18-643 Lecture 6: Hard vs Soft Logic

James C. Hoe
Department of ECE
Carnegie Mellon University
Housekeeping

• Your goal today: understand the difference between hard and soft logic

• Notices
  – Handout #3: lab 1, due noon, 9/22

• Readings (skim)
The Project Template

• Pick a “compute” application
• Pick a metric of merit
• Study implementation options on the Zedboard
  – a good software implementation must be one option
  – the rest is up to you
• Report findings

Keep in mind, you have optimistically 6 weeks; don’t forget you are taking other courses
DoF: you pick the application

- The problem could be
  - well studied (expect thoroughness and depth)
  - unproven (credit for honest attempts)
  Convince us it is 6 weeks of effort

- Something there is a reason to do on FPGAs

- Best if it is something you want to or have to do anyways

- Need to find and study (at least) 1 closely relevant research paper as starting point
DoF: you define the metric

• What you can study
  – performance (throughput or latency?)
  – cost (in terms of what?)
  – power and energy (how will you measure?)
  – design effort (what will you measure)
  – app-specific metrics (e.g., numerical accuracy)
  – composite metric: energy-delay-product, performance/watt, performance/$

• Must commit up-front
  – measurement procedure/benchmark
  – testable “good-enough” target condition
DoF: Platform

• You have the Zedboard
• You may substitute a reconfigurable platform you are already using (check with me first)
• You have access to more advanced platforms
  – risky learning curve to fit in 6 weeks
  – only if this plays into what else you are doing in life
DoF: Approach

• 1 option must be a good software-only baseline
• This is a “study”
  – do more than crank out implementations
  – think about what are the design choices
  – hypothesize the expected effects of your choices
  – corroborate hypothesis by implementation and evaluation

• Implementation approach:
  – no artificial bounds
  – how would you work in real-life?
  – if you have access, you can use it (including tools and IPs)

Convince us it is 6 weeks of effort
What makes a good project

• Interesting and/or important
• Not totally obvious *(to you)*
• You have special insights or interest
• Hard enough for 6 weeks
• Not too hard, too risky
• Most importantly, you should enjoy it

The above need not be an AND.
We now return you to our regularly scheduled program . . . .
FPGA Pro’s and Con’s

• Reasons for FPGAs
  – 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

• The price of FPGAs
  – high unit cost (not for high-volume products)
  – “~10x” overhead in area/speed/power/….
  – RTL-level design abstraction (relative to SW)
Hard vs Soft Processor Cores

- **Table 4.2: The Zynq Book**

<table>
<thead>
<tr>
<th>Processor</th>
<th>Configuration</th>
<th>DMIPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MicroBlaze 900LUT/700FF/2BRAM to 3800LUT/3200FF/6DSP/21BRAM</td>
<td>area optimized (3-stage)</td>
<td>196</td>
</tr>
<tr>
<td></td>
<td>perf. optimized (5-stage) with branch optimizations</td>
<td>228</td>
</tr>
<tr>
<td></td>
<td>perf. optimized (5-stage) without branch optimizations</td>
<td>259</td>
</tr>
<tr>
<td>ARM Cortex-A9</td>
<td>1GHz; both cores combined</td>
<td>5000</td>
</tr>
</tbody>
</table>

- **Table 4.3: The Zynq Book**

<table>
<thead>
<tr>
<th>Processor</th>
<th>Configuration</th>
<th>CoreMark</th>
</tr>
</thead>
<tbody>
<tr>
<td>MicroBlaze</td>
<td>125MHz; 5-stage (Virtex-5)</td>
<td>238</td>
</tr>
<tr>
<td>ARM Cortex-A9</td>
<td>1GHz; both cores combined</td>
<td>5927</td>
</tr>
<tr>
<td>ARM Cortex-A9</td>
<td>800MHz; both cores combined</td>
<td>4737</td>
</tr>
</tbody>
</table>

??from book
[Kuon and Rose, 2006]

- Altera Stratix II FPGA, 90nm
  - Quartus II “balanced”, “standard fit”
  - hard multipliers and memory blocks
- ST Micro 90nm standard cells
  - Synopsys “high-effort”, add scan chain
  - ST Micro memory compiler
  - Cadence place and route
- Basic Results
  - avg 21x/40x in area (w/wo using hard macros)
  - 3~4x critical path
  - ~12x dynamic power
Benchmarking

- Opencores and local designs
  - removed cases where FPGA and ASIC are more than 5% different in FF count (Bias?)

- Metrics evaluated
  - logic density
  - circuit speed
  - power consumption

[Table 1: Kuon and Rose, “Measuring the Gap between FPGAs and ASICs,” 2006]
Area Ratios

Differences attributed to “overhead” surrounding LUTs and FFs

[Table 2: Kuon and Rose, “Measuring the Gap between FPGAs and ASICs,” 2006]
Critical Path Ratios

Table 3&4: Kuon and Rose, “Measuring the Gap between FPGAs and ASICs,” 2006
Dynamic Power Ratios

[Table 5: Kuon and Rose, “Measuring the Gap between FPGAs and ASICs,” 2006]
Actual Mileage Varies

• Comparisons strongly affected by
  – exact design, FPGA/ASIC target, methodology
  – comparing less than “best-effort” designs can bias in either direction
  – design is not a point---a full comparison would have to be Pareto-front to Pareto-front

• Either
  – precise in a specific context, or
  – warm-fuzzy rule of thumb (~10x all around)
    \[ 2x \ll \text{“}10x\text{”} \ll 100x \]

A moving target with arch and process changes
Effects of Tuning

- FPGA design not a scaled version of ASIC design
  - different relative cost in logic vs. wires vs. mem
  - different relative speed in logic vs. wires vs. mem
  - also unique usage and operating characteristics

Designed-for-FPGA RTLs need different tuning
FPGA Wire Peculiarities

• Routing architecture over-provisioned to handle worst case
• In a “typical” design, wires appear cheaper relative to other resource types
  
  Best case is nearest-neighbor, regular grid

• Counterintuitively, you SHOULD use wider busses
  – consume unused “free” wires
  – compensate for lower frequency
FPGA Memory Peculiarities

• Large memory abnormally fast
• Large memory are “free” until your run-out
• Quantized memory options
  – jumps between FF-based vs. LUT-RAM vs. BRAMs
  – choose from fixed menu of sizes and aspect ratios
• Must manage RAM usage
  – don’t waste BRAM on small buffers
  – tune buffer sizes to natural granularities, e.g., zero incremental cost to go from 2Kb to 4Kb
  – pack buffers to share same physical array
FPGA Logic Peculiarities

- Logic slower than expected
- Sharp aberrations around hard macro use
  
  e.g., faster mult than add in Virtex-II

- CLB-mapped logic not divisible for pipelining
  - over-pipeline adds cycles without freq. increase
  - “sweetspot” frequency that is easy to reach but hard to exceed

- Design for performance
  - correct and maximal usage of hard macros
  - shallowly pipelined, wide datapath
E.g., FPGA- vs ASIC-tuned NoC on FPGA

- ASIC RTL from nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/Router
- FPGA RTL from www.ece.cmu.edu/calcm/connect/

**FPGA Resource Usage**
(same router/NoC configuration)

**Network Performance**
(uniform random traffic @ 100MHz)

- Avg. Packet Latency (in ns)
- Load (in Gbps)

- At same cost
- Same config

[Papamichael, ISFPGA 2012]
Soft-IPs need not be general purpose

- Reconfigurable fabric provides generality
- Soft-IPs should be maximally specialized to usage

**Ring** | **Fat Tree** | **Mesh** | **High Radix**
---|---|---|---
**Uniform Random Traffic**

<table>
<thead>
<tr>
<th>Load (in flits/cycle)</th>
<th>Latency (in cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>0.50</td>
<td>1</td>
</tr>
<tr>
<td>0.75</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

**90% Neighbor Traffic**

<table>
<thead>
<tr>
<th>Load (in flits/cycle)</th>
<th>Latency (in cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>0.50</td>
<td>1</td>
</tr>
<tr>
<td>0.75</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
It is not just FPGA vs ASIC

- **CPU**: highest-level abstraction / most general-purpose support
- **Multicore**: still high-level abstraction / general parallelism
- **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></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>M-M-Mult</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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)/mm² normalized to 40nm</th>
<th>GFLOP/J normalized to 40nm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Intel Core i7 (45nm)</td>
<td>96</td>
<td>0.50</td>
<td>1.14</td>
</tr>
<tr>
<td></td>
<td>Nvidia GTX285 (55nm)</td>
<td>425</td>
<td>2.40</td>
<td>6.78</td>
</tr>
<tr>
<td></td>
<td>ATI R5870 (40nm)</td>
<td>1491</td>
<td>5.95</td>
<td>9.87</td>
</tr>
<tr>
<td></td>
<td>Xilinx V6-LX760 (40nm)</td>
<td>204</td>
<td>0.53</td>
<td>3.62</td>
</tr>
<tr>
<td></td>
<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>FFT-2&lt;sup&gt;10&lt;/sup&gt;</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>GFLOP/s</th>
<th>(GFLOP/s)/mm²</th>
<th>GFLOP/J</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black-Scholes</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>
Tradeoff in Heterogeneity?

What to put on it?

Big Core

GPGPU

FPGA

Custom Logic

little core

little core

little core

little core

little core

little core
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:

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

- For small $f$, die area under-utilized

Base Core Equivalent (BCE) in [Hill and Marty, 2008]
Asymmetric Multicores

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

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

- \( r = \text{cost of fast core in BCE} \)
- \( \text{perf}_{\text{seq}} = \text{speedup of fast core over BCE} \)
- solve for optimal die area allocation given \( f \)
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, then usable U-core area $n - r \leq \frac{P}{\phi}$
  - if $B$ is total memory bandwidth expressed also as a multiple of BCEs, then usable U-core area $n - r \leq \frac{B}{\mu}$

$$Speedup = \frac{1}{1 - f} + \frac{f}{\frac{1}{\text{perf}_{seq}} + \mu \times (n - r)}$$
### $\phi$ 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>$\phi$</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>Nvidia GTX480</strong></td>
<td>$\phi$</td>
<td>0.77</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>1.83</td>
<td></td>
</tr>
<tr>
<td><strong>ATI R5870</strong></td>
<td>$\phi$</td>
<td>1.27</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>8.47</td>
<td></td>
</tr>
<tr>
<td><strong>Xilinx LX760</strong></td>
<td>$\phi$</td>
<td>0.31</td>
<td>5.68</td>
</tr>
<tr>
<td></td>
<td>$\mu$</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td><strong>Custom Logic</strong></td>
<td>$\phi$</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>

On equal area basis, 3.41x performance at 0.74x power relative a BCE.

Nominal BCE based on an Intel Atom in-order processor, 26mm$^2$ in a 45nm process.
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 (mm²)</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 (1x)</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
- 432mm² populated by an optimally sized Fast Core and U-cores of choice
Single-Prec. MMMult (f=99%)
Single-Prec. MMMult ($f=90\%$)

![Graph showing speedup for different technologies and technologies]

- SymMC
- AsymMC
- ASIC
- GPU R5870
- FPGA LX760

Technologies:
- Power Bound
- Mem Bound
Single-Prec. MMMult (f=50%)
Single-Prec. FFT-1024 ($f=99\%$)

[Graph showing speedup against technology node for different architectures: SymMC, AsymMC, ASIC, FPGA, LX760, GPU R5870. Legend includes Power Bound and Mem Bound.]
FFT-1024 (f=99%) if hypothetical 1TB/sec bandwidth
Parting Thoughts

• FPGAs pay an overhead for reconfigurability
  – significant but reducing
  – power and BW bottleneck can compress differences

• FPGAs differ from ASICs in more than then reconfiguration overhead---require distinct architecture and tuning

• {Multicore/GPU/FPGA} are all midpoints between CPU and ASIC extremes
  – none is a panacea
  – go with the easier option unless not good enough