The 48 core SCC Processor: A programmer’s view

Timothy G. Mattson*, Rob F. Van der Wijngaart, Michael Riepen, Thomas Lehnig, Paul Brett, Werner Haas, Patrick Kennedy, Jason Howard, Sriram Vangal, Nitin Borkar, Greg Ruhl, Saurabh Dighe

Intel Corporation

The Single-Chip Cloud Computer (SCC) experimental processor is a 48-core ‘concept vehicle’ created by Intel Labs as a platform for many-core software research.
Disclosure

- The views expressed in this talk are those of the speaker and not his employer.
- I am in a research group and know nothing about Intel products. So anything I say about them is highly suspect.
- This was a team effort, but if I say anything really stupid, it’s all my fault ... don’t blame my collaborators.

Note: The SCC processor is not a product today, and may never be a product. This is a research processor to explore software and architectural issues as we plot our many core future.
Agenda

• Motivation
  – The TeraScale Research Program

• The 48 core SCC processor
  – SCC Hardware and system software
  – Address spaces and Message passing on SCC
  – Results

• Next Steps
Parallel Hardware Trends
Top 500: total number of processors (1993-2010)

This is what the many core revolution looks like to HPC

Source: the “June lists” from www.top500.org
The many core design challenge

• **Scalable architecture:**
  – How should we connect the cores so we can scale as far as we need?

• **Software:**
  – How should “general purpose programmers” write software for scalable many-core architectures?

• **Power Management**
  – If programmers are given control over voltage and frequency, can they improve performance/watt?

• **Manufacturability**:
  – Can we use tiled architectures to reduce design/validation costs?

Intel’s “TeraScale” processor research program is addressing these questions with a series of Test chips ... two so far.
Agenda

• Motivation
  - The TeraScale Research Program

• The 48 core SCC processor
  - The SCC Platform (Hardware and system software)
    - Address spaces and Message passing on SCC
    - Results

• Next Steps
Hardware view of SCC

- 48 P54C cores, 6x4 mesh, 2 cores per tile
- 45 nm, 1.3 B transistors, 25 to 125 W
- 16 to 64 GB DRAM using 4 DDR3 MCs
- 2 Tb/s bisection bandwidth @ 2 Ghz

**Technology**

<table>
<thead>
<tr>
<th>Process</th>
<th>45nm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transistors</td>
<td>Die: 1.3B</td>
</tr>
<tr>
<td>Die Area</td>
<td>Tile: 48M</td>
</tr>
<tr>
<td>Die Area</td>
<td>567.1mm²</td>
</tr>
</tbody>
</table>

R = router, MC = Memory Controller, P54C = second generation Pentium® core, CC = cache cntrl.
Router Architecture

- Input Arbitration
- FIFO
- Route Pre-compute
- Switch Arbitration
- VC Allocation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>2GHz @ 1.1V</td>
</tr>
<tr>
<td>Latency</td>
<td>4 cycles</td>
</tr>
<tr>
<td>Link Width</td>
<td>16 Bytes</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>64GB/s per link</td>
</tr>
<tr>
<td>Architecture</td>
<td>8 VCs over 2 MCs</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>500mW @ 50°C</td>
</tr>
<tr>
<td>Routing</td>
<td>Pre-computed, X-Y</td>
</tr>
</tbody>
</table>

Source: Howard et. al. 48 core SCC processor, ISSCC 2010
Power breakdown

Full Power Breakdown
Total - 125.3W

- MC & DDR3-800: 19%
- Cores: 69%
- Routers & 2D-mesh: 10%
- Global Clocking: 2%

Clocking: 1.9W
Cores: 87.7W
MCs: 23.6W

Low Power Breakdown
Total - 24.7W

- MC & DDR3-800: 69%
- Cores: 21%
- Routers & 2D-mesh: 5%
- Global Clocking: 5%

Clocking: 1.2W
Cores: 5.1W
MCs: 17.2W

Source: Howard et. al. 48 core SCC processor, ISSCC 2010
Power and memory domains

**Power Control domains**

- 7 Voltage domains (6 4-tile blocks plus one for the on-die network)
- 24 tile clock frequency dividers
- One voltage control register ... so only one voltage change in flight at a time.

*How cores map to MC is under programmer control .... i.e. the memory-controller domain is configurable.*
**SCC Platforms**

- Three platforms for SCC and RCCE*
  - Functional emulator (on top of OpenMP)
  - SCC board with two “OS Flavors” ... Linux or Baremetal (i.e. no OS)

*RCCE: Native Message passing library for SCC

Third party names are the property of their owners.
Networking: TCP/IP

- We support the standard “Internet protocol stack”

<table>
<thead>
<tr>
<th>Application</th>
<th>network applications (ssh, http, MPI, NFS …)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport</td>
<td>host-host data transfer (TCP, UDP)</td>
</tr>
<tr>
<td>Network</td>
<td>routing of datagrams from source to destination (IP)</td>
</tr>
<tr>
<td>Link</td>
<td>data transfer between peers (Ethernet, rckmb)</td>
</tr>
<tr>
<td>Physical</td>
<td>More packets over physical network (on-die mesh)</td>
</tr>
</tbody>
</table>

- Rckmb … our implementation of the Link layer
  - Local write/remote read data transfer
  - Static mapping IP address $\leftrightarrow$ SCC core number
  - Uses IP packet to determine destination

- Seamless integration with Linux networking utilities

- Rckmb is based on NAPI
  - Interrupt shared between peer nodes
  - Provides for operation in polling mode

Source: Van der Wijngaart, Mattson and Haas, submitted to ACM OS review, 2010
Agenda

• Motivation
  – The TeraScale Research Program

• The 48 core SCC processor
  – The SCC Platform (Hardware and system software)
  – Address spaces and Message passing on SCC
  – Results

• Next Steps
SCC Address spaces

- 48 x86 cores which use the x86 memory model for Private DRAM

On-chip Memory

Off-chip Memory

Where is the physical Memory

Cache utilization

Shared off-chip DRAM (variable size)

Private DRAM

CPU_0

Private DRAM

CPU_47

Shared on-chip Message Passing Buffer (8KB/core)

Shared test and set register
SCC Address spaces: Msg. Pass. (RCCE) Configuration

- RCCE is a message passing library and thinks of the chip as a distributed memory platform ... we have no use for shared DRAM and instead emphasize private DRAM.
SCC Address spaces: Msg. Pass. (RCCE) Configuration

• RCCE is a message passing library and thinks of the chip as a distributed memory platform ... we have no use for shared DRAM and instead emphasize private DRAM.

To better understand how the Memory works on SCC, we will take a closer look at how RCCE is implemented.
How does RCCE work? Part 1

- Treat Msg Pass Buf (MPB) as 48 smaller buffers ... one per core.
- Symmetric name space ... Allocate memory as a collective op. Each core gets a variable with the given name at a fixed offset from the beginning of a core’s MPB.

```c
A = (double *) RCCE_malloc(size)
```
Called on all cores so any core can put/get(A at Core_ID) without error-prone explicit offsets

Flags allocated and used to coordinate memory ops
How does RCCE work? Part 2

- The foundation of RCCE is a one-sided put/get interface.
- Symmetric name space ... Allocate memory as a collective and put a variable with a given name into each core’s MPB.

... and use flags to make the put’s and get’s “safe”
Consequences of MPBT properties for RCCE:

- If data changed by another core and image still in L1, read returns stale data.
  - **Solution:** Invalidate before read.
- L1 has write-combining buffer; write incomplete line? expect trouble!
  - **Solution:** don’t. Always push whole cache lines
- If image of line to be written already in L1, write will not go to memory.
  - **Solution:** invalidate before write.

Message passing buffer memory is special ... of type MPBT
Cached in L1, L2 bypassed. Not coherent between cores
Cache (line) allocated on read, not on write.
Single cycle op to invalidate all MPBT in L1 ... Note this is not a flush

Discourage user operations on data in MPB. Use only as a data movement area managed by RCCE ... Invalidate early, invalidate often
Agenda

• Motivation
  – The TeraScale Research Program

• The 48 core SCC processor
  – The SCC Platform (Hardware and system software)
  – Address spaces and Message passing on SCC
  – Results

• Next Steps
Round-trip Latencies, 32 byte message

- Ping-pong between a pair of cores ... one fixed at a corner, the second core varied from “same tile” to opposite corner

5.10E-06
5.15E-06
5.20E-06
5.25E-06
5.30E-06
5.35E-06
5.40E-06
5.45E-06
5.50E-06
5.55E-06
5.60E-06

Latency (secs)

Network hops

Data fit to a straight line:

\[ T = 5 \text{ microsecs} + 30 \text{ nanosecs} \times \text{hops} \]

RCCE_Send/Recv ... 3 messages per transit, 6 for roundtrip

4 cycles per hop, 800 MHz router ... or \(2 \times 3 \times 4 / 0.8 \text{ GHz} = 30 \text{ nanosecs.}\)

533 MHz tile, 800 MHz Mesh, 800 MHz DDR3
Ping pong bandwidths

• Point to point bandwidth between cores 0 and 46 (8 network hops).

4 performance regions
1. MSG < 8KB: no L1 conflicts.
2. 8KB < MSG < 24 KB: increasing L1 conflicts
3. 24 KB < MSG < 256 KB: saturated L1; performance dictated by L2 bandwidth
4. 256 KB < MSG: saturated L2;

• Case A, C and D use 32 Byte flags. Cases A and B use the full MPB.
NAS Parallel benchmarks

1. **BT: Multipartition decomposition**
   - Each core owns multiple blocks (3 in this case)
   - update all blocks in plane of 3x3 blocks
   - send data to neighbor blocks in next plane
   - update next plane of 3x3 blocks

2. **LU: Pencil decomposition**
   - Define 2D-pipeline process.
     - await data (bottom+left)
     - compute new tile
     - send data (top+right)

Third party names are the property of their owners.
Problem size: Class A, 64 x 64 x 64 grid*

SCC processor 500MHz core, 800 MHz routers, 25MHz system interface, and DDR3 memory at 800 MHz.

* These are not official NAS Parallel benchmark results.

Third party names are the property of their owners.
Dense Linear Algebra on SCC
Elemental: A new C++ library for dense linear algebra

Torus-wrap/elemental mapping of matrices to a two-dimensional process grid. Runs on top of RCCE. Goal … a modern replacement for ScaLAPACK.

Ernie Chan, Bryan Marker, Jack Poulson, and Robert van de Geijn of UT Austin
RCCE vs. MPI over TCP/IP

Ping-pong core 00 to 46 (polling, MPB)

Bandwidth (MB/s)

Message size (bytes ... log₂ scale)

533 MHz tile, 800 MHz Mesh, 800 MHz DDR3
Agenda

• Motivation
  – The TeraScale Research Program

• The 48 core SCC processor
  – The SCC Platform (Hardware and system software)
  – Address spaces and Message passing on SCC
  – Results

• Next Steps
Remember our RCCE TCP/IP comparison?

Let’s take a closer look at this data large messages (on the next slide)
RCCE vs. MPI over TCP/IP

Ping-pong core 00 to 46 (polling, MPB)

Bandwidth (MB/s)

MPI Polling SRAM (MPB)

RCCE

Message size (bytes ... log₂ scale)

533 MHz tile, 800 MHz Mesh, 800 MHz DDR3
533 MHz tile, 800 MHz Mesh, 800 MHz DDR3

The story gets more complicated when we consider a TCP/IP solution that uses interrupts instead of polling.
Why is RCCE so much slower than MPI for large messages? Actually, all are slow for large messages.

- We know how to improve RCCE:
  - Pipeline packets for RCCE (~2x)
  - Use an optimized memcpy (~3x)

Should be ~½ Stream bandwidth. Stream runs ~60 MB/s.

533 MHz tile, 800 MHz Mesh, 800 MHz DDR3
Conclusions

- **RCCE software works**
  - RCCE’s restrictions (Symmetric MPB memory model and blocking communications) have not been a fundamental obstacle
  - Functional emulator is a useful development/debug device

- **SCC architecture**
  - The on-chip MPB was effective for scalable message passing applications
  - Software controlled power management works ... but it’s challenging to use because (1) granularity of 8 cores and (2) high latencies for voltage changes

- **Future work**
  - Improve MPI by (1) turning TCP parameters to SCC network and then (2) implement MPI over native SCC transport layer instead of TCP/IP.
  - Intel started a program in Q3’10 to collaborate with ~100 external research groups to explore the SCC chip. Stay tuned for results.