Programming a Million-Core Machine

Steve Furber
The University of Manchester
steve.furber@manchester.ac.uk
Bio-inspiration

• How can massively parallel computing resources accelerate our understanding of brain function?
• How can our growing understanding of brain function point the way to more efficient parallel, fault-tolerant computation?
• Brains demonstrate
  – massive parallelism (10^{11} neurons)
  – massive connectivity (10^{15} synapses)
  – excellent power-efficiency
    • much better than today’s microchips
  – low-performance components (~ 100 Hz)
  – low-speed communication (~ metres/sec)
  – adaptivity – tolerant of component failure
  – autonomous learning
Neurons

- multiple inputs, single output (c.f. logic gate)
- useful across multiple scales ($10^2$ to $10^{11}$)

Brain structure

- regularity
- e.g. 6-layer cortical ‘microarchitecture’
SpiNNaker project

- A million mobile phone processors in one computer
- Able to model about 1% of the human brain...
- ...or 10 mice!
Design principles

• Virtualised topology
  – physical and logical connectivity are decoupled

• Bounded asynchrony
  – time models itself

• Energy frugality
  – processors are free
  – the real cost of computation is energy
SpiNNaker node
SpiNNaker chip

Mobile DDR SDRAM interface

Multi-chip packaging by UNISEM Europe
48-node PCB
SpiNNaker platforms
103 machine: 864 cores, 1 PCB, 75W
105 machine: 103,680 cores, 1 cabinet, 9kW
104 machine: 10,368 cores, 1 rack, 900W (NB 12 PCBs for operation without aircon)

106 machine: 1M cores, 10 cabinets, 90kW
The networking challenge

- Emulate the very high connectivity of real neurons
- A spike generated by a neuron firing must be conveyed efficiently to >1,000 inputs
- On-chip and inter-chip spike communication should use the same delivery mechanism
Network – packets

• Four packet types
  – MC (multicast): source routed; carry events (spikes)
  – P2P (point-to-point): used for bootstrap, debug, monitoring, etc
  – NN (nearest neighbour): build address map, flood-fill code
  – FR (fixed route): carry 64-bit debug data to host

• Timestamp mechanism removes errant packets
  – which could otherwise circulate forever

Header (8 bits)  Event ID (32 bits)
T  ER  TS  0  -  P

Header (8 bits)  Address (16+16 bits)  Payload (32 bits)
T  SQ  TS  1  -  P  Dest  Srce
Network – MC Router

- All MC spike event packets are sent to a router
- Ternary CAM keeps router size manageable at 1024 entries (but careful network mapping also essential)
- CAM ‘hit’ yields a set of destinations for this spike event
  - automatic multicasting
- CAM ‘miss’ routes event to a ‘default’ output link

Event ID

<table>
<thead>
<tr>
<th>On-chip</th>
<th>Inter-chip</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 X 1 0 1</td>
<td>▶️ 000000010000010000 001001</td>
</tr>
</tbody>
</table>
Problem graph (circuit)
Problem: represented as a network of nodes with a certain **behaviour**...

Problem is split into two parts...  

**behaviour of each node** embodied as an interrupt handler in code...  

**abstract problem topology**...  

**compile, link**...  

**problem topology loaded into firmware routing tables**...  

**binary files loaded into core instruction memory**...

Our job is to make the model behaviour reflect reality

The code says "send message" but has no **control** where the output message goes
Bisection performance

- 1,024 links
  - in each direction
- ~10 billion packets/s
- 10Hz mean firing rate
- 250 Gbps bisection bandwidth
Partially-Ordered Event-Driven Systems

- A set of dynamical processes $P = \{P_i\}$
  - $S_i(t)$ is the state of $P_i$ at time $t$
- A set of event channels $E = \{E_j\}$
  - $E_j$ carries a time series of asynchronous impulses
  - generated by a process $E_j = e_j(P_j)$
- Hybrid model (biology): processes evolve $S_i = s_i(t, E^* \subseteq E)$
- Discrete model (*SpiNNaker*)
  - time can be abstracted into a series of (e.g.) 1ms events $E_t$
  - We can model each event atomically: $E_j \Rightarrow S_i := p_i(S_i, j)$
- In practice, on *SpiNNaker* event handling takes a finite time and may overlap subsequent events.
Event-driven software model

- **Packet event**
  - Buffer packet
  - Trigger DMA
  - DMA event
  - Update synapses
  - DMA transfer results (synaptic data copy)

- **Control flow**
  - Data flow

- **Timer event**
  - Update neurons
  - Synaptic inputs
Event-driven software model
PyNN design flow
PyNN integration

- LIF

- Izhikevich
PyNN integration

- Vogels-Abbott benchmark
  - 500 LIF neurons
SpiNNaker vision

The system is composed by
NENGO robot with place cells
• SpiNNaker:
  • 5M conn/s/ARM
• Spaun:
  • 2.5M neurons
  • ~100Hz firing rates
  • ~500 inputs/neuron
  • 125G conn/s
• Real-time Spaun:
  • 25,000 ARMs
  • 30x 48-node PCB
  • by end 2013?

Conclusions

- Brains represent a significant computational challenge
  - now coming within range?
- **SpiNNaker** is driven by the brain modelling objective
  - virtualised topology, bounded asynchrony, energy frugality
- The major architectural innovation is the multicast communications infrastructure
- We have working hardware & software
  - 48-node 864-ARM PCBs now
  - first multi-PCB systems now working