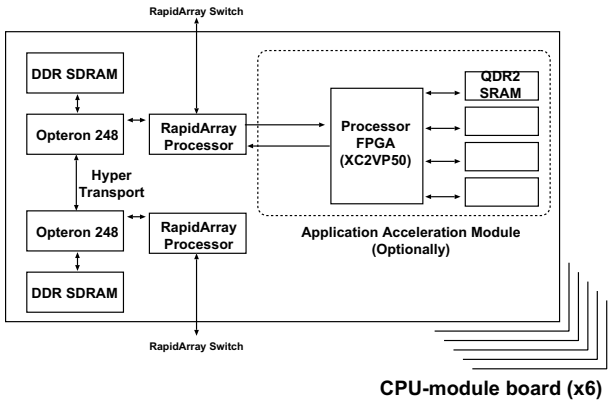

Figure 7. Cray XD1 Hardware Architecture.

Table II presents area and speed implementation results of these pipeline variations on two FPGA chips, namely Virtex2Pro-5(XC2VP70-5) chip and Spartan3-5 (XC3S5000-5) for different number of pipeline stages. This shows that increased pipelining in FPGAs comes at the expense of an increased use of flip-flops with little logic (LUT) overheads. FPGAs are however rich in flip-flops and their use comes at no extra slice cost (if the slice LUTs are used for the combinatorial path) in contrast to ASICs where

Table III
PERFORMANCE OF GENERATED PIPELINES (VIRTEX2PRO)

Model	$f_{\max}$ (MHz)	Size (LUT)	Memory (bit)	Multiplier (18x18MULT)	Stage
G5	150.5	2690	108k	3	42
G3	154.6	2020	108k	0	37
G5+	150.5	3097	432k	3	42

Table II
PERFORMANCE OF THE GENERATED PIPELINE (MODEL G5)

Virtex2Pro (-5)				Spartan3 (-5)			
stage	size		$f_{\max}$(MHz)	stage	size		$f_{\max}$(MHz)
	LUTs	FFs			LUTs	FFs	
10	3338	940	47.1	10	3366	940	33.1
12	3402	1089	57.1	12	3411	1089	51.6
13	3276	1225	78.4	13	3302	1243	67.4
15	3297	1360	78.8	14	2889	1207	60.0
16	2878	1324	81.3	16	2886	1321	80.2
17	2883	1351	89.7	18	2888	1428	79.4
18	2889	1477	85.0	19	2889	1554	77.5
19	2871	1564	88.8	20	2889	1572	70.7
21	2860	1659	108.4	21	2861	1659	73.3
22	2754	1747	107.0	22	2754	1747	90.8
23	2860	1826	110.3	24	3021	1875	86.6
24	3015	1875	110.7	25	3033	1950	92.0

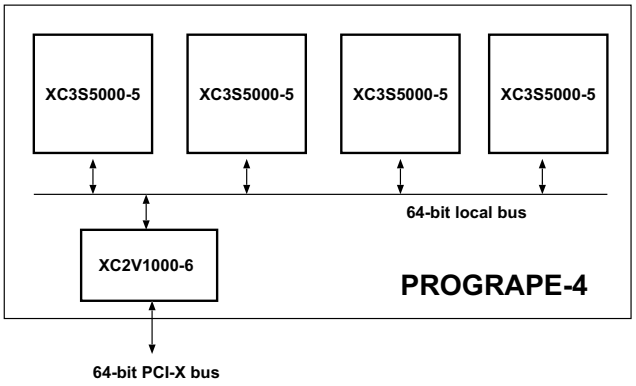

Figure 8. PROGRAPE-4 Hardware Architecture.

flip-flops are very costly.

IV. COMPARATIVE IMPLEMENTATION RESULTS

In an attempt to compare our FPGA implementations with alternative technologies, table IV shows implementation results of our gravitational force pipeline, with G5 accuracy, using ASIC, GPU and GPP technologies. Comparison criteria include speed performance, power consumption, performance in Gflops per chip, performance in Gflops per Watt, in addition to other information such as chip technology and year of development.

In the performance maesurement, we use a $O(N^2)$ of leap-flog scheme for time integration, in which the ratio of computation time for the access to the host or off chip becomes less than 1 such as the hierarchical tree algorithm or individual time-step scheme, the results strongly depend on the performance of host computer and communication speed or memory bandwidth, and the performance analysis becomes more complex than that described in this paper.

In the power maesurement, we use difference between idle and computing conditions. Firstly we measure the power of a system in the idle condition, and nextly we measure again the power of the system in computing, finally we obtain the power consumption by subtractiong the power values between the first and next conditions.

It is of course extremely difficult to perform totally fair comparisons between different technologies and there are various technologies ranging from 500 nm to 45 nm. However, if we accept that ASIC solutions for many-body simulations are extremely difficult to justify with the spiralling cost of state-of-the-art ASIC fabrication, then programmable and reconfigurable technologies are the implementation platforms of choice for these types of applications. We first note that GPU implementations offer the highest Gflops performance (the G92 GPU implementation is 11x faster than the Spartan3-based FPGA implementation), with the optimized Q6600 Core2Quad implementation (using SSE instructions, the Phantom-GRAPE implementation[1]) performing slightly better than the Spartan3-based FPGA implementation. However, if we adjust for the chip technology used (90nm vs. 65nm) we can expect state-of-the-art FPGA solutions to perform better than general purpose processors, although not by much. Given the relatively higher cost of FPGAs as well as their low level programming model (all of our designs were captured in VHDL), GPP solutions look more advantageous. However, the performance per Watt figure of the Core2Quad Q6600 implementation is 34x less than the Spartan3-based FPGA implementation. The performance per Watt figure of the G92-based GPU implementation is also 15x less than the Spartan3-based implementation. This means that for very high performance, large scale, many-

[1] `http://progrape.jp/phantom/`

Table IV
IMPLEMENTATION RESULTS (GRAPE-5 AND G5 MODEL)

	GRAPE-5	Bioler-3	Cray XD1	PROGRAPE-4	ASUS EN8800GTX	MSI N9800GTX+	Core2Quad Q6600	Atom 230
Device Chip	ASIC 300k gates	FPGA XC2VP70-5	FPGA XC2VP50-7	FPGA XC3S5000-5	GPU G80	GPU G92	CPU SSE	CPU SSE
Development Year	1997	2004	2004	2006	2007	2008	2007	2008
Chip technology	500 nm	130 nm	130 nm	90 nm	90 nm	65 nm	65 nm	45 nm
Chips/board	8	4	1	4	1	1	1	1
Pipelines/chip	2	16	10	16	N/A	N/A	N/A	N/A
Frequency (MHz)	80	133.3	120	100	1350	1890	2400	1600
Gflops/chip	24.3	81	45.6	60.8	470.8	687.1	70.3	6.35
Gflops/board	48.6	324.2	45.6	243.2	470.8	687.1	70.3	6.35
ratio of performance (against GRAPE-5)	1.0	6.7	0.9	5.0	9.7	14.1	1.45	0.13
Cost($) per board	N/A	15000	N/A	2400	790	268	200	74
Mflops per Cost	N/A	21.6	N/A	101	596	2663	352	85.8
Power Consumption per board (without host)	80 W	30 W	N/A	5 W	148 W	122 W	49 W	3.1 W
Power Consumption per chip	8 W	7.5 W	N/A	1.3 W	148 W	122 W	49 W	3.1 W
Gflops/Watt	0.61	11	N/A	49	3.2	5.6	1.43	2.05

body simulations, FPGAs could be a viable solution on an energy cost basis. Nonetheless, GPUs are much cheaper than FPGAs and have a relatively easier programming model. One could also argue that GPUs could well be clocked down to reduce their energy consumption, at a relatively lower performance penalty. We will conduct such experiments in future work.

V. CONCLUSION

Many-body simulations are characterised by a high degree of data and instruction parallelism as well as data locality. The use of parallel computing technologies in the form of FPGAs and SIMD architectures can thus lead to very large speed-ups compared to implementations running on sequential single processors. However, these technologies vary in their cost, programming abstraction level, and power consumption. Our experiments in the context of gravitational force calculations in many-body simulations showed that GPUs are very competitive in performance and performance per cost figures as they led to the highest performance figures. This combined with their lower cost and higher programming abstraction level, makes them a very attractive implementation platform for many-body simulations. Nonetheless, the performance per Watt figure favoured FPGAs with a factor of 15 to 1 compared to GPUs and 34 to 1 compared to GPPs. As such, we anticipate the use of FPGAs for large scale high performance simulations, where power consumption is a bottleneck, to be a possible niche market for FPGAs. Future work consists in extending the experiments conducted in this research through the use of dynamic frequency scaling for GPU and GPP implementations, as well as extending our hardware pipeline to molecular dynamics simulations where calculations are more complex than those involved in astrophysical applications and accuracy levels are higher.

REFERENCES

[1] D. Sugimoto, Y. Chikada, J. Makino, T. Ito, T. Ebisuzaki, and M. Umemura, "A Special-Purpose Computer for Gravitational Many-Body Problems," *Nature*, vol. 345, pp. 33–+, May 1990.

[2] J. Makino and M. Taiji, *Scientific Simulations with Special-Purpose Computers — The GRAPE Systems*. New York: John Wiley and Sons, 1998.

[3] T. Fukushige, M. Taiji, J. Makino, T. Ebisuzaki, and D. Sugimoto, "A Highly Parallelized Special-Purpose Computer for Many-Body Simulations with an Arbitrary Central Force: MD-GRAPE," vol. 468, pp. 51–61, Sep. 1996.

[4] R. Susukita, T. Ebisuzaki, B. G. Elmegreen, H. Furusawa, K. Kato, A. Kawai, Y. Kobayashi, T. Koishi, G. D. McNiven, T. Narumi, and K. Yasuoka, "Hardware accelerator for molecular dynamics: MDGRAPE-2," *Computer Physics Communications*, vol. 155, pp. 115–131, Oct. 2003.

[5] T. Narumi, R. Susukita, T. Ebisuzaki, G. D. McNiven, and B. G. Elmegreen, "Molecular dynamics machine: Special-purpose computer for molecular dynamics simulations," *Molecular Simulation*, vol. 21, pp. 401–415, 1999.

[6] M. Taiji, T. Narumi, Y. Ohno, N. Futatsugi, A. Suenaga, N. Takada, and A. Konagaya, "Protein Explorer: A Petaflops Special-Purpose Computer System for Molecular Dynamics

Simulations," in *Proc of Supercomputing '03*, Nov. 2003, (in CD-ROM).

[7] R. A. Gingold and J. J. Monaghan, "Smoothed particle hydrodynamics - Theory and application to non-spherical stars," vol. 181, pp. 375–389, Nov. 1977.

[8] J. J. Monaghan, "Smoothed particle hydrodynamics," *Annual review of astronomy and astrophysics*, vol. 30, pp. 543–574, 1992.

[9] M. Steinmetz, "GRAPESPH: cosmological smoothed particle hydrodynamics simulations with the special-purpose hardware GRAPE," *Mothly Notice of Royal Astronomical Society*, vol. 278, pp. 1005–1017, Feb. 1996.

[10] R. Klessen, "GRAPESPH with fully periodic boundary conditions - Fragmentation of molecular clouds," vol. 292, pp. 11–+, Nov. 1997.

[11] G. R. Liu and M. B. Liu, *Smoothed Paricle Hydrodynamics — a meshfree particle method.* Tuck Link: World Scientific, 2003.

[12] C. A. Brebbia, *The Boundary Element Method for Engineers*. London: Pentech Press, 1978.

[13] J. Barnes and P. Hut, "A Hierarchical O(NlogN) Force-Calculation Algorithm," *Nature*, vol. 324, pp. 446–449, Dec. 1986.

[14] S. K. Okumura, J. Makino, T. Ebisuzaki, T. Fukushige, T. Ito, D. Sugimoto, E. Hashimoto, K. Tomida, and N. Miyakawa, "Highly Parallelized Special-Purpose Computer, GRAPE-3," *PASJ*, vol. 45, pp. 329–338, Jun. 1993.

[15] A. Kawai, T. Fukushige, and J. Makino, "$7.0/Mflops Astrophysical N-Body Simulation with Treecode on GRAPE-5," in *Proc of Supercomputing '99 (Gordon Bell Prize winner)*, Nov. 1999, pp. 197–206.

[16] S. Warren, M, K. Salmon, J, J. Becker, D, P. Goda, M, and T. Sterling, "A 55 TFLOPS Simulation of Amyloid-forming Peptides from Yeast Prion Sup35 with the Special-purpose Computer System MDGRAPE-3," in *Proc. Supercomputing 97, in CD-ROM. IEEE, Los Alamitos, CA*, 1997.

[17] T. Hamada, T. Fukushige, A. Kawai, and J. Makino, "PROGRAPE-1: A Programmable Special-Purpose Computer for Many-Body Simulations," Apr. 1998, pp. 256–257.

[18] ——, "PROGRAPE-1: A Programmable, Multi-Purpose Computer for Many-Body Simulations," vol. 52, pp. 943–954, Oct. 2000.

[19] G. L. Lienhart, A. Kugel, and R. Männer, "Using Floating Point Arithmetic on FPGAs to Accelerate Scientific N-Body Simulations," Apr. 2002, pp. 182–191.

[20] N. Nakasato and T. Hamada, "Astrophysical Hydrodynamics Simulations on a Reconfigurable System," Apr. 2005.

[21] T. Hamada and N. Nakasato, "Massively Parallel Processors Generator for Reconfigurable System," Apr. 2005.

[22] T. Hamada, N. Nakasato, and T. Ebisuzaki, "A 236 Gflops Astrophysical Simulation on a Reconfigurable Super-Computer," in *SuperComputing 2005*, Seattle, Nov. 2005.

[23] T. Hamada, T. Fukushige, A. Kawai, and J. Makino, "PROGRAPE-1: A Programmable Special-Purpose Computer for Many-Body Simulations," in *ASSL Vol. 240: Numerical Astrophysics*, 1999, pp. 427–+.

[24] T. A. Cook, H. Kim, and L. Louca, "Hardware acceleration of n-body simulations for galactic dynamics," in *Proc. of SPIE Field Programmable Gate Arrays (FPGAs) for Fast Board Development and Reconfigurable Computing*, Oct. 1995, pp. 115–126.

[25] K. Tsoi, C. Ho, H. Yeung, and P. Leong, "An Arithmetic Library and its Application to the N-body Problem," Apr. 2004, pp. 68–78.

[26] L. Nyland, M. Harris, and J. Prins, "N-body simulations on a GPU," in *Proc of the ACM Workshop on General-Purpose Computation on Graphics Processors*, 2004.

[27] M. Harris, "GPGPU: General-Purpose Computation on GPUs," in *SIGGRAPH 2005 GPGPU COURSE. (http://www.gpgpu.org/s2005/)*, 2005.

[28] T. Hamada and T. Iitaka, "The Chamomile Scheme: An Optimized Algorithm for N-body simulations on Programmable Graphics Processing Units," 2007, astro-ph/0703100 (March 2007).

[29] T. Ito, J. Makino, T. Fukushige, T. Ebisuzaki, S. K. Okumura, and D. Sugimoto, "A Special-Purpose Computer forN-Body Simulations: GRAPE-2A," vol. 45, pp. 339–347, Jun. 1993.

[30] T. Hamada and N. Nakasato, "PGR : A Software Package for Reconfigurable Super-Computing," Aug. 2005, pp. 366–373.

[31] L. Zhuo and V. Prasanna, K, "High Performance Linear Algebra Operations on Reconfigurable Systems," in *Proc of Supercomputing '05*, Nov. 2005, (in CD-ROM).