A Triple Hybrid Interconnect for Many-Cores: Reconfigurable Mesh, NoC and Barrier

Heiner Giefers, Marco Platzner
Computer Engineering Group
University of Paderborn

{hgiefer, platzner}@upb.de
Motivation

• Parallel applications on many-core architectures show different communication requirements

• Today’s interconnect of choice is the packet switched NoCs
  - Scalable, allows for resource sharing
  - Broadcast and multicast are typically slow or expensive

• Can we improve performance by using a hybrid interconnect?
  - Select interconnect depending on communication pattern
Outline

I. Many-core architecture
   • Packet switched NoC
   • Reconfigurable mesh network
   • Barrier synchronization network
   • Resource utilization results
   • Communication API

II. Experimental results
   • Operand communication
   • Application case study
Triple Hybrid Interconnect for Many-Cores

- Packet switched NoC
- Reconfigurable mesh network
- Barrier synchronization network
Packet Switched NoC

- Network interface: fast simplex link
- Buffers: 16 entry FIFOs
- Routing: 2D dimension order routing (XY routing)
- Arbitration: static priorities, blocking
- Flow control: wormhole routing, handshake signals
  → 2 cycle flit latency per hop
Reconfigurable Mesh Network

- Based on the reconfigurable mesh model

- Cores can autonomously reconfigure their local switch pattern
  - Build up global topologies
  - NIC interprets a control-labeled word as switch configuration

- Native support for broadcast and multicast

- Investigated reconfigurable mesh FPGA implementations earlier
  - Picoblaze processing elements, e.g., [FPL’07] and [FPL’09]
  - Microblaze processing elements [ERSA’10]

→ Single cycle latency per hop
Barrier Synchronization Network

- Beneficial for parallel algorithms that require global synchronization

- Bit-level AND-reduce tree
  - Implemented in FPGA’s fast carry chains
  - Single-cycle evaluation

- Use the FSL capabilities to stall processors while waiting for a barrier
  - One \texttt{fslput} to signal a ‘1’ onto the AND-reduce tree
  - A subsequent \texttt{fslget} stalls the Microblaze until all nodes have reached the barrier
  - Only use FSL ports, no FIFO

→ 3-cycle latency for global barrier
Resource Utilization Results (XC5VLX110T)

- Tiled architecture
  - µB + 8-bit router + 32-bit switch + barrier core + 16kByte BRAM
  - Operate prototypes up to 30 tiles

- Switch is almost 2x as fast as a router

<table>
<thead>
<tr>
<th>Module</th>
<th>LUT</th>
<th>FF</th>
<th>DSP</th>
<th>BRAM</th>
<th>$f_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6x5 tiles</td>
<td>66175</td>
<td>35352</td>
<td>60</td>
<td>120</td>
<td>102 MHz</td>
</tr>
<tr>
<td></td>
<td>95%</td>
<td>51%</td>
<td>93%</td>
<td>81%</td>
<td></td>
</tr>
<tr>
<td>Router (8-bit)</td>
<td>385</td>
<td>286</td>
<td>0</td>
<td>0</td>
<td>247 MHz</td>
</tr>
<tr>
<td>Router (32-bit)</td>
<td>679</td>
<td>398</td>
<td>0</td>
<td>0</td>
<td>240 MHz</td>
</tr>
<tr>
<td>Switch (8-bit)</td>
<td>133</td>
<td>36</td>
<td>0</td>
<td>0</td>
<td>448 MHz</td>
</tr>
<tr>
<td>Switch (32-bit)</td>
<td>446</td>
<td>132</td>
<td>0</td>
<td>0</td>
<td>448 MHz</td>
</tr>
<tr>
<td>Barrier Core</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>519 MHz</td>
</tr>
</tbody>
</table>

¹ Only 64 DSPs available. One µB utilizes 3 DSPs for build-in HW-multiplier
### Communication API

<table>
<thead>
<tr>
<th>NoC</th>
<th>Reconfigurable Mesh</th>
<th>Barrier Net</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>send(T,D)</td>
<td></td>
<td></td>
<td>Reconfigure local switch element</td>
</tr>
<tr>
<td>sendMsg(T,*D,L)</td>
<td></td>
<td></td>
<td>Send single value</td>
</tr>
<tr>
<td>receive()</td>
<td></td>
<td>swconf(P)</td>
<td>Send message of L values</td>
</tr>
<tr>
<td>receiveMsg(*D,*S)</td>
<td></td>
<td>broadcastMsg(*D,L)</td>
<td>Receive a single value</td>
</tr>
<tr>
<td></td>
<td>broadcast(D)</td>
<td></td>
<td>Receive a message of L values</td>
</tr>
<tr>
<td></td>
<td>read()</td>
<td>sync</td>
<td>Set synchronization point</td>
</tr>
<tr>
<td></td>
<td>read(*D,L)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Use standard microblaze-gcc toolchain
- \( D \): 32-bit data (integer or float)
- \( S/T \): 8-bit source/target address
- \( P \): 4-bit reconfigurable mesh switch pattern
- \( L \): 8-bit length of message in 32-bit words
Outline

I. Many-core architecture
   • Packet switched NoC
   • Reconfigurable mesh network
   • Barrier synchronization network
   • Resource utilization results
   • Communication API

II. Experimental results
   • Operand communication
   • Application case study
Operand Communication

**Communication Pattern**

<table>
<thead>
<tr>
<th>Communication Pattern</th>
<th>Reconf.</th>
<th>NoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:n (broadcast)</td>
<td>74</td>
<td>1118</td>
</tr>
<tr>
<td>1:1 (node 0 to node 29)</td>
<td>74</td>
<td>108</td>
</tr>
<tr>
<td>Multicast on first row</td>
<td>69</td>
<td>297</td>
</tr>
<tr>
<td>n:1 (nodes 1..29 to node 0)</td>
<td>733</td>
<td>402</td>
</tr>
</tbody>
</table>

- NoC-broadcast or -multicast is handled by sending separate messages
- Overhead for setting up and consuming messages is relatively high
- NoC is more efficient for n:1 and n:m communication
Case Study: Jacobi Method

- Given a square system of linear equations $Ax = b$
- Iteratively solve $x^{(\mu+1)}_i = \frac{b_i}{a_{i,i}} - \sum_{k=1 \atop k \neq i}^{n} \frac{a_{i,k}}{a_{i,i}} x^{\mu}_k$ ($i = 1, 2, \ldots, n$)

$$
\begin{pmatrix}
 a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\
 a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\
 a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\
 a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4}
\end{pmatrix}
\begin{pmatrix}
 x^0_1 \\
 x^0_2 \\
 x^0_3 \\
 x^0_4
\end{pmatrix}
= 
\begin{pmatrix}
 b_1 \\
 b_2 \\
 b_3 \\
 b_4
\end{pmatrix}

1. Distribute $x^{\mu}_i$ on columns
2. All nodes compute products $a_{i,k} x^{\mu}_i$ in parallel
3. Send product to diagonal node of the same row
4. Diagonal nodes compute $x^{\mu+1}_i$
Case Study: Results

- Jacobi method trades on the NoC capabilities
- However, combined use of reconfigurable mesh, barrier and NoC brings additional performance
Summary and Outlook

• Approach: use different interconnects for different communication patterns in applications

• Introduced a triple hybrid interconnect

• Hybrid interconnect provides opportunities
  - NoC is advantageous for larger messages and dynamic workloads
  - Reconfigurable mesh features fast operand broad-/multicast
  - Barrier network brings fast global synchronization

• Jacobi method case study benefits from a combined use of all networks

• Outlook
  - Improve programming tool-flow: automate interconnect selection
  - Detailed analysis: energy consumption
  - Comparison: Hybrid interconnect vs. more complex NoC