Wideband Channelization for Software-Defined Radio on GPU

Vignesh Adhinarayanan Krzysztof Kepa, Wu Feng

Departments of CS & ECE
Roadmap

- The Computing Universe and OpenCL
  - Programming Model for Wideband Channelization
- Goals of Wideband Channelization
- Overall Results
- Detailed Profiling, Optimizations, and Results
- Publications
The Computing Universe

- **Diversity of Architectures**
  - Homogeneous
    - CPU, GPU, FPGA, etc.
  - Heterogeneous multi-/many-core
    - CPU + FPGA (Convey)
    - CPU + GPU (AMD & NVIDIA)
    - APU (AMD) or Xeon Phi (Intel)
    - Cell (effectively dead)

- **Diversity of Algorithms**
  - OpenDwarfs, a realization of Berkeley’s 13 dwarfs, where a dwarf is a fundamental computation & communication pattern

- **Diversity of Optimizations**

- **Challenges**
  - **Programmability**
    - Reduce learning curve. How?
  - **Portability**
    - Eliminate re-implementation
  - **Productivity**
    - Reduce “time to solution”

- **APU/CPU/GPU vs. FPGA**
  - [x]PU programmability, tools, and support superior to that of FPGA
  - FPGA requires
    - HW design knowledge
    - Manual optimizations for performance
    - Long implementation cycles
      - Hours/days instead of seconds/minutes
Addressing ....

**Programmability, Portability, and Productivity**

Solution? **OpenCL: Open Computing Language**

- Cross-platform standard for building parallel applications
  - Host CPU + accelerator
- Host application
  - Compiled once
- Computation kernel
  - Platform-independent
  - Compilation in the runtime
  - Computations defined for single processing element
OpenCL: Models (Continued)

- **Execution Model**
  - 3D index space
  - Work-groups
    - Group/Local/Global ID
  - Commands
    - Execution/Mem Access/ Sync

- **Memory Model**
  - Host memory
  - Global memory
  - Local memory
OpenCL: Models (Continued)

■ Programming Model
  ○ Host app uses OpenCL API
    ▪ Comm. / control / sync
  ○ Kernels specified in subset of C99
    ▪ Restricted
      ▪ No recursion
      ▪ No function pointers
    ▪ Extended
      ▪ Address-space qualifiers
      ▪ Synchronization
      ▪ Built-in functions

■ Related Work
  ○ Altera OpenCL
  ○ FCUDA
    ▪ CUDA → Autopilot C
  ○ OpenRCL
  ○ SOpenCL
Goals for Wideband Channelization on GPU

- An efficient mapping of the performance-critical stages of the PFB channelizer on a graphics processing unit (GPU).

- Realization of a number of optimization techniques for improving the performance on such devices and reducing associated data-transfer overheads.

- Evaluation of our implementation on mobile graphics processors, including the first such evaluation on integrated CPU+GPU devices.
Overall Results

- Up to
  - 43-fold speed-up for discrete mobile GPU (HD 6470M)
  - 27-fold speed-up for integrated mobile GPU (HD 6480G)

when compared to an optimized multi-threaded CPU implementation running on A4-3300M CPU.
Profile of the Polyphase Filter Bank

![Bar chart showing the percentage execution time for different number of channels.](chart.png)

- **Number of channels**: 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072
- **Execution Time**:%  
  - FIR Filtering
  - FFT
  - Channel Mapping

---

CHREC
NSF Center for High-Performance Reconfigurable Computing
General Optimizations

- System-Wide Optimization
  - Reduce data-transfer overhead

- GPU Optimizations
  - Increase GPU occupancy by batching
  - Transform the data layout
  - Use local memory and use constant memory
  - Reduce dynamic instruction count
  - Vectorization
  - Reduce branching
  - Use special instructions (e.g., FMA)
Results: Impact of Optimizations and Speed-Up

**Hardware Platforms**

<table>
<thead>
<tr>
<th>H/W Platform</th>
<th>A4-3300 M</th>
<th>HD 6480G</th>
<th>HD 6470M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Multicore CPU</td>
<td>Integrated GPU</td>
<td>Discrete GPU</td>
</tr>
<tr>
<td>CUs</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Cores</td>
<td>2</td>
<td>240</td>
<td>160</td>
</tr>
<tr>
<td>Clock (MHz)</td>
<td>1900</td>
<td>444</td>
<td>750</td>
</tr>
<tr>
<td>Mem Size (MB)</td>
<td>4096</td>
<td>512</td>
<td>1024</td>
</tr>
<tr>
<td>Mem Clock (MHz)</td>
<td>675</td>
<td>675</td>
<td>800</td>
</tr>
<tr>
<td>TDP</td>
<td>35W (combined)</td>
<td>9W</td>
<td></td>
</tr>
</tbody>
</table>

**Software Platform**

- Operating System: 64-bit Windows 7 OS
- SDK: AMD APP v2.8
- OpenCL: Version 1.1
- FFT Library: clAmdFft-1.10.274
Evaluation & Optimization of FFT

Application
- 1D FFT. 16-, 64-, and 256-pt (Spectral method dwarf)

Approach
- Apply optimizations in isolation and in concert to characterize machine-level behavior.

Testbed
- AMD Radeon HD 6970/7970
- NVIDIA Tesla C2075/K20c

Notable Insights
- In concert, (1) 99% of execution time is comprised of global memory data transfer, and (2) the optimal set of optimizations in concert is: register preloading, coalesced global access pattern, 8- or 16-byte vector access, scalar arithmetic, transposition via local memory, and constant memory usage.
Recent Publications

- Under Review
  - C. del Mundo, W. Feng, “Towards a Performance-Portable FFT Library for Heterogeneous Computing,” *IEEE IISWC ’13* (Submitted April 2013.)