Measures of Goodness

James C. Hoe
Department of ECE
Carnegie Mellon University
Housekeeping

• Your goal today: have respect for the many subtleties and dimensionalities of good hardware
  Digested from three 18-447 lectures

• Notices
  – Handout #3: lab 1, due noon, 9/23
  – Ultra96 ready for pick up (see Handout #3a)
  – Recitation starts this week, Wed 4:30~5:20

• Readings
  – 18-447 Spring 2019 Lectures 5, 12 and 23
Ultra96V2 Kit on Loan

- Each Kit contains: (2 kits per team)
  - 1x Ultra96-V2 development board
  - 1x 16GB microSD card with SD adapter
  - 1x Voucher for SDSoC license from Xilinx
  - 1x Quick Start Instruction card 2.1
  - 1x External 96Boards compliant power supply kit
  - 1x USB-to-JTAG/UART pod for Ultra96-V2

- You will treat it like your grade depended on it

- You will return all of it in perfect condition, or else
Looking Ahead

• Lab 1 (wk3/4): get cozy with Vivado
  – most important: learn logic analyzer and eclipse debugger
• Lab 2 (wk5/6): meet Vivado HLS
  – most important: decide if you would use it
• Lab 3 (wk7/8): hands-on with ARM and AFU
  – most important: have confidence it can work
• Project . . .
Performance is about time

- To the first order, performance $\propto \frac{1}{\text{time}}$

- Two very different kinds of performance!!
  - latency = time between start and finish of a task
  - throughput = number of tasks finished in a given unit of time (a rate measure)

- Either way, shorter the time, higher the performance, but . . .
Throughput ≠ 1/Latency

- If it takes $T$ sec to do $N$ tasks, throughput=$N/T$; latency$_1$=$T/N$?
- If it takes $t$ sec to do 1 task, latency$_1$=$t$; throughput=$1/t$?
- When there is concurrency, throughput≠1/latency

• Optimizations can tradeoff one for the other
  (think bus vs F1 race car)
Little’s Law

• $L = \lambda \cdot W$
  - $L$: number of customers
  - $\lambda$: arrival rate
  - $W$: wait time

• Fix any two, the third is decided

• E.g.,
  - AXI DRAM read: latency and # outstanding requests determine achieved BW (until peak)
  - in-order instruction pipeline: ILP and RAW hazard distance determine instruction throughput

In 643 terms:

# overlapped tasks
throughput
latency
Overhead and Amortization

• Throughput becomes a function of $N$ when there is a non-recurring start-up cost (aka overhead)

• E.g., DMA transfer on a bus
  – bus throughput $\text{raw} = 1$ Byte / (10^-9 sec) steadystate
  – 10^-6 sec to setup a DMA
  – throughput $\text{effective}$ to send 1B, 1KB, 1MB, 1GB?

• For start-up-time=$t_s$ and throughput $\text{raw} = 1/t_1$
  – throughput $\text{effective} = N / (t_s + N \cdot t_1)$
  – if $t_s \gg N \cdot t_1$, throughput $\text{effective} \approx N/t_s$
  – if $t_s \ll N \cdot t_1$, throughput $\text{effective} \approx 1/t_1$

  we say $t_s$ is “amortized” in the latter case
Latency Hiding

- What are you doing during the latency period?
- Latency = hands-on time + hands-off time
- In the DMA example
  - CPU is busy for the $t_s$ to setup the DMA
  - CPU has to wait $N \cdot t_1$ for DMA to complete
  - CPU could be doing something else during $N \cdot t_1$ to "hide" that latency
“Performance” is more than time
Under fixed power ceiling, more ops/second only achievable if less Joules/op?
Power = Energy / time

- Energy (Joule) dissipated as heat when “charge” move from VDD to GND
  - takes a certain amount of energy per operation, e.g., addition, reg read/write, (dis)charge a node
  - to the first order, energy $\propto$ work

*You care if on battery or pay the electric bill*

- Power (Watt=Joule/s) is rate of energy dissipation
  - more op/sec then more Joules/sec
  - to the first order, power $\propto$ performance

*Usually the problem is “thermal design power”*
Power and Performance not Separable

- Easy to minimize power if don’t care about performance
- Expect superlinear increase in power to increase performance
  - slower design is simpler
  - lower frequency needs lower voltage
- Corollary: Lower perf also use lower J/op (=slope from origin)

All in all, slower is more efficient in J/op and perf/Watt
Scale Makes a Difference

• Perf/Watt and J/op are normalized measures
  – hides the scale of problem and platform
  – recall, Watt $\propto \text{perf}^k$ for some $k>1$
• 10 GFLOPS/Watt at 1W is a very different design
  problem than at 1KW or 1MW or 1GW
  – say 10 GFLOPS/Watt on a <GPGPU,problem>
  – now take 1000 GPU GPUs to the same problem
  – realized perf is $< 1000x$ (less than perfect parallelism)
  – required power $> 1000x$ (energy to move data & heat)

In general be careful with normalized metrics
Design Tradeoff
Multi-Dimensional Optimizations

• HW design has many optimization dimensions
  – throughput and latency
  – area, resource utilization
  – power and energy
  – complexity, risk, social factors . . .

• Cannot optimize individual metrics without considering tradeoff between them, e.g.,
  – reasonable to spend more power for performance
  – converse also true (lower perf. for less power)
  – but never more power for lower performance
Pareto Optimality (2D example)

All points on front are optimal (can’t do better)  
How to select between them?
Application-Defined Composite Metrics

- Define scalar function to reflect desiderata---incorporate dimensions and their relationships

- E.g., energy-delay-(cost) product
  - smaller the better
  - can’t cheat by minimizing one ignoring others
  - what does it mean? why not energy\(^3\times\text{delay}^2\)?

- Floors and ceilings
  - real-life designs more often about good enough than optimal
  - e.g., meet a perf. floor under a power(cost)-ceiling (minimize design time, i.e., stop when you get there)
Which is Design Point is Best?
(runtime, energy, power, EDP)

Is B really lowest power?
Parallelism and Hardware Efficiency
Parallelism Defined

• $T_1$ (work measured in time):
  – time to do work with 1 PE
• $T_\infty$ (critical path):
  – time to do work with infinite PEs
  – $T_\infty$ bounded by dataflow dependence
• Average parallelism:
  \[ P_{\text{avg}} = \frac{T_1}{T_\infty} \]
• For a system with $p$ PEs
  \[ T_p \geq \max\{ \frac{T_1}{p}, T_\infty \} \]
• When $P_{\text{avg}} \gg p$
  \[ T_p \approx \frac{T_1}{p}, \text{aka “linear speedup”} \]

```
x = a + b;
y = b * 2
z = (x - y) * (x + y)
```
Linear Parallel Speedup

- Ideally, parallel speedup is linear with $p$

\[
speedup = \frac{\text{runtime}_{\text{sequential}}}{\text{runtime}_{\text{parallel}}}
\]

\[\propto \frac{1}{p}\]
This happens when $P_{\text{avg}} < p$; how else?
Amdahl’s Law

- If only a fraction $f$ is parallelizable by a factor of $p$

$$\text{time}_{\text{parallel}} = \text{time}_{\text{sequential}} \cdot \left( (1-f) + \frac{f}{p} \right)$$

$$\text{speedup} = \frac{1}{(1-f) + \frac{f}{p}}$$

- if $f$ is small, $p$ doesn’t matter
- even when $f$ is large, diminishing return on $p$; eventually “1-f” dominates
Data Movement not Free

• An algorithm has a cost in terms of operation count
  – runtime_{compute-bound} = # operations / FLOPS

• An algorithm also has a cost in terms of number of bytes communicated (ld/st or send/receive)
  – runtime_{BW-bound} = # bytes / BW

• Which one dominates depends on
  – ratio of FLOPS and BW of platform
  – ratio of ops and bytes of algorithm

• Average Arithmetic Intensity (AI)
  – how many ops performed per byte accessed
  – # operations / # bytes
Roofline Performance Model

[Williams & Patterson, 2006]

Attained Performance of a system (op/sec)

- \( \text{perf}_{\text{BW-bound}} = \text{AI} \cdot \text{BW} \)
- \( \text{perf}_{\text{compute-bound}} = \text{FLOPS} \)
- Runtime > max ( # op/FLOPS, # byte/BW) > #op \cdot \max(1/\text{FLOPS}, 1/(\text{AI} \cdot \text{BW}))
- \( \text{perf} = \min(\text{FLOPS}, \text{AI} \cdot \text{BW}) \)
Non-Ideal Speedup

$S$ vs $p$

How could this be?
Non-Ideal Speed Up

How could this be?
Parallelization Overhead

• Best parallel and seq. algo. need not be the same
  – best parallel algo. often worse at p=1
  – if \( \text{runtime}_{\text{parallel}}@p=1 = K \cdot \text{runtime}_{\text{sequential}} \) then best-case speedup = \( p/K \)

• Communication between PEs not instantaneous
  – extra time for the act of sending or receiving data as if adding more work \( (T_1) \)
  – extra time waiting for data to travel between PEs as if adding critical path \( (T_\infty) \)

If overhead grows with \( P \), speedup can even fall
Parallelization not just about speedup

- For a given functionality, non-linear tradeoff between power and performance
  - slower design is simpler
  - lower frequency needs lower voltage
  \[ \Rightarrow \text{For the same throughput, replacing 1 module by 2 half-as-fast reduces total power and energy} \]

Better to replace 1 of this by 2 of these; or N of these

Good hardware designs derive performance from parallelism
Parting Thoughts

• Need to understand performance to get performance!

• Good HW/FPGA designs involve many dimensions (each one nuanced)
  – optimizations involve making tradeoff
  – over simplifying is dangerous and misleading
  – must understand application needs

• Real-life designs have non-technical requirements

power and energy is first-class