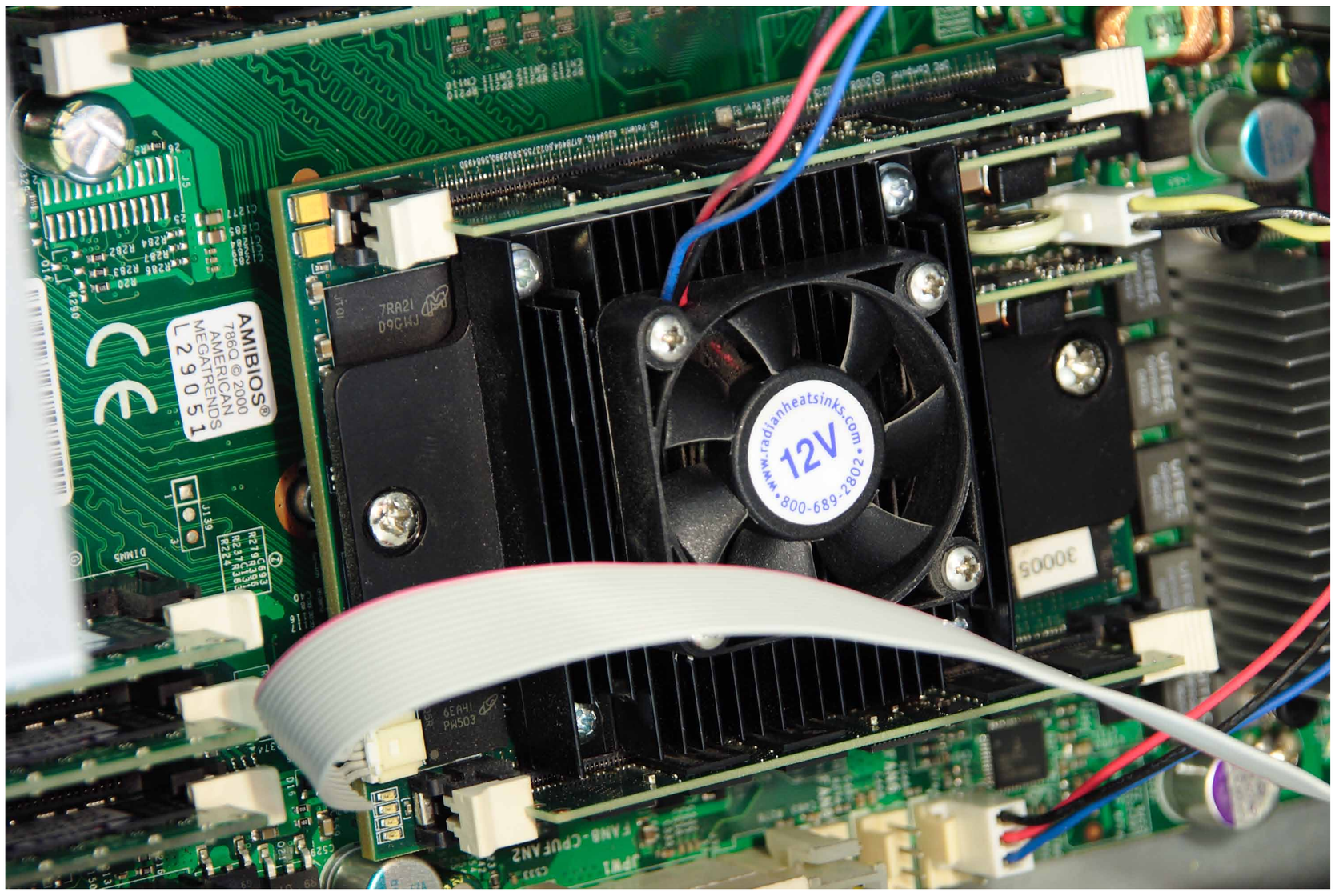
Collision Detection Hardware Optimised for Ray-Tracers

Muiris Woulfe
Michael Doyle
Michael Manzke

Graphics, Vision and Visualisation Group (GV2)
Trinity College Dublin
Ireland



DRC Accelium AC2030 module hosting a Xilinx Virtex-5 XC5VLX330-2 FPGA, used to prototype the microarchitecture

COLLISION DETECTION is the process of determining whether objects in a computer-simulated environment are intersecting. It is vital for a range of diverse applications. However, despite numerous developments, it is often challenging to perform accurate collision detection at interactive rates. To overcome this challenge, we propose a radical departure from the state of the art: a custom broad phase collision detection microarchitecture combined with software to achieve execution speeds greater than those attainable using software alone. The integration of this microarchitecture into a ray-tracer is also investigated.

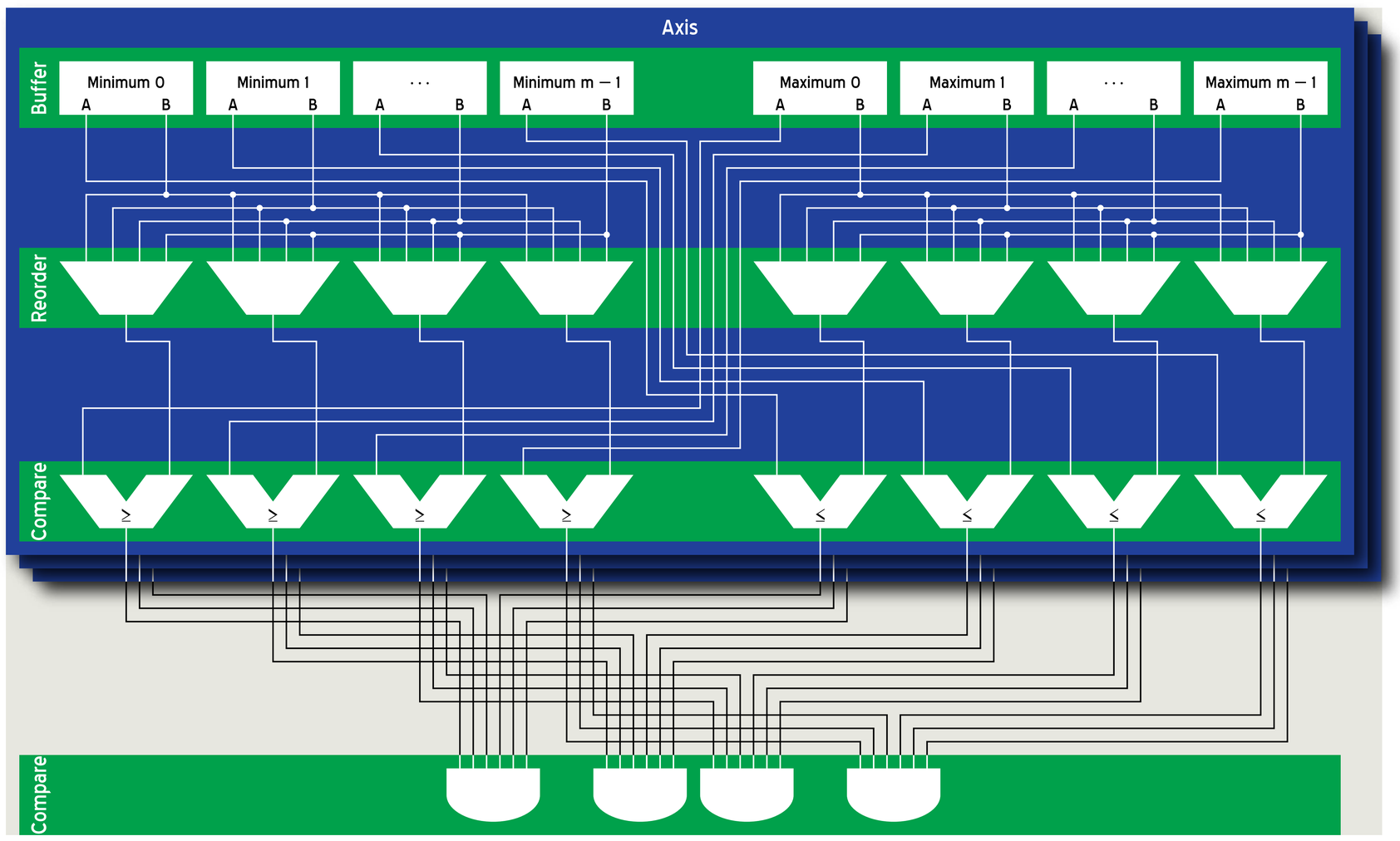
Microarchitecture

Since the primary means of gaining acceleration using a microarchitecture is through the exploitation of parallelism, we selected the embarrassingly parallel brute force all-pairs algorithm for our design. Axis-aligned bounding boxes (AABBs) were used as the bounding volumes.

The microarchitecture buffers the six data of each AABB using six $2m$-port memories, constructed from six m dual-port memories, each with address ports A and B. Afterwards, the AABBs are read from the buffer in a specific sequence. Originally, every A is set to 0, while the first B is set to 1 and the remaining Bs are set to 0. On the subsequent cycle, the second B is incremented to 1 and the remaining Bs retain their previous values. This pattern continues until the first B selects the address after the last AABB. At this stage, every A is set to 1, the first B is set to 2 and the remaining Bs are set to 1. This pattern continues until some A selects the second last AABB and the first B selects the address after the last AABB.

To perform the necessary comparisons, the data emitted from the buffer must first be reordered using multiplexers. To do this, multiplexer 0's selector is initialised to 1, multiplexer 1's is initialised to 2, multiplexer $m-2$'s is initialised to $m-1$ and multiplexer $m-1$'s is initialised to 0. On each cycle, each selector is incremented by 1 mod $m-1$. The sequence restarts each time the buffer's A inputs are modified.

Finally, the AABB collision test is performed using $3m$ greater-than-or-equal-to comparators and $3m$ less-than-or-equal-to comparators. The results from these comparators go to m logical AND gates. Each gate has six inputs corresponding to the six comparator results forming a single AABB pair.



Microarchitecture for broad phase collision detection, where m denotes the parallelism

Ray-Tracing

The outlined microarchitecture suffers from two potential disadvantages. Firstly, the number of objects supported may be significantly constrained by memory size. Secondly, the $O(n^2)$ complexity of the all-pairs algorithm may become significant with large quantities of objects, despite the effective exploitation of parallelism.

We deduced that spatial partitioning algorithms are an appropriate means of overcoming these disadvantages. These algorithms divide the environment into discrete cells, which can be individually processed by the microarchitecture. Additionally, we deduced that ray-tracing is liable to surpass rasterisation for the display of 3D images. Therefore, we decided to use the k-d tree, which is the standard ray-tracing spatial partitioning algorithm, to facilitate data reuse.

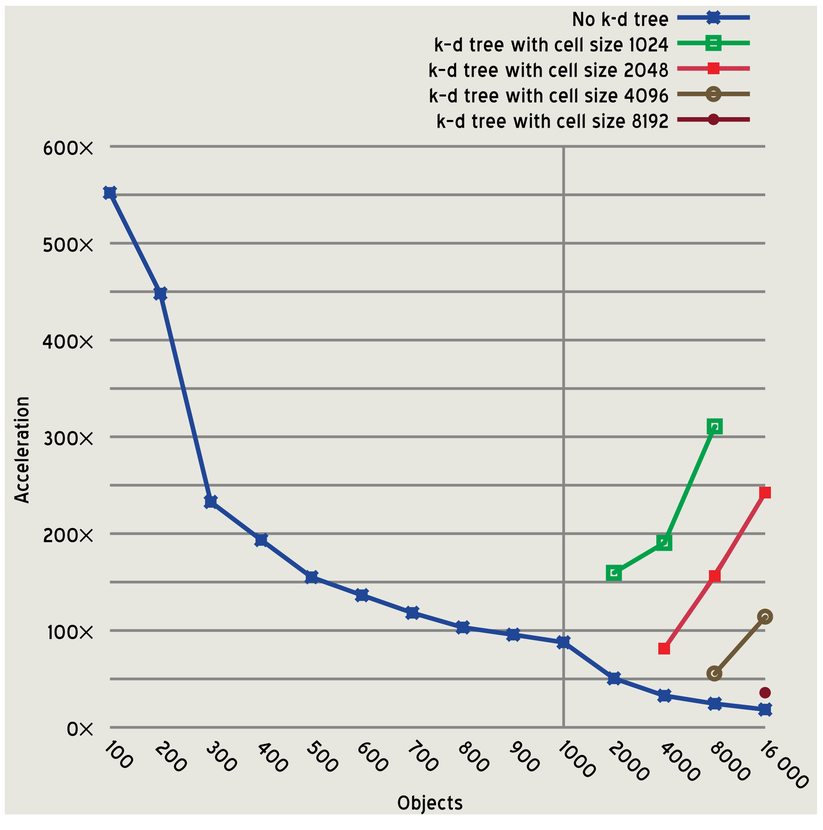
Results

We prototyped the microarchitecture using a DRC Accelium AC2030 module hosting a Xilinx Virtex-5 XC5VLX330-2 field-programmable gate array (FPGA). The associated software executed on a 2 GHz Quad-Core AMD Opteron 2350 computer with 8 GB RAM. This software simulated an enclosed cube with a variable quantity of objects, of different sizes and types, bouncing off each other and the walls of the cube.

The execution times were measured for the broad phase of the Bullet Physics SDK and for our microarchitecture, both including and excluding the k-d tree. For the k-d tree, a variety of cell sizes were used. These cell sizes specify the maximum quantity of objects that each leaf may contain.

The results demonstrate accelerations up to 552×. Moreover, the k-d tree is clearly beneficial for large quantities of objects as the acceleration of the 8000-object simulation increases from 24× to 311×. Therefore, it is evident that the collision detection microarchitecture provides substantial accelerations and that the k-d tree is an effective data structure for overcoming the microarchitecture's limitations in the context of a ray-tracer.

100-object simulation used to exercise the collision detection system



Accelerations, using a variety of k-d tree cell sizes