A Heterogeneous GASNet Implementation for FPGA-accelerated Computing

Ruediger Willenberg and Paul Chow
High-Performance Reconfigurable Computing Group
University of Toronto

October 9, 2014
FIELD-PROGRAMMABLE GATE ARRAYS

[Diagram of a grid of logic gates with connections highlighted]

October 9, 2014
High-Performance Reconfigurable Computing Group   ∙   University of Toronto
FIELD-PROGRAMMABLE GATE ARRAYS

© H.Roesner

© flickr / MadPhysicist
Hardwired FPGA functions

SRAM

Ser Des

ARM A9

ARM A9

SRAM

1GE MAC

DSP
Computing with FPGAs

• Fully customized dataflow and buffering

• Tightly coupled pipelining of computations

• Very low energy / computation ratio

• Computation cores can be switched out in \textit{msecs} with \textit{partial reconfiguration}
Example: Smith-Waterman DNA sequencing (Dynamic Programming)

49x – 980x speedup (I/O dependent) on Xilinx V4-LX160 FPGA vs. 2.2GHz AMD Opteron

(Storaasli/Cray 2009)
FPGA programmability drawbacks

• Very different thinking than software programming
• Established Hardware Description Languages (Verilog HDL, VHDL) are very low-level
• Interfaces are often specific to vendors, platforms, FPGA boards
• Implementation (compile-equivalent) takes hours
Programmability improvements

• Higher-level HDLs
  – SystemC, SystemVerilog, Bluespec, Chisel (UC Berkeley)

• High-Level-Synthesis (“C-to-gates”)
  – Very active area in research and industry, but:
  – Only useful if programmer understands hardware
Programmability improvements (2)

OpenCL as an interface to FPGAs

- Abstracts hardware interface as it does for GPUs
- Sets a programming model (memory, core hierarchy)
- Implies High-Level-Synthesis for actual code kernels
- But GPU model not a perfect fit for FPGA
  - Overconstrained dataflow
Classic accelerator model: Master-Slave
Our programming model philosophy

- Use a common API for Software and Hardware
Common SW/HW API

- CPU and FPGA components can initiate data transfers
- SW and HW components use similar call formats
- For distributed memory and message-passing, this was implemented by TMD-MPI (TMD: Toronto Molecular Dynamics)

Our contribution: Building a heterogeneous infrastructure for PGAS
Why again a common API?

• Easier development: SW Prototyping → Migration

• Model makes no distinction between CPUs and FPGAs (in terms of data communication, synchronization)

• FPGA-initiated communication relieves CPU (even more so for 1-sided comm.)

• FPGA-only systems (or 1 CPU + many FPGAs) can work efficiently
THeGASNet

- Toronto Heterogeneous GASNet
- Compatible software and hardware implementations of the GASNet Core API
- GASNet is widely used, well-defined PGAS API
- Core API’s “Active Message” types (S/M/L/LA) and handlers cover essential hardware needs
THeGASNet Hardware: GAScore

Memory

FPGA

CPU
App.c
GASNet.h

GAScore
RDMA

On-Chip Network
GAScore

• Remote memory communication engine (RDMA)
• Controlled through FIFOs (FSLs)
• Configuration parameters are the same as for GASNet Active Message function calls
  – Simple software layer for embedded CPUs
THeGASNet Hardware: GAScore

Memory

CPU

App.c

GASNet

RDMA

GAScore

FPGA

On-Chip Network
THeGASNet Hardware: GAScore

- Memory
- FPGA
- Custom Hardware
- GAScore
- RDMA
- On-Chip Network
THeGASNet Hardware: PAMS

- Memory
- FPGA
- Custom Core
- GAScore
- RDMA
- On-Chip Network
Programmable Active Message Sequencer

- Controls custom hardware operations
- Handles reception/transmission of Active Messages
- Custom hardware ops and Active Messages are initiated based on:
  - Custom hardware state
  - Number of received messages with a specific code
  - Amount of received data
- (Re-)programmable through Active Messages
- Future: Reconfigure Computation Core by AM
THeGASNet Software (PC/ARM)

main()

TCP/IP Thread

PCIe Thread

Application Thread 0

Handler Thread 0

Application Thread 1

Handler Thread 1
THeGASNet packet format

Identical for:
- Thread-to-thread
- TCP/IP
- PCIe
- On-chip network
MapleHoney - Heterogeneous Cluster

![Diagram of MapleHoney Heterogeneous Cluster](image_url)
MapleHoney - Single FPGA
## Microbenchmarks: FPGA

### Short message latency (ns)

<table>
<thead>
<tr>
<th>FPGA hops</th>
<th>1-way</th>
<th>2-way</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>160</td>
<td>330</td>
</tr>
<tr>
<td>1</td>
<td>240</td>
<td>490</td>
</tr>
<tr>
<td>2</td>
<td>320</td>
<td>550</td>
</tr>
</tbody>
</table>

### Long message latency (ns)

<table>
<thead>
<tr>
<th>FPGA hops</th>
<th>4 Bytes</th>
<th>4 kBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>540</td>
<td>22555</td>
</tr>
<tr>
<td>1</td>
<td>620</td>
<td>22635</td>
</tr>
<tr>
<td>2</td>
<td>700</td>
<td>22715</td>
</tr>
</tbody>
</table>
# Microbenchmarks: FPGA

### Simple barrier latency (ns)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency to node 0</td>
<td>870</td>
<td></td>
</tr>
<tr>
<td>Latency to node 0 and back</td>
<td>1960</td>
<td></td>
</tr>
</tbody>
</table>

### “Staggered” barrier latency (ns)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency to node 0</td>
<td>620</td>
<td></td>
</tr>
<tr>
<td>Latency to node 0 and back</td>
<td>1480</td>
<td></td>
</tr>
</tbody>
</table>
### Microbenchmarks: PC, TCP/IP, 1GEth

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full barrier latency (4x 4 Threads)</td>
<td>21.6 ms</td>
</tr>
<tr>
<td>2-way Short message latency</td>
<td>9.3 ms</td>
</tr>
<tr>
<td>1-way 4 kByte Long message latency</td>
<td>6.2 ms</td>
</tr>
<tr>
<td>Average Bandwidth 64MByte transfer</td>
<td>0.84 Gbit/s</td>
</tr>
<tr>
<td>Average Bandwidth 1GByte transfer</td>
<td>0.87 Gbit/s</td>
</tr>
</tbody>
</table>

High variation on latency measurements except barrier => Thread scheduling issues?
## Microbenchmarks: PC <-> FPGA (PCIe)

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-way Short message latency</td>
<td>15.7 us</td>
</tr>
<tr>
<td>1-way 4 kB Long latency PC -&gt; FPGA</td>
<td>32.4 us</td>
</tr>
<tr>
<td>1-way 4 kB Long latency FPGA -&gt; PC</td>
<td>23.6 us</td>
</tr>
<tr>
<td>Sustained PCIe bandwidth PC -&gt; FPGA</td>
<td>175.1 Mbyte/s</td>
</tr>
<tr>
<td>Sustained PCIe bandwidth FPGA -&gt; PC</td>
<td>172.3 Mbyte/s</td>
</tr>
</tbody>
</table>
Application example: Jacobi Heat Transfer

Iterative PDE solver with stencil computation pattern

Cell iteration

Node division

Communication pattern
Step I: Design PC application

• Design SPMD application using THeGASNet

• Problem size: 45760x45760 cells on 16 nodes/threads

• 11880x11880 cells per node/thread: ~1 GB storage

• Debug and profile application on PC platform
Step II: Migrate to MicroBlaze

- Using the same THeGASNet headers and compiler family (gcc), differences should be minimal.
- Static instead of dynamic memory allocation; shared segment pre-defined.
- OS functionality like timers may need to be changed (abstracted in platform header).
- Command-line arguments to application converted to compile-time arguments (application unchanged).
Step III: Replace MicroBlaze with HW

- External connections (Memory bus and GAScore interface) stay the same
- Custom core only does computation
- PAMS needs to be configured for communication in-between computation phases
- Keeping one SW GASNet thread on PC to
  - Move data into FPGA memory
  - Program PAMS
  - Output benchmark time
# Performance comparison

Time for one complete iteration (averaged)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 PCs, 16 threads</td>
<td>4.54s</td>
</tr>
<tr>
<td>16 MicroBlazes</td>
<td>222.86s</td>
</tr>
<tr>
<td>16 Custom FPGA cores</td>
<td>5.69s</td>
</tr>
</tbody>
</table>

Memory-bound: No FPGA gain!
Conclusion

GASNet SW API + GAScore (+PAMS) enable

• Universal abstraction for FPGA interfaces
• Simple heterogeneous communication and synchronization
• Simplified kernel migration to accelerators
• Still have to pick the right problems for FPGA speedup!
Outlook: THePaC++
Heterogeneous C++ PGAS library

C++ PGAS Application
C++ PGAS Library
GASNet Library

Compile

CPU-based Host
THeGASNet

Dynamic generation

Manual or HLS
Static generation

PAMS
Custom FPGA Hardware

October 9, 2014
High-Performance Reconfigurable Computing Group • University of Toronto
Thank you for your attention!

Questions?