

# ECE 471 – Embedded Systems

## Lecture 22

Vince Weaver

<http://www.eece.maine.edu/~vweaver>

[vincent.weaver@maine.edu](mailto:vincent.weaver@maine.edu)

25 November 2014

# Announcements

- Project groups status report due:
  1. One-sentence summary of project
  2. What hardware you plan to use for input/output and board?
  3. Have you acquired/tested the hardware? Does it work?
  4. Are you willing to present on Tuesday rather than Thursday?



# Homework 8 Review

- C-coding
  - why mili-celsius?
  - check for error (no segfault)
  - scanf()
  - strcmp / strncmp
    - can't do compare with ==
    - string1 == string2
    - excess ampersands
    - No need for & in front of strings. Complicated



- assuming `char[36]=Y` or `N`
  - `strtok()`
  - skip `t=` part. Pointer, so `string+2`
  - `strstr()`, `strtod`, `atoi`, `atof`
  - remember to `fclose`
- Why `Vdd`. provide enough current needs extra transistor.  
ground if want to bus power. 85C if try.
  - shell script



# Low-Level Linux/Project digressions



# Wii Nunchuck

- Fairly easy to interface
- Put onto i2c bus. Device 0x52
- Send handshake to initialize. Use longer one (0xf0/0x55/0xfb/0x00) not the simpler one you might find(0x40/0x00). This works on generic nunchucks and possibly also disables encryption of results.
- To get values, send 0x00, usleep a certain amount, and read 6 bytes. This includes joy-x, joy-y, accelerometer



x/y/z and c and z button data. More info can be found online.

byte0 = joy-x, byte1 = joy-y, byte2 = top8 acc x, byte3 = top8 acc y, byte4 = top8 acc z, byte 5 is bottom 2 z,y,x then button c and z (inverted?)



# Linux and Keyboard

- Old ps/2 keyboard just a matrix of keys, controlled by a small processor. Communication via a serial bus. Returns "keycodes" when keypress and release and a few others.
- Many modern keyboards are USB, which requires full USB stack. To get around needing this overhead (for BIOS etc) support bit-bang mode. OS usually has abstraction layer that supports USB keyboards same as old-style





- Linux assumes “CANONICAL” input mode, i.e. like a teletype. One line at a time, blocking input, wait until enter pressed.
- You can set non-CANONICAL mode, async input, and VMIN of 1 to get reasonable input for a game. Arrow keys are reported as escape sequences ( ESCAPE-[A for up, for example).
- Even lower-level you can access “RAW” mode which gives raw keycode events, etc.
- There are libraries like ncurses that abstract this a bit.



Also GUI and game libraries (SDL).



# DVFS



# CPU Scaling

- Intel SpeedStep
- Enhanced speed step. Change  $V$  and  $F$  at different points. Slower to change frequency if  $V$  not changed first. Bus clock keeps running even as PLL shut down 10ms transition
- AMD PowerNow! (laptop) Cool'n'Quiet (desktop)
- VIA PowerSaver/LongHaul – Fine grained DVFS



- p4-clockmod – mainly for thermal management, skip clocks, hurt performance without saving energy (throttling)
- IBM EnergyScale
- Transmeta LongRun – leakage varies due to process variation Longrun2 monitors performance/leakage and varies Vdd and Vt



# Governors

- ondemand – dynamically increase frequency if at 95% of CPU load  
introduced in 2.6.9
- performance – run CPU at max frequency
- conservative – increase frequency if at 75% of load
- powersave – run CPU at minimum frequency
- userspace – let the user (or tool) decide



# Governors – cont

- Various tunables under `/sys/devices/system/cpu`
- Can trigger based on ACPI events (power plug in, lid close)
- Laptop tools
- `cpufreq-info` and `cpufreq-set`  
Need to be root



# User Governors

- typically can only update once per second
- ondemand people claim it reacts poorly to bursty behavior
- Powernowd – scale based on user and sys time
- cpufreqd
- Obsolete with introduction of “ondemand” governor?





# Sources of Info for Governors

- System load
- performance counters
- input from user?



# TurboBoost

- Nehalem/Ivy Bridge/Sandy Bridge (AMD has similar Turbo CORE)
- Some Core2 had similar “Intel Dynamic Acceleration”
- Kicks in at highest ACPI Pstate
- “Dynamic Overclocking”



# TurboBoost – from HotChips 2011 Slides

- Monitors power, current, thermal limits, overclocks
- 100 uarch events, leakage function of temp and voltage
- P1: guaranteed stable state  
P0: turbo boost, maximum possible
- 12 temp sensors on each core
- PECL – an external microcontroller, used to control fans, package power



# TurboBoost example

- From Wikipedia Intel\_Turbo\_Boost article
- Core i7-920XM
- Normal freq 2.0GHz
- 2/2/8/9 – number of 133MHz steps above with 4/3/2/1 cores active
- 2.26GHz, 3.06GHz, 3.20GHz



# Non-x86 Power Saving



# IBM EnergyScale

- Thermal reporting
- Static and Dynamic Power Save
- “Power Folding” – reduce the number of CPUs reported to the OS until they are all busy
- Power Capping (like RAPL)
- Fan Control – Avoid “over-cooling”



- Processor Nap – 2ms to wake up
- Processor Winkle (as in Rip Van) – 10-20ms to wake up, 95% of power



# ARM Cortex A9 (Pandaboard)

- Cortex-A9 Technical Reference Manual, Chapter 2.4 Power Management
- Energy Efficient Features
  - Accurate branch prediction (reduce number of incorrect fetch)
  - Physically addressed caches (reducing number of cache flushes)
  - Use of micro TLBs





- caches that use sequential access information? reduce accesses to tags
- small instruction loops can operate without access icache
- Potentially separate power domains for CPU logic, MPE (multi-media NEON), and RAMs
- Full-run mode
- Run with MPE disabled
- Run with MPE powered off



- Standby – entered with `wfi` instruction. Processor mostly shutdown except part waiting for interrupt
- Dormant – caches still powered
- Shutdown



# Pandaboard Power Stats

- Wattsuppro: 2.7W idle, seen up to 5W when busy
- <http://ssvb.github.com/2012/04/10/cpuburn-arm-cortex-a9.html>
- With Neon and CPU burn:

Idle system	550 mA	2.75W
cpuburn-neon	1130 mA	5.65W
cpuburn-1.4a (burnCortexA9.s)	1180 mA	5.90W
ssvb-cpuburn-a9.S	1640 mA	8.2W



# Notes on Process Technology

- 65nm – 2006  
p4 to core2, IBM Cell  
1.0v, High-K dielectric, gate thickness a few atoms  
193/248nm light (UV)
- 45nm – 2008  
core2 to nehalem  
large lenses, double patterning, high-k
- 32nm – 2010



## sandybridge to westmere immersion lithography

- 22nm – 2012 ivybridge, haswell  
oxide only 0.5nm (two silicon atoms)  
fin-fets
- 14nm and smaller – ??  
Extreme UV (13.5nm light, hard-vacuum required)?  
Electron beam?



# Notes on Process Technology

- TI-OMAP cell phone processor (more or less discontinued by TI, big layoffs in 2012)  
Beagle Board and Gumstix OMAP35?? – 65nm
- OMAP4460 (Pandaboard) 45nm
- Cortex A15 28nm
- Rasp-pi BCM2835 – 45nm?



# CPU Power and Energy

- Became a trendy thing to research in 1999-2002
- Before that usually concern was with performance.
- These days energy results are often reported as a core part of any architectural proposal, not as a separate issue.
- The results discussed here are academic and may or may not be implemented in actual chips.



# Thermal Concerns Too

Power density exceed hot plate, approaching rocket nozzle





# Methodologies Used in These Papers

It varies, but many of these are from simulations (sometimes validated). Anything from SPICE to “cycle-accurate” simulators.



# DVFS

- Voltage planes – on CMP might share voltage planes so have to scale multiple processors at a time
- DC to DC converter, programmable.
- Phase-Locked Loops. Orders of ms to change. Multiplier of some crystal frequency.
- Senger et al ISCAS 2006 lists some alternatives. Two phase locked loops? High frequency loop and have programmable divider?



- Often takes time, on order of milliseconds, to switch frequency. Switching voltage can be done with less hassle.



# When can we scale CPU down?

- System idle
- System memory or I/O bound
- Poor multi-threaded code (spinning in spin locks)
- Thermal emergency
- User preference (want fans to run less)

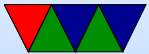


# Adaptive Body Biasing

- Related to but not always considered part of DVFS
- Control voltage applied to body
- Change the threshold voltage
- Reduces leakage but slows performance



# Energy Delay / Energy Delay Squared



# Measuring Power and Energy



# Why?

- New, massive, HPC machines use impressive amounts of power
- When you have 100k+ cores, saving a few Joules per core quickly adds up
- To improve power/energy draw, you need some way of measuring it





# Energy/Power Measurement is Already Possible

## Three common ways of doing this:

- Hand-instrumenting a system by tapping all power inputs to CPU, memory, disk, etc., and using a data logger
- Using a pass-through power meter that you plug your server into. Often these will log over USB
- Estimating power/energy with a software model based on system behavior



# Existing Related Work

Plasma/dposv results with Virginia Tech's PowerPack



# Powerpack

- Measure at Wall socket: WattsUp, ACPI-enabled power adapter, Data Acquisition System
- Measure all power pins to components (intercept ATX power connector?)
- CPU Power – CPU powered by four 12VDC pins.
- Disk power – measure 12 and 5VDC pins on disk power connector



- Memory Power – DIMMs powered by four 5VDC pins
- Motherboard Power – 3.3V pins. Claim NIC contribution is minimal, checked by varying workload
- System fans



# Shortcomings of current methods

- Each measurement platform has a different interface
- Typically data can only be recorded off-line, to a separate logging machine, and analysis is done after the fact
- Correlating energy/power with other performance metrics can be difficult
- PAPI (Performance API) is a cross-platform, open-source library for gathering performance-related data
- PAPI-C interface makes adding new power measuring



components straightforward

- PAPI can provide power/energy results in-line to running programs



# More PAPI benefits

- One interface for all power measurement devices
- Existing PAPI code and instrumentation can easily be extended to measure power
- Existing high-level tools (Tau, VAMPIR, HPCToolkit, etc.) can be used with no changes
- Easy to measure other performance metrics at same time



# RAPL

- **R**unning **A**verage **P**ower **L**imit
- Part of an infrastructure to allow setting custom per-package hardware enforced power limits
- User Accessible Energy/Power readings are a bonus feature of the interface





# How RAPL Works

- RAPL is *not* an analog power meter
- RAPL uses a software power model, running on a helper controller on the main chip package
- Energy is estimated using various hardware performance counters, temperature, leakage models and I/O models
- The model is used for CPU throttling and turbo-boost, but the values are also exposed to users via a model-specific register (MSR)



# Available RAPL Readings

- `PACKAGE_ENERGY`: total energy used by entire package
- `PP0_ENERGY`: energy used by “power plane 0” which includes all cores and caches
- `PP1_ENERGY`: on original Sandybridge this includes the on-chip Intel GPU
- `DRAM_ENERGY`: on Sandybridge EP this measures DRAM energy usage. It is unclear whether this is just the interface or if it includes all power used by all the DIMMs too



# RAPL Measurement Accuracy

- Intel Documentation indicates Energy readings are updated roughly every millisecond (1kHz)
- Rotem et al. show results match actual hardware

Rotem et al. (IEEE Micro, Mar/Apr 2012)



# RAPL Accuracy, Continued

- The hardware also reports minimum measurement quanta. This can vary among processor releases. On our Sandybridge EP machine all Energy measurements are in multiples of 15.2nJ
- Power and Energy can vary between identical packages on a system, even when running identical workloads. It is unclear whether this is due to process variation during manufacturing or else a calibration issue.



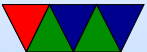
# NVML

- Recent NVIDIA GPUs support reading power via the NVIDIA Management Library (**NVML**)
- On Fermi C2075 GPUs it has milliwatt resolution within  $\pm 5W$  and is updated at roughly 60Hz
- The power reported is that for the entire board, including GPU and memory



# Beagle Board

- Has a 0.1 Ohm resistor you can measure voltage across to get current usage.
- Can read both sides of this using an on-chip ADC via i2c to calculate this with a complex set of equations.



# PandaBoard

- Google this, people show which SMD pins to probe.



# Raspberry Pi

- People using external power meters.

