GPU VSIPL: High Performance VSIPL Implementation for GPUs

Andrew Kerr, Dan Campbell*, Mark Richards, Mike Davis

andrew.kerr@gtri.gatech.edu, dan.campbell@gtri.gatech.edu,
mark.richards@ece.gatech.edu, mike.davis@gtri.gatech.edu

High Performance Embedded Computing (HPEC) Workshop

24 September 2008

Distribution Statement (A): Approved for public release; distribution is unlimited

This work was supported in part by DARPA and AFRL under contracts FA8750-06-1-0012 and FA8650-07-C-7724. The opinions expressed are those of the authors.
Signal Processing on Graphics Processors

- GPUs original role: turn 3-D polygons into 2-D pixels...
  - ...Which also makes them cheap & plentiful source of FLOPs
    - Leverages volume & competition in entertainment industry
    - Primary role highly parallel, very regular
    - Typically <$500 drop-in addition to standard PC
- Outstripping CPU capacity, and growing more quickly
  - Peak theoretical ~1TFlop
  - Power draw: 280GTX = 200W  Q6600 = 100W
  - Still making improvements in market app with more parallelism, so growth continues
GPGPU (Old) Concept of Operations

- Arrays ➔ Textures
- Render polygons with the same pixel dimensions as output texture
- Execute a fragment program to perform desired calculation
- Move data from output buffer to desired texture

Now we have compute-centric programming models…
… But they require expertise to fully exploit
VSIPL - Vector Signal Image Processing Library

- Portable API for linear algebra, image & signal processing
- Originally sponsored by DARPA in mid ’90s
- Targeted embedded processors – portability primary aim
- Open standard, Forum-based
- Initial API approved April 2000

- Functional coverage
  - Vector, Matrix, Tensor
  - Basic math operations, linear algebra, solvers, FFT, FIR/IIR, bookkeeping, etc
VSIPL & GPU: Well Matched

- VSIPL is great for exploiting GPUs
  - High level API with good coverage for dense linear algebra
  - Allows non experts to benefit from hero programmers
  - Explicit memory access controls
  - API precision flexibility

- GPUs are great for VSIPL
  - Improves prototyping by speeding algorithm testing
  - Cheap addition allows more engineers access to HPC
  - Large speedups without needing explicit parallelism at application level
GPU-VSIPL Implementation

• Full, compliant implementation of VSIPL Core-Lite Profile
• Fully encapsulated CUDA backend
  • Leverages CUFFT library
  • All VSIPL functions accelerated
• Core Lite Profile:
  • Single precision floating point, some basic integer
  • Vector & Sxalar, complex & real support
  • Basic elementwise, FFT, FIR, histogram, RNG, support
  • Full list: http://www.vsipl.org/coreliteprofile.pdf

• Also, some matrix support, including vsip_fftm_f
CUDA Programming & Optimization

CUDA Optimization Considerations

- Maximize occupancy to hide memory latency
- Keep lots of threads in flight
- Carefully manage memory access to allow coalesce & avoid conflicts
- Avoid slow operations (e.g. integer multiply for indexing)
- Minimize synch barriers
- Careful loop unrolling
- Hoist loop invariants
- Reduce register use for greater occupancy

"GPU Performance Assessment with the HPEC Challenge" – Thursday PM
GPU VSIPL Speedup: Unary

- nVidia 8800GTX Vs Intel Q6600
- Speedup vs Vector Length
- Speedup ranges from 320X to 20X
GPU VSIPL Speedup: Binary

nVidia 8800GTX Vs Intel Q6600

Speedup vs Vector Length

- vmul
- vadd
- vdiv
- vsub
- cvadd
- cvsub
- cmul
- cvjmul

Speedup: 25X, 40X
GPU VSIPL Speedup: FFT

nVidia 8800GTX Vs Intel Q6600

Vector Length

- Real to Complex
- Complex to Complex
GPU VSIPL Speedup: FIR

Filter Length 1024

nVidia 8800GTX
Vs
Intel Q6600

157X
Application Example: Range Doppler Map

- Simple Range/Doppler data visualization application demo
- Intro app for new VSIPL programmer
- 59x Speedup TASP ➔ GPU-VSIPL
- No changes to source code

<table>
<thead>
<tr>
<th>Section</th>
<th>9800GX2 Time (ms)</th>
<th>Q6600 Time (ms)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admit</td>
<td>8.88</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Baseband</td>
<td>67.77</td>
<td>1872.3</td>
<td>28</td>
</tr>
<tr>
<td>Zeropad</td>
<td>23.18</td>
<td>110.71</td>
<td>5</td>
</tr>
<tr>
<td>Fast time FFT</td>
<td>47.25</td>
<td>5696.3</td>
<td>121</td>
</tr>
<tr>
<td>Multiply</td>
<td>8.11</td>
<td>33.92</td>
<td>4</td>
</tr>
<tr>
<td>Fast Time FFT⁻¹</td>
<td>48.59</td>
<td>5729.04</td>
<td>118</td>
</tr>
<tr>
<td>Slow time FFT, 2x CT</td>
<td>12.89</td>
<td>3387</td>
<td>263</td>
</tr>
<tr>
<td>log10</td>
<td>.</td>
<td>²</td>
<td>22.2</td>
</tr>
<tr>
<td>Release</td>
<td>54.65</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total:</td>
<td>293.52</td>
<td>17299.42</td>
<td>59</td>
</tr>
</tbody>
</table>
GPU-VSIPL: Future Plans

- Expand matrix support
- Move toward full Core Profile
- More linear algebra/solvers
- VSIPL++
- Double precision support
Conclusions

• GPUs are fast, cheap signal processors
• VSIPL is a portable, intuitive means to exploit GPUs
• GPU-VSIPL allows easy access to GPU performance without becoming an expert CUDA/GPU programmer
• 10-100x speed improvement possible with no code change

• Not yet released, but unsupported previews may show up at: http://gpu-vsipl.gtri.gatech.edu