Hindawi
International Journal of Reconfigurable Computing
Volume 2018, Article ID 6784319, 14 pages
https://doi.org/10.1155/2018/6784319

Research Article

RP-Ring: A Heterogeneous Multi-FPGA Accelerator

Shuaizhi Guo, Tianqi Wang, Linfeng Tao, Teng Tian, Zikun Xiang, and Xi Jin

University of Science and Technology of China, Hefei, China

Correspondence should be addressed to Xi Jin; jinxi@ustc.edu.cn

Received 21 August 2017; Revised 5 November 2017; Accepted 17 January 2018; Published 4 April 2018

Academic Editor: Michael Hübner

To reduce the cost of designing new specialized FPGA boards as direct-summation MOND (Modified Newtonian Dynamics) simulator, we propose a new heterogeneous architecture with existing FPGA boards, which is called RP-ring (reconfigurable processor ring). This design can be expanded conveniently with any available FPGA board and only requires quite low communication bandwidth between FPGA boards. The communication protocol is simple and can be implemented with limited hardware/software resources. In order to avoid overall performance loss caused by the slowest board, we build a mathematical model to decompose workload among FPGAs. The dividing of workload is based on the logic resource, memory access bandwidth, and communication bandwidth of each FPGA chip. Our accelerator can achieve two orders of magnitude speedup compared with CPU implementation.

1. Introduction

N-body simulations have been widely used in scientific and engineering applications. Problems in astrophysics, semiconductor device simulation, molecular dynamics, plasma physics, and fluid mechanics require efficient N-body simulation methods [1]. The problem can be described as follows. The topic gives the initial positions and velocities of N particles, demanding updating their positions and velocities every T time steps. Nonetheless, the size of the N-body simulation (N) is always limited by the available computational resources, and the increasing need for larger system simulations requires more efficient computational methods. So many researchers have been interested in faster algorithms for large-scale particle simulation and invented some efficient algorithms, such as Barnes and Hut algorithms that reduce the computation complexity to $O(N \log N)$ and FMM algorithms which have a computation complexity of $O(N)$ [2]. However, these algorithms use some approximation and are complex to parallelize. Direct-summation N-body algorithm computes the interaction between particles in an accurate way and is quite convenient for parallelization. What is more, direct-summation is a fundamental building-block for other algorithms [3], so a lot of high performance direct-summation computational platforms emerge these years. In the modified Newton dynamics simulation project of Yunnan Observatories, Chinese Academy of Sciences, we want to work out such a platform that can make use of all the resources that we have in lab and to meet the demand for power and performance at the same time.

1.1. Background. Computational solutions for N-body simulation can be categorized as CPU, GPU, ASIC, and FPGA according to the computing unit. Furthermore, these technologies vary in their cost, programming abstraction level, and power consumption [4]. There has been one thorough study on x86-based or power-PC-based tuning [5] and several papers on GPU cluster implementation [2, 6]. The GRAPE ("GRAvity piPE") project built ASIC-based high performance computing solutions for gravitational force calculations where the calculation of particle interactions was calculated by an ASIC chip in the form of a fully pipelined hardwired processor dedicated to gravitational force calculation [7, 8]. Hamada et al. used the Bioler-3 system to implement an FPGA-based gravitational force computing accelerator [4]. These studies have shown that CPUs' performance is limited and ASIC offers no advantages; GPUs are competitive in performance and performance per cost; the performance per Watt figure favoured FPGA [4].

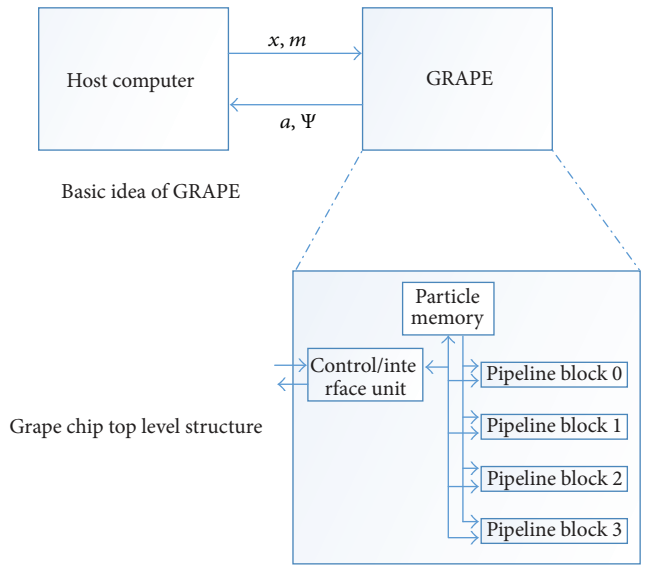


Figure 1: Basic idea of GRAPE and its top-level structure.

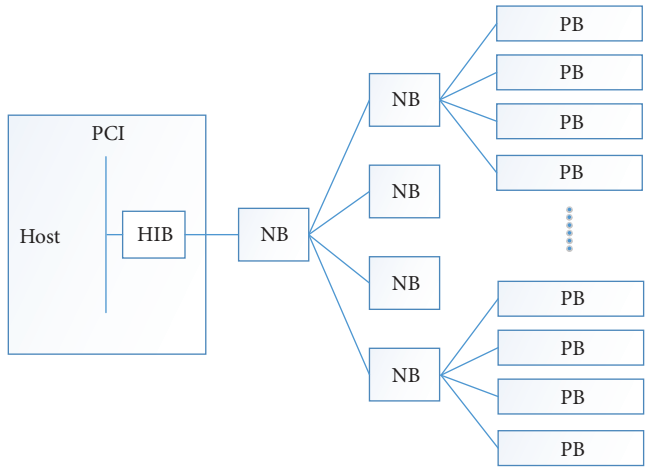


Figure 2: GRAPE computing cluster.

1.2. Related Work and Challenge. Figure 1 shows a basic structure of a hardware accelerated solution for N-body simulations. It consists of a host computer and an acceleration coprocessor for potential calculation. The host computer performs all other calculations, for example, position and velocity upgrade. More specifically, the coprocessor consists of a number of pipelines and the particle information stored in particle memory. The control/interface unit receives instruction from host computer and controls the pipeline to acquire particle information from particle memory to calculate potential [7, 9].

Figure 2 shows how to build a computing cluster with GRAPE. To build a cluster with 16 GRAPE boards, we need one host-interface board (HIB), five network boards (NBs), and 16 processor boards (PBs). Each NB has one uplink and four downlinks. Thus, the 16 PBs are connected to the host computer through two-level tree network of NBs. NB and HIB handle the communication between PB and the host computer [9]. With the increase in the amount of PBs, the demand for NBs increases rapidly. It means that the interconnection overhead of building a large computing cluster is unacceptable and the interconnection problem becomes a challenge of building large computing cluster.

1.3. Motivation. We want to work out an accurate numerical computation method based on MOND theory. MOND

(Modified Newtonian Dynamics) theory is an alternative for the popular Dark Matter (DM) theory, which successfully explains the distribution of force in an astronomical object from observed distributions of baryonic matters [10].

MOND's numerical algorithm is different from traditional N-body simulation's method, so the GRAPE is not suitable for our mission. MOND theory is based on potential calculation and can be described as follows: given the density distribution of baryonic matters $\rho_b(x)$, try to figure out the final potential $\Phi(x)$. The final potential is influenced by the distribution of two kinds of matter: $\rho_b(x)$ and $\rho_{\mathrm{ph}}(x)$, where $\rho_b(x)$ is the density distribution of baryonic matters including stars and gasses and $\rho_{\mathrm{ph}}(x)$ is the density distribution of phantom dark matter, which is the theoretical hypothesis of MOND [11]. The final gravity potential $\Phi(x)$ is given by the classical Poisson equation:

$$\nabla^2\Phi(x) = 4\pi G\left(\rho_b(x) + \rho_{\mathrm{ph}}(x)\right). \tag{1}$$

$\rho_{\mathrm{ph}}(x)$ is given by the $\rho_b(x)$'s potential $\phi(x)$ as in the following equation:

$$\rho_{\mathrm{ph}} = \frac{\nabla\cdot\left[\mu\left(\nabla\phi/a_0\right)\nabla\phi(x)\right]}{4\pi G}. \tag{2}$$

Finally, $\rho_b(x)$'s potential $\phi(x)$ is given by the classical Poisson equation:

$$\nabla^2\phi(x) = 4\pi G\rho_b(x). \tag{3}$$

Different from the traditionally direct-summation N-body algorithm, MOND requires a more time-consuming potential calculation, whose computation complexity is $O(N^2)$.

With limited project budget, we choose to use the FPGA-based direct-summation algorithm. Instead of designing new specialized boards, we reuse existing ones in order to reduce the overhead. The scale of MOND simulation is limited by the available computational resources. A single FPGA chip does not provide enough logical resource, so the multi-FPGA solution seems to be the only choice. Major contributions of our work are as follows:

(1) In order to accelerate direct-summation N-body simulation we propose an extensible heterogeneous multi-FPGA solution called RP-ring (reconfigurable processor ring) to utilize existing multiple different FPGA boards. The experiment shows that our implementation achieved two orders of magnitude speedup compared with high-end CPU implementation.

(2) In order to prevent the slowest board from dragging the overall performance down, we propose a model about how to decompose workload among FPGAs and optimize logic resource allocation. To improve the whole system's performance, this model shall divide workload based on the logic resource, memory bandwidth, and communication bandwidth of each FPGA board and allocate logic resource among potential calculation pipeline, DMA/FIFO, and other modules.

The remainder of this paper is organized as follows: Section 2 presents background information on the algorithm of MOND theory numerical simulation. The following section will present the architecture of RP-ring. Then we will present the model, which guides the decomposition of workload and logic resource's allocation. After that, we will present our hardware implementation result on heterogeneous multi-FPGA and compare it with other implementation on various GPU boards and CPUs as well as ASIC implementation. These results will then be discussed before conclusions are drawn.

2. Direct-Summation Algorithm

MOND numerical simulation is a variant of N-body simulation; the calculation can be described in the following five steps [11]:

(1) With the known baryonic matter distribution $\rho_b(x)$, calculate gravity potential $\phi(x)$ according to (3).

(2) Calculate the phantom dark matter distribution $\rho_{\mathrm{ph}}(x)$ with (2) by finite difference

(3) Solve the Poisson equation (1) to get the final potential $\Phi(x)$.

(4) Calculate the acceleration and velocity with the final potential by $\Phi(x)$ finite difference.

(5) Calculate the location of each particle in the next time step.

Steps (2), (4), and (5) have a computation complexity of $O(N)$, so using CPU to do the tasks serially will not influence the performance. Steps (1) and (3) have a computation complexity of $O(N^2)$, so we focus on accelerating them. In direct-summation algorithm, Steps (1) and (3), which calculate gravity potential, we can use the solution of the Poisson equation:

$$\phi\left(\vec{r}_i\right) = m_i\sum_{j\neq i}\frac{m_j\left(R_{ij}^2 + 1.5\epsilon^2\right)}{\left(R_{ij}^2 + \epsilon^2\right)^{(3/2)}}. \tag{4}$$

Therefore, in the following article we use FPGA to construct potential calculation pipeline and propose the RP-ring solution to build a larger multi-FPGA system. It should be pointed out that this work is not limited to MOND theory numerical simulation. It can be extended conveniently to other direct-summation N-body simulations.

3. Architecture

As (4) shows, the accumulation of the potential acting on each particle by all other particles in the system is mutually independent. We can calculate several particle-pairs' potential simultaneously, so the existing ASIC implementation represented by GRAPE puts several potential calculating pipeline in the chip to find the best degree of parallelism. In Section 1, we have analyzed the existing work and find out their bottleneck. Now we come to the RP-ring.

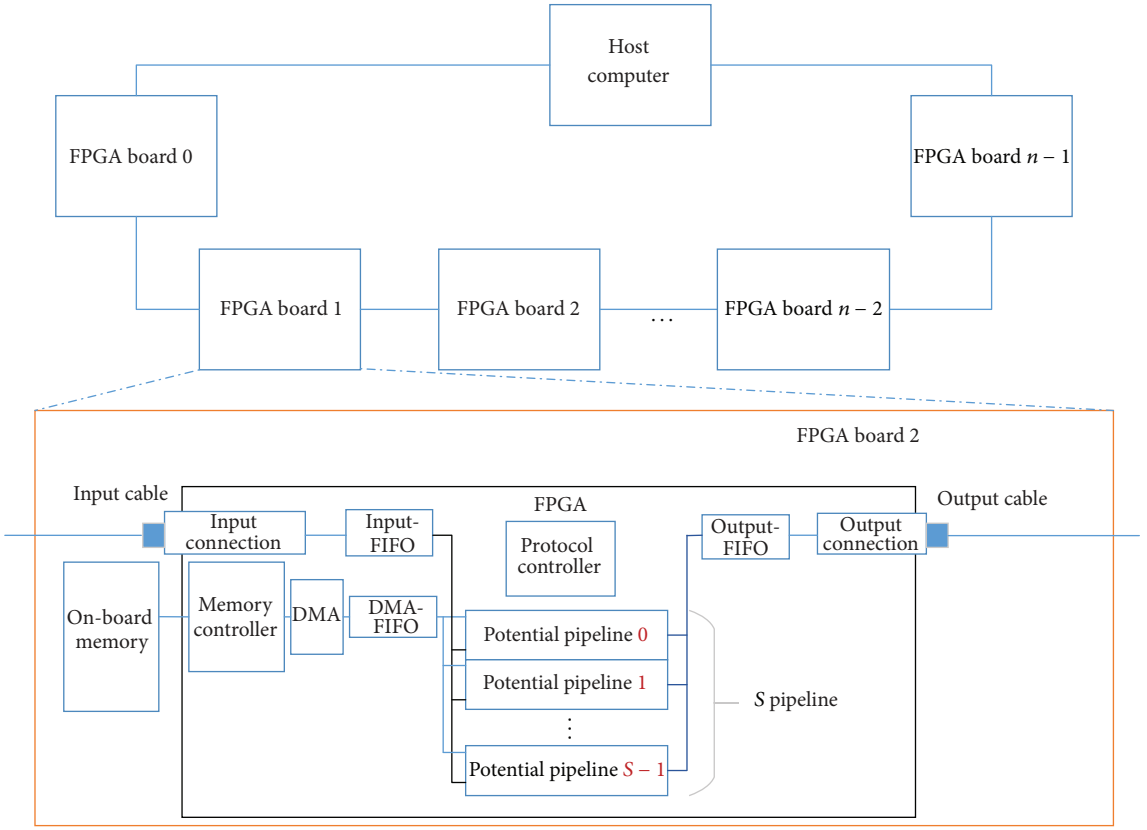


Figure 3: The architecture of RP-ring.

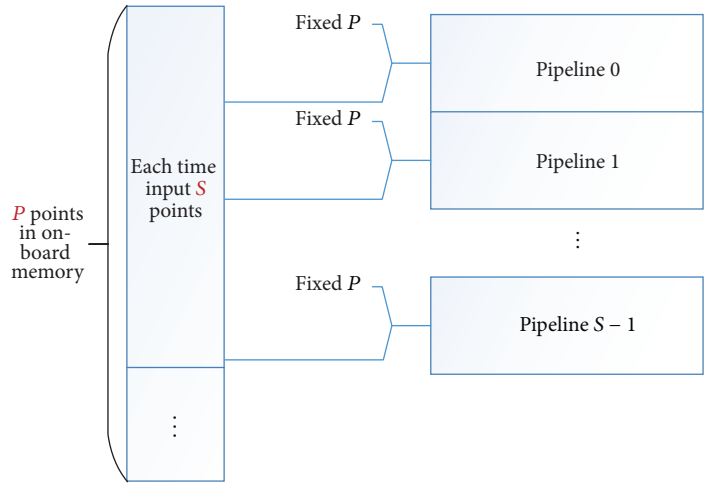


Figure 4: The input of potential pipeline.

3.1. RP-Ring Solution. Figure 3 illustrates a ring network consisting of N FPGA boards and a host computer. Each FPGA board has on-board memory and is connected with previous/next FPGA board with cable. There are s pipelines, DMA, memory controller, and other modules in an FPGA chip. All these modules are controlled by protocol controller. The potential pipelines have two input ports, one connected to Input-FIFO and the other connected to DMA-FIFO. In order to reuse the local particle information, we fix the data from the Input-FIFO, which is received from previous board, and traverse the local particle information as Figure 4 shows. In the following paragraphs, we will explain RP-ring's control flow and data flow.

3.1.1. Control Flow. As shown in Figure 3, each FPGA board's control flow has the following features:

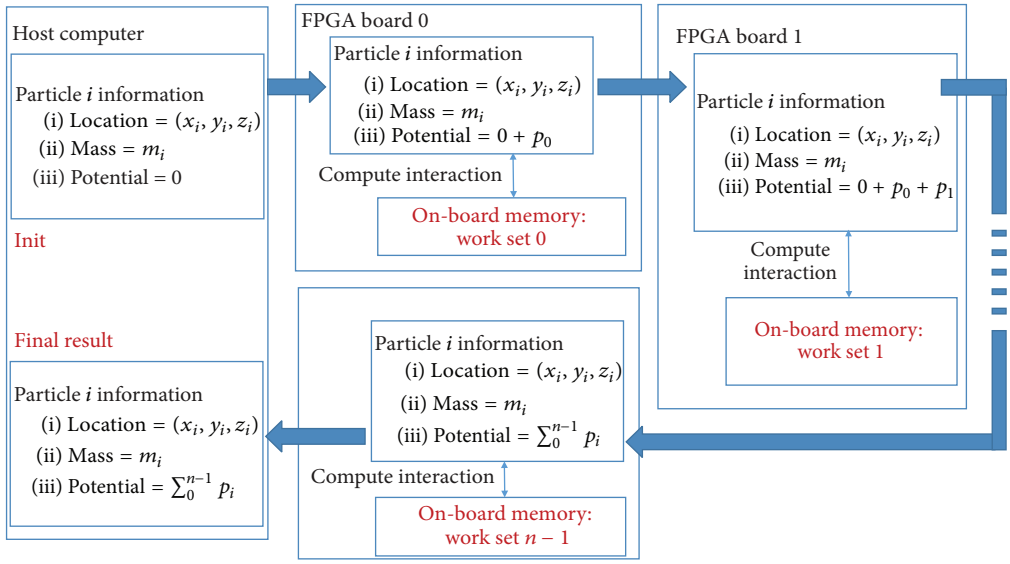


Figure 5: The data flow of RP-ring.

(1) Obtain the results from the previous FPGA board and put data into the Input-FIFO.

(2) There is DMA on-board memory to get local particle information and put it into the DMA-FIFO.

(3) The pipelines gain data from Input-FIFO and DMA-FIFO, calculate the potential, and then write the result into Output-FIFO

(4) Read data from Output-FIFO, and send it to the next FPGA board through output connection.

3.1.2. Data Flow. Figure 5 shows the data flow of RP-ring. The information of the ith particle consists of its location $\mathbf{x} = (x_i, y_i, z_i)$, mass m_i, and potential p_i which is initialized to zero. When the ith particle's information reaches the zero board, the FPGA board calculates the interaction between the particle and the local particles work set 0 and stores the intermediate results $p_0 + 0$ into the potential field. When the ith particle's information reaches the first board, the FPGA board calculates the interaction between the particle and the local particles work set 1 and stores the intermediate results $p_1 + p_0 + 0$ into the potential field. Therefore, when the ith particle's information flows through the whole ring and returns to the host computer, the potential field reaches the final result $\sum_i^{n-1} p_i$. Then the host computer can continue to follow up operation.

The RP-ring solution, we propose in this paper, can avoid the problem we mentioned in Section 1. It implements an extensible heterogeneous multi-FPGA solution. Different from the GRAPE's tree network, RP-ring utilizes ring topology network. Each FPGA board's on-board memory stores a portion of particle information. During the process, each board receives particle information from the previous board, then calculates the interaction with local particle information, and finally sends the result to the next board in the ring network. For each particle's information, when it flows through the whole boards in the ring, the calculation of interaction with all other particles is finished. The advantages of this solution are as follows:

(1) In RP-ring, when the whole working set flows through the ring network once, the calculation of interaction is completed. The amount of data that needs to be transported between FPGA boards is reduced, and the demand for communication bandwidth is also reduced.

(2) The ring network topology is simpler than the tree network in GRAPE cluster. There is no need for additional network board.

(3) The interconnection protocol is quite simple. It requires little overhead to implement protocols no matter the software or hardware. Thus, we can save more resources to construct potential calculation pipeline.

3.2. Potential Pipeline. Potential pipeline is designed based on Poisson equation (4). To make each stage of the pipeline have similar latency, (4) is rewritten as

$$\begin{aligned}
\phi\left(\vec{r}_i\right) &= m_i \sum_{j \neq i} \frac{m_j \left(R_{ij}^2 + 1.5\epsilon^2\right)}{\left(R_{ij}^2 + \epsilon^2\right)^{(3/2)}} \\
&= m_i \sum_{j \neq i} \frac{m_j}{R_{ij}^2 + \epsilon^2} * \frac{R_{ij}^2 + 1.5\epsilon^2}{\sqrt{R_{ij}^2 + \epsilon^2}} \\
&= m_i \sum_{j \neq i} \frac{m_j}{R_{ij}^2 + \epsilon^2} * \frac{R_{ij}^2 + 1.5\epsilon^2}{R_{ij}^2 + \epsilon^2} * \sqrt{R_{ij}^2 + \epsilon^2} \\
&= m_i \sum_{j \neq i} \frac{m_j}{R_{ij}^2 + \epsilon^2} * \sqrt{R_{ij}^2 + \epsilon^2} * \left(\frac{0.5\epsilon^2}{R_{ij}^2 + \epsilon^2} + 1\right).
\end{aligned} \tag{5}$$

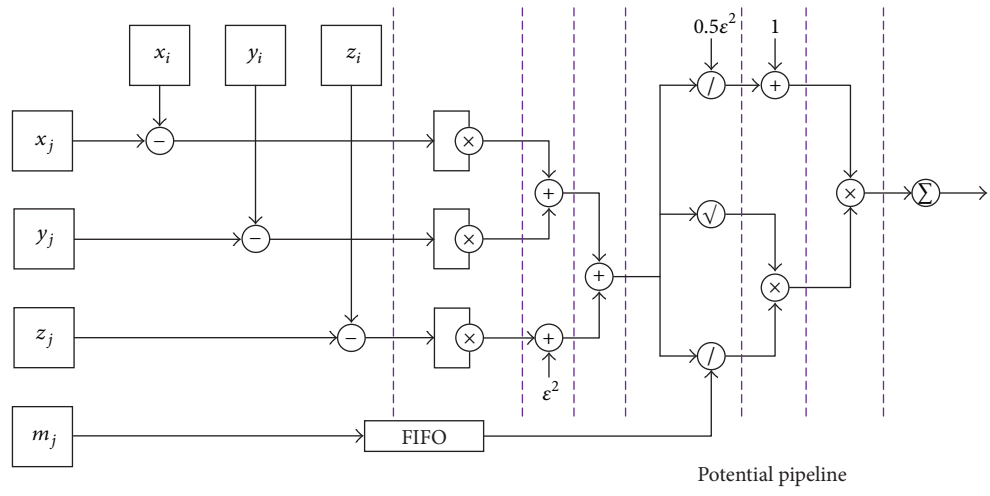


Figure 6: Potential pipeline's detail.

Figure 6 shows the design of potential pipeline. According to the complexity of different operation, the addition, substraction, and multiplication units are set to the same latency cycle as the division and square root units. In our design, x, y, z, m are presented in IEEE 754 floating-point format.

3.3. System Optimization. From the above, the ring network may result in the slowest board dragging the overall performance down, so it is important to balance the time-consumption (T_i) of particle's information flowing through each board. Processing capacity of each board varies, but by decomposing the workload based on their own capacity we can adjust T_i to improve the whole system's performance efficiently. (See suitable work set i in Figure 5.) The following section will discuss this problem in detail with a mathematical model.

4. Model

The purpose of this mathematical model is, given multiple FPGA boards with known parameters, how to decompose the workload among them and choose their parameters of potential calculation pipeline, so that the whole system's maximum throughput can be obtained.

4.1. Symbol Conventions. Assume that the scale of simulation is N, and we have n FPGA boards.

$$N = \sum_{i}^{n} N_i. \tag{6}$$

N_i is the workload assigned to the ith FPGA board. It can be seen from the RP-ring's architecture that each board's processing capacity depends on the number of potential calculation pipelines and their operating frequency. Suppose that the ith FPGA board contains s_i pipelines and their operating frequency is f_i; then the performance of the board P_i has

$$P_i \propto s_i \cdot f_i. \tag{7}$$

In order to maximize the whole system's throughput, we just need to allocate the workload among the FPGA boards in a proportional way according to their processing capacity, so the problem is converted to how to choose s_i and f_i, s.t. $P = \sum_i P_i$ maximum. Additionally, f_i is a function of s_i; that is,

$$f_i = g(s_i). \tag{8}$$

Therefore, in this model, s_i is the only free variable. Finally, the problem is rewritten as

$$\begin{aligned} &N_i = \alpha_i N, \quad \alpha_i = \frac{P_i}{\sum_i P_i} = \frac{s_i f_i}{\sum_i P_i} = \frac{s_i \cdot g(s_i)}{\sum_i s_i \cdot g(s_i)} \\ &\text{st.} \quad P = \sum_i P_i, P_i \text{ maximum.} \end{aligned} \tag{9}$$

4.2. Constraint. Furthermore, there are three constraints in this model:

(1) FPGA logic resource constraint,

(2) memory access bandwidth constraint,

(3) communication bandwidth constraint.

FPGA logic resource constraint: in each FPGA, the logic resource consumption of FIFO, DMA, memory controller, input/output interconnection, and potential pipeline is smaller than the maximum amount of resources that FPGA can provide. Suppose that LUT^i, FF^i, BRAM^i, and DSP^i are the amounts of LUT, Flip-Flop, BRAM, and DSP Slice that the ith FPGA provides; we use vector $\mathbf{R^i}$ to present them, $\mathbf{R^i} = (\text{LUT}^i, \text{FF}^i, \text{BRAM}^i, \text{DSP}^i)$, $\mathbf{R^i_{\text{FIFO}}}$ is the resource consumption of the ith board's FIFO, and so on. Then we have

$$\mathbf{R^i} \geq \mathbf{R^i_{\text{FIFO}}} + \mathbf{R^i_{\text{DMA}}} + \mathbf{R^i_{\text{MemCtrl}}} + \mathbf{R^i_{\text{IOCtrl}}} + \mathbf{R^i_{\text{Pipeline}}}. \tag{10}$$

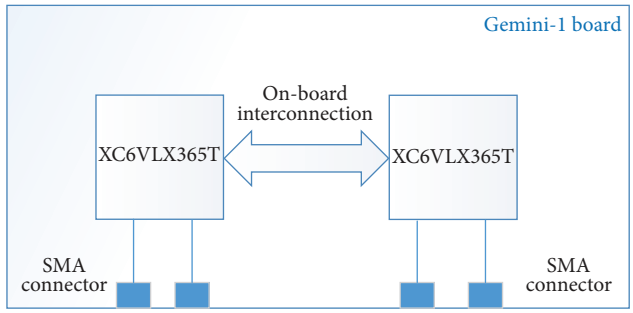


Figure 7: Top-level view of Gemini-1.

Apparently, $\mathbf{R}^{\mathbf{i}}_{\mathrm{FIFO}}$, $\mathbf{R}^{\mathbf{i}}_{\mathrm{DMA}}$, $\mathbf{R}^{\mathbf{i}}_{\mathrm{Pipeline}}$, $\mathbf{R}^{\mathbf{i}}_{\mathrm{MemCtrl}}$, and $\mathbf{R}^{\mathbf{i}}_{\mathrm{IOCtrl}}$ depend on the pipeline needed data bandwidth and can be seen as a function of s_i. That is to say,

$$\begin{aligned}\mathbf{R}^{\mathbf{i}} \geq\ & h^i_{\mathrm{FIFO}}(s_i) + h^i_{\mathrm{DMA}}(s_i) + h^i_{\mathrm{MemCtrl}}(s_i) \\ & + h^i_{\mathrm{IOCtrl}}(s_i) + h^i_{\mathrm{Pipeline}}(s_i).\end{aligned} \tag{11}$$

Memory access bandwidth constraint: the input data of potential pipeline come from previous board's result and local memory's particle information. We fixed the data from the previous board and traverse the local particle information, so half of the potential pipelines' input bandwidth is borne by the memory access bandwidth. That is to say,

$$\mathrm{BW}^i_{\mathrm{Mem\_max}} \geq \frac{1}{2}\mathrm{BW}^i_{\mathrm{pipeline}}. \tag{12}$$

In (12), $\mathrm{BW}^i_{\mathrm{Mem\_max}}$ is the maximum memory access bandwidth of the ith board, and $\mathrm{BW}^i_{\mathrm{pipeline}}$ is the input data bandwidth summation of all the potential pipelines in the ith board. Apparently, $\mathrm{BW}^i_{\mathrm{pipeline}}$ is proportional to the number of potential pipelines. That is to say, $\mathrm{BW}^i_{\mathrm{pipeline}} \propto s_i$, so (12) can be written as

$$\mathrm{BW}^i_{\mathrm{Mem\_max}} \geq \frac{1}{2}\mathrm{BW}^i_{\mathrm{pipeline}} = \frac{1}{2}k \cdot s_i, \tag{13}$$

where k is a constant.

Communication bandwidth constraint: as described in memory access bandwidth constraint, the ith board's communication bandwidth between previous board and next board should be greater than one over N_i of the half of $\mathrm{BW}^i_{\mathrm{pipeline}}$. That is to say,

$$\begin{aligned}\mathrm{BW}^i_{\mathrm{Input\_max}} &\geq \frac{1}{N_i} \cdot \frac{1}{2}\mathrm{BW}^i_{\mathrm{pipeline}} = \frac{1}{2N_i}k \cdot s_i \\ \mathrm{BW}^i_{\mathrm{Output\_max}} &\geq \frac{1}{N_i} \cdot \frac{1}{2}\mathrm{BW}^i_{\mathrm{pipeline}} = \frac{1}{2N_i}k \cdot s_i\end{aligned} \tag{14}$$

In conclusion, based on the target function and the three constraints, when the parameter of the boards and the needed functional relation are given, solving the optimization problem can guide us on how to decompose the work load among the boards and how to choose their parameters of potential calculation pipeline.

5. Implementation

In this section, we will demonstrate our implementation under RP-ring solution and its performance parameters in our MOND theory numerical simulation project.

Table 1 shows our existing boards and their parameters. The experiment demonstrates that under RP-ring solution, we can,

(1) based on the boards' feature, select software or hardware to implement the interconnection protocol,

(2) according to the boards' resource, choose different interconnection media.

Thus, this solution has good flexibility and scalability and is compatible with heterogeneous multi-FPGA.

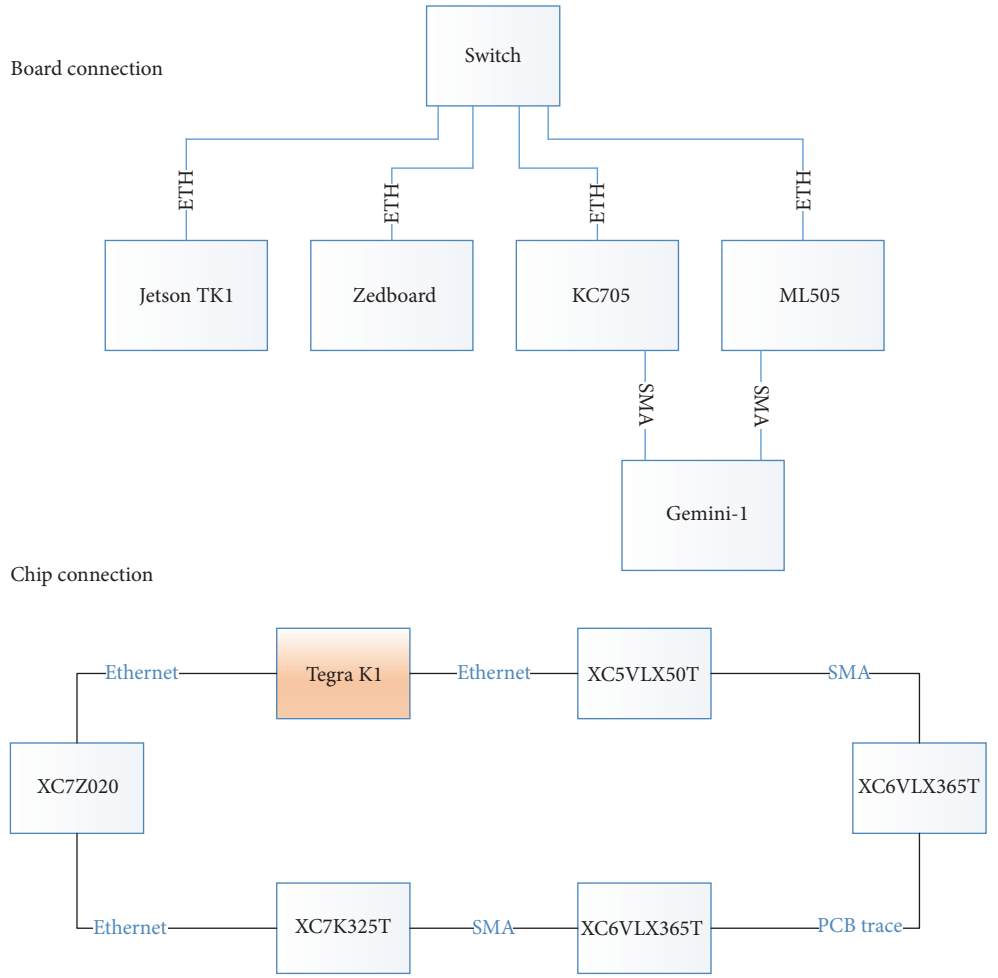
In Table 1, Jetson-TK1 is a NVIDIA Application Processor board, which is used as host computer. Zedboard, KC705, and XUPV5 are Xilinx Evaluation Kits for Zynq-7000, Kintex-7, and Virtex-5. Gemini-1 is our design FPGA board for prototyping. Figure 7 shows the top-level structure of Gemini-1. Gemini-1 has two pieces of XC6VLX365T Virtex-6 FPGA; they are connected through PCB trace. Each Virtex-6 FPGA chip has SMA connector to transport data.

5.1. Topology. Figure 8 shows connection between the boards and connection between chips. In the ring network, Tegra K1 is used as host computer, and XC7Z020, XC7K325T, XC5VLX110T, and XC5VLX365T are connected through ethernet, SMA cable, or PCB trace. Figure 9 is the picture of real product.

5.2. Data Structure. The provisions of the RP-ring's particle information format are as shown in Figure 10. The location, mass, and potential field store the particle's three-dimensional coordinates, mass, and the provisional result of potential, as well as the field of Tag record FPGA boards that the particle information has passed. Once the information passes an FPGA board, the corresponding position in Tag filed is set.

Table 1: The parameters of the boards.

Board	Main chip	Hard CPU	Logic cells	Flip flop	Block RAM	DSP slice	Memory access bandwidth (theoretical)	Connection
Zedboard	XC7Z020	Cortex A9 Dual Core	85,000	106,400	560 Kb	220	8.5 GB/s	Ethernet
KC705	XC7K325T	No	326,080	407,600	4000 Kb	840	12.8 GB/s	Ethernet and SMA
XUPV5	XC5VLX110T	No	46,080	28,800	1728 Kb	48	3.2 GB/s	Ethernet and SMA
Gemini-1	XC6VLX365T ×2	No	364,032 each	455,040 each	14,976 Kb each	576 each	12.8 GB/s each	SMA
Jetson-TK1 (host computer)	Tegra K1	Cortex A15 Quard Core	—	—	—	—	—	Ethernet



(i) UTP: ethernet
(ii) SMA: subminiature version A

Figure 8: The experiment's connection.

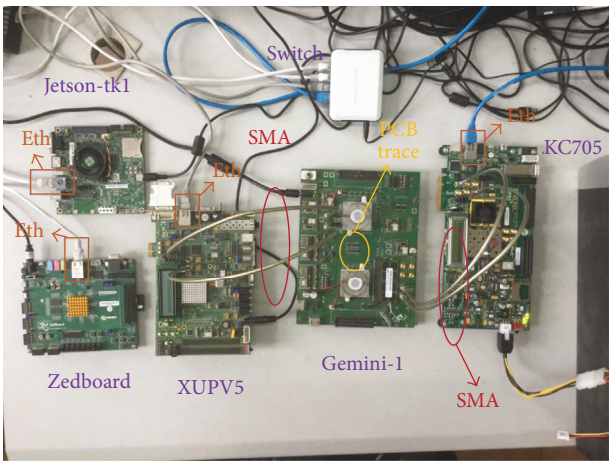


Figure 9: Boards.

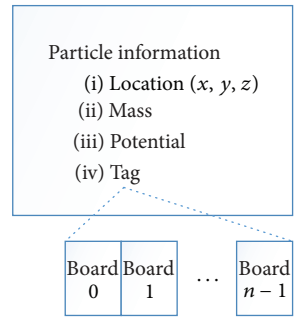


Figure 10: The data structure of particle information.

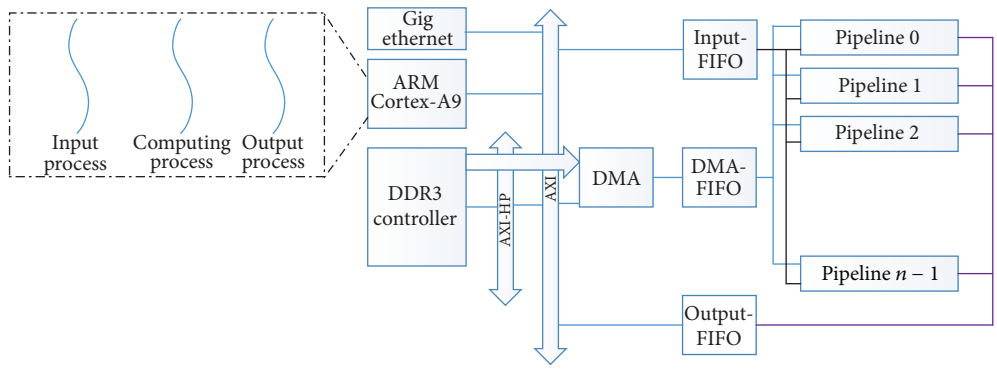


Figure 11: Protocol's software implementation.

When all of the bits are set, the potential field contains the final result.

5.3. Protocol

5.3.1. Software Implementation. For the FPGA with integrated CPU, like Zynq-7000, the interconnection protocol can be implemented with software as Figure 11 shows. Figure 11 shows its system architecture: multiple pipelines are instantiated in FPGA; two input ports of each pipeline are connected to Input-FIFO and DMA-FIFO; the output port of each pipeline is connected to Output-FIFO. The CPU creates three processes: Input Process, Computing Process, and Output Process.

Input Process. Control Gigabyte Ethernet receives particle information from previous board and stores it to Input-FIFO. Moreover, Input Process handles the situation of retransmission, Input-FIFO's fullness, and so on.

Computing Process. Control DMA traverses local particle information from DDR3. Control pipelines compute the potential based on the particle information in Input-FIFO and local information in DMA-FIFO and then write the result to Output-FIFO.

Output Process. Control Gigabyte Ethernet Sends the result in Output-FIFO to the next board. Moreover, Output Process handles the situation of retransmission, next board's Input-FIFO's fullness, and so on.

5.3.2. Hardware Implementation. For the FPGA without integrated CPU, like XC7K325T, the interconnection protocol can be implemented with hardware as Figure 12 shows. Because the interconnection media can be ethernet cable, SMA cable, or PCB trace, the input/output connection can be ethernet controller, SMA controller, or SelectIO controller. In Figure 12, a protocol FSM replaces the role of CPU. It controls the input connection, receives the data from previous board, stores the message in the Input-FIFO, and controls the DMA traverse local data, and pipelines finish the calculation, control the output connection, and send the result in Output-FIFO to the next board.

6. Experimental Result

6.1. Logical Resource Consumption. Table 2 shows the resource utilization of each FPGA board. Particularly, Zynq FPGA has hard DDR3 controller and Gigabit Ethernet controller, and Virtex-5 FPGA has hard DDR2 and MAC. Thus, these modules do not require extra logic resource. Virtex-5 does not have existing DMA IP, so we design a simplified version.

6.2. Communication Bandwidth Consumption. In order to measure the communication bandwidth consumption between boards, we add counters to record the data traffic on the interconnection. Figure 13 shows the counters in the system and the interconnection's notation. These counters will work after each packet passes through the path. At the

Table 2: Resource consumption.

	Gemini-1				KC705			
	LUT	REG	BRAM	DSP	LUT	REG	BRAM	DSP
FIFO	265	129	4	0	265	112	4	0
DMA	2753	4078	8	0	1322	1680	8	0
IN/OUT	450	530	0	0	1479	2082	0	0
MEM	5889	5882	0	0	14016	9019	2	0
PIPELINE	209493	301565	5	480	185509	273149	5	736
TOTAL	*227520*	*455040*	*416*	*576*	*203800*	*407600*	*445*	*840*
	Zedboard				XUPV5			
	LUT	REG	BRAM	DSP	LUT	REG	BRAM	DSP
FIFO	265	112	4	0	47	57	4	0
DMA	0	0	0	0	777	516	0	0
IN/OUT	0	0	0	0	2226	2307	0	0
MEM	0	0	0	0	0	0	0	0
PIPELINE	45014	80403	2	171	10552	15945	0	64
TOTAL	*53200*	*106400*	*140*	*220*	*69120*	*69120*	*128*	*64*

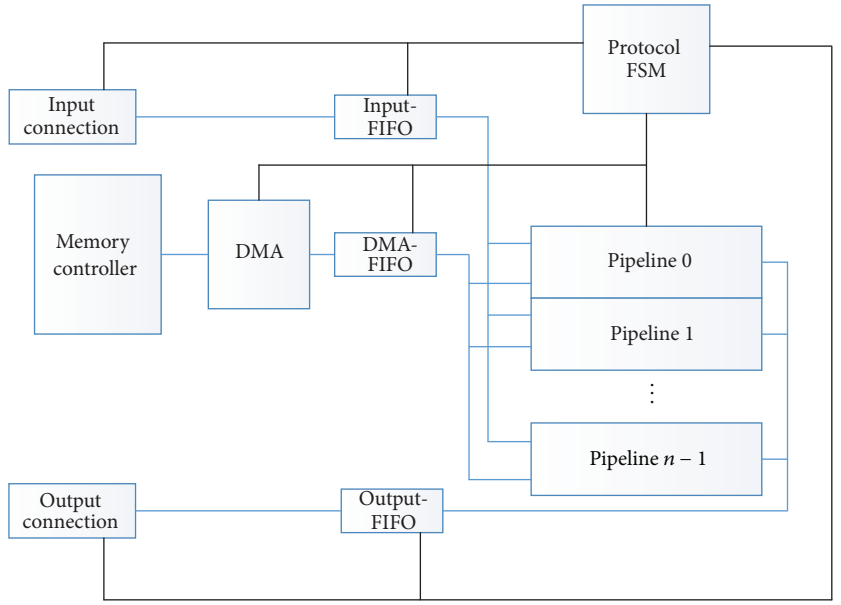


Figure 12: Protocol's hardware implementation.

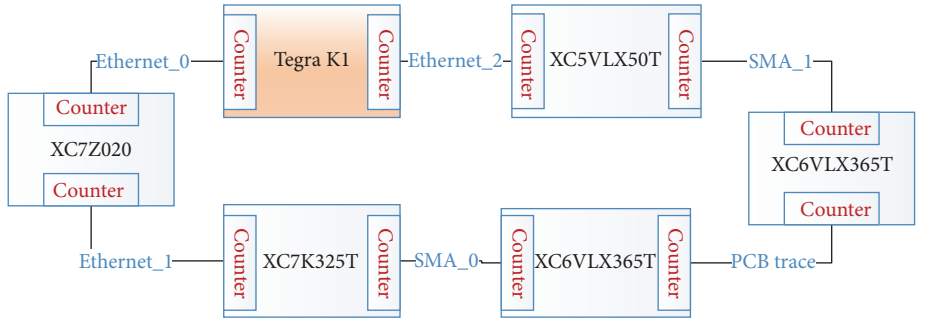


Figure 13: Counters' location and interconnection's notation.

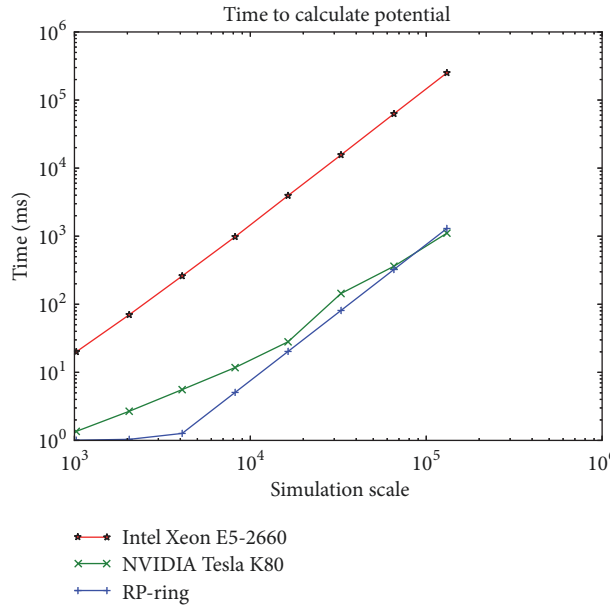


Figure 14: Software performance comparison.

Table 3: Communication bandwidth consumption.

Interconnection	Data traffic (MB)	Time (ms)	Bandwidth measured (MB/s)	Bandwidth theory (MB/s)
Ethernet 0	15.733	1297.2	12.128	1000
Ethernet 1	15.641	1297.2	12.057	1000
Ethernet 2	15.847	1297.2	12.216	1000
SMA 0	2.911	1297.2	2.244	3125
SMA 1	2.926	1297.2	2.256	3125
PCB trace	2.903	1297.2	2.238	6400

Table 4: The number of potential pipelines.

Board	Pipeline	Frequency	GFlops	MPair/s
Zedboard	8	200 MHz	32.9	866.6
KC705	32	344.9 MHz	227.2	5977.8
XUPV5	2	224.3 MHz	9.234	242.9
Gemini-1	32×2	266.1 MHz	$175.3 * 2$	$4612.0 * 2$
Total (theory)	106	—	620.1	16311.3
Total (experiment)	106	—	503.5	13244.7

end of the whole experiment, we can read out the value of each counter and divide it by the time of calculation. By this way, we can confirm that our data has no loss in the communication and give out the specific throughput bandwidth of each FPGA boards.

Table 3 shows the communication bandwidth consumption result during calculating 131,072 particles' potential, the time of which is 1297.113 ms. Because of the solution's ring topologies, the consumption of communication bandwidth is very low.

6.3. Performance Comparison. Table 4 shows the number of potential pipelines in each FPGA board, operation frequency, and their performance. The conversion between GFlops and MPair/s is based on Atsushi Kawai's work [16]. Based on the above results, we calculate the whole system's theoretical parameters. Finally, we list the system's experimental result.

6.3.1. Comparison with Software. We choose CPU and GPU solutions as the control groups of our work. Fabian's RAMSES code is a widely used method for MOND simulation [11]. Therefore, we choose their method as the reference software implementation and use Intel Xeon E5-2660 to run different scale test. Based on RAMSES's QUMOND method and Nitin's direct N-body kernels to simulate the same work set on NVIDIA's Tesla K80 GPU, Figure 14 shows the time taken to calculate potential in different platform. The benefit from the structure of FPGA is that RP-ring is faster than other solutions at the beginning. In the right-end, the NVIDIA results appear to be better than the RP-ring, because as the number of particles increases, our platform is approaching its theoretical maximum performance (503.5 of 620.1 GFlops), and Tesla K80 has not reached its limits. As for the CPU solution, it uses much more time compared to RP-ring and GPU, limited by its poor capability of parallel computing. When simulating a system with 131072 particles, our work is 193 times faster than Xeon E5-2660 CPU and can achieve similar performance to Tesla K80.

Table 5: Hardware performance comparison.

Implement	Main chip	GFlops
GRAPE-4 cluster [7]	ASIC	1080
GRAPE-6 cluster [9]	ASIC	1349
GRAPE-8 board [8]	ASIC	960
Lienhart et al.'s work [12]	FPGA	3.9
Spurzem et al.'s work [13]	FPGA	4.3
Hamada et al.'s Bioler-3 [4]	FPGA	324.2
GPU cluster [14]	GPU	781
Sozzo et al.'s work [15]	FPGA	46.55
Our Work	FPGA	503.5

6.3.2. Comparison with Hardware. Table 4 shows a number of hardware implementation details of N-body simulation. Junichiro's GRAPE-4 cluster uses 1692 pipeline and achieves 1.08 TFlops [7]. And he builds a compute cluster with GRAPE-6 chips whose peak performance is 1.349 TFlops [9]. Furthermore, his GRAPE-8 achieves 960 GFlops with just one board [8]. Reference [14] showed a solution with GPU whose performance is 781 GFlops.

Some FPGA solutions are also listed in the Table 5. Lienhart et al. use FPGA to achieve 3.9 GFlops' performance [12]. Spurzem et al.'s solution has the performance of 4.3 GFlops [13], Hamada et al.'s work reaches 324.2 GFlops by a board with 5 FPGA chips [4], and Sozzo et al.'s work makes a solution with 46.55 GFlops' performance [15].

7. Conclusion and Discussion

In this paper, we proposed an extensible solution: RP-ring, which is used for heterogeneous multi-FPGA-based direct-summation N-body simulation, and a model to decompose workload among each FPGA. RP-ring tries to use existing FPGA boards rather than designing new specialized boards to reduce cost. The solution can be expanded conveniently with any heterogeneous FPGA boards and the communication bandwidth requirement is quite low, so that the communication protocol could be designed to be simple and consume few resource. The model considers the constraint of FPGA's logic resource, memory access bandwidth, and communication bandwidth to divide workload reasonably and optimize the whole system's performance. We also build a heterogeneous multi-FPGA system based on RP-ring and use it for MOND theory's numerical simulation. The experimental result shows that the low cost multi-FPGA system is 193 times faster than high-end CPU implementation and achieves similar performance to high performance GPU.

Disclosure

An earlier version of this work was presented as a poster at 2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was sponsored by Huawei Innovation Research Program (YB2015090102); support from Huawei Technologies Co., Ltd., is gratefully acknowledged.

References

[1] S. Bhatt, M. Chen, C.-Y. Lin et al., "Abstractions for parallel N-body simulations," in *Proceedings of the Scalable High Performance Computing Conference (SHPCC-92)*, pp. 38–45, IEEE.

[2] T. Hamada, T. Narumi, R. Yokota, K. Yasuoka, K. Nitadori, and M. Taiji, "42 TFlops hierarchical N-body simulations on GPUs with applications in both astrophysics and turbulence," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, (SC '09)*, New York, NY, USA, November 2009.

[3] S. Harfst, A. Gualandris, D. Merritt, R. Spurzem, S. P. Zwart, and P. Berczik, "Performance analysis of direct N-body algorithms on special-purpose supercomputers," *New Astronomy*, vol. 12, no. 5, pp. 357–377, 2007.

[4] T. Hamada, K. Benkrid, K. Nitadori, and M. Taiji, "A comparative study on ASIC, FPGAs, GPUs and general purpose processors in the $O(N^2)$ gravitational N-body simulation," in *Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems (AHS '09)*, pp. 447–452, San Francisco, Calif, USA, July 2009.

[5] N. Arora, A. Shringarpure, and R. W. Vuduc, "Direct n-body kernels for multicore platforms," in *Proceedings of the 38th International Conference on Parallel Processing, ICPP-2009*, pp. 379–387, Austria, September 2009.

[6] I. Zecena, M. Burtscher, T. Jin, and Z. Zong, "Evaluating the performance and energy efficiency of n-body codes on multicore CPUs and GPUs," in *Proceedings of the 2013 IEEE 32nd International Performance Computing and Communications Conference, IPCCC 2013*, USA, December 2013.

[7] J. Making, M. Taiji, T. Ebisuzaki, and D. Sugimoto, "Grape-4: A massively parallel special-purpose computer for collisional n-body simulations," *The Astrophysical Journal* , vol. 480, no. 1, pp. 432–446, 1997.

[8] J. Makino and H. Daisaka, "GRAPE-8—an accelerator for gravitational N-body simulation with 20.5Gflops/W performance," in *Proceedings of the 24th International Conference for High Performance Computing, Networking, Storage and Analysis (SC '12)*, Salt Lake City, Utah, USA, November 2012.

[9] J. Makino, T. Fukushige, and M. Koga, "A 1.349 Tflops simulation of black holes in a galactic center on GRAPE-6," in *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, pp. 43-43, Dallas, Tex, USA, November 2000.

[10] B. Famaey and S. S. McGaugh, "Modified newtonian dynamics (MOND): observational phenomenology and relativistic extensions," *Living Reviews in Relativity*, vol. 15, article 10, 2012.

[11] F. Lüghausen, B. Famaey, and P. Kroupa, "Phantom of RAMSES (POR): a new Milgromian dynamics N-body code," *Canadian Journal of Physics*, vol. 93, no. 2, pp. 232–241, 2014.

[12] G. Lienhart, A. Kugel, and R. Männer, "Using floating-point arithmetic on FPGAS to accelerate scientific N-Body simulations," in *Proceedings of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 2002*, pp. 182–194, USA, April 2002.

[13] R. Spurzem, P. Berczik, G. Marcus et al., "Accelerating astrophysical particle simulations with programmable hardware

(FPGA and GPU)," *Computer Science - Research and Development*, vol. 23, no. 3-4, pp. 231–239, 2009.

[14] A. Castellĺő, R. Mayo, and J. Planas, "Exploiting Task-Parallelism on GPU Clusters via OmpSs and rCUDA Virtualization Trustcom/BigDataSE/ISPA," in *Proceedings of the IEEE Trustcom/BigDataSE/ISPA*, pp. 160–165, 2015.

[15] E. D. Sozzo, L. D. Tucci, and M. D. Santambrogio, "A highly scalable and efficient parallel design of N-body simulation on FPGA," in *Proceedings of the 31st IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2017*, pp. 241–246, USA, June 2017.

[16] A. Kawai and T. Fukushige, "$158/GFLOPS astrophysical N-body simulation with reconfigurable add-in card and hierarchical tree algorithm," in *Proceedings of the ACM/IEEE Conference on Supercomputing (SC '06)*, 2006.