Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2016, Article ID 4592780, 10 pages
http://dx.doi.org/10.1155/2016/4592780

Research Article

An Accelerating Solution for N-Body MOND Simulation with FPGA-SoC

Bo Peng,[1,2] Tianqi Wang,[1,2] Xi Jin,[1,2] and Chuanjun Wang[3,4,5]

[1]*Key Laboratory of Strongly Coupled Quantum Matter Physics, Chinese Academy of Sciences, School of Physical Sciences, University of Science and Technology of China, Hefei, Anhui 230026, China*
[2]*Hefei Branch Center of National ASIC Design Engineering Technology Research Center, Institute of Advanced Technology, University of Science and Technology of China, Hefei, Anhui 230600, China*
[3]*Yunnan Observatories, Chinese Academy of Sciences, Kunming 650216, China*
[4]*Key Laboratory for the Structure and Evolution of the Celestial Objects, Chinese Academy of Sciences, Kunming 650216, China*
[5]*University of Chinese Academy of Sciences, Beijing 100049, China*

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

Received 21 December 2015; Accepted 17 May 2016

Academic Editor: Michael Hübner

As a modified-gravity proposal to handle the dark matter problem on galactic scales, Modified Newtonian Dynamics (MOND) has shown a great success. However, the N-body MOND simulation is quite challenged by its computation complexity, which appeals to acceleration of the simulation calculation. In this paper, we present a highly integrated accelerating solution for N-body MOND simulations. By using the FPGA-SoC, which integrates both FPGA and SoC (system on chip) in one chip, our solution exhibits potentials for better performance, higher integration, and lower power consumption. To handle the calculation bottleneck of potential summation, on one hand, we develop a strategy to simplify the pipeline, in which the square calculation task is conducted by the DSP48E1 of Xilinx 7 series FPGAs, so as to reduce the logic resource utilization of each pipeline; on the other hand, advantages of particle-mesh scheme are taken to overcome the bottleneck on bandwidth. Our experiment results show that 2 more pipelines can be integrated in Zynq-7020 FPGA-SoC with the simplified pipeline, and the bandwidth requirement is reduced significantly. Furthermore, our accelerating solution has a full range of advantages over different processors. Compared with GPU, our work is about 10 times better in performance per watt and 50% better in performance per cost.

1. Introduction

Modified Newtonian Dynamics (MOND) is an alternative proposal to popular dark matter (DM) theory, accounting for the missing mass problem in astrophysics. To study the outskirts of disk galaxies, N-body MOND simulation is an essential task, namely, to simulate the dynamic evolution of an astronomical system consisting of multiple celestial objects (denoted as N-body), where the interacting philosophy of each pair object obeys the MOND proposal.

Gravitational N-body problem is traditionally explored with computer simulations, which involves massive nonlinear calculations for the MOND potentials. Yet it has been limited due to both the nonlinearity and the large N-scale in a long time. In 2010, Mordehai Milgrom proposed a new formulation of MOND, named quasi-linear MOND (QUMOND) [1]. Based on the QUMOND formulation, the calculation of MOND potential contains 3 steps: the first and the last are potential calculations similar to classical gravity calculations, whereas the second step is to calculate the phantom dark matter (PDM) distribution. The drawback of nonlinearity is ameliorated by QUMOND, while the large N-scale is still a big challenge, which brings a tremendous computation complexity $\mathcal{O}(N^2)$. Besides the development of algorithms optimizing the computation complexity, the hardware acceleration of the arithmetic calculation unit is another effective solution to accelerate the N-body MOND simulation and thus is the subject of the present paper.

Hardware accelerators for N-body MOND simulation abound. In early 1990s, based on the methodology of

application specific integrated circuit (ASIC), a series of specific processors, named GRAPE (GRAvity PipE), were proposed for the calculation of particle-particle interaction in the N-body problem. From 1991 to 2012, GRAPE experienced a development from version 1 to version 8 [2]. A single GRAPE-8 chip integrates 48 pipeline processors and provides a peak performance of 480 Gflops (12000 Mpairs/s) in total. Besides ASIC like GRAPE, FPGA and GPU are also popular choices to accelerate the N-body simulation. Similar with ASIC, the main idea of FPGA-based accelerators is to customize parallel pipelines utilizing programmable gate arrays; nevertheless GPU-based accelerators implement a calculating parallelization through thousands of cores. In 2006, Kawai and Fukushige proposed two FPGA-based add-in cards [3], which were applied to astrophysical N-body simulations with the hierarchical tree algorithm; they achieved a performance of 80.9 Gflops (2128 Mpairs/s) with 16 133 MHz pipelines being used. In 2007, Portegies Zwart et al. proposed a GPU accelerator for $\mathcal{O}(N^2)$ gravitational N-body simulations [4]; their results indicated that the GeForce 8800GTX GPU had a 10-time speedup compared to Intel Xeon CPUs.

During the simulation, parameters of these N objects are correlated. Therefore in most situations where ASIC, FPGA, or GPU is utilized, the accelerator is implemented as an add-in card, relying on a host computer to deal with the data dispatching. However, the potential calculation consumes most of the simulation time, which means that processors in the host computer are mostly idle during the simulation. Wang et al. provided statistical results of a CPU-GPU hybrid parallel strategy for an $\mathcal{O}(N \log N)$ cosmological simulation; they revealed that, in nearly 75 percents of the total time, CPUs were in a waiting state [5]. Moreover, this card-host structure usually requires extra energy and spaces, which leads to a big waste in energy, money, and space. Especially, in N-body MOND simulation, the two-step potential calculation brings a double waste with the same simulation scale. These disadvantages motivate us to turn to FPGA-SoC, which combines the embedded low-power processor and FPGA, exhibiting a high integration.

In this paper, by utilizing the FPGA-SoC, we propose a highly integrated accelerating solution for N-body MOND simulations. Besides data dispatching, the embedded processor is also responsible for the PDM distribution calculation in the 2nd step. Considering that the number of pipelines an FPGA integrates directly affects the accelerator performance, a modification is made to the typical potential summation pipeline, that is, to make full use of the DSP48E1 in Xilinx 7 series FPGA, so as to reduce the logic resource occupation of each pipeline. What is more, we optimize the data flow from memory to pipelines based on the particle-mesh scheme, which contributes to a reduction of the bandwidth requirement and improves the performance of our solution significantly. At last, we test our solution in the low-cost Zynq-7020 all programmable SoC.

The rest of this paper is organized as follows. The background of MOND and N-body MOND simulation is briefly introduced in the next section. We discuss our motivation and contribution in Section 3. Then we describe the particle-mesh scheme and make an illustration of the system architecture of our solution in Section 4. Experimental results are presented in Section 5. Finally, we conclude this paper in Section 6.

2. N-Body MOND Simulation

2.1. Modified Newtonian Dynamics. Modified Newtonian Dynamics (MOND) can be interpreted as a modification to the law of gravity. It is an alternative proposal to popular dark matter (DM) theory, accounting for the missing mass problem in astrophysics. Both MOND and DM elegantly fit the rotation curve of spiral galaxies. However, there exist some challenges for the DM-based model; the biggest one is that the tight scaling relations cannot be understood [6], while MOND provides a good explanation to it, as well as details of the rotation curve [7].

Modified Newtonian Dynamics (MOND) was firstly proposed by Milgrom in 1983 [7]. The Milgrom proposal is as follows: the acceleration g is Newtonian g_N in the strong gravity field but begins to deviate from it around a critical acceleration a_0 and converges to the weak field limit:

$$g = \sqrt{g_N a_0} \quad \text{when } g \ll a_0. \tag{1}$$

Here $a_0 \cong 10^{-10}\,\mathrm{m/s^2}$ is Milgromian characteristic acceleration constant, g denotes the Milgromian gravitational acceleration, and g_N indicates the Newtonian one.

In conventional Newtonian dynamics, we have the classical Poisson equation:

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

where $\phi(x)$ is the Newtonian potential and $\rho_b(x)$ is the density distribution of baryonic matters including the star and the gas. Linear equation (2) can be calculated by a typical Poisson solver. However, when describing MOND in this form, it leads to nonlinear generalization of the Newtonian Poisson equation [8]:

$$\nabla\left[\mu\left(\frac{|\nabla\Phi(x)|}{a_0}\right)\nabla\Phi(x)\right] = 4\pi G\rho_b(x), \tag{3}$$

where $\Phi(x)$ is the distribution of MOND potential and the μ-function is an interpolating function, representing the MOND gravity in the transitional zone from Newtonian to the weak field limit, and is usually adopted as follows:

$$\mu(y) = \frac{\sqrt{1+4/y}}{2} - 1. \tag{4}$$

Equation (3) is hard to solve due to its nonlinearity.

Therefore, in the weak field limit, we have the so-called quasi-linear MOND (QUMOND) [1], as written in

$$\nabla^2\Phi(x) = 4\pi G\rho_b(x) + \nabla\cdot\left[\mu\left(\frac{|\nabla\phi|}{a_0}\right)\nabla\phi(x)\right]. \tag{5}$$

Here $\phi(x)$ indicates the Newtonian potential.

For simplification, denote

$$\rho_{\mathrm{ph}}(x) = \frac{\nabla\cdot\left[\mu\left(|\nabla\phi|/a_0\right)\nabla\phi(x)\right]}{4\pi G}, \tag{6}$$

and (5) can be rewritten as

$$\nabla^2\Phi(x) = 4\pi G\left(\rho_b(x) + \rho_{\mathrm{ph}}(x)\right). \quad (7)$$

Analogous to dark matter, the so-called phantom dark matter (PDM) is introduced, and $\rho_{\mathrm{ph}}(x)$ can be interpreted as its density distribution [9]. MOND potential (7) has a similar formulation like (2), and it can be regarded as a modification to the Newtonian potential. Both MOND potential (7) and Newtonian potential (2) can be solved by existing Poisson solvers.

2.2. N-Body MOND Simulation. An N-body simulation problem is to study the interactions of each pair object and further simulate the dynamic evolution of an astronomical system, which consists of multiple celestial objects (denoted as N-body). N-body MOND simulation is to calculate the changes of particle properties in a galaxy with time under MOND; namely, the interacting philosophy of each pair object obeys the MOND proposal. These properties include potential, velocity, and position, to name but a few. In this paper, we focus on the potential distribution $\Phi(x)$ at a fixed time, with the known baryonic matter distribution $\rho_b(x)$. By using the evaluated $\Phi(x)$, the current acceleration can be worked out according to $g = -\nabla\Phi$. Particle properties at next time can be further calculated. For an individual particle, the MOND potential calculation comprises three steps:

(1) with the known baryonic matter distribution $\rho_b(x)$, calculating Newtonian potential according to (2);

(2) calculating the phantom dark matter (PDM) distribution with (6);

(3) solving modified Poisson equation (7) to get the final MOND potential.

The N-body MOND simulation has a double potential computation compared to the typical Newtonian simulation, which challenges the performance of accelerators. Noticing that the formulation of (2) is similar to that of (7), thus the first and third steps can be conducted by the same accelerator.

3. Motivation and Contribution

Table 1 provides the practical time cost of MOND calculation on the Intel i5 processor; the calculation is conducted through 3 steps described in Section 2. With the increase of the simulation scale N, the computation time of MOND simulation reveals a quadratical increase. Thus it is both crucial and necessary to accelerate N-body MOND simulations. What is more, Table 1 shows that the time consumption of the 2nd step is just a minority, while the bottleneck of MOND simulation is the potential calculation, both Newtonian and MOND. This fact enables our keystone that accelerations are only applied to potential calculations of steps (1) and (3), whereas the 2nd step is conducted by the common processor, other than an specific accelerator.

Besides pursuing a high calculation speed, there also exist other essential considerations, like the power consumption, the economic cost, and so on. Most existing accelerating solutions utilize a card-host scheme; the accelerator is implemented as an add-in card, relying on external host processors to deal with the data dispatching. However, as mentioned in Section 1, external host processors are mostly idle during the simulation. Hence this card-host scheme would lead to a big waste in CPU resources, electrical energies, economic consumptions, and even space occupations. This motivates us to propose a highly integrated and low-power accelerating solution for N-body simulations, by utilizing the FPGA-SoC, which integrates both embedded low-power processors and FPGA.

Table 1: Time cost of MOND calculation on Intel i5 processor.

Number of particles	Time cost of Newtonian potential (s)	Time cost of PDM distribution (s)	Time cost of MOND potential (s)
10000	2.062	0.001	2.065
20000	8.551	0.003	8.536
50000	55.09	0.012	55.68

The calculation speed of an FPGA-based accelerator is mainly decided twofold: one is the number of pipelines the FPGA integrates and the other is the throughput of each pipeline; here the pipeline throughput is limited by two factors, the data bandwidth and the frequency pipelines work on. To obtain a high calculation speed, we are motivated to analyze the physical model of potential calculation and optimize the pipeline design, thus to pursue less logic resource occupations, lower data bandwidth demands, and higher operating frequencies.

In this paper, we focus on a highly integrated accelerating solution for N-body simulation in MOND. Our contributions are as follows.

(1) The utilization of FPGA-SoC makes the solution highly integrated.

(2) We propose optimized summation pipelines for the calculation of Newtonian and MOND potentials, in which the square term is conducted by three DSP48E1s in Xilinx 7 series FPGAs, such that the logic resource occupation of each pipeline is reduced and more pipelines are implemented.

(3) Based on the particle-mesh scheme, the data flow from memory to pipelines is optimized: the space coordinates of each object are automatically calculated out, other than being transferred, which benefits the reduction of data bandwidth.

(4) We conduct extensive experiments to test our proposed solution. The results show that 9 optimized pipelines can be implemented in an Zynq-7020 FPGA; if we utilize the typical pipeline, the number of pipelines that can be implemented in the same FPGA is 7, and the HLS (high level synthesis) tool only produces 4 pipelines.

4. System Design

4.1. Physical Model. We choose the nearest grid point (NGP) scheme, one of the particle-mesh strategies, as our basic

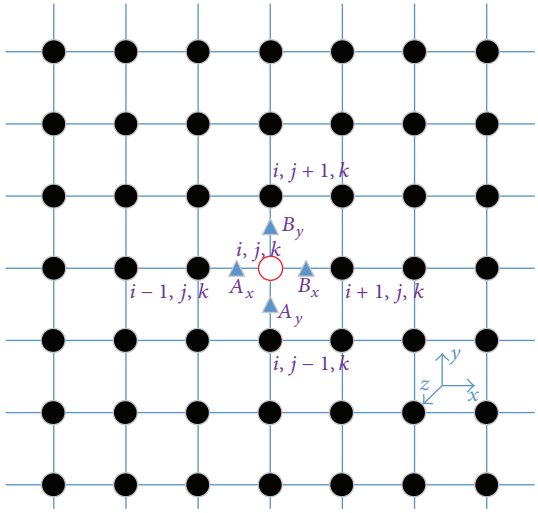


FIGURE 1: Illustration of the discretization scheme in the x-y plane.

physical model. Based on the NGP scheme, particles in the galaxy object are interpolated onto a mesh, and each particle is approximatively supposed to be located at the closest point in the mesh, as depicted in Figure 1.

In the particle-mesh scheme, the baryonic matter distribution ρ_b is right equivalent to the mass m_i for each particle i in the mesh. Under these circumstances, (2) can be directly solved by calculating the potential summation of every particle-pair like the following equation:

$$\phi\left(\vec{r}_i\right) = m_i \sum_{j \neq i} \frac{m_j \left(R_{ij}^2 + 1.5\epsilon^2\right)}{\left(R_{ij}^2 + \epsilon^2\right)^{(3/2)}}, \tag{8}$$

where ϵ is a proportional constant for adjusting the result at small radii and R_{ij} indicates the Euclidean distance between a pair of particles:

$$\begin{aligned} R_{ij}^2 &= \left(\vec{r}_i - \vec{r}_j\right)^2 \\ &= \left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2. \end{aligned} \tag{9}$$

With the particle-mesh scheme described in Figure 1, the distribution of PDM (6) can be approximatively calculated by finite difference, as written in [1, 9]

$$\begin{aligned} \rho_{\mathrm{ph}}^{ijk} = \frac{1}{4\pi G h} \Big[& \mu_{B_x} \left(\nabla\phi\right)_{B_x,x} - \mu_{A_x} \left(\nabla\phi\right)_{A_x,x} \\ & + \mu_{B_y} \left(\nabla\phi\right)_{B_y,y} - \mu_{A_y} \left(\nabla\phi\right)_{A_y,y} + \mu_{B_z} \left(\nabla\phi\right)_{B_z,z} \\ & - \mu_{A_z} \left(\nabla\phi\right)_{A_z,z} \Big]. \end{aligned} \tag{10}$$

Here $(\nabla\phi)_{A_x,x}$ is the x-component of $\nabla\phi$ at point A_x. When we write ρ_b as m_i, ρ_{ph} can be written as Δm_i.

In N-body MOND simulations, astronomers care about both the overall movement trend of a galaxy and the details of the inner galaxy. The former indicates that the mesh should handle a full coverage of the MOND gravity field, while the latter means that we need to reduce the greed size so as to obtain a higher spatial resolution and simulate more objects in the galaxy. If we set the resolution too high, it may lead to memory overflow and unacceptable time consumption. To address this problem, multistage particle-mesh scheme is optional. It is possible to change the resolution because, at a point far away from the galaxy, the galaxy can be seen as one particle with the total mass of the galaxy (Figure 2). Thus we can apply a higher resolution when it comes to the inner galaxy and a lower resolution for the outer space.

In this paper, for simplification, we set a fixed resolution. Technically, the computation flow is as follows:

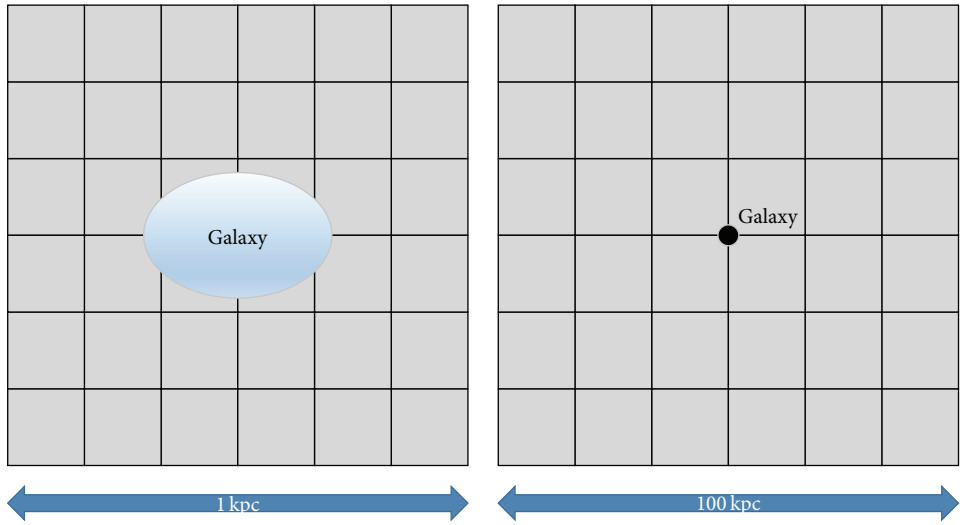
(1) building a mesh with a coverage of the galaxy and its outskirts as shown in Figure 1; the mesh is divided into multiple grids according to the fixed resolution, with particles attached to grid nodes;

(2) solving Newtonian potential, PDM distribution, and the final MOND potential through the three steps in Section 2.

4.2. Architecture of Accelerating Solution. Figure 3 provides the architecture of our accelerating solution, which consists of 8 parts: the embedded ARM Cortex-A9 processor, the DDR controller, the command and state bus, the data bus, registers, DMA-FIFO groups, the pipeline local controller, and potential calculation pipelines. All units are integrated in one chip.

Different from conventional accelerating solutions, we utilize the embedded ARM processor to deal with the data dispatching. Besides, the ARM processor also conducts all the lightweight computing tasks, including initializing the particle-mesh scheme, monitoring the status of DMA and pipelines, and calculating the PDM distribution. To maximize the bandwidth between memory and accelerating pipelines, the control stream and the data stream are splitted. A 32-bit AXI bus is implemented to transfer commands and states, and the data is transferred through a 64-bit AXI high performance bus and two DMA-FIFO groups. Each pipeline is concentrated to deal with one fixed particle. The potential of the fixed particle is actually the summation of $N - 1$ potential components; a potential component indicates the potential caused by another particle in this system. Thus the responsibility of the pipeline is to conduct the potential component calculation and the accumulation of these $N-1$ potential components. To prevent from calculating the invalid or meaningless potential component, namely, the potential caused by the fixed particle itself, in the local controller, a transfer counter and several register groups are coordinated. Each register group stores the information of the fixed particle for corresponding pipeline. The transfer counter is used to count the data transferred to exclude the self-calculations (Figure 4).

The issue of calculation bottleneck arises mainly in the potential summation. To handle this problem, we make endeavours 3-fold: pipeline stage splitting, pipeline simplification, and bandwidth reducing.



(a) Higher resolution ratio (b) Lower resolution ratio

Figure 2: Illustration of multistage particle-mesh scheme.

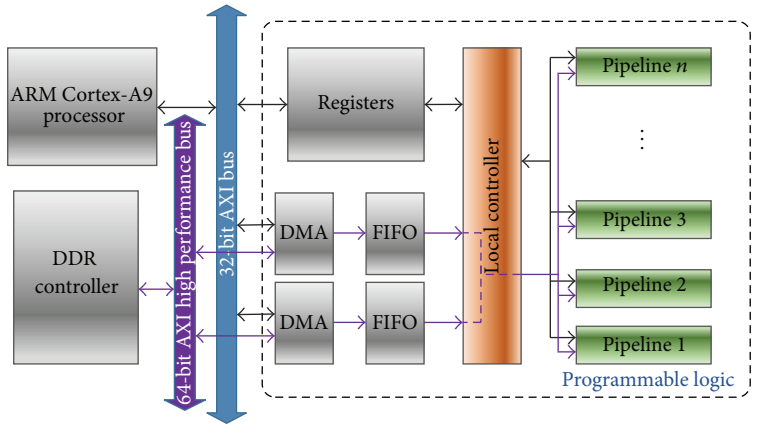


Figure 3: System architecture.

(1) Pipeline Stage Splitting. When splitting stages for a pipeline, it is important to ensure that every stage of the pipeline has the same latency cycle. If computation units in the same stage differ in latency, designers can either introduce additional registers to buffer results of lower latency units or reduce the higher latency cycle. This usually means a waste in the logic resource or frequency. As to our pipelines, floating calculation units are generated with the Xilinx floating-point IP generator. Due to the complexity of different calculations, the addition, subtraction, and multiplication units are set to the same latency cycle, while the division and square root units are set to another latency cycle. To make sure that every stage of the pipeline has the same latency cycle, (8) is transformed as follows:

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

$$= m_i \sum_{j \neq i} \frac{m_j}{R_{ij}^2 + \epsilon^2} * \frac{R_{ij}^2 + 1.5\epsilon^2}{\sqrt{R_{ij}^2 + \epsilon^2}}$$

$$= m_i \sum_{j \neq i} \frac{m_j}{R_{ij}^2 + \epsilon^2} * \frac{R_{ij}^2 + 1.5\epsilon^2}{R_{ij}^2 + \epsilon^2} * \sqrt{R_{ij}^2 + \epsilon^2}$$

$$= m_i \sum_{j \neq i} \frac{m_j}{R_{ij}^2 + \epsilon^2} * \sqrt{R_{ij}^2 + \epsilon^2} * \left(\frac{0.5\epsilon^2}{R_{ij}^2 + \epsilon^2} + 1\right). \tag{11}$$

(2) Pipeline Simplification. The upper part of Figure 5 shows the potential summation pipeline using the typical method like that presented in [2, 10]. When implementing the pipeline on Xilinx 7 series FPGAs, to make full use of the resource

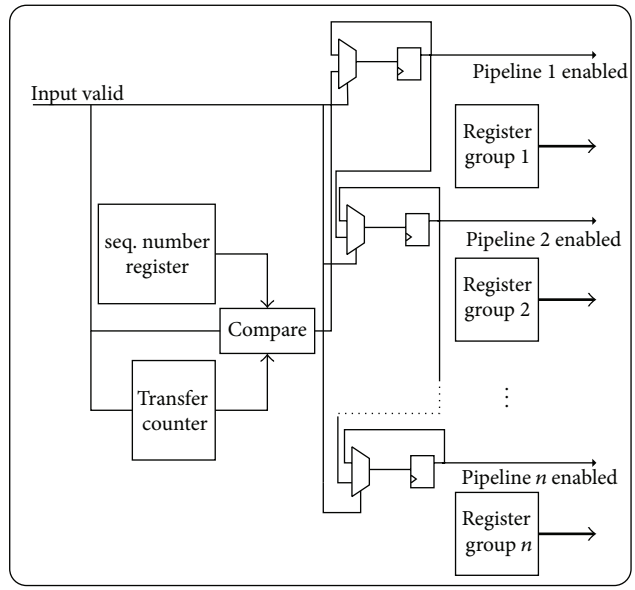


Figure 4: Partial diagram of controller module.

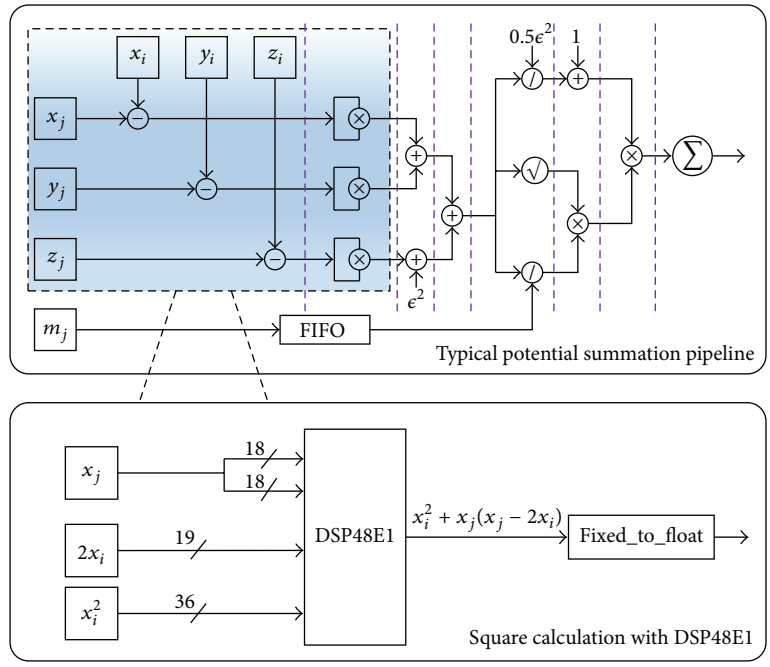


Figure 5: Up: typical potential summation pipeline with stage optimization; dotted line shows the dividing of pipeline stages. Down: modifying the logic in shadow box with DSP48E1.

of FPGA, we modify the pipeline as shown in the lower part of Figure 5. Take the x-dimension as an example; the square term $(x_i - x_j)^2$ can be equivalently rewritten as

$$\left(x_i - x_j\right)^2 = x_i^2 + x_j^2 - 2x_i x_j = x_i^2 + x_j\left(x_j - 2x_i\right). \quad (12)$$

It is right conform to the calculation format $(D - A)B + C$ of DSP48E1. Motivated by this fact, we apply 3 DSP48E1s to calculate the squared Euclidean distance. Then, the result will be transferred to floating point, adapting to the remainder parts of the pipeline, which are kept the same with the typical method as shown in Figure 5.

(3) Bandwidth Reducing. Based on the particle-mesh scheme, we can reduce the bandwidth from memory to pipelines. Theoretically, for each particle, we need three dimensional coordinates and the mass to calculate the potential component. Note that, in particle-mesh scheme, the coordinates of each particle are indicated as $(0, 0, 0), (0, 0, 1), \ldots, (0, 0, Z_{\max})$, $(0, 1, 0), \ldots, (X_{\max}, Y_{\max}, Z_{\max})$. The pipeline conducts $N - 1$

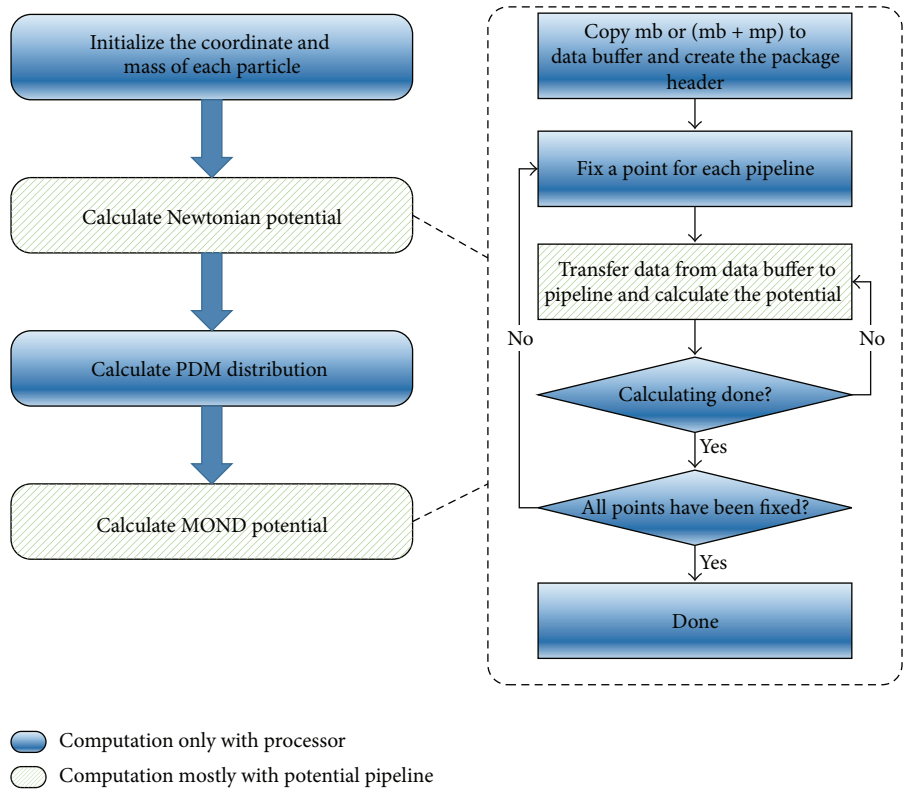


Figure 6: Flow chart of MOND simulation.

potential component calculation in a particular order. Thus, with the upper limits X_{max}, Y_{max}, and Z_{max}, the pipeline controller can automatically calculate the coordinates. Through this optimization, the data needed to be transferred is only the particle mass, so the bandwidth requirement can be significantly reduced.

4.3. Working Flow. The working flow of N-body MOND simulation with our accelerating solution is as described in Figure 6. Firstly, parameters of every particle in the particle-mesh scheme are initialized through the ARM processor, including the coordinates and mass. Secondly, data are transferred to the accelerator to calculate the Newtonian potential. Then, results of the previous step are used to calculate the PDM distribution $\rho_{ph}(x)$. In particle-mesh, $\rho_{ph}(x)$ is discontinuous and can be described as Δm_i for an individual particle i. Note that there does not really exist Δm_i, and it is only an intermediate to assist calculating of MOND potential in a form similar to the typical Newtonian formula. Next, m_i is updated with Δm_i. Finally, modified data are transferred to the accelerator for the MOND potential calculation. The dashed line box in Figure 6 shows the detailed work flow of the potential calculation.

In our solution, we define a structure to store the information of every particle, named grid, which is described in Algorithm 1. The x, y, and z coordinates and mass of baryon are decided by physics model, and the rest variables need to be solved by the processor and accelerator. In our solution, the grid structure will be initialized by the ARM processor.

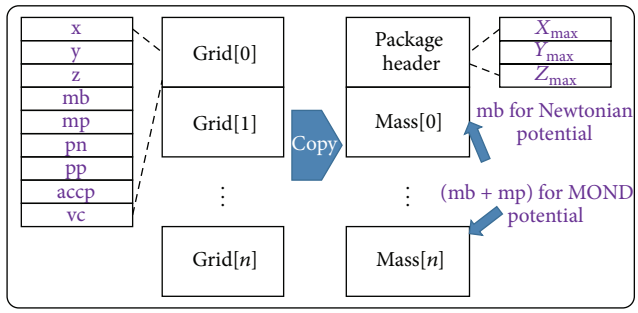


Figure 7: Illustration of memory copy operation.

However, this structure is not efficient when transferring data from memory to pipelines, since the data to be transferred is not consecutive. To avoid this problem, data to be transferred are copied to a fixed consecutive space before starting the potential calculation, as shown in Figure 7. A package header which contains X_{max}, Y_{max}, and Z_{max} will be transferred at the beginning of each operation.

5. Experiment and Result

5.1. Experiment Setup. To test our accelerating solution, we choose the Zedboard as the experiment platform. The Zedboard utilizes a Zynq-7020 FPGA-SoC, which consists of a dual-core ARM Cortex-A9 processor as the processing

```
typedef struct
{
    int    x;        //x coordinate
    int    y;        //y coordinate
    int    z;        //z coordinate
    float  mb;       //mass of baryon
    float  mp;       //mass of phantom dark matter
    float  pn;       //potential of Newton
    float  pp;       //potential of phantom dark matter
    float  accp;     //acceleration of phantom dark matter
    float  vc;       //circular velocity
}Grid;
```

Algorithm 1

Table 2: Zynq-7020 resource utilization.

Entity	LUTs	Registers	DSP48E1s
Pipeline (simplified)	3826 (7%)	7467 (7%)	15 (7%)
Pipeline (typical)	4417 (8%)	7761 (7%)	24 (11%)
Pipeline (HLS)	7014 (13%)	12132 (11%)	24 (11%)
Accelerator (9 simplified pipelines)	45014 (85%)	80403 (76%)	171 (78%)
Accelerator (7 typical pipelines)	39616 (74%)	64777 (61%)	196 (89%)

system (PS) and a Xilinx 7-series FPGA including 220 DSP48E1s as the programmable logic (PL). The maximum frequency of AXI bus between PS and PL is 250 MHz. What is more, 512 MB DDR3 with 32-bit interface is included on the Zedboard.

5.2. Resource Utilization. Table 2 makes a comparison between the typical pipeline and modified pipeline as shown in Figure 5, on resource utilization and the maximum frequency. We also use HLS (high level synthesis) tool to generate the pipeline and give the result in Table 2. All results are generated by Vivado 2015.2 with default options. Notice that as it is difficult to generate the accumulator by HLS tool, all the results of pipeline do not contain the accumulator. In addition, we give the total resource utilization result of an accelerator with 9 simplified pipelines or 7 typical pipelines.

From Table 2, we can get that the typical pipeline needs extra 15% LUTs and 60% DSP48E1s than the simplified pipeline. Furthermore, compared with the typical one, our simplified pipeline has achieved a more balanced utilization between different logic resources, especially in Zynq-7020. As a result, 2 more pipelines can be integrated when using simplified pipeline. Moreover, when using pipelines generated by HLS, only 4 pipelines can be implemented.

5.3. Comparison and Discussion. Suppose that an astronomical system contains $32 \times 32 \times 32$ particles that is right interpreted in a $32 \times 32 \times 32$ particle-mesh. Using the particle-particle algorithm, we calculate the MOND potential distribution of this system through different hardware platforms

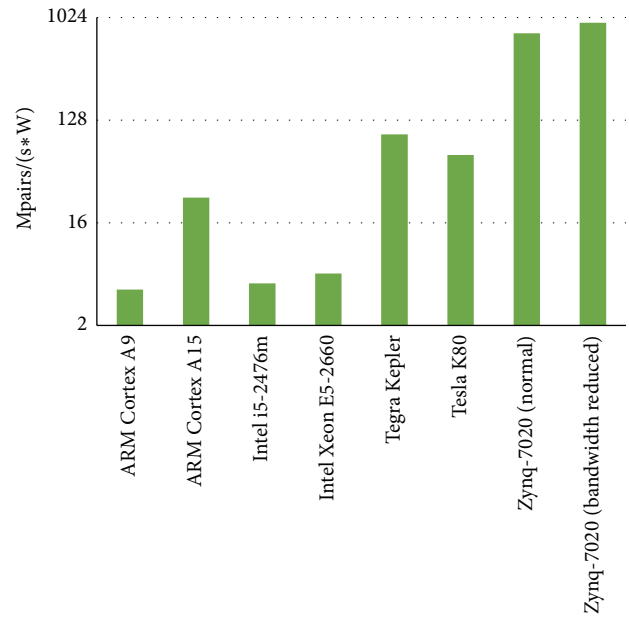


Figure 8: Comparison on performance per watt.

including CPU, GPU, and Zynq-7020, record the practical run time, and further make a comparison.

The same C source code is running in different types of CPUs without multicore parallelization, so the results only reflect the performance of a single core. We also run a well-optimized CUDA code in both embedded GPU and high performance GPU. For the accelerator proposed in this paper, we test two schemes. Both are running at 142 MHz, with 9 pipelines integrated; the difference is that one scheme utilizes the bandwidth reducing method mentioned in Section 4, while the other does not. All results are given in Table 3. Besides, we also give the power consumption for each device. As most power consumption parameters are from product manuals, these items are only for qualitative reference. The comparison on performance per watt is provided in Figure 8, with the vertical axis using a logarithmic scale.

From Table 3 we can evaluate the contributions of offloading the coordinate calculation to the FPGA logic. Without

Table 3: Calculating the potential in $32 \times 32 \times 32$ particle-mesh with different devices.

Computing unit	Type	Frequency (MHz)	The number of processor cores or CUDA cores or pipelines	s/frame	Mpairs/s	Power (watt)	Speedup
ARM Cortex-A9	Embedded CPU	667	2	345.7	3.106	1.5	1
ARM Cortex-A15	Embedded CPU	2300	4	32.24	33.30	≈5	10.7
Intel i5-2476m	CPU	1600	2	26.97	39.81	17	12.8
Intel Xeon E5-2660	CPU	2200	8	15.75	68.17	95	21.9
Tegra Kepler	Embedded GPU	950	192	5.592	192.0	<2	61.8
Tesla K80	GPU	562	$2 * 2496$	0.1701	6312	≈100	2032.2
Zynq-7020 (normal)	FPGA	142.8	9	1.107	970.0	1.3	312.3
Zynq-7020 (bandwidth reduced)	FPGA	142.8	9	0.853	1200.5	1.3	386.5

this optimization, the actual performance is 970 Mpairs/s with 9 pipelines, while the theoretical performance is 1285.2 Mpairs/s when the frequency is 142.8 MHz. The bottleneck is the bandwidth between memory and pipelines, which is 1724 MB/s under this situation. It is corresponding to the result of paper [11], in which Sadri et al. demonstrated that the full duplex throughput between memory and programmable logic of Zynq-7020 was about 1708.5 MB/s. However, by offloading the coordinate calculation to the FPGA logic, the actual test performance rises to 1200 Mpairs/s, close to the theoretical value. This optimization is also beneficial in reducing the requirement of memory space, as only mass instead of mass and coordinates needs to be stored in memory.

What is more, by analyzing Table 3 and Figure 8, two more points can be concluded. Firstly, compared with CPUs, our accelerating solution is better in both performance and performance per watt. Secondly, our solution exhibits a higher competitiveness than GPU in both performance per watt and performance per cost. For performance per watt, our solution is about 10 times better than GPUs like Tesla K80. This result is similar to the result presented in 2009 [10]. However, another conclusion in paper [10] is that GPU shows about 26 times better than FPGA in performance per cost. In our test, the Tesla K80 is about $4000, a Tegra K1 board with 192 Kepler cores is about $200, and the FPGA board is $395. This means that our solution shows about 50% higher than GPU in performance per cost.

6. Conclusion

In this paper, we propose a highly integrated accelerating solution based on FPGA-SoC for N-body MOND simulation. Through making full use of DSP48E1 in Xilinx 7 series FPGA, we simplify the classical potential summation pipeline to reduce the utilization of logic resources. Two more pipelines can be integrated in Zynq-7020 with our simplified pipeline. What is more, we reduce the bandwidth requirement between memory and pipelines based on the characteristics of particle-mesh scheme, which can also reduce the requirement on memory space significantly. The experimental results indicate that the performance of our solution is close to the theoretical value. It also shows that our accelerator is much better than CPU in both performance and performance per watt, about 10 times better in performance per watt when compared with GPU Tesla K80, and about 50% better than a GPU-based solution in performance per cost. Future work in this area involves formatting and testing the performance of an accelerating cluster combined with several FPGA-SoCs.

Competing Interests

The authors declare that they have no competing interests.

References

[1] M. Milgrom, "Quasi-linear formulation of MOND," *Monthly Notices of the Royal Astronomical Society*, vol. 403, no. 2, pp. 886–895, 2010.

[2] 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.

[3] 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.

[4] S. F. Portegies Zwart, R. G. Belleman, and P. M. Geldof, "High-performance direct gravitational N-body simulations on graphics processing units," *New Astronomy*, vol. 12, no. 8, pp. 641–650, 2007.

[5] Y. Wang, Y. Dou, S. Guo, Y. Lei, and D. Zou, "CPU-GPU hybrid parallel strategy for cosmological simulations," *Concurrency Computation Practice and Experience*, vol. 26, no. 3, pp. 748–765, 2014.

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

[7] M. Milgrom, "A modification of the Newtonian dynamics as a possible alternative to the hidden mass hypothesis," *The Astrophysical Journal*, vol. 270, pp. 365–370, 1983.

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

[9] 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.

[10] 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.

[11] M. Sadri, C. Weis, N. Wehn, and L. Benini, "Energy and performance exploration of Accelerator coherency port using Xilinx ZYNQ," in *Proceedings of the 10th FPGAworld Conference (FPGAworld '13)*, Stockholm, Sweden, 2013.