ECE 571 – Advanced Microprocessor-Based Design Lecture 22

Vince Weaver

`http://web.eece.maine.edu/~vweaver`

`vincent.weaver@maine.edu`

25 April 2017

Announcements

- Presentations next week
- Second Exam next Class

Exam Review

Can bring one 8.5x11" page of notes

- Prefetch
  - Difference between hw/sw prefetch
  - Power/performance impact of prefetcher
- Virtual Memory
  - What is it good for?
  - Physical vs Virtual Memory
  - TLB purpose
- DRAM

- SRAM vs DRAM
- NVRAM

- System Energy
  - Which parts of the system contribute power usage
  - How can the O/S reduce power being used
- Mobile Energy
  - Are mobile power usage concerns different than System Power concerns?

Reading 1

ARM Reveals Cortex-A72 Architecture Details
by Andrei Frumusanu

http://www.anandtech.com/show/9184/arm-reveals-cortex-a72-architecture-details

Cortex-A72 Announced Feb 2015, this article from April 2015.
Is there a newer Cortex-A chip?

Highest Performance

- CoreLinux, Mali, MMU-400, NIC-400, ELA-500 (embedded logic analyzer)
- A72 direct successor to A57. Names purely marketing.
- How much performance from design, how much from shrink to 14nm/16nm process? FinFET
- Process size is getting more and more hand-wavy
- 3.5x performance of Cortex A15 in smartphone envelope
- 75% less energy for same workload
- A57 had problems running at peak frequency without

having to scale down.

- 16nm FF POP? (finfet package-over-package?)
- Note they are comparing against Cortex-A15 not A57 directly. A15@28nm, A57@20nm 1.9x better, A57@16nm 2.6x better, A72@16nm=3.5x better. Normalize, 1.3 times better than A57?
- Sustained 750mW operating at 2.5GHz

Next-Generation Performance

- Performance, interesting mix of benchmarks. SPEC2006, Stream, LMBench, bunch of others.
- 16-30% improvement in IPC
- Performance, Power, Area
- Better branch predictor "sophisticated new, reduce energy from mispredict and speculation" Bypasses completely if doing a bad job
- Fancier decode, ARM64 instruction fusion. Lots of power optimization

- 3-way L1-icache at direct mapped power?
- 5-wide dispatch
- Advanced FP/SIMD unit: 3-cycle FMUL, 3-cycle FADD, 6-cycle FMAC, 2-cycle CVT ???, radix-16 FP divider
- radix-16 division?
- improved CRC unit
- L1/L2 bandwidth improved by 30%. Sophisticated prefetcher? Lots of power optimization
- L1 way predictor (reduce hit power)
- Power-optimized RAM organization
- Extensive power in L2-idle?

Reading 2

A walk through of the Microarchitectural improvements in Cortex-A72

`https://community.arm.com/groups/processors/blog/2015/05/04/a-walk-through-of-the-microarchitect`

Blog Posting

Single core FP performance (For me, linpack/cores)

Type	Cores	ARM Reported	My own measurement (LINPACK)
Cortex-A8	1	1.0	1.0 (BBB: 0.068)
Cortex-A9	2	?	7.0 (Panda: 0.95/2)
Cortex-A9	4	?	??
Cortex-A15	8	11x	20x (Chrome: 5.44/4)
Cortex-A53			16x (Dragon: 2.2/2)
Cortex-A57	?	12x	58x (Jetson: 16.0/4)
Cortex-A57		15x	
Cortex-A72			

- Tradeoff – larger branch predictor can cost more power but reduce speculation so save energy

- Cache– found way for 3-way cache to have similar power to direct mapped
- TLB – can turn off high bits when assuming locality?
- Branch predictor – optimize for close branches?
- Micro-op handling?
- Lots of power optimization
- Better prefetcher
- L1 dcache-hit predictor
- Power optimization in L2-idle case
- Big-little with Cortex-a53

Jetson TX-1

NVIDIA Announces Jetson TX1 - A Tegra X1 Module & Development Kit
by Ryan Smith

Jetson TX-1

- We used this board for the homeworks, released in November 2015.
- There's now a Jetson-TX2
- Nintendo Switch has a Tegra X1 in it
- Tegra X1 SoC
- 64-bit ARM Cortex A57 CPUs (4 cores)

- “1-TFLOP” 256-core Maxwell class GPU
  When run double-precision hpl_CUDA get around 7 GFLOPS which is even less than the CPU gets, although the fan doesn’t come on at all.
- 4GB LPDDR4 memory (25.6 GB/s)
- Full featured ITX-size I/O carrier board: gigabit Ethernet, wifi, USB, GPIOs, camera, PCIe slot
- 10W of power
- PCIe, Ethernet

Nvidia

- Push for machine learning?
- Drive PX? – auto-driving cars

Maxwell GPU

The NVIDIA GeForce GTX 980 Review: Maxwell Mark 2
by Ryan Smith

Maxwell GPU Page 1

- Article said it was maxwell-like, so an article on Maxwell
- Kepler, Maxwell, Pascal
- At this point (2014?) stuck at 28nm process

Maxwell GPU Page 2

- Mobile-first strategy, and scaling up
- GFLOPS/W from 15-30? (Best in my lab currently around 2 for a high-end haswell-ep server), best ARM is jetson at 1.2
- Managed to double flops/w over kepler while staying in same process
- Grid layout of the GPU

- Only texture units and FP64 CUDA cores are shared, which is greatest power saving
  Shared resources good, but only if using them, and crossbar sucks lots of power
- Other power features: better scheduler, fewer units
- Increase cache size. Larger static power, but can reduce data that has to go over memory bus
- Transistor optimization (not many details)

Maxwell-2 GPU Page 3

- ROP = render output unit
- Color compression, drastically reduce bandwidth needed

Maxwell-2 GPU Page 4

- Direct3d features
- HDMI 2.0 needed for 4k displays at 60Hz.
  3840 pixels * 2160 (8.3 Megapixels)
  HDMI 2.0 allows up to 18 Gbit/s
- VR-direct
- Latency reduction, from 50ms to 40ms
- antialiasing

More GPU Stuff

- What is meant by a FLOP? 64/32/16-bit

Reading

Where is the energy spent inside my app? Fine Grained Energy Accounting on Smartphones with Eprof
by Pathak, Hu, and Zhang

Intro

- Apps limited by battery life
- energy consumption is important
  1. Track at level of “program entities” (i.e. thread, routines, etc)
  2. Track various components
  3. Need to map power used to the entities
- cell phones do not have built-in power meters.
- asynchronous power behavior: power draw might not be related to running app:

1. Tail power (GPS, wifi, sdcard) (ref 4,5)
2. persistent power wakelocks
   lock keeps phone from sleeping. can extend beyond a routine
3. “exotic” components: camera and GPS start when switched on by one and continue until switched off (even if app switched in doesn’t use it)

- eprof fine-grained energy profiler ”first” for modern
- cellphones on Android and Windows Mobile
- Leverages fine-grained power modeling from [4]
- system call driven finite-state-machine?

- last-trigger policy?
- look at 6 of the most 10 popular apps in android market
- Surprising results:
  - ad modules consume 65-75%
  - clean termination of TCP sockets 10-50% of energy
  - tracking user data 20-30%
  - Actual application only 20-30%
  - Majority energy in 3G, Wifi, GPS. I/O energy important CPU optimization might be waste of time.
  - asynchronous power behavior important "I/O energybundles" component stays in high power state

Much of I/o energy triggered by small amount of routines

- Accounting granularity

Accounting Granularity

- Granularity: a process, a thread, a subroutine, a system call
- Threads are important as often apps often have third-party threads running
- System calls often trigger different power states, so track them
- They track call stacks when system calls made. Periodically poll things. And log threadID at context switch

Asynchronous Power Behavior

- Lots of components (CPU, mem, sdcard, wifi, nic, 3G, bluetooth, GPS, camera, accelerometer, digital compass, lcd, touch sensor, microphone, speakers) and many use as much power as CPU

- Tail power: a routine can start up a high-power component but it does not shut off until long after the component is done

- Wakelocks: some things must keep the phone from sleeping. i.e sync, etc. source of bugs

- Exotic components: app can turn on something like CPU and it stays on after context switch, even if other app not using it
- Measuring power current drawn through battery.

Accounting Policies on Cellphones

- How to attribute energy use to what caused it
- example, send some data. Ramps up the 3G radio for 2.5s before actually start. 61uAH. But still draws power for 6s after completed
- Why uAH? Batteries often rated in AH (Charge). Not the same as Joules (energy) AH not take into account voltage. Voltage can change while battery discharging.
- If multiple processes use the radio, how split up the cost? Weights?

- Last-trigger policy: attribute tail to last routine that would trigger.
- If multiple system calls using component, try to split up evenly among them
- High-rate components: not cover RAM or OLED.
- RAM modeled with LLC performance counter. Interesting they try to use perf_event
- OLED power can be calculated based on the *colors* on the screen. Can screen-scrape and estimate

Eprof

- Android and Windows Mobile (only describe Android in this paper)
- SDK Routine tracing: Android Dalvik VM provides runtime tracing
- NDK Routine Tracing: native development kit
- System Call Tracing: use ADB (Android Debugger) logging. Also modify bionic C library

- Modify framework to do syscall tracing without having to modify programs (no source)
- Accounting and Data Presentation

Evaluation/Related Work

- Split-time accounting. Samples, accounts power to whatever running at sample time?
- Accuracy, split-time is worst as it attributes asynch power to pid0
- Overhead, both power and performance

Applications/Benchmarks

- 3G, no room for wifi results
- Android Browser
  Google triggers GPS. 53/31/16 CPU/3g/GPS
  34 threads
- Angry Birds
  time in ads, etc
- Free Chess
  ads

- NYTimes
  Obfuscated code, ads
- Finding energy bugs
  Found wakelock bug which was reported and fixed

Optimizing I/O Energy using Bundles

- I/O consumes most of energy
- I/O energy in a few bundles
- Very few routines perform I/O