Hardware and Software Acceleration for N-Body Problem

An FPGA based approach for accelerating Barnes-Hut Tree

Li Wei-Wen Advisor: Liu Ren-Shuo Group Number: B496

Abstract

N-body simulations had been important in research fields like astrophysics and chemistry. It is infeasible to conduct analytical prediction on multiple bodies interactions. Therefore, simulating individual bodies movement and their interactions relies heavily on numerical method.

Nevertheless, N-Body simulations are computationally expensive. Each iteration requires $O(n^2)$ computation for force calculation. Thus, there has been numerous approach for accelerating such computations. In our work, we present the possibility for two direct summation FPGA kernel to operate on a Tree Based N-body simulation.

Introduction

For classical formulation, to quantify multiple bodies, we need the information of their positions, velocities, and mass. To evolve the system, we may conduct Taylor expansion. and for simplicity, we truncated it to the second term, so that the evolution of position relies on velocities, and the evolution of velocities relies on their acceleration. That is the Euler Scheme:

$$x(t+\Delta t) = x(t) + v(t)\Delta t$$
$$v(t+\Delta t) = v(t) + a(t)\Delta t$$

To compute the evolution of a certain bodies, we need its interaction with n-1 bodies. Taking gravitational system for example, the force is computed as follows

$$\mathbf{F_i(t)} = \sum_j^{n-1} \mathbf{F_{ij}(t)} = \sum_j^{n-1} \frac{Gm_im_j}{(r+\epsilon)^2}\hat{\mathbf{r}} = \sum_j^{n-1} \frac{Gm_im_j}{|\mathbf{r_i}-\mathbf{r_j}+\epsilon|^3}(\mathbf{r_i}-\mathbf{r_j})$$

There had been numerous attempts to accelerate the N-Body problem. One of which is the Barnes-Hut Tree. It construct a tree of the bodies and split the bodies until they are separated in individual leaf nodes.

Main Objectives and Problem Formulation

Our primary goal is to accelerate the force computation of the Tree Algorithm using FPGA kernels. We provide two main changes on the original Barnes-Hut tree.

1. Each body is not separated until only one body is left a single cell. Instead, we set a maximal number of bodies m for each block. By doing so, the depth of the tree is reduced.
2. The modified algorithm avoids traversing the tree for individual bodies by directly using a list of leaf nodes

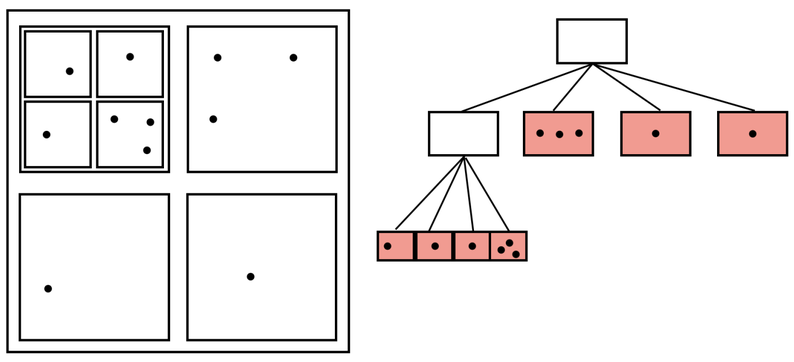

Figure 1: Modification on the Barnes-Hut tree shown in 2D. The reddish color denotes the leaf nodes of the tree. The change is that we allow one cell holding multiple bodies.

Methods

In order to construct large scale design, I compose the force computing kernel using Vitis HLS (High Level Synthesis). I utilize the two kernels to accelerate the Barnes-Hut method.

For experiment, I tested the acceleration design on AWS F1 instances. Then, I experience with the best parameters for the size of the tile and the maximum number of bodies in the cell.

Proposed Hardware Implementations

We separate the force computation in the modified tree into two categories: the local force computation of bodies within the leaf, and the remote computation between bodies and leaf cells.

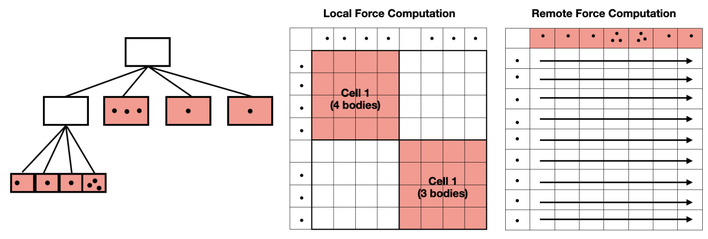

Figure 2: Separating local force and remote force. And conducting the two with different FPGA kernels

Remote Computation

For remote computation, we utilize the kernel developed by Del Sozzo et al. It set up a tiling approach which provides efficient pipelining and parallelism, but neglects the symmetric of force that could be extracted. However, since the remote force are asymmetric, it perfectly suits our algorithm.

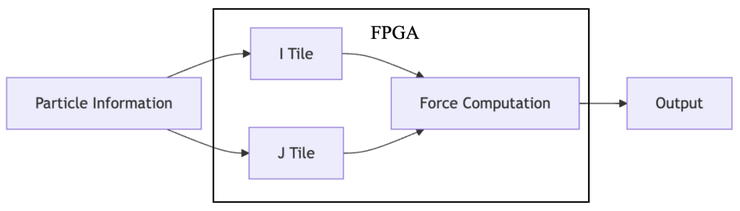

Figure 3: Schematic view of Del Sozzo el al. It tiles the information of bodies to reach scalability. The force computation is conduct in the FPGA and integration is left in the CPU.

Local Computation

In order to make use of the symmetric of force in the local force computation, I choose the kernel described in Menzel et al. Menzel et al. organize force computation to be arrays of 2x2 processing elements. Thus eliminate the difficulties when pipelining a triangular-shaped force computing scheme when we naively compute the symmetric force.

j \ i	0	1	2	3	4	5	6	7	8	9
0	0	+	+	+	+	+	+	+	+	+
1	-x	0	x	x	x	x	x	x	x	x
2	-x	-	0	+	+	+	+	+	+	+
3	-x	-	-x	0	x	x	x	x	x	x
4	-x	-	-x	-	0	+	+	+	+	+
5	-x	-	-x	-	-x	0	x	x	x	x
6	-x	-	-x	-	-x	-	0	+	+	+
7	-x	-	-x	-	-x	-	-x	0	x	x
8	-x	-	-x	-	-x	-	-x	-	0	+
9	-x	-	-x	-	-x	-	-x	-	-x	0

Figure 4: Figure from Menzel et al. It creates a kernel that can be pipelined by HLS easily. Each 2x2 PE computed 2 pairs of forces of the bodies and reverse them to get the other 2 pair of forces.

Conclusion

The result is yet to be fully implemented and requires further development. It is difficult for a single person to work out the whole thing in a short amount of time. But utilizing the different advantage of each kernel, our design should provide a good insight for attacking the N-Body problem.