POETS - a generic event-based simulation engine

(III of IV) 7 February 2019

Andrew Brown
Department of Electronics and Computer Science,
University of Southampton, SO17 1BJ, UK

Abstract:

In Seminar I, we talked about the general idea of event-based simulation, using massive multi-core architectures. In Seminar II, we described an implementation of this concept, the SpiNNaker engine, and showed *how* it does *what* it does, in the highly specialised domain of neural simulation. In this third talk, we explore another machine - a POETS engine - which is itself a generalisation of the concepts embodied in SpiNNaker. POETS can be exploited for a much wider application portfolio than SpiNNaker.

The capabilities and costs of conventional parallel computing are well known, and as hardware architectures become ever larger - and more powerful - the *relative* cost of message choreography (the responsibility of the application designer) threatens to become rate-limiting. Traditionally - compared to raw compute - messages are expensive, and this has a significant impact on the way software architects think: information has to be bundled up into as few, large messages as possible, and this inevitably warps the way the system is structured.

Imagine a system with an arbitrarily large number of processors, where core-core communication is of negligible cost..... *What* might we do with it? *How* might we do *anything* with it? In the (complex and multi-dimensional) spectrum of compute capabilities, this imaginary system lies a long way away from the cluster representing the 'conventional' parallel supercomputer. Reasonably, POETS is not such a system: it has finite physical limits, although it is realised using FPGA technology, so everything is mutable. It does, however, in its current embodiment, lie far closer to our fictitious machine than the resource-consuming behemoths common in supercomputer laboratories today.

Event-driven machines - of which POETS and SpiNNaker are existence proofs - open the door to a new class of computing technique: event-driven computing. To exploit the capabilities of this type of architecture requires a new way of formalizing problems: you cannot simply port an existing codebase, however elegantly structured, to an event-based system and expect to get any of the advantages it offers. It is necessary to go back to the underlying mathematical foundations and re-cast the problem in a manner that is capable of exploiting the capabilities of the hardware. (Automating this - creating a compiler to target an event-driven architecture - is an unsolved research problem.)

In this talk, we firstly describe the design decisions underlying, and structure of, a POETS engine and then discuss in some detail an example of the use of event-based techniques in solving a real problem:

- **Space-filling neural synthesis:** The generation of *arbitrary* neural topologies is not difficult, but overlaying these structures with the attributes of geometry, whilst ensuring that they do not (self) intersect, is computationally hard - and adding vasculature makes the problem more so. Here we describe ways in which event-based techniques can be used to accelerate the growth of non-intersecting synthetic neural structures. *Why* do we want to do this? The outputs of a simulator are of no use unless there is a reasonable expectation - and demonstration - that the results are credible. Generating sensible test circuits for simulators such as SpiNNaker is an important aspect of their journey from laboratory curiosity to a useful, useable research tool.