Ultra-Fast NoC Emulation on a Single FPGA

Thiem Van Chu, Shimpei Sato, and Kenji Kise
Tokyo Institute of Technology
Contributions

Methodologies for emulating Network-on-Chip (NoC) architectures with up to 1000s of nodes on a single FPGA

Up to 16384 nodes

Load-latency graph

Cycle-accurate & 5000x simulation speedup over BookSim\textsuperscript{1)}, a widely used software simulator

High end PC
Core i7 4770 CPU
32GB RAM

BookSim\textsuperscript{1)}: several days

Proposal: several minutes

Virtex-7 FPGA

\textsuperscript{1)}http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim
Multi/Many-Core Architectures Have Become Mainstream

1000s cores in near future architectures

Network-on-Chip (NoC)

- The interconnection network of many-core architectures
- NoC simulation plays a vital role in designing many-core architectures

A many-core architecture with 2D Mesh NoC
Software Simulators

- Flexible and easy to debug
- **Too slow** to simulate large architectures
- **Parallelization is non-trivial**
  - Without sacrificing accuracy, only a limited degree of parallelization can be achieved

<table>
<thead>
<tr>
<th>Typical Simulation Time of 1000s-Core Architectures</th>
<th>Simple Models</th>
<th>Moderate Models</th>
<th>Detailed Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Several <em>days</em></td>
<td></td>
<td>Several <em>months</em></td>
<td>Several <em>years</em></td>
</tr>
</tbody>
</table>
FPGA-Accelerated Simulation

- Avoids impractical designs
- Possible to reuse RTL code
- Can achieve an ultra-fast simulation speed
  - Many operations can be simulated simultaneously in a tick of FPGA’s clock
  - Adding detail to a model requires more hardware, but does not necessarily degrade performance
FPGA-Accelerated Simulation

**Challenges**

- **Single FPGA**
  - Limited resources
- **Multiple FPGAs**
  - More resources
  - More complex
  - Off-chip communication

---

Traffic Generation

- Unit 1
- ... ...
- Unit N

Source Queue 1

Source Queue N

Network

- Router
- ... ...
- Router

Software:
Dynamic memory allocation

**Memory constraint**

Must be *very large* to ensure that no generated packet is dropped
Proposals

Method 1: Decoupling Time Counters

Traffic Generation Unit 1

... Source Queue 1

Traffic Generation Unit N

... Source Queue N

Network

Router

Router

Must be very large to ensure that no generated packet is dropped

Eliminate the memory constraint

Method 2: Time-Multiplexing

Logical clusters

Physical cluster

Simulate architectures with 1000s of cores on a single FPGA
Emulation Model

Three basic components
- **Router**: a state-of-the-art pipelined router architecture
- **Traffic generator**: generates and injects synthetic workloads
- **Traffic sink**: collects performance characteristics

![The primary router model](image-url)
Emulation Model: 2D Mesh

- **Packet source**: models injection processes (e.g. Bernoulli process)
- **Source queue**: every packet generated by the packet source is stored in the source queue until it can enter the network
- **Flit generator**: models traffic patterns (e.g. uniform random)
Decoupling Time Counters

Conventional approach
- Every packet source is synchronized with the network

The source queues must be very large to cope with the case when the packet sources generate so many packets that the network becomes very congested.
Decoupling Time Counters

Proposal

- Each packet source, as well as the network, has its own time counter and operates based on a separate state machine.

The state transitions are based on the status of the source queues and the relationship between the time counters.
Decoupling Time Counters

- **W** Waiting
- **R** Running

**State Machine of Packet Source i**

- \( PS_i \) intends to generate a packet but \( SQ_i \) is full
- One space in \( SQ_i \) becomes available

No packet is dropped

But...

Some packets may not be generated on time

\[ T \]

Packet Source 1

\[ T \]

Packet Source N

Source Queue 1

Network

Router

Router
Decoupling Time Counters

Waiting

Running

State Machine of the Network

\[ \forall i \text{ } SQ_i \text{ is empty } \& \text{ } T_i < T \]

\[ \exists i \text{ } SQ_i \text{ is empty } \& \text{ } T_i < T \]

\[ \forall i \text{ } SQ_i \text{ is NOT empty } \text{ OR } T_i = T \]

∀i either \( SQ_i \) contains at least one packet or the time of \( PS_i \) is synchronized with the time of the network

The specified synthetic workload is simulated accurately

\( T \)
Time-Multiplexing

To complete one cycle of the network, the physical cluster sequentially emulates a number of logical clusters.
Time-Multiplexing

- Combinational logic and block RAMs (BRAMs) are utilized much more efficiently because they can be shared between many NoC nodes
- Example: 128x128 mesh NoC

Direct implementation
5x128x128 = 81,920 BRAMs

Using time-multiplexing
< 500 BRAMs

Method for translating to emulation code
... Please see the paper

Input buffers can be implemented using BRAMs
Datapath

- Physical cluster emulates different logical clusters using different states loaded from state memory
- In buffer and out buffer store data passed between logical clusters

1. Load a state & Emulate the corresponding logical cluster
2. Store the updated state
Datapath

- Physical cluster emulates different logical clusters using different states loaded from state memory
- **In buffer** and **out buffer** store data passed between logical clusters
Example

Need $D_0^i$, $D_2^i$, $D_5^i$

$D_0^i$: In Buffer

$D_2^i$, $D_5^i$: Out Buffer

Emulated
Emulating
To be emulated

Logical clusters

Emulation cycle

$D_0^i$, $D_1^i$ $D_7^i$

$D_0^{i+1}$, $D_1^{i+1}$
Evaluation and Analysis

- 128 × 128 mesh NoC (16,384 nodes) on a Xilinx VC707 board
- Four NoC designs
  - 5-stage 2-VC: canonical 5-stage pipelined VC router architecture with 2 VCs per port
  - 5-stage 1-VC: canonical 5-stage pipelined VC router architecture with 1 VC per port
  - 4-stage 2-VC: canonical 4-stage pipelined VC router architecture with 2 VCs per port
  - 4-stage 1-VC: canonical 4-stage pipelined VC router architecture with 1 VC per port
- Three configurations
  - 4-phy (2 × 2): use four nodes to emulate the entire 128 × 128 mesh network
  - 16-phy (4 × 4): use 16 nodes to emulate the entire 128 × 128 mesh network
  - 32-phy (8 × 4): use 32 nodes to emulate the entire 128 × 128 mesh network
- Metrics
  - Hardware usage
  - Verification against BookSim, a widely used cycle-accurate software simulator
  - Simulation performance: speedup over BookSim
## Configuration Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology</td>
<td>128x128 mesh <em>(16,384 nodes)</em></td>
</tr>
<tr>
<td>Router architecture</td>
<td>5-stage pipelined VC router <em>or</em> 4-stage pipelined VC router (employing look-ahead routing)</td>
</tr>
<tr>
<td># of VCs per port</td>
<td>2 <em>or</em> 1</td>
</tr>
<tr>
<td>Routing algorithm</td>
<td>Dimension-order (XY)</td>
</tr>
<tr>
<td>Flow control</td>
<td>Credit-based</td>
</tr>
<tr>
<td>VC/Switch allocator</td>
<td>Separable output first</td>
</tr>
<tr>
<td>Arbiter type</td>
<td>Fixed priority</td>
</tr>
<tr>
<td>Flit size</td>
<td>25-bit <em>or</em> 22-bit</td>
</tr>
<tr>
<td>VC size</td>
<td>4-flit</td>
</tr>
<tr>
<td>Packet length</td>
<td>8-flit</td>
</tr>
<tr>
<td>Injection process</td>
<td>Bernoulli</td>
</tr>
<tr>
<td>Traffic pattern</td>
<td>Uniform random</td>
</tr>
<tr>
<td>Source queue length</td>
<td>8-entry</td>
</tr>
</tbody>
</table>
Simulate of a same 128 × 128 NoC design by using 4 physical nodes (4-phy), 16 physical nodes (16-phy), and 32 physical nodes (32-phy)

Use more physical nodes \(\rightarrow\) Consume more registers & logic

Only a minor difference in the number of occupied BRAMs
Accuracy

- The proposed methods do not affect the simulation accuracy
  - Synthetic workloads can be modeled accurately without using a large amount of memory
  - No compromise in simulation accuracy is made

Verification

- Compare the output results in simulating 4 NoC designs of the FPGA-based emulator and BookSim
  - 5-stage 2-VC
  - 5-stage 1-VC
  - 4-stage 2-VC
  - 4-stage 1-VC
Verification: Proposal vs BookSim

**Solid Lines: Proposal (FPGA-based)**
**Dotted Lines: BookSim (Software-based)**

Nearly identical
Simulation Performance

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology</td>
<td>128x128 mesh</td>
</tr>
<tr>
<td>Router architecture</td>
<td>5-stage</td>
</tr>
<tr>
<td># of VCs per port</td>
<td>2</td>
</tr>
</tbody>
</table>

The drop is caused by stalling the emulated network in the first proposed method which helps to eliminate the memory constraint.

**Speedup over BookSim**

Flit Injection Rate (flits/node/cycle)  
(Amount of Network Activity)
Simulation Performance

Simulation Speedup over Booksim

- 4-phy: 702x
- 16-phy: 2782x
- 32-phy: 5490x

High end PC
- Core i7 4770 CPU
- 32GB RAM

BookSim: ≈ 5.5 days

Proposal: ≈ 1.5 minutes

Virtex-7 FPGA
Conclusions & Future Work

Conclusions

- Two methods are proposed to enable ultra-fast and accurate emulation of large-scale NoC architectures on a single FPGA
- *More than 5000x simulation speedup over BookSim* is achieved when emulating an 128x128 NoC with state-of-the-art router architectures

Future work

- Support full-system simulations
- Support a wide range of benchmarks/workloads
Q & A

Thank you!