A communication infrastructure for a million processor machine

Andrew D Brown
Electronics and Computer Science
University of Southampton
SO17 1BJ UK
+44 2380 593374
adb@ecs.soton.ac.uk

Steve B Furber
Computer Science
University of Manchester
M13 9PL UK
+44 161 2756129
steve.furber@manchester.ac.uk

Jeff S Reeve
Electronics and Computer Science
University of Southampton
SO17 1BJ UK
+44 2380 592784
jsr@ecs.soton.ac.uk

Peter R Wilson
Electronics and Computer Science
University of Southampton
SO17 1BJ UK
+44 2380 594162
prw@ecs.soton.ac.uk

Mark Zwolinski
Electronics and Computer Science
University of Southampton
SO17 1BJ UK
+44 2380 593528
mz@ecs.soton.ac.uk

John E Chad
Biological Sciences
University of Southampton
SO17 1BJ UK
+44 2380 594292
j.e.chad@soton.ac.uk

Luis Plana
Computer Science
University of Manchester
M13 9PL UK
+44 161 2756194
lplana@cs.manchester.ac.uk

David R Lester
Computer Science
University of Manchester
M13 9PL UK
+44 161 2755762
david.r.lester@manchester.ac.uk

ABSTRACT
SpiNNaker (Spiking Neural Network architecture) is a massively parallel computing machine, comprised of a million ARM9 cores. These are realised on 50000 chips, 20 cores/chip. While it could be classed as a MIMD machine, there is no unifying bus structure, and there is no attempt to maintain cross-system memory coherency. Inter-core communication is brokered by a fast message-passing system, built in and managed at the hardware level - thus there is an inevitable tension between speed and flexibility.

The message passing infrastructure was designed to be fast and have a high bandwidth; a consequence of this design decision is that the effective data payload is only 32 bits/packet. Whilst this is ample for a wide range of applications, whilst the system is initialising, it is necessary to transport relatively large and sophisticated data structures across the system. This can be slow and cumbersome, and makes some form of internal self-organisation extremely attractive. This is described in outline here.

Categories and Subject Descriptors
C.1.3 [**Computer Systems Organisation**]: Other architecture styles.

General Terms
Algorithms, Design, Reliability, Experimentation, Languages.

Keywords
Multi-core, self-organisation.

1. INTRODUCTION
Generalising outrageously, the class of problems for which SpiNNaker is ideally suited contain any problem that can be represented as a mesh, where node-node interactions are local (in the topological sense). This definition is broad: it covers network simulations (including large networks of neurons), finite element problems, ray-tracing, and many-body interaction problems, to name but a few.

2. THE SYSTEM
The computational ensemble of a million cores is not isotropic. The sheer scale of the system demands that some type of hierarchy exists: each SpiNNaker node contains 20 cores, of which one is selected as the 'monitor' core. Each of these 20 cores has its own local memory, plus they share access to some node-local RAM, six fast I/O ports and 128 MByte of node-local off-chip SDRAM - see figure 1. Each chip is connected (via the IO ports) to six physically adjacent nodes. The configuration of choice at this level is currently a hexagonal mesh, although there is nothing sacred about this. One of the node-local central resources on each chip is an ethernet port; approximately one in 5000 of these is connected to an external 'conventional' computing system.

Thus there exists a shallow hardware hierarchy - although in steady state, any core in any node may communicate with any

other core in any node, whilst the system is configuring, it is obliged to make use of the hardware hierarchy provided: worker core → monitor core → ethernet monitor core → outside world.

Three types of packet support inter-core communication: Nearest neighbour (NN) packets, which allow *monitor* cores to communicate with their nearest neighbour monitors, via the fixed hardware I/O links; point to point (P2P) packets, which allow arbitrary pairs of monitors to communicate, and multicast (MC) packets, which allow arbitrary pairs of cores to communicate. The NN protocol is entirely hardware, and is available from power-up, but the latter two require significant bootstrap processing to establish.

Once in place, however, (by a mechanism nor described here), the next problem is the mapping of our problem mesh into the physical processor mesh.

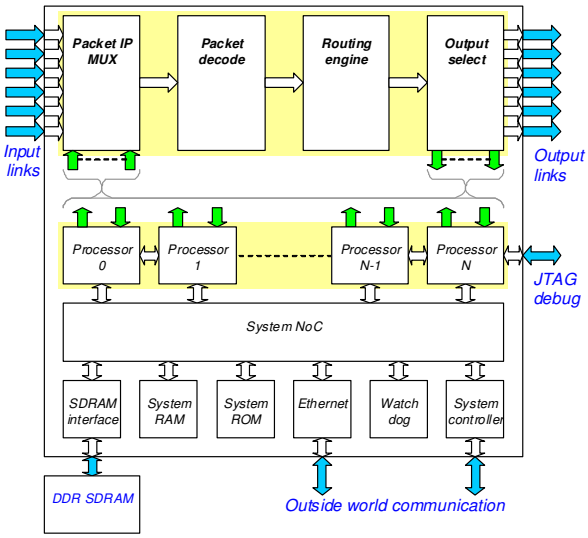

Figure 1: SpiNNaker internal architecture

2.1 Self-organisation

Conventional place and route techniques (generally considered largely a solved problem by the late 1980s) require - at some stage - for some part of the system to have an overview of the entire problem space. Where this is not feasible, hierarchical decomposition allows the problem to be solved as a series of sub-problems, and the solution stitched together; the decrease of quality of solution is not considered important.

For the P2P infrastructure, we have 50000 routing tables (one for each chip), each containing 50000 entries. Thus we need to compute and load into SpiNNaker 2.5GBytes of data to create this packet network.

For the MC problem, using neural network simulation as an example, we will typically be mapping 10^9 neurons to 10^6 cores (each core can support the emulation of 1000 neurons), and (assuming the average neural fan-in to be a biologically realistic 10^4), establish 10^{13} neural interconnects, which are then required to be loaded into 50000 MC route tables. These numbers indicate that we cannot even hold the data in a 32-bit offline machine, let alone reason about them in anything other than linear time.

Fortunately, we have an alternative: a million core parallel processing machine, which, with careful planning, can actually generate the results where they need to be.

The computational process by which this is performed is best outlined by analogy. Consider the problem mesh as a set of mutually repulsive particles; spread them randomly on a two-dimensional plane; allow the system to relax into a configuration of minimum potential energy, energy being defined as some suitable non-linear function of inter-particle distance. (Inconveniences such as local energetic minima can be dealt with as in simulated annealing.)

Mapping this solution technique onto the pair of discrete networks that is the processor:problem mesh, we allow each problem node to broadcast (via the NN packet protocol) a "field" (some scalar value that is decreased geometrically with every node hop.) Each node also accumulates an integral of all fields broadcast from every other node. Although the notion of slope is not easily defined in a discrete mesh with an arbitrary topology, the notion of relative size of (topologically) adjacent integrals is easy to capture - if a problem node detects a lower field integral on a processor node adjacent to the one to which it is currently mapped, it migrates there.

Thus the system (i.e. the problem:processor node map) will "relax" to a configuration of minimum energy, which corresponds to a maximal separation of problem nodes within the processor mesh.

Note this is an asymptotic process, and these are never realised comfortably on discrete networks - they can oscillate. Whilst this in itself is not an issue (we are attempting to solve a discrete placement and mapping problem, not solve a field), the reliable *detection* of this oscillation, and the identification of a steady state solution (the two are not the same thing) remain problematical.

3. ACKNOWLEDGMENTS

This work is supported by the Engineering and Physical Science Research Council (EPSRC), ARM Ltd, Silistix Ltd, and Thales Ltd, UK.