Building Networks with Open, Reconfigurable Hardware

1st Asia NetFPGA Developer’s Workshop
Monday, June 14, 2010

John W. Lockwood
CEO & Lead Consultant, Algo-Logic Systems

With input from:
NetFPGA Group, Stanford University

Hardware and tools available for university programs thanks to grants, donations, and/or partnerships from:

[List of logos and names of companies and organizations]
NetFPGA’s Defining Characteristics

• **Line-Rate**
  – Processes back-to-back packets
    • Without dropping packets
    • At full rate of Gigabit Ethernet Links
  – Operating on packet headers
    • For switching, routing, and firewall rules
  – And packet payloads
    • For content processing and intrusion prevention

• **Open-source Hardware**
  – Similar to open-source software
    • Full source code available
    • BSD-Style License
  – But harder, because
    • Hardware modules must meeting timing
    • Verilog & VHDL Components have more complex interfaces
    • Hardware designers need high confidence in specification of modules
NetFPGA = Networked FPGA

A line-rate, flexible, open networking platform for teaching and research

Network Interface Card
Hardware Accelerated Linux Router
IPv4 Reference Router
Traffic Generator
Openflow Switch
List of All Projects
Add Your Project
NetFPGA Reference Platform

NetFPGA

= Software running on a standard PC

+ A hardware accelerator built with Field Programmable Gate Array driving Gigabit network links
NetFPGA System
Hardware & Software Components

User-Space Software

Linux Kernel Driver

PCI Bus

NetFPGA Hardware: Logic Gates, SRAM, & DRAM

CPU Queues

Flip/Flops & SRAM

SRAM MAC Queues,

PHY

Gigabit Ethernet

SCONE

PW-OSPF

Java GUI

Driver

DMA

Registers

nf2c0

nf2c1

nf2c2

nf2c3

ioctl

user data path

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC

MAC
Open Platform, Low-cost Hardware

- NetFPGA Cards & Pre-built Systems
  - Available from 3rd Party Vendors
- Or, build your system from parts
  - Details in the on-line Guide
Stanford’s Rackmount Deployment

Stanford NetFPGA Cluster (NFC)
Internetconnect-side View

Statistics
- Rack of 40
  - 1U PCs
  - NetFPGAs
- Managed
  - Power,
  - Console
  - VLANs
- Provides 160 Gbps of full line-rate processing bandwidth
UCSD’s NetFPGA Cluster
Inter-module Communication

Module i

Packet memory

Table memory

Module i+1

data
ctrl
wr
rdy
Modular Architecture

NetFPGA

Register Bus
Packet Bus

Register I/O over PCI

RxQ
Packet Processing

Module$_1$

Registers

Module$_2$

Registers

...

Module$_n$

Registers

Packet Processing

TxQ

From Ethernet

DMA from host

To Ethernet

DMA to host
NetFPGA Base Reference Router

SRAM → Forward Table Logic → Input Arbiter → Output Port Lookup → Output Queues → MAC RxQ, CPU RxQ, MAC RxQ, CPU RxQ, MAC RxQ, CPU RxQ, MAC RxQ, CPU RxQ, MAC RxQ, CPU RxQ, MAC RxQ, CPU RxQ

Output Queues → Packet Buffer Logic → DRAM

Input Arbiter → MAC TxQ, CPU TxQ, MAC TxQ, CPU TxQ, MAC TxQ, CPU TxQ, MAC TxQ, CPU TxQ, MAC TxQ, CPU TxQ
Reference Router Control Software

Click the Details tab of the Quickstart window

This is the reference router pipeline – a canonical, simple-to-understand, modular router pipeline
Output Queues module
(from Details tab)

Configuration details

Statistics
Enhanced Router Pipeline

Event Capture
to capture output queue events (writes, reads, drops)

Rate Limiter
to create a bottleneck
NetFPGA Hardware Block Diagram

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been... and then open the file again. If the red x still appears you may have to delete the image and then insert it again.

Your hardware specified in Verilog source code connected to components of the Reference Router circuits and cores.

Four Gigabit Ethernet Interfaces

Board-Board Interconnect

Control, PCI Interface

Host computer

Linux OS - NetFPGA Kernel driver

User-defined software networking applications
Hardware Description Languages

• **Concurrent**
  – By default, Verilog statements evaluated concurrently

• **Express fine grain parallelism**
  – Allows gate-level parallelism

• **Provides Precise Description**
  – Eliminates ambiguity about operation

• **Synthesizable**
  – Generates hardware from description
FPGA Look-Up Tables

Combinatorial logic is stored in Look-Up Tables (LUTs)

- Also called Function Generators (FGs)
- Capacity is limited only by number of inputs, not complexity
- Delay through the LUT is constant

Diagram From: Xilinx, Inc
Finite State Machines

Inputs (X) \[ \rightarrow \]

Outputs (Z)

- \( \lambda (X, S(t)) \) [Mealy]
- \( \lambda (S(t)) \) [Moore]

\[ S(t+1) = \delta(X, S(t)) \]

State Storage

Combinational Logic

Next State
Field Programmable Gate Arrays

CLB
- Primitive element of FPGA

Routing Module
- Global routing
- Local interconnect

Macro Blocks
- Block Memories
- Microprocessor

I/O Block
NetFPGA Reference Code & Modules

NF2 Directory Tree

- **bin** (scripts for running simulations and setting up the environment)
- **bitfiles** (contains the bitfiles for all projects that have been synthesized)
- **lib** (stable common modules and common parts needed for simulation/synthesis/design)
- **projects** (user projects, including reference designs)
Structure of Libray Source Code

lib
  C (common software and code for reference designs)
  java (contains software for the graphical user interface)
  Makefiles (makefiles for simulation and synthesis)
  Perl5 (common libraries to interact with reference designs and aid in simulation)
  python (common libraries to aid in regression tests)
  scripts (scripts for common functions)
  verilog (modules and files that can be reused for design)
Structure of Project Source Code

- **projects**
  - **doc** (project specific documentation)
  - **include** (contains file to include verilog modules from lib, and creates project specific register defines files)
  - **regress** (regression tests used to test generated bitfiles)
  - **src** (contains non-library verilog code used for synthesis and simulation)
  - **sw** (all software parts of the project)
  - **synth** (contains user .xco files to generate cores and Makefile to implement the design)
  - **verif** (simulation tests)
Test-Driven Designs

• **Regression tests**
  – Have repeatable results
  – Define the supported features
  – Provide clear expectation on functionality

• **Example: Internet Router**
  – Drops packets with bad IP checksum
  – Performs Longest Prefix Matching on destination address
  – Forwards IPv4 packets of length 64-1500 bytes
  – Generates ICMP message for packets with TTL <= 1
  – Defines how packets with IP options or non IPv4
  … and dozens more …

  *Every feature is defined by a regression test*
Welcome to the Worldwide Community

NetFPGA @ SIGCOMM - Seattle, WA

NetFPGA @ SIGMETRICS - San Diego, CA

EuroSys - Glasgow, Scotland, U.K.

Workshop in Beijing, China

Workshop in Bangalore, India
NetFPGA Deployments

- Over 1,350 NetFPGA users with 1,300+ cards deployed at 150+ universities in 17 Countries worldwide

Worldwide Hardware Deployments - Feb 2010
NetFPGA Hardware in North America

USA Deployments - Feb 2010
NetFPGA Hardware in Europe

European Deployments - Feb 2010
NetFPGA Hardware in Asia

China, Korea, Japan, Taiwan, and India Deployments - Feb 2010
NetFPGA Deployments in Korea
## Contributed projects

<table>
<thead>
<tr>
<th>Project (Title &amp; Summary)</th>
<th>Base Version</th>
<th>Status</th>
<th>Organization</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4 Reference Router</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Guide</td>
</tr>
<tr>
<td>Quad-Port Gigabit NIC</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Guide</td>
</tr>
<tr>
<td>Ethernet Switch</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>Buffer Monitoring System</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Guide</td>
</tr>
<tr>
<td>Hardware-Accelerated Linux Router</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Guide</td>
</tr>
<tr>
<td>DRAM-Router</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>DRAM-Queue Test</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>Packet Generator</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>OpenFlow Switch</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>NetFlow Probe</td>
<td>1.2</td>
<td>Functional</td>
<td>Brno University</td>
<td>Wiki</td>
</tr>
<tr>
<td>AirFPGA</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>Fast Reroute &amp; Multipath Router</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>NetThreads</td>
<td>1.25</td>
<td>Functional</td>
<td>University of Toronto</td>
<td>Wiki</td>
</tr>
<tr>
<td>Precise Traffic Generator</td>
<td>1.25</td>
<td>Functional</td>
<td>University of Toronto</td>
<td>Wiki</td>
</tr>
<tr>
<td>URL Extraction</td>
<td>2.0</td>
<td>Functional</td>
<td>Univ. of New South Wales</td>
<td>Wiki</td>
</tr>
<tr>
<td>zFilter Sprouter (Pub/Sub)</td>
<td>1.2</td>
<td>Functional</td>
<td>Ericsson</td>
<td>Wiki</td>
</tr>
<tr>
<td>Windows Driver</td>
<td>2.0</td>
<td>Functional</td>
<td>Microsoft Research</td>
<td>Wiki</td>
</tr>
</tbody>
</table>

As of Feb 2010

.. And more on [http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/ProjectTable](http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/ProjectTable)
<table>
<thead>
<tr>
<th>Windows Driver</th>
<th>2.0</th>
<th>Functional</th>
<th>Microsoft Research</th>
<th>Wiki?</th>
</tr>
</thead>
<tbody>
<tr>
<td>RED</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>Open Network Lab</td>
<td>2.0</td>
<td>Functional</td>
<td>Washington University</td>
<td>Wiki</td>
</tr>
<tr>
<td>DFA</td>
<td>2.0</td>
<td>Functional</td>
<td>UMass Lowell</td>
<td>Wiki?</td>
</tr>
<tr>
<td>G/PaX</td>
<td>2.7</td>
<td>Functional</td>
<td>Xilinx</td>
<td>Wiki</td>
</tr>
<tr>
<td>RCP Router</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>Deficit Round Robin (DRR)</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>OpenFlow-MPLS Switch</td>
<td>2.0</td>
<td>Functional</td>
<td>Ericsson</td>
<td>Wiki</td>
</tr>
<tr>
<td>PTP-enabled Router</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>Vlan Tag Handler</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>Port Aggregator</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford University</td>
<td>Wiki</td>
</tr>
<tr>
<td>IP Lookup w/Blooming Tree</td>
<td>1.25</td>
<td>In Progress</td>
<td>University of Pisa</td>
<td>Wiki</td>
</tr>
<tr>
<td>KOREN Testbed</td>
<td>2.7</td>
<td>In Progress</td>
<td>Chungnam-Korea</td>
<td>Wiki</td>
</tr>
<tr>
<td>Virtual Data Plane</td>
<td>1.2</td>
<td>In Progress</td>
<td>Georgia Tech</td>
<td>Wiki</td>
</tr>
<tr>
<td>Deficit Round Robin (DRR) Input Arbiter</td>
<td>1.2</td>
<td>In Progress</td>
<td>Universidade Federal do Rio Grande do Sul (Brazil)</td>
<td>Wiki</td>
</tr>
<tr>
<td>Counter Diffs</td>
<td>2.0</td>
<td>Functional</td>
<td>Stanford (Lu, Jianying)</td>
<td>Wiki</td>
</tr>
<tr>
<td>Tunneling NIC</td>
<td>2.0</td>
<td>In Progress</td>
<td>Stanford</td>
<td>Wiki</td>
</tr>
<tr>
<td>Multicore Prototype for Real-Time Switching</td>
<td>1.0</td>
<td>Available on Request</td>
<td>University of Waterloo</td>
<td>Wiki</td>
</tr>
<tr>
<td>End-to-End Ethernet Authorization</td>
<td>2.0</td>
<td>In Progress</td>
<td>Euskal Herriko Unibertsitateko</td>
<td>Wiki</td>
</tr>
<tr>
<td>Ultra-high Speed Congestion-control</td>
<td>2.0</td>
<td>In Progress</td>
<td>University of North Carolina</td>
<td>Wiki</td>
</tr>
</tbody>
</table>

... And more on [http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/ProjectTable](http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/ProjectTable)
Other Network FPGA Platforms

- **NetFPGA 1G**
  - 4 * 1 GigE
  - 4 Gbps

- **FPX**
  - 2 * OC-48
  - 4.8 Gbps

- **Upcoming 10G**
  - 4 * 10GE
  - 40 Gbps
Motivation for Network Security

• **Viruses can be costly to businesses**
  – Annoyance to average users
  – Use networks to propagate

• **Sample Attacks**
  – Nimda, Code Red, Slammer
  – MSBlast
    • Infected over 350,000 hosts
  – SoBigF
    • Infected 1 million users in first 24 hours
    • Infected > 200 million in the first week
    • Caused an estimated $1 billion in damages to repair.

• **End-systems difficult to maintain**
  – Operating systems become outdated
  – Users introduce new machines on network
Example of SOC Firewall-FPX Platform

FPX Block Diagram

- **Reference**

- **More Information:**
  - http://www.reprogrammablenetworks.com/
Regular Expression Matching

• **String Matching**
  – Allow for matching of fixed-length strings
    • HEX(683063423739) finds SoBig.F

• **Regular Expressions**
  – Allows for a wide range of matches
    • Case variations: (W|w)(A|a)(R|r)(H|h)(O|o)(L|l)
    • Wild-card characters: Albert ? Einstein
    • Strings of wildcards: A.* Einstein
  – Content may begin anywhere in packet

• **Morphable Patterns**
  – Will require systems that can evolve
GUI to Select Search Strings
Compiler to build Parallel Scan Engines

Diagram showing a network of Dispatcher and Flow Collector nodes connected with DFA (Deterministic Finite Automaton) blocks labeled RE:1 DFA, RE:2 DFA, RE:3 DFA, and RE:n DFA.
Content Matching Module

Regular Expression (RE) Matching Circuit

- dataen_outappl
- d_outappl
- sof_outappl
- eof_outappl
- sod_outappl
- tca_outappl
- clk
- reset
- enable

- dataenappl
- dappl
- sofappl
- eofappl
- sodappl
- tcaappl
- Content
- Matching
- Vector
- ready
Sample ModelSim Waveform

[TTL <> 0] [Src IP] [Dest IP]
Throughput Comparison

• Sed was run on different Linux PCs
  – Dual Intel Pentium III @ 1 GHz
    • 13.7 Mbps when data is read from disk
    • 32.72 Mbps when data is read from memory
  – Alpha 21364 @ 667 MHz
    • 36 Mbps when data is read from disk
    • 50.4 Mbps when data is read from memory

• Software results are 40x slower than FPsed
String Processing Benchmarks
Associative Match in FPGA TCAM

- **Content**: Content Vector = “00000011” (binary) = x”03” (hex)
- **Source IP Address**: 128.252.5.5 (dotted.decimal)
- **Destination IP Address**: 141.142.2.2 (dotted.decimal)
- **Source Port**: 4096 (decimal) = 1000 (hex)
- **Destination Port**: 80 (decimal) = 50 (hex)
- **Protocol**: TCP (6)
TCAM Function in FPGA Hardware
Data Buffering & Per-flow Queuing

• **Flow Buffering**
  – Maintain linked list of packets
  – Track head and tail pointers for each list
  – Track free memory

• **Queue Management**
  – Enqueue packet
  – Dequeue packet

• **Schedule Traffic Flows**
Flow Buffering

Flow State

- Head Pointer
- Packet Reads = 0
- Packet Writes = 3
- Tail Pointer

Flow State Diagram:

- A: $M[x]$
- B: $M[y]$
- C: $M[z]$
- D: $M[v]$

Other Flow State

- Head Pointer

Reserved Empty Slot

[implementation dependent]
Queue Management

Data

Flow Buffer

SRAM Controller

Enqueue FSM

Queue Manager

Packet Scheduler

Dequeue FSM

Flow ID

Tail

Next Tail

Head

Next Head

Packet Scheduler

P0

P1

P2

P3

Enqueue FSM

Dequeue FSM
Architecture of the SOC Firewall

Xilinx XCV2000E FPGA

Payload Scanner
(FCCM’03)

Payload Match Bits
TCAM Filter

Flow ID

Extensible Module(s)

Interfaces to Off-Chip Memories

SDRAM 2 Controller

Flow Buffer

SDRAM Controller

Packet Scheduler

Queue Manager

Free List Manager

SDRAM 1 Controller

Layered Internet Protocol Wrappers
(FPL’01)
Resulting SOC-Firewall FPGA Layout

- Layered Protocol Wrappers
- Per-flow Queuing
- Region for Extensible Plug-in Modules
- TCAM Header Filtering
- Payload Filtering
- Memory Controller
- Packet Store Manager
- Regular Expression Filtering
- Algo-Logic
Upcoming NetFPGA 10G (4 * 10 GE)

QDRII+ SRAM
3x 36bit interfaces, 300MHz+
(each i/f: 1x K7R643684MFC30)

Xilinx Virtex5
XCV5TX240T-2
FG1759

RLDRAM II
2x 32bit interfaces, 300MHz+

NetFPGA 10G

PCIe x8, Gen1 endpoint edge connector

2x Samtec x10 Connector

SFP+ Cage
SFI 10Gbps
PHY AEL2005
XAU1 4 GTXs

SFP+ Cage
SFI 10Gbps
PHY AEL2005
XAU1 4 GTXs

SFP+ Cage
SFI 10Gbps
PHY AEL2005
XAU1 4 GTXs

SFP+ Cage
SFI 10Gbps
PHY AEL2005
XAU1 4 GTXs

PCIe 8 GTXs

10 GTXs

2 x Samtec x10 Connector
40 Gbps NetFPGA PCB
Algo-Logic Systems

- Founded by Dr. John W. Lockwood, the Algo-Logic ® team has extensive experience building routers, data center switches, and network processing circuits in ASICs and FPGA logic. Algo-Logic specializes in mapping network algorithms into hardware logic. The founders are experts in developing, documenting, and prototyping logic and systems of reprogrammable networks.

- Web
  - http://Algo-Logic.com
- Email
  - Solutions@Algo-Logic.com
- Phone
  - (650) 395-7026
- Fax
  - (650) 498-8296

- Office Address
  - 530 Lytton Ave
    Second Floor
    Palo Alto, CA  94301
Consulting Services

• **Design and Build Hardware-Accelerated Network Systems**

  – Network Architecture
    • Data-center networks
    • Network security
    • Content-aware networks
    • On-chip interconnect

  – Performance Analysis
    • Mathematical system modeling
    • Real and synthetic trace simulation
    • Live measurements in hardware prototypes
Design Services

• **Implementations of Network Algorithms**
  - Hardware Logic in synthesizable high-level HDLs, Verilog, and VHDL
  - Ultra low latency processing
  - System-level software with APIs in C and C++

• **Architectures for Next-Generation Networks**
  - Data-center networks
  - Trading-floor networks
  - Network security
  - Content-aware networks

• **Systems Architecture and Development**
  - FPGA design
  - Verification and test bench development
  - Customization of IP cores
  - System-level Integration

*Design, develop, and verify line-rate network processing systems at multi-Gbps rates.*
Extended Training

- Customized training services to help your team build your own network systems in hardware.
  - Experience in teaching the world-wide NetFPGA tutorials
  - Customized training specifically tailored for your company’s skill level and requirements
  - Hands-on training with live hardware systems
1st Asia NetFPGA Developer Workshop

June 13-14, 2010 @ KAIST, Daejeon, Korea
http://fif.kr/AsiaNetFPGAws

• Program Committee Chairs
  – John W. Lockwood
    • (Algo-Logic Systems)
  – Seung-Joon Seok
    • (Kyungnam University)

• Advisory
  – Nick McKeown
    • (Stanford University)
  – Dae Young Kim
    • (Chungnam University)

• Local Arrangement Co-Chairs
  – Sue Moon
    • (KAIST)
  – Jaeyong Lee
    • (Chungnam National Univ.)

• Many Thanks
  – Xilinx
  – NetFPGA Group
  – Sponsors:
Program Committee Members

• Jun Bi  
  – (Tsinghua University)
• Gordon Brebner  
  – (Xilinx)
• Michael Ciesla  
  – (UNSW)
• Yu-Huang Chu  
  – (Chunghwa Telecom)
• Sun Moo Kang  
  – (NIA)
• Eric Keller  
  – (Princeton University)
• Jongwon Kim  
  – (GIST)
• Taekyoung Kwon  
  – (Seoul National Univ.)
• Martin Labrecque  
  – (University of Toronto)
• Yukuen Lai  
  – (Chung-Yuan Christian Univ.)
• Jaeyong Lee  
  – (Chungnam National Univ.)
• John W. Lockwood  
  – (Algo-Logic Systems)
• Kevin Xie  
  – (Xilinx)
• Sue Moon  
  – (KAIST)
• Andrew, W. Moore  
  – (Cambridge University)
• Akihiro Nakao  
  – (University of Tokyo)
• Seung Yeob Nam  
  – (Yeungnam University)
• Seung-Joon Seok  
  • (Kyungnam University)
• Seung-Yong Park  
  – (Yonsei University)
• Hwangjun Song  
  – (POSTECH)
• Charlie Wiseman  
  – (Washington University)
• Chong Ho Yoon  
  – (Korea Aerospace Univ.)
• Martin Zadnik  
  – (Brno University of Tech.)
• Hongyi Zeng  
  – (Stanford University)
• Young Cho  
  – (USC-ISI)
Welcome