N-Body Simulation Project Proposal

Adib Khondoker (aak2250), Moises Mata (mm6155), Kristian Nikolov (kdn2117),
Isaac Trost (wit2102) , Robert Pendergrast (rlp2153)

Spring 2025

The primary goal of this project is to develop a hardware accelerator to simulate and display the kinematics of a variable number of celestial bodies. Also known as the N-Body problem, these types of calculations are used extensively in the fields of astrophysics and aerospace when determining the trajectories and orbits for spacecraft and planets. Since the gravitational forces on a celestial body can be calculated independently at any given moment, 'solving' an N-Body problem is well suited for hardware-based parallelization.

The primary function of the hardware accelerator will be to receive a list of parameters defined by the user that characterize each individual simulation. The hardware then quickly calculates an array of positions for each body at a constant time rate using one of the simulation calculation methods detailed in the following section. As these positions are calculated, they are sent to the software, which will subsequently display the bodies on the VGA display. Each celestial body will be represented as a colored circle with a radius pixel count proportional to its respective mass. The bodies will be displayed against a black background.

N-Body Positional Calculation

The primary task of the hardware will be accelerating the N-Body calculations. An exceedingly simple way of doing these calculations is the All-Pairs method. In this method, each body has the gravitational forces it

experiences due to every other body calculated, then summed and used to modify its velocity vector at every timestep. The calculations done for every single body can be done simultaneously, reducing the run time from O(n^2) to O(n) as long as the number of bodies does not exceed the number of available threads. We may calculate them in 3-dimensions even though the software may render the simulation in 2D to simplify the work necessary.

While our FPGA-accelerated All-pairs method computes instantaneous forces between bodies, numerical integration translates the forces into moving bodies that satisfy the coupled differential equations of motion. We evaluated three integration methods on their compatibility with FPGA hardware:

1. Euler's method: A simple method that has a single position/velocity update per timestep, but has error $O(t)$, for a timestep *t*. This error accumulates quite quickly and offers a poor simulation for conservation of energy, making Euler's method inadequate for our purposes.
2. Runge-Kutta 4 (RK4): This method provides high accuracy through four intermediate calculations per timestep. This however, comes with a 4x computational overhead which may be memory inefficient for our hardware.
3. Leapfrog Integration: This integrator provides exact energy conservation with a cost O(N), and an error on order $O(t^2)$ for a timestep t (which is smaller than $O(t)$ when $t<1$). The structure of the integration method is a staggered update sequence, which lends itself nicely to FPGA pipelining capabilities. The leapfrog integrator for position and velocity is presented below:

$$v_{n+1/2} = v_{n-1/2} + a_n \Delta t$$

$$x_{n+1} = x_n + v_{n+1/2} \Delta t$$

Hardware-Software Interface

One of the primary functions of the hardware-software interface is the sending of the user-defined parameters from software to be processed in the hardware accelerator. These parameters include the number of bodies (N), their respective masses, M_i, their starting coordinates (X_i,Y_i), and their starting velocities (Vx_i, Vy_i). These values will either be chosen from a predefined list of simulation parameters, or individually input by the user.

Additionally, a small hardware module will be needed to display the graphics. A second interface with the graphics module will be needed to display objects and trails. The software will feed objects size, colors, current position, and a handful of past positions, and the graphics modules will draw those objects on the screen.

Hardware Implementation

The N-body orbital dynamics computational accelerator will utilize the FPGA chip for gravitational force calculations, while the ARM core will handle user interactions and timestep navigation.

The accelerator will be designed using multiple processing blocks that perform pairwise gravitational force computations in parallel. Each block will be implemented as a systolic-array pipelined arithmetic unit, with stages dedicated to distance calculation, inverse square root approximation, force vector multiplication, and accumulation. The DSP blocks in the FPGA will be used for high-speed arithmetic operations. The FPGA will operate in a hardware-accelerated loop, continuously updating positions, velocities, and accelerations at each timestep.

Our implementation will be optimized to use block RAM (BRAM) for temporary storage, therefore fixed point representation will be used for our

calculations. A circular buffer will also be implemented for the joystick to scroll through timesteps. If there is a need to use floating point representation for higher precision or a need for more memory to cycle through our timesteps, the off chip SDRAM will need to be used along with the DMA engine to transfer data quickly.

The ARM core will also be responsible for managing the joystick controlled timestep navigation. All necessary parameters from keyboard input will be loaded onto the BRAM, with the arm core reading joystick input and deciding whether to move forward in the simulation or load a previous time step from the circular buffer that the accelerator has calculated. Communication between the FPGA chip and the arm core will be handled through the AXI bus.

This implementation will be developed in SystemVerilog, synthesized using Quartus Prime, and verified with ModelSim simulations.

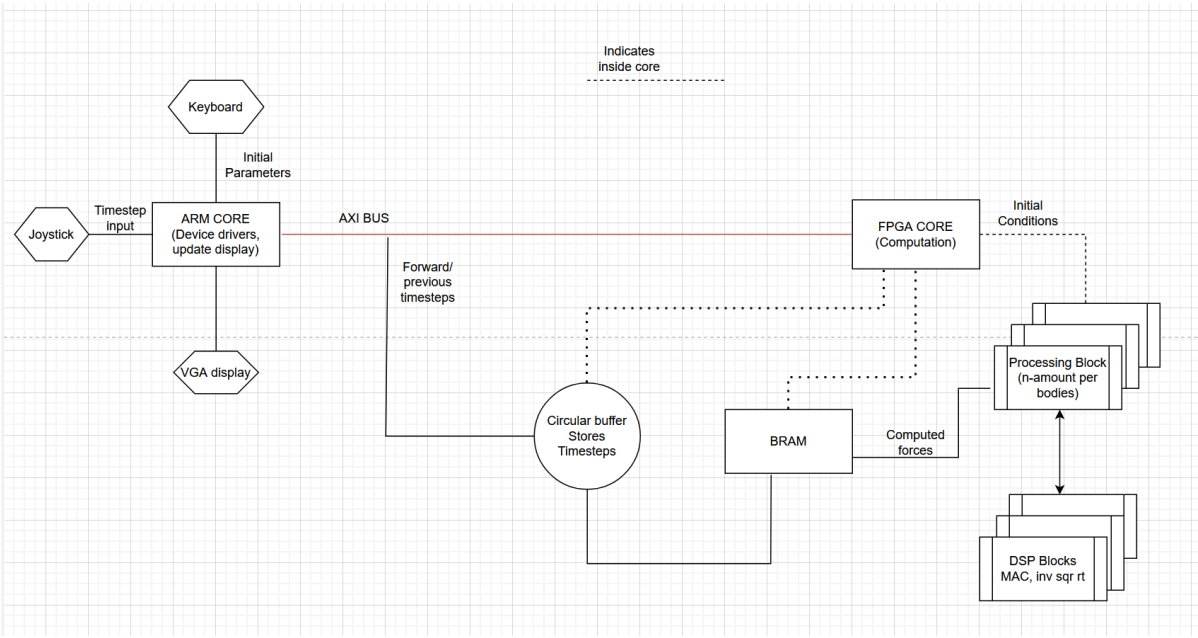

Figure 1: Hardware Implementation Architecture

User Interface

At the top of the VGA display, there will be a navigation menu that will allow the user to input the parameters of the simulation. To navigate this menu, a joystick or 3D design mouse will be used. The joystick will be responsible for moving side-to-side through the menu bar, as well as to place celestial objects onto the screen at the start of each simulation.

Additionally, while the simulation is running the joystick will be used to control the simulation's timestep, allowing the user to move the simulation both forward and backwards through time. In addition to the joystick, a standard usb keyboard will be attached to enter the simulation's numerical parameters such as the number of bodies, their respective masses, and the length of each simulation.

Major Tasks

- Specify performance capabilities of N-body simulation (e.g. 2D vs 3D, acceleration algorithm choice, time step size, etc.)
- Implement hardware accelerator in SystemVerilog to be capable of performing all simulation calculations and VGA display updates.
- Develop a Linux device driver for the accelerator.
- Software for handling the User Interface, as well as device drivers for USB input: keyboard, joystick and VGA cable.

References

Camphuijsen et Al. "Visualizing the Few Body Problem." 6 Nov. 2015
Gupta et Al. "FPGA Accelerated N-body Simulations." 14 Oct. 2023
Quinn et Al. "Time stepping N-body simulations." 3 Oct. 1997