18-447 Lecture 27: Hardware Acceleration

James C. Hoe
Department of ECE
Carnegie Mellon University
Housekeeping

• Your goal today
  – see why you should care about accelerators
  – know the basics to think about the topic

• Notices
  – Lab4, due this week
  – HW5, past due

• Readings
  – *Amdahl's Law in the Multicore Era, 2008* (optional)
  – *Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs?* 2010 (optional)
“HW Acceleration” is nothing new!

• What needed to be faster/smaller/cheaper/lower-energy than SW has always been done in HW
  – we go to HW when SW isn’t good enough because “good” HW can be more efficient
  – we don’t go to HW when SW is good enough because “good” HW takes more work
• When we say “HW acceleration”, we always mean efficient and not just correct
Computing’s Brave New World

Microsoft Catapult
[MICRO 2016, Caulfield, et al.]

Google TPU
[Hotchips, 2017, Jeff Dean]
How we got here . . . .

1970~2005

2005~??
Moore’s Law without Dennard Scaling

Under fixed power ceiling, more ops/second only achievable if less Joules/op?
Future is about **Performance/Watt and Ops/Joule**

This is a sign of desperation . . . .
Why is Computing Directly in Hardware Efficient?
Why is HW/FPGA better?

no overhead

• A processor spends a lot of transistors & energy
  – to present von Neumann ISA abstraction
  – to support a broad application base (e.g., caches, superscalar out-of-order, prefetching, . . .)

• In fact, processor is mostly overhead
  – ~90% energy [Hameed, ISCA 2010, Tensilica core]
  – ~95% energy [Balfour, CAL 2007, embedded RISC ]
  – even worse on a high-perf superscalar-OoO proc

Computing directly in application-specific hardware can be 10x to 100x more energy efficient
Why is HW/FPGA better? efficiency of parallelism

- For a given functionality, non-linear tradeoff between power and performance
  - slower design is simpler
  - lower frequency needs lower voltage

⇒ For the same throughput, replacing 1 module by 2 half-as-fast reduces total power and energy

Good hardware designs derive performance from parallelism
Software to Hardware Spectrum

- **CPU**: highest-level abstraction / most general-purpose support
- **GPU**: explicitly parallel programs / best for SIMD, regular
- **FPGA**: ASIC-like abstraction / overhead for reprogrammability
- **ASIC**: lowest-level abstraction / fixed application and tuning
ASIC isn’t always ultimate in performance

• Amdahl’s Law: \( S_{\text{overall}} = 1 / ( (1-f) + f/S_f ) \)
• \( S_{f-\text{ASIC}} > S_{f-\text{FPGA}} \) but \( f_{\text{ASIC}} \neq f_{\text{FPGA}} \)
• \( f_{\text{FPGA}} > f_{\text{ASIC}} \) (when not perfectly app-specific)
  – more flexible design to cover a greater fraction
  – reprogram FPGA to cover different applications

[based on Joel Emer’s original comment about programmable accelerators in general]
Tradeoff in Heterogeneity?
Amdahl’s Law on Multicore

- A program is rarely completely parallelizable; let’s say a fraction $f$ is perfectly parallelizable.
- Speedup of $n$ cores over “sequential”

$$Speedup = \frac{1}{(1-f) + \frac{f}{n}}$$

- But, “sequential” above determined by how many cores to dice an area into

Base Core Equivalent (BCE) in [Hill and Marty, 2008]
http://research.cs.wisc.edu/multifacet/amdahl/

more smaller cores ← size of cores in BCE → fewer larger cores

16x 1-BCE cores

1x 16-BCE core

assume perf grows with sqrt of area
Asymmetric Multicores

- Pwr/area-efficient “slow” BCEs vs pwr/area-hungry “fast” core
  - fast core for sequential code
  - slow cores for parallel sections
- [Hill and Marty, 2008]

\[
\text{Speedup} = \frac{1}{\frac{1-f}{\text{perf}_{\text{seq}}} + \frac{f}{(n-r) + \text{perf}_{\text{seq}}}}
\]

- \( r \) = cost of fast core in BCE
- \( \text{perf}_{\text{seq}} \) = speedup of fast core over BCE
- solve for optimal die allocation
http://research.cs.wisc.edu/multifacet/amdahl/
Heterogeneous Multicores
[Chung, et al. MICRO 2010]

\[
\text{Speedup} = \frac{1}{1 - f + \frac{f}{\text{perf}_{\text{seq}}(n-r)}}
\]

[Hill and Marty, 2008] simplified

- \( f \) is fraction parallelizable
- \( n \) is total die area in BCE units
- \( r \) is fast core area in BCE units
- \( \text{perf}_{\text{seq}}(r) \) is fast core perf. relative to BCE

For the sake of analysis, break the area for GPU/FPGA/etc. into units of **U-cores** that are the same size as BCEs. Each U-core type is characterized by a relative performance \( \mu \) and relative power \( \varphi \) compared to a BCE.
Modeling Power and Bandwidth Budgets

- The above is based on area alone
- Power or bandwidth budget limits the usable die area
  - if $P$ is total power budget expressed as a multiple of a BCE’s power,
    \[ n - r \leq \frac{P}{\phi} \]
  - if $B$ is total memory bandwidth expressed as a multiple of BCEs,
    \[ n - r \leq \frac{B}{\mu} \]

\[
\text{Speedup} = \frac{1}{1 - \frac{f}{\text{perf}_{\text{seq}}}} + \frac{f}{\mu \times (n - r)}
\]
### \( \varphi \) and \( \mu \) example values

<table>
<thead>
<tr>
<th></th>
<th>MMM</th>
<th>Black-Scholes</th>
<th>FFT-2(^10)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nvidia GTX285</strong></td>
<td>( \varphi )</td>
<td>0.74</td>
<td>0.57</td>
</tr>
<tr>
<td></td>
<td>( \mu )</td>
<td>3.41</td>
<td>17.0</td>
</tr>
<tr>
<td><strong>Xilinx LX760</strong></td>
<td>( \varphi )</td>
<td>0.31</td>
<td>0.26</td>
</tr>
<tr>
<td></td>
<td>( \mu )</td>
<td>0.75</td>
<td>0.26</td>
</tr>
<tr>
<td><strong>Custom Logic</strong></td>
<td>( \varphi )</td>
<td>0.79</td>
<td>4.75</td>
</tr>
<tr>
<td></td>
<td>( \mu )</td>
<td>27.4</td>
<td>482</td>
</tr>
</tbody>
</table>

Nominal BCE based on an Intel Atom in-order processor, 26mm\(^2\) in a 45nm process.
## Case Study [Chung, MICRO 2010]

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>GPUs</th>
<th>FPGA</th>
<th>ASIC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Intel Core i7-960</td>
<td>Nvidia GTX285</td>
<td>ATI R5870</td>
<td>Xilinx V6-LX760</td>
</tr>
<tr>
<td>Node</td>
<td>45nm</td>
<td>55nm</td>
<td>40nm</td>
<td>40nm</td>
</tr>
<tr>
<td>Die area</td>
<td>263mm²</td>
<td>470mm²</td>
<td>334mm²</td>
<td>-</td>
</tr>
<tr>
<td>Clock rate</td>
<td>3.2GHz</td>
<td>1.5GHz</td>
<td>1.5GHz</td>
<td>0.3GHz</td>
</tr>
</tbody>
</table>

### Single-prec. floating-point apps

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M-M-Mult</td>
<td>MKL 10.2.3 Multithreaded</td>
<td>CUBLAS 2.3</td>
<td>CAL++</td>
<td>hand-coded</td>
</tr>
<tr>
<td>FFT</td>
<td>Spiral.net Multithreaded</td>
<td>CUFFT 2.3 3.0/3.1</td>
<td>-</td>
<td>Spiral.net</td>
</tr>
<tr>
<td>Black-Scholes</td>
<td>PARSEC multithreaded</td>
<td>CUDA 2.3</td>
<td>-</td>
<td>hand-coded</td>
</tr>
</tbody>
</table>
"Best-Case" Performance and Energy

<table>
<thead>
<tr>
<th>Device</th>
<th>GFLOP/s actual</th>
<th>(GFLOP/s)/mm² normalized to 40nm</th>
<th>GFLOP/J normalized to 40nm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Core i7 (45nm)</td>
<td>96</td>
<td>0.50</td>
<td>1.14</td>
</tr>
<tr>
<td>Nvidia GTX285 (55nm)</td>
<td>425</td>
<td>2.40</td>
<td>6.78</td>
</tr>
<tr>
<td>ATI R5870 (40nm)</td>
<td>1491</td>
<td>5.95</td>
<td>9.87</td>
</tr>
<tr>
<td>Xilinx V6-LX760 (40nm)</td>
<td>204</td>
<td>0.53</td>
<td>3.62</td>
</tr>
<tr>
<td>same RTL std cell (65nm)</td>
<td>---</td>
<td>19.28</td>
<td>50.73</td>
</tr>
</tbody>
</table>

- CPU and GPU benchmarking is compute-bound; FPGA and Std Cell effectively compute-bound (no off-chip I/O)
- Power (switching+leakage) measurements isolated the core from the system
- For detail see [Chung, et al. MICRO 2010]
## Less Regular Applications

<table>
<thead>
<tr>
<th></th>
<th>GFLOP/s</th>
<th>(GFLOP/s)/mm²</th>
<th>GFLOP/J</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FFT-2¹⁰</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Core i7 (45nm)</td>
<td>67</td>
<td>0.35</td>
<td>0.71</td>
</tr>
<tr>
<td>Nvidia GTX285 (55nm)</td>
<td>250</td>
<td>1.41</td>
<td>4.2</td>
</tr>
<tr>
<td>ATI R5870 (40nm)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Xilinx V6-LX760 (40nm)</td>
<td>380</td>
<td>0.99</td>
<td>6.5</td>
</tr>
<tr>
<td>same RTL std cell (65nm)</td>
<td>952</td>
<td>239</td>
<td>90</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Mopt/s</th>
<th>(Mopt/s)/mm²</th>
<th>Mopt/s/J</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Black-Scholes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Core i7 (45nm)</td>
<td>487</td>
<td>2.52</td>
<td>4.88</td>
</tr>
<tr>
<td>Nvidia GTX285 (55nm)</td>
<td>10756</td>
<td>60.72</td>
<td>189</td>
</tr>
<tr>
<td>ATI R5870 (40nm)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Xilinx V6-LX760 (40nm)</td>
<td>7800</td>
<td>20.26</td>
<td>138</td>
</tr>
<tr>
<td>same RTL std cell (65nm)</td>
<td>25532</td>
<td>1719</td>
<td>642.5</td>
</tr>
</tbody>
</table>
Combine Model with ITRS Trends

<table>
<thead>
<tr>
<th>Year</th>
<th>2011</th>
<th>2013</th>
<th>2016</th>
<th>2019</th>
<th>2022</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>40nm</td>
<td>32nm</td>
<td>22nm</td>
<td>16nm</td>
<td>11nm</td>
</tr>
<tr>
<td>Core die budget (mm(^2))</td>
<td>432</td>
<td>432</td>
<td>432</td>
<td>432</td>
<td>432</td>
</tr>
<tr>
<td>Normalized area (BCE)</td>
<td>19</td>
<td>37</td>
<td>75</td>
<td>149</td>
<td>298 (16x)</td>
</tr>
<tr>
<td>Core power (W)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Bandwidth (GB/s)</td>
<td>180</td>
<td>198</td>
<td>234</td>
<td>234</td>
<td>252 (1.4x)</td>
</tr>
<tr>
<td>Rel pwr per device</td>
<td>1X</td>
<td>0.75X</td>
<td>0.5X</td>
<td>0.36X</td>
<td>0.25X</td>
</tr>
</tbody>
</table>

- 2011 parameters reflect high-end systems of the day; future parameters extrapolated from ITRS 2009
- 432mm\(^2\) populated by an optimally sized Fast Core and U-cores of choice
Single-Prec. MMMult (f=99%)
Single-Prec. MMMult (f=90%)
Single-Prec. MMMult ($f=50\%$)

![Graph showing speedup for different technologies and process nodes. The graph compares SymMC, AsymMC, ASIC, GPU R5870, and FPGA LX760. The x-axis represents different technology nodes: 40nm, 32nm, 22nm, 16nm, and 11nm. The y-axis represents speedup. The graph includes lines for Power Bound and Mem Bound.]
Single-Prec. FFT-1024 (f=99\%)
FFT-1024 (f=99%) 

if 1TB/sec memory bandwidth

![Graph showing speedup for different technologies and process nodes.](image-url)
You will be seeing more of this

- Performance scaling requires improved efficiency in Op/Joules and Perf/Watt
- Hardware acceleration is the most direct way to improve energy/power efficiency
- Need better hardware design methodology to enable application developers (without losing hardware’s advantages)
- Software is easy; hardware is hard?

*Hardware isn’t hard; perf and efficiency is!!!