Path Tracing on Massively Parallel Neuromorphic Hardware

P. Richmond\textsuperscript{1} and D.J. Allerton\textsuperscript{1}
\textsuperscript{1}Department of Automatic Control Systems Engineering, University of Sheffield

Abstract
Ray tracing on parallel hardware has recently benefit from significant advances in the graphics hardware and associated software tools. Despite this, the SIMD nature of graphics card architectures is only able to perform well on groups of coherent rays which exhibit little in the way of divergence. This paper presents SpiNNaker, a massively parallel system based on low power ARM cores, as an architecture suitable for ray tracing applications. The asynchronous design allows us to demonstrate a linear performance increase with respect to the number of cores. The performance per Watt ratio achieved within the fixed point path tracing example presented is far greater than that of a multi-core CPU and similar to that of a GPU under optimal conditions.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Hardware Architecture—Parallel processing I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Ray Tracing

1. Introduction
Ray tracing offers a significant departure from traditional rasterized graphics with the promise of more naturally occurring lighting effects such as soft shadows, global illumination and caustics. Understandably this improved visual realism comes at a large computational cost. Traditionally ray tracing has been realised in software with high performance supercomputer solutions addressing the issue of performance [GP90]. The increased flexibility of the Graphics Processing Unit (GPU) has more recently opened up the possibility of massively parallel ray tracing on consumer hardware [PBMH02, PBD\textsuperscript{+}10]. Essential to the success of this work has been the ability to exploit thread level coherency within the wide SIMD architecture. Typically this requires the grouping of similar threads via the use of generated hierarchical data structures [FS05]. Due to the issue of ray coherency and the requirement of efficient algorithms for generating hierarchical data structures on the fly, GPU ray-tracing has failed to achieve the same orders of magnitude speed-up achieved in other GPU accelerated domains such as molecular dynamics. This situation has led to the use of energy hungry supercomputing clusters of GPUs for high performance ray tracing applications.

Aside from pure supercomputing performance, energy efficiency and cooling requirements are quickly becoming a key consideration with respect to supercomputing metrics. The SpiNNaker architecture is a massively parallel (up to a million cores) and highly interconnected architecture which considers power efficiency as a primary design goal. Originally designed for the purpose of simulating large neuronal models in real time, the architecture is based on low power (passively cooled) asynchronous ARM processors with programmable cores. This paper examines the potential of the SpiNNaker architecture in its current form for the purposes of ray tracing. More specifically this paper describes the implementation and performance of a path tracer realised using fixed point integer arithmetic [HRB\textsuperscript{+}09] suitable for the ARM based architecture of SpiNNaker. Consideration is given to the energy efficiency of the performance results which demonstrate a significant performance per Watt ratio.

2. The SpiNNaker architecture
The SpiNNaker hardware architecture consists of a number of independently functional and identical Chip-Multiprocessors each consisting of 18 ARM 968 cores running at 200MHz. Each core has its own dedicated tightly coupled memory, holding 32KB of instructions and 64KB of data. Each chip also contains 1Gbit of shared SDRAM connected via a DMA controller which uses an asynchronous “system” Network on Chip (NoC) to replace the requirement of a traditional on chip bus. The system NoC implies a Globally Asynchronous and Locally Synchronous (GALS) timing model [PFT\textsuperscript{+}07] allowing each chip to run in its own timing domain. Similarly, a second “Communication” NoC...