Application development co-design for FPGA-accelerated data center HPC servers

Mihai Lazarescu
Luciano Lavagno
Outline

• CPU trends, energy efficiency
• Toolset objectives and approach
• OpenCL to FPGA: good!
• Toolset flow
• Preliminary results
• Wrap-up
CPU trends

- Uptrend: transistor count

- Capped:
  - Power
  - Frequency
  - Perf./thread

- Efficient development flows?
Improve silicon efficiency

**GOPS/W**

- CPU ...................... 1
- GP-GPU .................. 3
- Accelerators
  - Software .............. 6
  - **FPGA .................. 30**
- Hardware IP ........... \(>100\)
- Bio ....................... 1000

[Ruch IBM 2011]
Toolset objectives and approach

• Get SW-like NRE costs with HW efficiency by:
  – Integrating advanced HW High-Level Synthesis (HLS) tools in a SW compilation flow for HW accelerators
  – Accepting a variety of concurrent models for better learn time and adoption by SW engineers
  – Using HLS to reduce HW design time (mostly verification time)
  – Improving Result Quality with manual and automated DSE

• Map SW on FPGA to:
  – Reduce run-time energy consumption
  – Reduce production cost (reusable components)
Why open source?

- OS builds community
  - Foster the use and fruitful exchanges of ideas
- OS fosters Academy-Industry cooperation
  - Both value creators, in synergistically complementary ways
- OS supports industry
  - Lowers (SMEs) entry costs
  - Creates jobs (also for students)
Multi-language input

• **Problem**: what high-level behavioral model for RTL synth?
  – C, C++, SystemC, Simulink/Stateflow, CUDA, OpenCL are successful to some extent, no definite winner

• **Objective**: don't learn a new language
  – Faster and cheaper adoption by software engineers
  – Development speed up by verification in domain-specific lang.

• **Solution**: C++/SystemC just as intermediate representation
  Domain-specific model ➤ C++/SystemC ➤ HLS tools
OpenCL HLS to FPGAs

- Data centers: lots of energy for computing and cooling
- Many data center-typical algorithms *embarrassingly parallel* (e.g., search, image and speech recognition)
  - Already *efficiently coded in parallel languages*
- FPGA implementation vs. CPU/GP-GPU programs:
  - Low energy
  - Good performance
  - Preserve HW reuse (reconfigurable by application)
    - Preserve reusability
    - Reduce dark silicon
Execution with FPGA accelerators

- **FPGA**: very high energy efficiency, dynamic reconfig.
- **OpenCL**: extreme parallelism, simple programming model
- **Dynamic resource allocation**: runtime FPGA reconfiguration
- **Global memory**: shared to all CPUs and FPGAs in cluster
  - No global cache coherency (efficiency)
OpenCL programming model

- Kernels are functional computation units
  - Mapped to CPU, GP-GPU or FPGA
- Kernels are split in *independent workgroups*
  - Run-time mapped to resources (best resource/performance trade-off)
- Workgroups are made of *synchronized workitems*
  - Share local memory (SRAM)
- Memory hierarchy:
  - Global DRAM, shared by kernels and host code
  - Local SRAM, shared by workitems (+ private registers)
- Code parallelization and optimized use of memory hierarchy already solved by SW engineer
SDAccel optimization flow

1. Un-optimized Kernels
   - Capture Memory Accesses Only
     - Optimize Memory Access
       - Build & Run on Board
         - Performance Goal Met? (Yes/No)
         - Incorporate New Memory Accesses into Kernels
           - Partially Optimized Kernels
   - Host Code
   - Partially Optimized Kernels
     - Optimize Data Path
       - Build & Run on Board
         - Compare Results to Test Kernel
           - Performance Goal Met? (Yes/No)
           - Fully Optimized Kernels

Speed-ups:
- 100x to 1000x
- 2x to 10x

Bologna, October 7-9, 2016
OpenCL to FPGA

• Both Xilinx and Altera support OpenCL with:
  – Workgroup replication, for best performance/resource trade-off
  – Pipelined workitems, for efficient HW implementation
  – Automate Design Space Exploration for:
    • Loops within a workitem
    • Local memory optimization

• Xilinx SDAccel
  – OpenCL functional debugging
  – Cost/performance analysis
  – Manual Design Space Exploration
    • Requires HW design expertise
    • To automate
Open Source OpenCL kernels

• Sponsored by Xilinx via University grants
  – To develop an OpenCL-based FPGA acceleration ecosystem
• Large library of Open Source OpenCL host and kernel code:
  – Optimized for FPGA implementation
  – Includes synthesis scripts
• Reference implementations for key areas:
  – Machine learning (e.g., neural networks, k-nearest neighbors)
  – Financial algorithms (e.g., Black Scholes, Heston)
  – Graph algorithms (e.g., Floyd Warshall, Dijkstra)
  – Database operations (e.g., sort, join)
Preliminary application examples

• Financial algorithms, e.g., Black-Scholes and Heston
  – Monte Carlo parallel simulations: local memory, not global
  – FPGA performance and energy much better than GP-GPU

• Machine learning, e.g., k-nearest neighbors
  – Limited by global memory bandwidth (GP-GPUs are typically better)
  – FPGAs use less energy and have better performance (if streaming)

• Sorting, e.g., bitonic sorting
  – Limited by global memory bandwidth (GP-GPUs are typically better)
  – FPGAs use less energy
Heston model of financial markets

- FPGA is competitive since global memory is not used

<table>
<thead>
<tr>
<th>platform</th>
<th>t(ns)</th>
<th>power(W)</th>
<th>energy/step(nJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTX 960</td>
<td>0.604</td>
<td>120</td>
<td>72</td>
</tr>
<tr>
<td>K4200</td>
<td>0.663</td>
<td>105</td>
<td>70</td>
</tr>
<tr>
<td>Virtex 7</td>
<td>1.424</td>
<td>12</td>
<td>17</td>
</tr>
</tbody>
</table>
K-nearest neighbors

- FPGA best due to streaming & on-chip global memory

<table>
<thead>
<tr>
<th>Platform</th>
<th>Time</th>
<th>Power(W)</th>
<th>Energy(J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTX 960</td>
<td>930ms</td>
<td>120</td>
<td>111.6</td>
</tr>
<tr>
<td>K4200</td>
<td>3110ms</td>
<td>108</td>
<td>335.88</td>
</tr>
<tr>
<td>Virtex 7</td>
<td>1ms</td>
<td>3</td>
<td>0.0039</td>
</tr>
</tbody>
</table>
Summary

- OpenCL and FPGAs very promising for data center HPC
- Excellent energy efficiency, good performance
- May need FPGA-specific high-level optimization, e.g.
  - Exploit global memory access bursts
- Encouraging results for different domain applications
  - Easier DSE than for other (less embarrassingly parallel) models
  - Dynamic resource management is key to data center and HPC use

ECOSCALE project: http://www.ecoscale.eu/