Spatial computation on a homogeneous, many-core architecture

Daniel Bates, Alex Bradbury, Andreas Koltes and Robert Mullins
Motivation: ever-changing tradeoffs

Heterogeneity won't be optimal forever

• Rising complexity
• Rising transistor costs

Source: Feature dimension reduction slowdown, by Handel Jones
Related work

Many projects in this field:

- Elm
- Raw/Tilera
- Rigel
- TRIPS
- ...

None have the same focus on flexibility and cooperation between cores.
Loki architecture: key features

- 4-stage, in-order, single-issue pipeline
- Deep network integration
  - No memory access stage
- Indirect network access
Loki architecture: key features

FETCH  DECODE  EXECUTE  WRITE BACK

NETWORK

Memory: addu r10, r11, r12 → 3
Loki architecture: key features

- **FETCH**: addu r10, r11, r12 → 3
- **DECODE**
- **EXECUTE**
- **WRITE BACK**: NETWORK

Memory: addu r10, r11, r12 → 3
Loki architecture: key features

FIND
addu r10, r11, r12 → 3

DECODE
RF
(3,4,2)

EXECUTE

WRITE BACK
NETWORK

NETWORK
Loki architecture: key features

FETCH  DECODE  EXECUTE  WRITE BACK

NETWORK

tile 3, core 4, channel 2

r10 = 8

8 → (3,4,2)

NETWORK
Loki architecture: key features

FETCH

DECODE

EXECUTE

WRITE BACK

NETWORK

8 → (3,4,2)
Previous work: execution patterns
Previous work: flexibility

- Mappings within one tile (8 cores)
- Complementary to this work
- Now exploring mappings across multiple tiles
This work: case studies

- Three kernels: sorting, FFT, matrix multiplication
- Explore different mappings
- Choose energy-performance-area tradeoff
Methodology

TSMC 40LP → Place & route, extraction

TSMC 40LP → HDL

HDL → Place & route, extraction

Place & route, extraction → Testbench and energy characterisation

Testbench and energy characterisation → Multiple regression analysis (R)

Multiple regression analysis (R) → Energy model

C code → Loki compiler (LLVM/Clang)

C code → Simulator (SystemC)

Loki compiler (LLVM/Clang) → Simulator (SystemC)

Simulator (SystemC) → Event counts

Event counts → Energy

Energy → Time

Time → Area

Area → Energy

Energy model → Energy

Energy model → Time

Time → Energy

Energy → Time

Time → Area

Area → Energy
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Loki</th>
<th>ARM1176</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Technology</strong></td>
<td>40nm LP</td>
<td>40nm LP</td>
</tr>
<tr>
<td><strong>Frequency/MHz</strong></td>
<td>435</td>
<td>700</td>
</tr>
<tr>
<td><strong>Area/mm² (core + L1)</strong></td>
<td>~0.125</td>
<td>~1</td>
</tr>
<tr>
<td><strong>Energy per instruction/pJ</strong></td>
<td>~5-20</td>
<td>~140</td>
</tr>
<tr>
<td><strong>Floating point support</strong></td>
<td>Software only</td>
<td>HW double-precision</td>
</tr>
</tbody>
</table>
Sorting

Critical path: 9 instructions

Graph showing energy consumption over time for two different processors: loki and arm1176. The graph indicates a comparison of energy efficiency over time with input data consisting of $2^{16}$ integers.
Triggered instructions (Parashar et al, ISCA 2013)

compare:
  when (!p0 and !in1.finished and !in2.finished)
  p1 = in1.read() < in2.read()
  p0 = 1

send1:
  when (p0 and p1)
  out.write(in1.read())
  p0 = 0

send2:
  when (p0 and !p1)
  out.write(in2.read())
  p0 = 0
Features communication-centric architectures can borrow:

- Blocking communication
- Peek operation
- Data streams
- Triggered branch

Critical path: 4 instructions
FFT

\[
\begin{array}{cccccccc}
0 & 4 & 2 & 6 & 1 & 5 & 3 & 7 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
\quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
\end{array}
\]

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array}
\]
Software coherence

- Take the SWEL approach (Pugsley et al, PACT 2010)
  - Cache data in the lowest-level shared cache
  - Requires L1 bypass
- Also looking into configurable cache hierarchies
Matrix multiplication - vector

Input: two 128 x 128 integer matrices
Matrix multiplication - scalarised

![Diagram showing matrix multiplication and energy consumption over time for different core configurations.](image)
Matrix multiplication – systolic array
Summary

- We can use large numbers of cores effectively
- Promising performance and energy figures
- A step towards a fully-homogeneous system
- Flexible communication is key
What's next?

- Memory system
- Network protocols
- Compiler optimisations
- Programmable accelerators
- Test chip (2015)
  - 4 x 4 tiles x 8 cores/tile = 128 cores
  - Hoping to share development boards
- Operating system
Spatial computation on a homogeneous, many-core architecture

Daniel Bates, Alex Bradbury, Andreas Koltes and Robert Mullins

Daniel.Bates@cl.cam.ac.uk

www.cl.cam.ac.uk/~rdm34/loki/

PRISM-2