N-Body Simulation using FPGA

Kumar Ayush (140260016), Sandesh Kalantre (140260012)

April 15, 2016

Abstract

An N-body simulation approximates the motion of particles, often specifically particles that interact with one another through some type of physical forces. Computing forces for each pair of particles has $O(n^2)$ complexity. Optimizing this computation for specialised hardware has improved speed by a full order of magnitude.[1] We implement and analyze the naive N-body simulation algorithm on a Terasic DE0-Nano board equipped with Altera Cyclone IV FPGAs.

1 Introduction

In physics, the n-body problem is the problem of predicting the individual motions of a group of celestial objects interacting with each other gravitationally.

The N-body problem simulates the evolution of a system of N bodies, where the force exerted on each body arises due to its interaction with all the other bodies in the system. The simulation proceeds over time steps, each time computing the net force on every body and thereby updating its position and other attributes.

Currently there are many computation methods and implementations to simulate the N-body problem. The straightforward approach is to directly compute pair-wise forces between bodies in each time step. In each step, the net force on each body is calculated through combining all Newton gravitational forces between that body and all the other bodies in the system. The system is then updated by moving each body to it's new position.

Psuedocode for Naive Algorithm

```
For each timestep
    For each body i
        For all other bodies j
            Calculate Pairwise force F(i,j)
    Move body for each step
    End
End
```

This naive implementation has $O(n^2)$ complexity. In general, direct methods using numerical integration require on the order of $\frac{n^2}{2}$ computations to evaluate the potential energy over all pairs of particles, and thus have a time complexity of $O(n^2)$. For simulations with many particles, the $O(n^2)$ factor makes large-scale calculations especially time consuming.

A number of approximate methods have been developed that reduce the time complexity relative to direct methods:

1. **Barnes-Hut** : Collision-less methods used when close encounters among pairs are not important and distant particle contributions do not need to be computed to high accuracy. The potential of a distant group of particles is computed using a multipole expansion of the potential. This approximation allows for a reduction in complexity to $O(N \log(N))$.

2. **Fast multipole** methods take advantage of the fact that the multipole-expanded forces from distant particles are similar for particles close to each other.

2 N-body simulation on a FPGA

It is fairly obvious that N-body simulation is highly parallel and therefore a standard approach using a microprocessor may not yield the most efficient solution in terms of time complexity. A hardware level implementation of the simulation is expected to our perform a software level calculation.

We use the naive implementation of the force calculation in our project to express the feasibility and power of using an FPGA for such a high parallel computation.

3 Force Calculation

The force computation is the central step in simulation of the n-body problem. It can be easily implemented at the hardware level on an FPGA and offers significant speed boost over a processor sequentially which has to use it's ALU to perform such calculations.

- The positions of the particles are represented as 32-bit floating point numbers according to the IEEE Standard for Floating-Point Arithmetic (IEEE 754).
- The force calculation is done by entity *NBS_Force*. It takes in input the positions of two particles and outputs the force between the two particles.
- The calculation begin by calculation of a relative vector between the two particles.
- The components of the relative vector are squared individually and added.
- Fast-inverse square root algorithm is used to calculate $\frac{1}{r}$.
- The masses of the two bodies are used to output the final force.

Schematically the working of *NBS_Force* can be represented as:

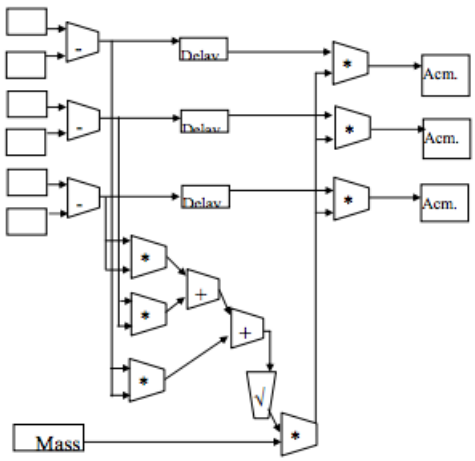

Figure 1: Force Calculation

We then use the naive algorithm presented in section 1 to proceed with the N-body simulation.

The positions of the particles are updated using the standard Euler method for solving ODEs.

4 Output display to a VGA port

We use a standard 15 pin VGA to display the positions of the particles on a screen. The schematic for the position of a single particle is as follows:

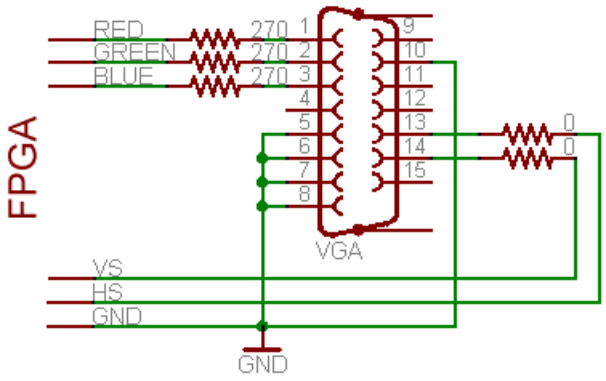

Figure 2: VGA Pin schematic (Image taken from fpga4fun.com)

- The positions of the particles are scaled to the width of the screen before displaying.
- We use color to represent the third co-ordinate(z) of the particle.

5 Conclusion

In conclusion, an FPGA offers a feasible and easy way to implement highly parallel problems. The inclusion of FPGA support presents significant performance speedup over pure software implementations

References

[1] Guangdeng Liao, Scott Siowry *Parallel Hardware/Software Implementation of the N-body Problem on a Supercomputer.*

[2] Jean-Pierre Deschamps, Gustavo D. Sutter, Enrique Cantó *Guide to FPGA Implementation of Arithmetic Functions.*