CHiMPS
A C-level Compilation Flow for Heterogeneous CPU-FPGA Architectures

Andrew Putnam
Susan Eggers, Dave Bennett, Eric Dellinger, Jeff Mason, Prasanna Sundararajan

University of Washington                     Xilinx Research Labs
Motivation

How do we make reconfigurable computing work for HPC and scientific computing users?

<table>
<thead>
<tr>
<th></th>
<th>HPC Programmers</th>
<th>FPGA Programmers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Background</strong></td>
<td>Biology, Physics, Finance...</td>
<td>Electrical Engineering</td>
</tr>
<tr>
<td><strong>Languages</strong></td>
<td>C, Fortran, Matlab</td>
<td>HDLs (Verilog, VHDL)</td>
</tr>
<tr>
<td><strong>Optimization</strong></td>
<td>Cache tuning, Coarse-grain parallelization</td>
<td>Pipelining, state machines, very fine-grained parallelization</td>
</tr>
<tr>
<td><strong>Programming Model</strong></td>
<td><strong>Shared Memory</strong></td>
<td><strong>Streaming</strong></td>
</tr>
</tbody>
</table>

**HPC Programmer Wishlist**

- Very low barrier to entry
- Re-use existing code

- Compiler/hardware handle parallelism
- Code portable to future hardware
CHiMPS Goals

- C-level FPGA compiler
- Work with largely unmodified source code
- Run on existing systems – just add the FPGA
- Automatically, efficiently support the C memory model
- Better performance than CPU-only execution
- Lower Power

Emphasis on ease-of-use over maximum performance
Related Work – C for FPGAs

- HDLs with C syntax
  - Catapult-C, Hardware-C
- Predictable memory access
  - Impulse-C, Stream-C, NAPA-C, Nimble
- Embedded
  - C2H, Garp, xPilot
- Different Language
  - Mitrion-C, SA-C, ROCCC

None of these efficiently support shared-memory programming model
Target Platform

- Must be what HPC / Scientific users already use
- Multi-socket, multi-processor motherboard
- Runs commodity software, operating systems
Heterogeneous CPU-FPGA Platforms

- Full-featured commodity CPU for most tasks
- Use FPGA for performance-critical code
- Fast communication by sharing native processor bus
- CPU and FPGA are peers (not master-slave)
Compilation Flow

- Looks like a standard C compiler
- Largely unmodified C source code
- Compilation time from C to VHDL: 10-60 seconds
- 2-step flow makes CTL portable across platforms
void matmul(long* a, long* b, long* c, long sz)
{
  long i, j, k;
  for (i = 0; i < sz; i++)
  {
    long offset = i * sz;
    long* row = a + offset;
    long* out = c + offset;
    for (j = 0; j < sz; j++)
    {
      long* col = b + j;
      out[j] = 0;
      for (k = 0; k < sz; k++)
      {
        out[j] += row[k] * col[k*sz];
      }
    }
  }
}
Problems with a Single Cache

- Long Wires
Problems with a Single Cache

- Long Wires
- Arbitration latency

- SWM has 42 reads, 10 writes  (without resource sharing)
Problems with a Single Cache

- Long Wires
- Arbitration latency
- Sharing conflicts
for( k=0; k < sz; k++ ) {
    out[j] += row[k] * col[k*sz];
}
for( k=0; k < sz; k++ ) {
    out[j] += row[k] * col[k*sz];
}
for( k=0; k < sz; k++ ) {
    out[j] += row[k] * col[k*sz];
}
CHiMPS Many-Cache Model

```c
for( k=0; k < sz; k++ ) {
    out[j] += row[k] * col[k*sz];
}
```
for( k=0; k < sz; k++ ) {
    out[j] += row[k] * col[k*sz];
}
Many-Cache Benefits

- Shorter wires
- Lower latency
- Fewer sharing conflicts
- Customize for access type
- Supports applications with the shared memory model
## Results: Xilinx ACP

### Processor
<table>
<thead>
<tr>
<th>Intel Xeon 7300</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cores</strong></td>
</tr>
<tr>
<td><strong>Frequency</strong></td>
</tr>
<tr>
<td><strong>FSB Frequency</strong></td>
</tr>
</tbody>
</table>

### FPGA
<table>
<thead>
<tr>
<th>Xilinx Virtex-5 LX110T</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logic Cells</strong></td>
</tr>
<tr>
<td><strong>Slices</strong></td>
</tr>
<tr>
<td><strong>BRAMs (36kBit)</strong></td>
</tr>
<tr>
<td><strong>Distributed RAM</strong></td>
</tr>
<tr>
<td><strong>Multipliers / DSPs</strong></td>
</tr>
<tr>
<td><strong>Peak Power</strong></td>
</tr>
</tbody>
</table>
Performance

7.8x performance boost over CPU
Power

4.3x lower power than CPU
Performance per Watt

23.3x better performance per Watt
Summary

- Discussed Many-Cache: a simple FPGA programming model for HPC programmers

- 7.8x mean performance boost over Xeon CPU
- 4.3x lower power consumption
- 23.3x better performance per Watt

- Multiple banks, multiple caches are key to FPGA performance

- FPGAs programmed with CHiMPS can efficiently support C and the shared-memory programming model

HPC Wish List
- Very low barrier to entry
- Re-use existing code
- Compiler/hardware handle parallelism
- Code portable to future hardware
Questions