CHO: Towards a Benchmark Suite for OpenCL FPGA Accelerators

3rd Inter. Workshop on OpenCL 2015
Geoffrey Ndu, Javier Navaridas & Mikel Lujan
Talk Outline

1. Overview of FPGAs
2. Compiling OpenCL to FPGAs
3. CHO Benchmark Suite
4. Case Study: Compiling unmodified code to FPGA
Motivation and Objectives

CHO is a benchmark suite for OpenCL FPGA
http://it302.github.io/cho

- Developed in AXLE Project
  - Advanced Analytics for Extremely Large European Databases
  - We accelerate database algorithms on FPGA
- There is no open OpenCL FPGA benchmark
  - Software benchmarks are too large and complex
- Benchmarking is important
  - Allows compiler writers to evaluate ideas qualitatively
  - Enables users benchmark diverse frameworks
What are FPGAs

<table>
<thead>
<tr>
<th>Truth table</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

LookUp Table

8-to-1 MUX

SRAM Cells
What are FPGAs

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>ab + c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

LookUp Table

SRAM Cells

8-to-1 MUX

f
What are FPGAs

### Truth table

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>(ab + c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### LookUp Table

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

8-to-1 MUX

SRAM Cells

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

f = 0
A Modern FPGA

- General-purpose I/Os
- Logic Fabric
- DSP Blocks
- BRAMs
- PCI Express
- 10G Ethernet
- High-Speed Serial Transceivers
How FPGAs outperform CPUs and GPUs

(a) ARM9 bit reversal
(b) FPGA bit reversal
OpenCL on FPGAs
CHO Benchmark Suite

- Work-in-progress
- Currently is a port of CHStone
- CHStone is the de-facto C HLS benchmark
  - HLS is hardware programming using algorithmic description
  - Easier than Verilog and VHDL type languages
  - Often trade-off usability/productivity for quality
- 12 ‘diverse’ and non-trivial applications
- Written in OpenCL task parallel model
## CHO’s 12

<table>
<thead>
<tr>
<th>Domain</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Arithmetic</strong></td>
<td>dfadd</td>
<td>IEC/IEEE double-precision floating-point addition</td>
</tr>
<tr>
<td></td>
<td>dfdiv</td>
<td>IEC/IEEE double-precision floating-point division</td>
</tr>
<tr>
<td></td>
<td>dfmul</td>
<td>IEC/IEEE double-precision floating-point multiplication</td>
</tr>
<tr>
<td></td>
<td>dfsin</td>
<td>Double-precision floating-point sine function</td>
</tr>
<tr>
<td><strong>Media</strong></td>
<td>adpcm</td>
<td>Adaptive differential pulse code encoder &amp; decoder</td>
</tr>
<tr>
<td></td>
<td>gsm</td>
<td>GSM residual pulse excitation/long term prediction coding</td>
</tr>
<tr>
<td></td>
<td>jpeg</td>
<td>JPEG image decoder</td>
</tr>
<tr>
<td></td>
<td>motion</td>
<td>Motion vector decoding for MPEG-2</td>
</tr>
<tr>
<td><strong>Cryptography</strong></td>
<td>aes</td>
<td>Implementation of Advanced Encryption Standard</td>
</tr>
<tr>
<td></td>
<td>blowfish</td>
<td>Blowfish Encryption Algorithm</td>
</tr>
<tr>
<td></td>
<td>sha</td>
<td>Secure Hash Algorithm</td>
</tr>
<tr>
<td><strong>Miscellaneous</strong></td>
<td>MIPS</td>
<td>Simplified MIPS processors</td>
</tr>
</tbody>
</table>
Source Level Characterization CHO

- Characterize to make sure applications non-trivial
- Built clang AST tool for analysing sources
  - Clang is front end for the C-type languages
  - Uses LLVM as back-end
- Tool walks AST and classifies nodes
- Useful for finding expensive operators
# Source-Level Characteristics

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Type</th>
<th>LoC</th>
<th>Functions</th>
<th>Variables</th>
<th>Statements</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scalar</td>
<td>Aggregate</td>
<td>for</td>
<td>if</td>
</tr>
<tr>
<td>adpcm</td>
<td>32-bit int</td>
<td>463</td>
<td>21</td>
<td>103</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>aes</td>
<td>32-bit int</td>
<td>667</td>
<td>10</td>
<td>42</td>
<td>25</td>
<td>16</td>
</tr>
<tr>
<td>blowfish</td>
<td>8-bit char</td>
<td>495</td>
<td>4</td>
<td>38</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>dfadd</td>
<td>64-bit int</td>
<td>352</td>
<td>81</td>
<td>2</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>dfdiv</td>
<td>64-bit int</td>
<td>280</td>
<td>19</td>
<td>96</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dfmul</td>
<td>64-bit int</td>
<td>350</td>
<td>16</td>
<td>74</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dfsin</td>
<td>64-bit int</td>
<td>611</td>
<td>31</td>
<td>178</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>gsm</td>
<td>32-bit int</td>
<td>310</td>
<td>12</td>
<td>45</td>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td>jpeg</td>
<td>32-bit int</td>
<td>1061</td>
<td>31</td>
<td>204</td>
<td>30</td>
<td>34</td>
</tr>
<tr>
<td>mips</td>
<td>32-bit int</td>
<td>215</td>
<td>1</td>
<td>17</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>motion</td>
<td>32-bit int</td>
<td>436</td>
<td>13</td>
<td>59</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>sha</td>
<td>64-bit char</td>
<td>178</td>
<td>8</td>
<td>42</td>
<td>15</td>
<td>10</td>
</tr>
</tbody>
</table>
IR-Level Characterization

- Compiler often translate software IR to HDL (verilog)
- IR-level provide missing info at source level
  - E.g. Num of loops left after optimization
- LLVM IR seems to be popular choice nowadays
  - LLVM is an open source modular compiler kit
  - Altera SDK is LLVM-based
- Built Clang/LLVM tool for IR-Level characterization
## IR-Level Characteristics

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Number Basic Blocks</th>
<th>Number Instructions</th>
<th>Number Loops</th>
</tr>
</thead>
<tbody>
<tr>
<td>adpcm</td>
<td>95</td>
<td>2240</td>
<td>19</td>
</tr>
<tr>
<td>aes</td>
<td>186</td>
<td>5685</td>
<td>23</td>
</tr>
<tr>
<td>blowfish</td>
<td>58</td>
<td>5441</td>
<td>10</td>
</tr>
<tr>
<td>dfadd</td>
<td>199</td>
<td>1421</td>
<td>1</td>
</tr>
<tr>
<td>dfdiv</td>
<td>115</td>
<td>1212</td>
<td>3</td>
</tr>
<tr>
<td>dfmul</td>
<td>90</td>
<td>846</td>
<td>1</td>
</tr>
<tr>
<td>dfsin</td>
<td>469</td>
<td>4216</td>
<td>2</td>
</tr>
<tr>
<td>gsm</td>
<td>149</td>
<td>1493</td>
<td>18</td>
</tr>
<tr>
<td>jpeg</td>
<td>661</td>
<td>10261</td>
<td>109</td>
</tr>
<tr>
<td>mips</td>
<td>46</td>
<td>472</td>
<td>3</td>
</tr>
<tr>
<td>motion</td>
<td>1743</td>
<td>15121</td>
<td>326</td>
</tr>
<tr>
<td>sha</td>
<td>79</td>
<td>1364</td>
<td>30</td>
</tr>
</tbody>
</table>
Case Study with CHO

- How easy to compile non-trivial algorithms on FPGAs
  - Especially when I/O interfacing is involved
- Could unmodified software OpenCL run on FPGA
Synthesizing CHO

- Altera OpenCL SDK 14.1
- Nallatech P385-A7 FPGA accelerator card
  - Stratix V FPGA
  - 8GB DDR3 RAM
  - 8-lane Gen 3 PCIe
  - 1/2 length and height card
- Unmodified software applications
## Summary of Synthesis

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Synthesizable</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>adpcm</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>aes</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>blowfish</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>dfadd</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>dfdiv</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>dfmul</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>dfsin</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>gsm</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>jpeg</td>
<td>✗</td>
<td>Front-end error</td>
</tr>
<tr>
<td>mips</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>motion</td>
<td>✗</td>
<td>Front-end error</td>
</tr>
<tr>
<td>sha</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
## Difference of Major Versions

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Synthesizable</th>
<th>Notes</th>
<th>Altera SDK 14.1</th>
<th>Synthesizable</th>
<th>Notes</th>
<th>Altera SDK 13.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>adpcm</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Adpcm</td>
</tr>
<tr>
<td>aes</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Adpcm</td>
</tr>
<tr>
<td>blowfish</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Blowfish</td>
</tr>
<tr>
<td>dfadd</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dfadd</td>
</tr>
<tr>
<td>dfdiv</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dfdiv</td>
</tr>
<tr>
<td>dfmul</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dfmul</td>
</tr>
<tr>
<td>dfsin</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dfsin</td>
</tr>
<tr>
<td>gsm</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gsm</td>
</tr>
<tr>
<td>jpeg</td>
<td>✗</td>
<td>Front-end error</td>
<td></td>
<td></td>
<td></td>
<td>Peg</td>
</tr>
<tr>
<td>mips</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mips</td>
</tr>
<tr>
<td>motion</td>
<td>✗</td>
<td>Front-end error</td>
<td></td>
<td></td>
<td></td>
<td>Motion</td>
</tr>
<tr>
<td>sha</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sha</td>
</tr>
</tbody>
</table>

Front-end error

Frontend crash

Backend error
## CPU vs FPGA

- Compare performance FPGA vs CPU
- Kernels not optimized for FPGA
- Some applications failed on the FPGA
  - 3 produced incorrect results
  - 2 never finished execution

<table>
<thead>
<tr>
<th>CPU</th>
<th>Intel Xeon CPU E31245 @ 3.30GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Size</td>
<td>16 GB DDR3</td>
</tr>
<tr>
<td>OpenCL SDK</td>
<td>Intel Kernel Builder for OpenCL 1.4.0.117</td>
</tr>
</tbody>
</table>

CPU Platform Features
Concluding

- Straightforward to compile software OpenCL to FPGA
- Kernels most likely need to be tuned for FPGA
- Compiler is getting better
- Working on CHO 2.0
  - Data parallel OpenCL kernels
  - More modern application e.g. database algorithms
  - Tuned for FPGAs
The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 318633. We wish to thank Altera University Program for their software donation.