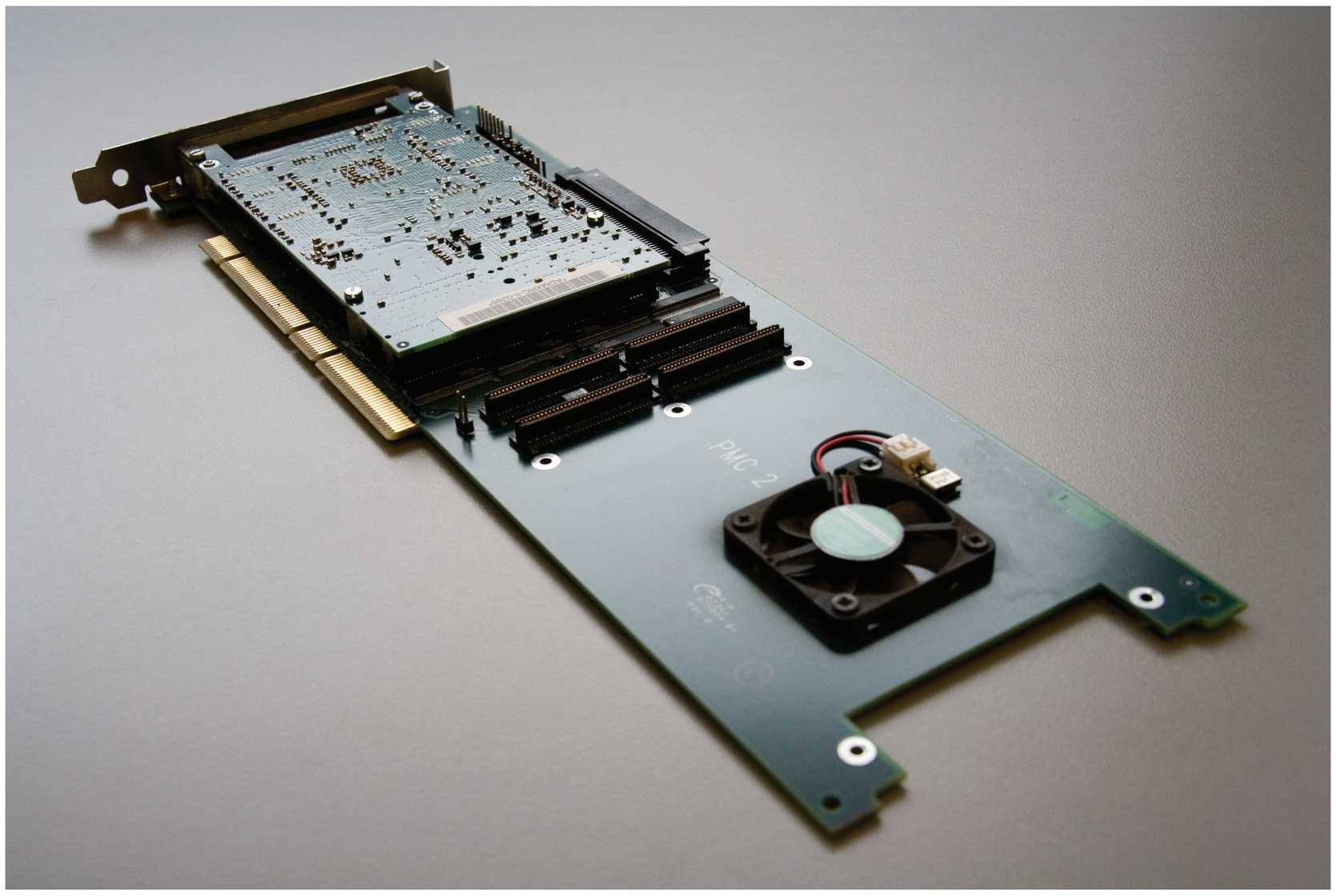
HARDWARE ACCELERATED BROAD PHASE COLLISION DETECTION FOR INTERACTIVE ENTERTAINMENT APPLICATIONS

Muiris Woulfe
John Dingliana
Michael Manzke



The Alpha Data ADM-XRC-II PCI board hosting a Xilinx Virtex-II XC2V6000-4 FPGA that was used to implement the microarchitecture

COLLISION DETECTION is a vital component of many applications, particularly in the field of interactive entertainment. Under a scheme referred to as hybrid collision detection, this component is divided into two distinct phases – the broad phase and the narrow phase. The broad phase quickly tests all objects for collision using an approximate algorithm, while the narrow phase checks the potential collisions returned by the broad phase using a more precise algorithm.

Since the broad phase tests all objects for collision, it can potentially be a limiting factor in many applications, despite the creation of algorithms that exploit coherence, as the worst case time for all broad phase algorithms remains $\Omega(n^2)$. For this reason, we put forward an alternative solution that offers a radical departure from the current state of the art.

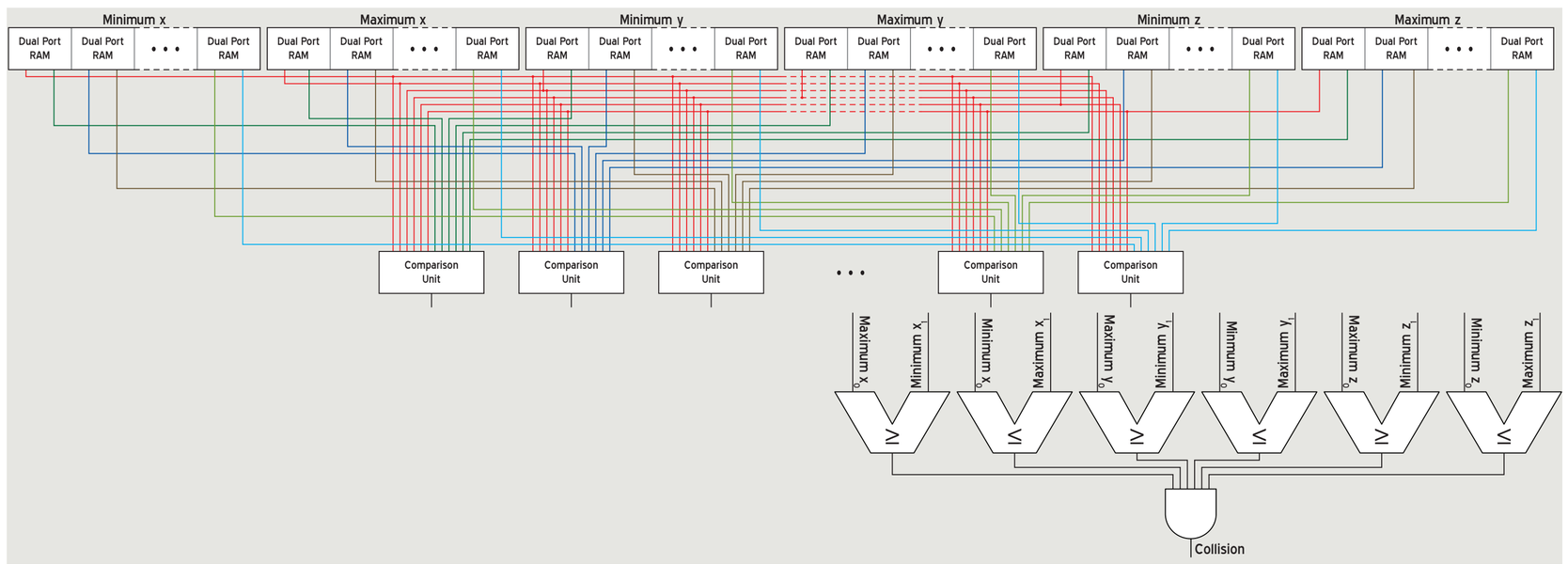
We propose the creation of a hardware coprocessor with a custom microarchitecture specifically tuned to execute broad phase collision detection routines efficiently. For prototyping, we have used a Field-Programmable Gate Array (FPGA), which is an Integrated Circuit (IC) whose internal logic may be dynamically reconfigured. However, our microarchitecture has been designed to be equally amenable to platforms capable of supporting larger and more complex designs, such as Application-Specific Integrated Circuits (ASICs) whose logic must be determined before the device can be fabricated.

MICROARCHITECTURE

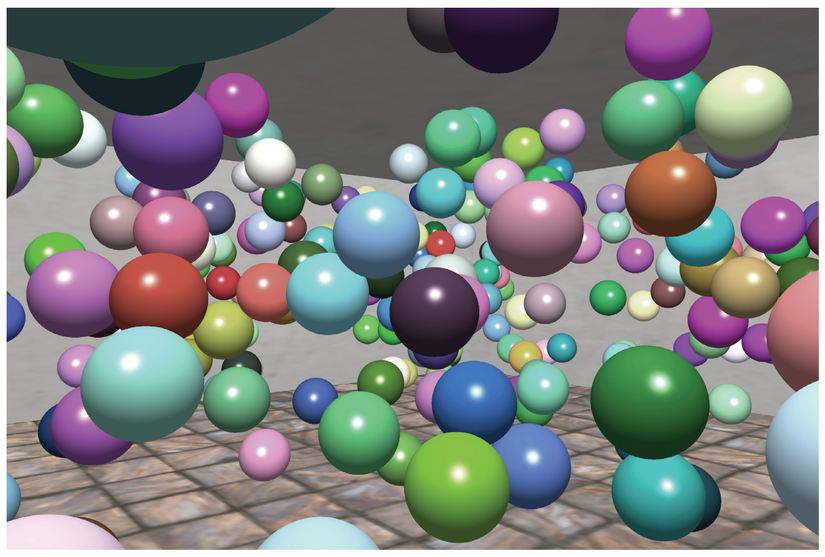
Prior to implementing the microarchitecture, it was necessary to determine the algorithm most amenable to hardware implementation. Since the primary means of gaining acceleration through a hardware coprocessor is by the exploitation of parallelism, the highly parallel brute force algorithm was selected for prototyping. Axis-Aligned Bounding Boxes (AABBs) were used as the bounding volume.

The main module of the microarchitecture is the comparator. It begins with a $2m$ width memory. This can be constructed using m dual port RAMs with the same data replicated across each RAM, logically creating a $2m$ port RAM. This $2m$ port RAM allows $2m$ data to be outputted in a single clock cycle, instead of the typical two data. $2m$ data facilitate a greater degree of parallelism in the comparison logic, allowing for greater acceleration. In total, six $2m$ width RAMs are required for storing the minimum and maximum x, y and z data.

When reading from the RAMs, a complex scheme is used to ensure each of the necessary comparisons take place. The first address input is set to the initial address and the remaining $2m-1$ address inputs are set to the $2m-1$ successive addresses. In the subsequent cycles, the first address input is kept constant while the remaining $2m-1$ address inputs are each incremented by $2m-1$ until one of these address inputs reaches the maximum address used in the system. When this occurs, the first address is incremented by 1 and the remaining $2m-1$ address inputs are set to the $2m-1$ addresses subsequent to the new first address. The process continues until the first address input reaches one less than the maximum address and the remaining $2m-1$ address inputs reach the maximum address.



The microarchitecture of the comparator module

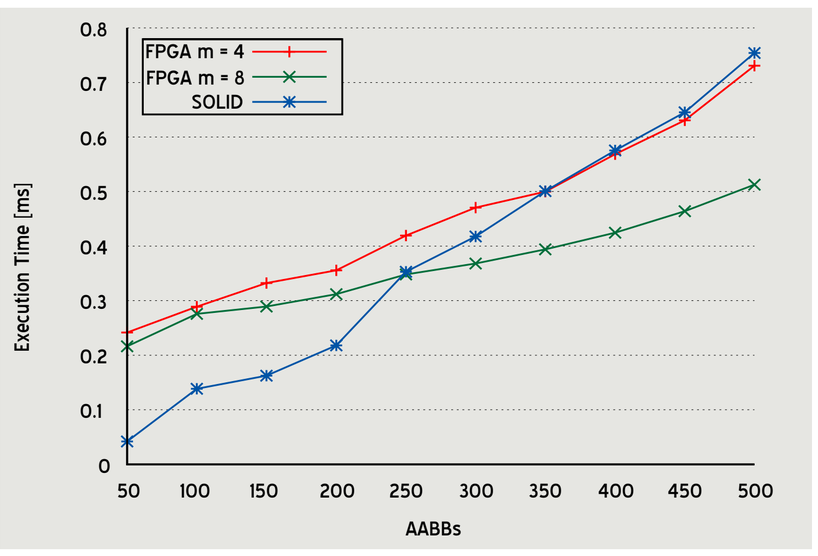


The simulation that exercised the collision detection microarchitecture and SOLID

The comparison unit is implemented using six floating-point cores with each core performing one comparison. The results from the six cores are ANDed to determine whether a collision occurred.

RESULTS

We implemented the microarchitecture on an Alpha Data ADM-XRC-II PCI board that hosts a Xilinx Virtex-II XC2V6000-4 FPGA. A software application that utilises the microarchitecture executed on an Intel Pentium 4 1.8 GHz computer with 2 GB RAM. Both the FPGA and the computer are of the same era.



Execution times for the microarchitecture and SOLID (m denotes the level of parallelism in the microarchitecture)

The application simulates a closed box with a variable number of spheres bouncing off each other and the walls of the box.

The execution time of the simulation, including the communication overhead between the host computer and the FPGA, was measured for a variable number of spheres and a variable degree of parallelism. This was compared with the execution time of the broad phase component of the standard SOLID collision detection library executing on the same computer.

The results indicate that small quantities of objects are decelerated due to the significant communication overhead. However, an acceleration of up to 1.5× is achievable using the FPGA coprocessor to simulate 500 objects. Due to a design limitation, greater quantities of objects have not yet been simulated, although it is evident that an even greater acceleration will be achieved once the design has been modified to accommodate these larger simulations.

gv2
graphics • vision • visualisation

Supported by
embarkinitiative
Investing in People and Ideas

http://www.cs.tcd.ie/~woulfem/