Background

**Computer Architecture**

**What's Computer Architecture?**

*Computer Architecture* is the science and art of selecting and interconnecting hardware components to create *computers* that meet functional, performance and cost goals. Computer architecture is *not* about using computers to design buildings.

http://www.cs.wisc.edu/arch/www/

IPSJ SIG-ARC (The IPSJ Special Interest Group on Computer Architecture)

Its’ name changed from *Computer Architecture* to *System Architecture*. 
Background

Computer Architecture -> Computing Architecture

School of Computing

International Symposium on Microarchitecture (MICRO) 2014

Keynote II

08:00 - 09:00 Keynote II

Keynote Title: The End of Moore’s Law - Again

Keynote Speaker: Trevor Mudge, University of Michigan

Session Chair: Eme Ozer

Is Moore’s Law Dead—Again

Cambridge, England
December 16th, 2014

Trevor Mudge
Background

◆ Is Moore’s Low Dead - Again

Sanity Check on Exponentials

- VLSI growths don’t follow the (log)arithmic to 2013 is incredible relative
  to 2005 or 1990. If we plot with a linear scale, we see a different trend.
- In 2005, the lattice spacing of silicon is 18.5 A and 18.5 A.
- In 2005, the lattice spacing of silicon is 18.5 A and 18.5 A.
- In 2005, the lattice spacing of silicon is 18.5 A and 18.5 A.
- In 2005, the lattice spacing of silicon is 18.5 A and 18.5 A.
- In 2005, the lattice spacing of silicon is 18.5 A and 18.5 A.
- In 2005, the lattice spacing of silicon is 18.5 A and 18.5 A.

Summary—There’s lots of room at the bottom

- In the long term, we will see new technologies that will continue
  the trend of low cost, high density, high performance, and high
  performance.
- The candidates are probably radiation in materials.
- The candidates are probably radiation in materials.
- The candidates are probably radiation in materials.
- The candidates are probably radiation in materials.
- The candidates are probably radiation in materials.
- The candidates are probably radiation in materials.
- More likely to see something like
  - Intel Xeon Phi
  - X86
  - TILE-Mx

Multi/Many-Core Architectures Have Become Mainstream

1000s cores in near future architectures

- 60+ cores
- 100 cores
- Multi/many-core

40 Years of Microprocessor Trend Data

- Transistors (thousands)
- Single-Thread Performance (SpecINT x 10^3)
- Frequency (MHz)
- Typical Power (Watts)
- Number of Logical Cores

Original data up to the year 2010 collected and plotted by M. Horowitz, F. Lambe, O. Shafranov, K. Duquette, L. Hammond, and C. Gatton.

New pt and data collected for 2010-2015 by R. Raja.
Intel Skylake 2015/08

- 4 core

Intel Xeon Phi

Table 2. Intel® Xeon Phi™ Product Family Specifications

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Form Factor &amp; Board Solution</th>
<th>Number of Cores</th>
<th>Frequency (GHz)</th>
<th>Peak Double Precision (GFLOPS)</th>
<th>Peak Memory Bandwidth (GB/s)</th>
<th>Memory Capacity (GB)</th>
<th>Interconnect Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>3120P</td>
<td>PCIe, Passive</td>
<td>57</td>
<td>1.1</td>
<td>1003</td>
<td>240</td>
<td>6</td>
<td>N/A</td>
</tr>
<tr>
<td>3120A</td>
<td>PCIe, Active</td>
<td>32</td>
<td>1.1</td>
<td>1003</td>
<td>240</td>
<td>6</td>
<td>N/A</td>
</tr>
<tr>
<td>5110P</td>
<td>PCIe, Passive</td>
<td>60</td>
<td>1.053</td>
<td>1011</td>
<td>320</td>
<td>8</td>
<td>N/A</td>
</tr>
<tr>
<td>5120DP</td>
<td>Dense form factor, None</td>
<td>60</td>
<td>1.053</td>
<td>1011</td>
<td>320</td>
<td>8</td>
<td>N/A</td>
</tr>
<tr>
<td>7110P</td>
<td>PCIe, Passive</td>
<td>61</td>
<td>1.238</td>
<td>1200</td>
<td>352</td>
<td>16</td>
<td>Peak turbo frequency 1.33 GHz</td>
</tr>
<tr>
<td>7120X</td>
<td>PCIe, None</td>
<td>61</td>
<td>1.238</td>
<td>1200</td>
<td>352</td>
<td>16</td>
<td>Peak turbo frequency 1.33 GHz</td>
</tr>
</tbody>
</table>
Intel Single-Chip Cloud Computer

Kalray MPPA 256 core processor

- High processing performance
- 700 GOPS – 230 GFLOPS SP
- Low power consumption
- High execution predictability
- High-level programming models
- PCI Gen3, Ethernet 10G, NoCX

Available since November 2012
Microsoft Catapult (FPGA Accelerator)

FPGA (Field Programmable Gate Array)

Intel bought Altera (2015/6)
コンピューティングシステム・アーキテクチャに関する研究と展開

計算機の研究・開発をメインとしてきたコンピュータ・アーキテクチャだけでなく、計算(コンピューティング)を重視するシステムアーキテクチャの研究が重要となってきた。
また、コンピューティングに対する要求の多用化は、ASICとして実現されてきたハードウェアをFPGAなどの柔らかいハードウェアへと置き換えつつある。
本講演では、コンピューティングシステム・アーキテクチャ技術と今後の展開を議論したい。
FPGA
(Field Programmable Gate Array)

Programmable hardware?

◆ Combinational Circuit

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Programmable hardware

- Sequential Circuit

![Sequential Circuit Diagram]

FPGA

- LUT (Lookup Table)
- Register
- Connection block, switch box

- Configuration
  - SRAM
ScalableCore System: Scalable Many-core Emulator
Slow Simulation Speed of Software Simulator

- Slow down simulation speed in software simulator in C++ with the increasing the number of target cores
  - Software simulator is very slow! (slow down 1000x ~ )
  - And, to achieve the scalable speed is difficult!

Motivation & Direction

- To achieve **scalable** simulation speed
  - = Keep simulation speed in simulations of large number of cores
- How to scale the simulation speed?
  - In this study, our target architecture is tile architecture with 2D mesh network

Map the target processor into multiple FPGAs
Our Solution: **ScalableCore System**

- Multiple FPGA units compose whole the target processor

**Target Many-core**

- ScalableCore System

**ScalableCore System**

- Mapping to Multiple FPGAs

**Our Original FPGA Boards**

- *We developed them from scratch!* 
  
  - **ScalableCore Unit**
    - FPGA+SRAM board
    - Xilinx Spartan-6 XC6SLX16
    - 512KB SRAM (8bit, 1-port read/write)
    - Configuration ROM
  
  - **Memory Unit**
    - FPGA+DRAM board
    - Xilinx Spartan-6 XC6SLX16
    - 16MB DRAM
    - Configuration ROM
It’s Scalable!

- 1 (1x1) ScalableCore Unit

It’s Scalable!!

- 16 (4x4) ScalableCore Units
It’s Scalable !!!

- 64 (8x8) ScalableCore Units

It’s Scalable !!!!

- 128 (16x8) ScalableCore Units
ScalableCore System 3.3 for 100-Nodes

- Memory Unit (for DRAM Controller):
  - FPGA+DRAM board

- ScalableCore Unit (for Processor Core):
  - FPGA+SRAM board

Node Micro Architecture of Target

- **Core**
  - MIPS32 ISA, 5-stage, Single-issue, In-order
  - No FPU Support (Future Work)
  - 2-Memory-ports (Inst, Load/Store)

- **DMA Controller**
  - 2-Memory-ports (32-bit DMA Read, 32-bit DMA Write)

- **Router**
  - 5-I/O, 4-stage (NRC/VA, SA, ST, LT)
  - 2-Virtual Channels, FIFO size=4, Credit-base flow control

- **Local Memory**
  - Access latency=1, 512KB, 32-bit
  - 4-Memory-ports (Inst, Load/Store, DMA Read, DMA Write)
Each FPGA board has own clock

Local Barrier Synchronization

- Handshaking with only 4 neighbor FPGAs
- Constant overhead of the handshaking
- Achieves scalable simulation speed

Cycle 1

- Sending to Unit 0
- Sending to Unit 1
- Sending to Unit 2
- Sending to Unit 3
- Receiving from Unit 0
- Receiving from Unit 1
- Receiving from Unit 2
- Receiving from Unit 3

Cycle 2

- Sending to Unit 0
- Sending to Unit 1
- Sending to Unit 2
- Sending to Unit 3
- Receiving from Unit 0
- Receiving from Unit 1
- Receiving from Unit 2
- Receiving from Unit 3

Wait!
Virtual Cycle

- Multiple FPGA clock cycles to 1 target clock cycle
  - Virtual hardware by using simple FPGA equipment

  Drive the circuit of target components
  Process the memory accesses

Simulation Speed Evaluation

- Environment
  - ScalableCore system 3.3 (FPGA-based simulator of M-Core)
    - Freq.: 40MHz (SerDes: 80MHz)
  - SimMc (Software simulator of M-Core)
    - Intel Corei7 870, Memory 4GB, gcc4.5.2 (-O3), Ubuntu Server 11.04

- # Node
  - 16 (4x4), 36 (6x), 64 (8x8), 100 (10x10)
Simulation Speed [KHz]

- ScalableCore System achieves constant simulation frequency: Good **weak scaling**
- With # target core increases, relative speed increases!!
  - With # target core increases, relative speed increases!!
  - In 100-Node, ScalableCore system runs at 129x faster

![Simulation Speed Graph](image)

---

**High-speed FPGA Accelerator for Integer Sorting**

![High-speed FPGA Accelerator Image](image)

---

*E. KIS, TOKYO TECH*
32-bit Integer Sorting

◆ Desktop computer
  ▶ Intel Core i7, 3.4GHz
  ▶ DDR3 16GB memory
  ▶ Microsoft Visual C++ Compiler 2013, /Ox optimization

◆ Baseline
  ▶ Merge sort
    32 mega elements, **4.12** seconds
    256 mega elements, **32.8** seconds

<table>
<thead>
<tr>
<th># of Elements</th>
<th>32M</th>
<th>64M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Sequence</td>
<td>Random</td>
<td>Sorted</td>
</tr>
<tr>
<td>QuickSort</td>
<td>2.754sec</td>
<td>&gt;2hours</td>
</tr>
<tr>
<td>MergeSort</td>
<td><strong>4.121</strong>sec</td>
<td>1.562sec</td>
</tr>
</tbody>
</table>

Table 1. Execution Time of Quick Sort and Merge Sort

Motivation

◆ Is an FPGA sorting at 200MHz faster than a software running on PC at 3.4GHz?
  ▶ Frequency slowdown of FPGA is 17!

◆ If the FPGA sorting is faster than PC, how fast it is?
  ▶ Is it effective if it achieves **1.2x** speedup in the real world?
  ▶ Is it effective if it achieves **2x** speedup in the real world?
  ▶ Is it effective if it achieves **4x** speedup in the real world?

◆ Our target speedup is ...
FPGA–based Custom Computing Approach

Proposed Sorting Accelerator

- Using the following sorting architectures
  - The sorting network
  - The merge sorter tree
The Sorting Network*

A sorting architecture composed of wires and comparators

Example: Sorting 4 values in the network

> Smaller and larger values are carried to the top and bottom

![Bubble sort network with 4 inputs and 4 outputs](image)


The Merge Sorter Tree*

A data path that executes merge process

![4-way merge sorter tree](image)

* Dirk Koch et al. FPGASort, FPGA’11
The Merge Sorter Tree

- Sorting process in the merge sorter tree
  - The data sequences in the leftmost FIFOs must be sorted

Cycle N | Cycle N+1 | Cycle N+2
---|---|---
3 | 5 | x: Invalid Value
5 | 7 |
3 | 8 |
1 | 9 | 2 | 11
2 | 10 | 2

Data Path of the Proposed Sorting Accelerator
Sorting 256 Elements from 256 to 1

- The generated initial data sequence is stored in the external memory (DRAM)

Sorting 256 Elements from 256 to 1

- Initialization is done
Sorting 256 Elements from 256 to 1

- The data is sent to Sorting Network

Sorting Network can sort 16 elements
- The initial data sequence turns into 16 sorted data sequences (units) by passed through this network

This is sorted

16 … 3 2 1 32 … 19 18 17 … 256 … 243 242 241
The data passed through the network is stored in Input Buffer, and sent to Merge Sorter Tree.

The root of the tree emits sorted data sequences.
Sorting 256 Elements from 256 to 1

- The data sequence composed of 16 units turns into 4 units

| 256 | 255 | 254 | 253 | 252 | ... | 128 | 67 | 66 | 65 | 192 | 131 | 130 | 129 | 256 | 195 | 194 | 193 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

This is sorted (64 elements)

Sorting 256 Elements from 256 to 1

- The data is stored in the external memory
Sorting 256 Elements from 256 to 1

- This data is not fully sorted yet...
  - This data has to be sent to Merge Sorter Tree again

Sorting 256 Elements from 256 to 1

- The data is read from DRAM and sent to Sorting Network
Sorting 256 Elements from 256 to 1

- In this time, this network is a mere data path because portions of the data sequence are already sorted.

The data passed through the network is stored in Input Buffer, and sent to Merge Sorter Tree.
Sorting 256 Elements from 256 to 1

- The root of the tree emits sorted data sequences

- The data is stored in the external memory
This data is fully sorted!!

- The data is fully sorted by passed through the network and the tree as required
  - \( \log_2 \text{of ways} \times \text{# of elements}/16 \)

The fully sorted data is sent to Host PC

- To verify that the accelerator accurately works
Data Path of the Accelerator with the Duplicated Merge Sorter Tree

- Duplication of the merge sorter tree

**Effectiveness**
- To sort data sequences in parallel
  - The sorting logic throughput is improved

Initial Data Sequence:

```
256 ..., 194, 193, 192, ..., 130, 129, 128, ..., 66, 65, 64, ..., 2, 1
```

Sorting the data in parallel

Executing merge process in a tree

Sorting is done!!!
Hardware Setup (1/2)

- Implementation Platform
  - Xilinx FPGA VC707 Evaluation Kit

![Hardware Setup Diagram]

- This kit originally has 1GB DDR3 SO-DIMM (800MHz/1600Mbps) memory

Evaluation:
Sorting Performance of 8-way/8-parallel

- In a case of random data sequence
  - sustainable performance is 70.2% of the estimated one
    - 10.06x faster than merge sort
    - 8.01x faster than quick sort
Discussion

- Our FPGA sorting at 200MHz achieved 8x faster than a software implementation.
  - Frequency 200MHz is 17x slower than PC at 3.4GHz
    - note: DRAM frequency for FPGA and PC is the same
  - FPGA implementation is 136x efficient
  - So, total speedup of 136/17 = 8

- We have a feasible plan to enhance the speedup over 30x.
  - multiple-outputs sorter cell

Open Sourced

- FACE is available on GitHub
  - [https://github.com/monotone-RK/FACE](https://github.com/monotone-RK/FACE)
- Currently, FACE can work on Xilinx FPGA VC707 Evaluation Kit
  - We will try to port another environment if you have requests and if possible
Fast and Accurate Emulation of Large-scale Network on Chip Architectures on a Single FPGA

Contributions

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

Cycle-accurate & ?? x simulation speedup over BookSim\(^1\), a widely used software simulator

5? , 50?, 500?, 5000?, 50000?

High end PC
Core i7 4770 CPU
32GB RAM

BookSim\(^1\): 5.5 days

Proposal:
Virtex-7 FPGA

\(^1\)http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim

Load-latency graph

Up to 16384 nodes
Contributions

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

Cycle-accurate & 5000x simulation speedup over BookSim\(^1\), a widely used software simulator

High end PC
Core i7 4770 CPU
32GB RAM

BookSim\(^1\): 5.5 days

Proposal: 1.5 minutes

FPGA
Virtex-7 FPGA

Network-on-Chip (NoC)

- A promising interconnection network of many-core architectures
- NoC simulation plays a vital role in designing many-core architectures
NoC Model

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

NoC 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)
NoC Simulations by Software

- 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

FPGA-Accelerated NoC Simulation

- Not flexible and hard to debug 😞
- 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 necessary degrade performance
FPGA-Accelerated NoC Simulation

**Challenges**

- **Single FPGA**
  - Limited resources

- **Multiple FPGAs**
  - Off-chip memory
  - More resources
  - More complex
  - Off-chip communication

**Memory constraint**

Traffic Generation Unit 1

Source Queue 1

Network

Router

Router

Traffic Generation Unit N

Source Queue N

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

Software:
Dynamic memory allocation

Our latest FPGA accelerator system

- ScalableCore: SmartCore ready FPGA accelerator
  - 128 (16 X 8) nodes configuration
- SmartCore concept was verified on this real many-module system
FPGA-Accelerated NoC Simulation

**Challenges**

**Single FPGA**
- Limited resources

- More resources

**Multiple FPGAs**
- Off-chip memory

- More complex

- Off-chip communication

**Memory constraint**
- Must be very large to ensure that no generated packet is dropped

**Proposals**

**Method 1: Decoupling Time Counters**

**Method 2: Time-Multiplexing**

- **Logical clusters**

- **Physical cluster**

- Eliminate the memory constraint

**Simulate architectures with 1000s of cores on a single FPGA**
Method 1: 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

PS<sub>i</sub> intends to generate a packet but SQ<sub>i</sub> is full

One space in SQ<sub>i</sub> becomes available

No packet is dropped

But...
Some packets may not be generated on time

 packets

State Machine of Packet Source <sub>i</sub>

Packet Source 1

source Queue 1

Network

Router

Packet Source N

source Queue N

Running

Waiting

∀<sub>i</sub> SQ<sub>i</sub> is NOT empty OR T<sub>i</sub> = T

∀<sub>i</sub> either SQ<sub>i</sub> contains at least one packet or the time of PS<sub>i</sub> is synchronized with the time of the network

The specified synthetic workload is simulated accurately

source Queue 1

Network

Router

Router

Packet Source N

source Queue N

Running

Waiting
Method 2: Time-Multiplexing

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

- **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.
Evaluation and Analysis

- 128 x 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 x 2): use four physical nodes to emulate the entire 128 x 128 mesh network
  - 16-phy (4 x 4): use 16 physical nodes to emulate the entire 128 x 128 mesh network
  - 32-phy (8 x 4): use 32 physical nodes to emulate the entire 128 x 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>Topology</th>
<th>128x128 mesh (16,384 nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router architecture</td>
<td>5-stage pipelined VC router or 4-stage pipelined VC router (employing look-ahead routing)</td>
</tr>
<tr>
<td># of VCs per port</td>
<td>2 or 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 or 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>
**Hardware Usage**

Simulation 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)

**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)**

- Proposal 5-stage 2-VC
- BookSim 5-stage 2-VC
- Proposal 5-stage 1-VC
- BookSim 5-stage 1-VC
- Proposal 4-stage 2-VC
- BookSim 4-stage 2-VC
- Proposal 4-stage 1-VC
- BookSim 4-stage 1-VC

Nearly identical

**Simulation Performance**

<table>
<thead>
<tr>
<th>Topology</th>
<th>128x128 mesh</th>
</tr>
</thead>
<tbody>
<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.
Conclusions of this 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
ARM ベース SoC 設計ワークショップ

日時：2016年12月17（土）午前9時30分～午後5時
場所：〒152-8500 東京都目黒区大岡山2丁目12-1
東京工業大学 大岡山キャンパス 西棟3号館1F 1001会議室

講演者
Dr. Sean Hong, Brent Lei (ARM ユニバーシティプログラム)

速記講義（東京工業大学・発表内容）

ARM ユニバーシティプログラムは、東京工業大学において、大学教員および学生のための1-dayワークショップを開始いたします。ワークショップの目的は、ARM Cortex-M3 Design Start ソフトウェアツールを使用したSoCをどのように設計・実装するかを理解することです。

このワークショップでは、ARM Keil MDK ソフトウェアツールを使用して、低消費電力のARM Cortex-M3 プロセッサのためのプログラミングを学びます。また、参加者はARMAD アダプターの歴史的見解をどのように設計し、ARM Nexys4 FPGA ボードにどのように実装するかを学びます。

Summary

◆ FPGA

◆ ScalableCore System: Scalable Many-core Emulator
  ▶ weak scaling
  ▶ 100x faster than CPU

◆ High-speed FPGA Accelerator for Integer Sorting
  ▶ streaming
  ▶ 8x faster than CPU

◆ Fast and Accurate Emulation of Large-scale Network on Chip Architectures on a Single FPGA
  ▶ drastic speedup