18-643 Lecture 1: Welcome, why are you here?

James C. Hoe
Department of ECE
Carnegie Mellon University

Housekeeping

• Your goal today: decide if you are coming back . . .
• Notices (all handouts on Blackboard)
  – Handout #1: syllabus
  – Handout #1a: academic integrity statements
  – Handout #2: lab 0, due noon, 9/8
  – Complete survey on Blackboard, due noon, 9/5
• Readings
  – Ch 1, Reconfigurable Computing
Field Programmable Gate Arrays: in the beginning

- I/O pins
- Programmable routing
- Programmable lookup tables (LUT) and flip-flops (FF)
  aka “soft logic” or “fabric”

A Quite Wondrous Device

- Make an ASIC from your desk all by yourself
  - no manufacturing NRE (non-recurring eng.) cost
  - faster design time: try out increments as you go
  - less validation time: debug as you go at full speed / can also patch after shipping
- But
  - high unit cost (not for high-volume products)
  - “~10x” overhead in area/speed/power/....
  - RTL-level design abstraction
- Somewhere between ASICs and processors
FPGA “Growing Pains”

- Real designers make ASICs
- It is not programmable if it is not “C”
  - until 2005, CPUs were fast and getting faster
  - after 2005, GPGPU happened
- Where are the killer apps?
  - performance demanding but not too demanding
  - enough volume but not too high
  - high-concurrency but not totally regular
  - functionalities evolve quickly but not too quickly

FPGA Killer Apps Over Time

- 1990s: glue logic, embedded cntrl, interface logic
  - reduce chip-count, increase reliability
  - rapid roll-out of “new” products
- 2000s: DSP and HPC
  - strong need for performance
  - abundant parallelism and regularity
  - low-volume, high-valued
- 2010s: communications and networking
  - throughput performance
  - fast-changing designs and standards
  - price insensitive
  - $value in field updates and upgrades

What is in store for 2020?
“Age of Expansion”

[Fig 8, S. M. Trimberger, “Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology.”]

Fast-forward through Moore’s Law

2015: 16nm FINFET interposer die stacking

[Fig 1, S. M. Trimberger, “Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology.”]
“Age of Accumulation”

[Fig 11, S. M. Trimberger, “Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology.”]

FPGA in its Modern Form

Strong Interest now in Computing

- **Microsoft** has FPGAs in datacenters (Bing, Azure, Brainwave)
  - they are not talking about 1 or 2 (or even 1k or 2k)
  - try googling also
    “<<big-cloud-company-X>> FPGA datacenter”
- **IBM** and **Intel** want to sell you CPUs with cache-coherent FPGA accelerators
  - BTW, Intel bought Altera
  - also, Micron bought Convey
- You can buy FPGA fabric IP to add to your chip

Yes, something is going on! But what?

---

Moore’s Law without Dennard Scaling

To get more performance must do more operations/second for less Joules/second
Future is about
Performance/Watt and Ops/Joule

This is a sign of desperation . . .

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
## 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>263mm$^2$</td>
<td>470mm$^2$</td>
<td>334mm$^2$</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

<table>
<thead>
<tr>
<th>M-M-Mult</th>
<th>MKL 10.2.3 Multithreaded</th>
<th>CUBLAS 2.3</th>
<th>CAL++</th>
<th>hand-coded</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>Spiral.net Multithreaded</td>
<td>CUFFT 2.3</td>
<td>-</td>
<td>Spiral.net</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0/3.1</td>
<td></td>
<td></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>MMM</th>
<th>Device</th>
<th>GFLOP/s actual</th>
<th>(GFLOP/s)/mm$^2$ 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>
<td></td>
</tr>
<tr>
<td>Nvidia GTX285 (55nm)</td>
<td>425</td>
<td>2.40</td>
<td>6.78</td>
<td></td>
</tr>
<tr>
<td>ATI R5870 (40nm)</td>
<td>1491</td>
<td>5.95</td>
<td>9.87</td>
<td></td>
</tr>
<tr>
<td>Xilinx V6-LX760 (40nm)</td>
<td>204</td>
<td>0.53</td>
<td>3.62</td>
<td></td>
</tr>
<tr>
<td>same RTL std cell (65nm)</td>
<td>---</td>
<td>19.28</td>
<td>50.73</td>
<td></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 2⁻¹⁰</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/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>

ASIC isn’t always ultimate in performance

- Amdahl’s Law: \( S_{overall} = \frac{1}{(1-f) + f/S_f} \)
- \( S_{FASIC} > S_{F-FPGA} \) but \( f_{ASIC} \neq f_{FPGA} \)
- \( f_{FPGA} > f_{ASIC} \) (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]
Digress Further: Other Looming Dooms

- Memory Wall
  - Moore’s Law scaled DRAM in capacity not speed
  - relative to logic, DRAM looks slower and slower
  - dark-silicon from data starvation?

- Complexity Wall
  - designer productivity grew slower than Moore’s Law on logic capacity
  - design team grew exponentially to make up
  - more transistors than a team can make all work?
    (see Mythical Man-Month)

Go Over Blackboard and Syllabus
Be Forewarned

- This is still a “young” course
  - no course out there exactly like this one
  - the topic area is “unsettled” and in transition
- This is a hard course
  - 4 “training” labs; 1 large open-ended project
  - 1 midterm (1st half)
  - weekly paper reviews (2nd half)
  - you must speak up in class
- I am assuming
  - you are into computer hardware
  - you know RTL
  - you are willing to suffer for performance