OpenSoC Fabric
An On-Chip Network Generator

Farzad Fatollahi-Fard, Dave Donofrio, George Michelogiannakis, John Shalf
<table>
<thead>
<tr>
<th>Motivation</th>
<th>OpenSoC Fabric</th>
<th>What is Chisel?</th>
<th>OpenSoC Fabric Breakdown</th>
<th>Conclusion and Future Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>OpenSoC Fabric</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Results</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Motivation
Why Are We Doing This?

› Want to build and model candidate future HPC chip multiprocessors

Parallelism is growing at exponential rate

Data movement dominates power costs

Network topology greatly affects application performance

An analysis of on-chip interconnection networks for large-scale chip multiprocessors
ACM Transactions on computer architecture and code optimization (TACO), April 2010
What Interconnect Provides the Performance? Is it Open Source?

What tools exist to answer these questions?
What tools exist for SoC research

What tools do we have to evaluate large, complex networks of cores?

- **Software models**
  - Fast to create, but plagued by long runtimes as system size increases

- **Hardware emulation**
  - Fast, accurate evaluate that scales with system size but suffers from long development time

A complexity-effective architecture for accelerating full-system multiprocessor simulations using FPGAs. FPGA 2008
## Comparison of NoCs

### Software Tools

| NoC       | Language     | Accuracy     | Verification | Drawbacks                                                      |
|-----------|--------------|--------------|--------------|                                                               |
| Booksim   | C++          | Cycle-Accurate | RTL          | Long runtimes limit simulation size                           |
| Garnet    | C++ (GEM5)   | Event-Driven | Other Simulators | Not fast enough for larger simulations (1K+ cores)          |
| NoCTweak  | SystemC      | Cycle-Accurate | RTL          | Long runtimes limit simulation size                           |
| PhoenixSim| OMNeT++      | Event-Driven | Other Simulators | For Photonics on-chip networks                               |
| Topaz     | C++ (GEM5)   | Cycle-Accurate | Other Simulators | Not fast enough for larger simulations (1K+ cores)         |
## Comparison of NoCs

### Hardware Tools

<table>
<thead>
<tr>
<th>NoC</th>
<th>Language</th>
<th>Features</th>
<th>Open Source?</th>
<th>Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford NoC Router</td>
<td>Verilog</td>
<td>Long list of Verilog parameters</td>
<td>Yes</td>
<td>- Hard to configure</td>
</tr>
<tr>
<td></td>
<td>Bluspec</td>
<td>Completely customizable via website</td>
<td>Yes (noncommercial)</td>
<td>- Designed for FPGAs</td>
</tr>
<tr>
<td>CONNECT</td>
<td>SystemVerilog</td>
<td></td>
<td></td>
<td>- Designed for ARM cores (not design space exploration)</td>
</tr>
<tr>
<td>ARM CoreLink</td>
<td>Pre-generated IP</td>
<td>Up to clusters of 48 cores</td>
<td>No</td>
<td>- For “small” designs - Cache Coherent</td>
</tr>
<tr>
<td>Arteris FlexNoC</td>
<td>Pre-generated IP</td>
<td>Tool optimized for VLSI design</td>
<td>No</td>
<td>- Full parameters unknown</td>
</tr>
</tbody>
</table>
Motivation

OpenSoC Fabric

What is Chisel?

OpenSoC Fabric Breakdown

Results

Conclusion and Future Work
OpenSoC Fabric
An Open-Source, Flexible, Parameterized, NoC Generator

- Part of the CoDEx tool suite
- Written in Chisel
- Dimensions, topology, VCs all configurable
- Fast functional C++ model for functional validation
- Verilog based description for FPGA or ASIC
  - Synthesis path enables accurate power / energy modeling
Current Status
Version 1.1.2 Released

- Multiple Topologies
  - Mesh
  - Flattened Butterfly
- Wormhole Flow Control
- Virtual Channels
- Run both through ASIC and FPGA tools
- Available for download
  - www.opensocfabric.org
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation</td>
<td>1</td>
</tr>
<tr>
<td>OpenSoC Fabric</td>
<td>2</td>
</tr>
<tr>
<td>What is Chisel?</td>
<td>3</td>
</tr>
<tr>
<td>OpenSoC Fabric Breakdown</td>
<td>4</td>
</tr>
<tr>
<td>Results</td>
<td>5</td>
</tr>
<tr>
<td>Conclusion and Future Work</td>
<td>6</td>
</tr>
</tbody>
</table>
Chisel: A New Hardware DSL
Using Scala to construct Verilog and C++ descriptions

- Chisel provides both software and hardware models from the same codebase
- Object-oriented hardware development
  - Allows definition of structs and other high-level constructs
- Powerful libraries and components ready to use
- Working processors fabricated using chisel
Recent Chisel Designs

Chisel code successfully boots Linux

• First tape-out in 2012
• Raven core taped out in 2014 – 28nm
Chisel Overview

How does Chisel work?

- Not “Scala to Gates”
- Describe hardware functionality
- Chisel creates graph representation
  - Flattened
- Each node translated to Verilog or C++
OpenSoC – Top Level Diagram
OpenSoC – Functional Hierarchy

- **Network Interface**
  - AXI
  - AHB
  - FIFO
- **Router**
- **Top-Level**
- **Injection/Ejection**
- **Routing Function**
  - Mesh
  - Flattened Butterfly
  - Torus
- **Allocator**
- **Arbiter**
  - Round Robin
  - Priority
  - Cyclic
Configuring

Parameters

- **OpenSoC configured at run time through *Parameters* class**
  - Declared at top level, sub modules can add / change parameters tree

- **Not limited to just numerical values**
  - Leverage Scala to pass functions to parameterize module creation
    - Example: Routing Function constructor passed as parameter to router
Configuring Parameters

- All OpenSoC Modules take a Parameters class as a constructor argument

- Setting parameters:
  - `parms.child("MySwitch", Map( ("numInPorts"->Soft(8)), ("numOutPorts"->Soft(3)) ))`

- Getting a parameter:
  - `val numInPorts = parms.get[Int]("numInPorts")`
Developing
Incredibly Fast Development Time

- Modules have a standard interface that you inherit
- Development of modules is very quick
  - Flattened Butterfly took 2 hours of development

abstract class VCRouter(parms: Parameters) extends Module(parms) {
  val numInChannels = parms.get[Int]("numInChannels")
  val numOutChannels = parms.get[Int]("numOutChannels")
  val nunVCs = parms.get[Int]("numVCs")
  val io = new Bundle {
    val inChannels = Vec.fill(numInChannels) { new ChannelVC(parms) }
    val outChannels = Vec.fill(numOutChannels) { new ChannelVC(parms).flip() }
  }
}

class SimpleVCRouter(parms: Parameters) extends VCRouter(parms) {
  // Implementation
}
OpenSoC – Functional Hierarchy
OpenSoC – Top Level Modules

- Stiches routers together
- Assigns routers individual ID
- Assigns Routing Function to routers
- Passes down Arbitration scheme
- Connections Injection and Ejection Queues for network endpoints
Results – Traffic Patterns

Uniform Random Traffic

- OpenSoC
- Booksim

4x4 DOR
Single Concentration
Dual Virtual Channel
Mesh Network

Average Packet Network Latency (clocks)

Packet Injection Rate %
## Results – Average Latency

<table>
<thead>
<tr>
<th>Compared to Booksim</th>
<th>OpenSoC Fabric (Software)</th>
<th>OpenSoC Fabric (Hardware)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uniform</strong></td>
<td>+1.86%</td>
<td>+8.37%</td>
</tr>
<tr>
<td><strong>Tornado</strong></td>
<td>+0.84%</td>
<td>+0.42%</td>
</tr>
<tr>
<td><strong>Transpose</strong></td>
<td>+7.37%</td>
<td>+8.29%</td>
</tr>
<tr>
<td><strong>Neighbor</strong></td>
<td>+0.84%</td>
<td>+6.28%</td>
</tr>
<tr>
<td><strong>Bit Reverse</strong></td>
<td>+1.85%</td>
<td>+10.6%</td>
</tr>
</tbody>
</table>
Results – Latency and Utilization

Nearest Neighbor Traffic Pattern
## Results – Application Traces

<table>
<thead>
<tr>
<th>Compared to Booksim</th>
<th>OpenSoC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AMR Avg latency</strong></td>
<td>-2.42%</td>
</tr>
<tr>
<td><strong>MiniDFT Avg latency</strong></td>
<td>-28.3%</td>
</tr>
<tr>
<td><strong>AMG Avg latency</strong></td>
<td>+16.3%</td>
</tr>
<tr>
<td><strong>AMR Execution time</strong></td>
<td>-2.19%</td>
</tr>
<tr>
<td><strong>MiniDFT Execution time</strong></td>
<td>-5.25%</td>
</tr>
<tr>
<td><strong>AMG Execution time</strong></td>
<td>+130.8%</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Motivation</td>
<td>1</td>
</tr>
<tr>
<td>OpenSoC Fabric</td>
<td>2</td>
</tr>
<tr>
<td>What is Chisel?</td>
<td>3</td>
</tr>
<tr>
<td>OpenSoC Fabric Breakdown</td>
<td>4</td>
</tr>
<tr>
<td>Results</td>
<td>5</td>
</tr>
<tr>
<td>Conclusion and Future Work</td>
<td>6</td>
</tr>
</tbody>
</table>
Future additions
Towards a full set of features

- Upgrade OpenSoC Fabric to use Chisel 3
- A collection of topologies and routing functions
- Standardized interfaces at the endpoints
- Power modeling in the C++ model
Conclusion

- This is an open-source community-driven infrastructure
  - We are counting on your contributions
Acknowledgements

- UCB Chisel
- US Dept of Energy
- Laboratory for Physical Sciences
- Ke Wen
- Columbia LRL
- John Bachan
More Information

http://opensocfabric.org