ECE 571 – Advanced Microprocessor-Based Design Lecture 5

Vince Weaver

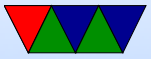
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

2 February 2016

Announcements

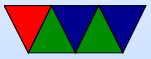
- HW#2 was posted
- HW#1 Mostly graded



HW#1 Review

- bzip2 benchmark – what does it do?
- 19 billion instructions +/- 400 or so
  (this is test input maybe?)
- 13 billion cycles +/- 6 million?
- Reversed: similar – HW2 will show you why I asked that
- Perf record: 3.5s,

```
66.48%  bzip2    bzip2            [.] mainSort
17.45%  bzip2    bzip2            [.] BZ2_compressBlock
 6.42%  bzip2    bzip2            [.] mainGtU.part.0
 6.12%  bzip2    bzip2            [.] handle_compress.isra.2
 0.70%  bzip2    bzip2            [.] add_pair_to_block
```

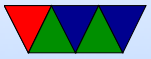


- Valgrind, 1m10.189s == roughly 20 times slower?

```
11,291,448,187  ???:mainSort [/opt/ece571/401.bzip2/bzip2]
 3,381,835,437  ???:BZ2_compressBlock [/opt/ece571/401.bzip2/bzip2]
 2,138,813,059  ???:handle_compress.isra.2 [/opt/ece571/401.bzip2/bz
 1,958,107,443  ???:mainGtU.part.0 [/opt/ece571/401.bzip2/bzip2]
   165,396,105  ???:BZ2_blockSort [/opt/ece571/401.bzip2/bzip2]
   140,068,091  ???:add_pair_to_block [/opt/ece571/401.bzip2/bzip2]
```

- Gprof, also 3.5s
  Different results, using function entry instead of exact instruction count for sampling?

```
time   seconds   seconds    calls   s/call   s/call  name
 71.77      2.16     2.16       53     0.04     0.04  mainSort
 18.94      2.73     0.57       53     0.01     0.05  BZ2_compressBl
  6.98      2.94     0.21    12223     0.00     0.00  default_bzallo
  1.00      2.97     0.03     1272     0.00     0.00  BZ2_hbMakeCode
  0.66      2.99     0.02  1856468     0.00     0.00  add_pair_to_bl
```

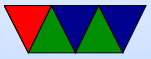


- Skid instructions – mov is more likely than sub?

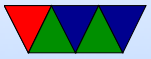
```
perf annotate:
  1.14  5f0:   mov    (%r10),%edx
  0.56         lea    (%rdx,%r13,1),%eax
  0.80         movzbl (%r15,%rax,1),%eax
  3.29         sub    %r9d,%eax
```

instructions:pp

```
perf annotate:
  0.78  5f0:   mov    (%r10),%edx
  0.88         lea    (%rdx,%r13,1),%eax
  3.14         movzbl (%r15,%rax,1),%eax
  0.99         sub    %r9d,%eax
  0.52         cmp    $0x0,%eax
  0.58        jne    689
  0.90         movslq %ebx,%rax
```



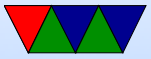
Power and Energy



Definitions and Units

People often say Power when they mean Energy

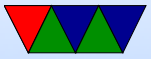
- Energy – Joules, kWH (3.6MJ), Therm (105.5MJ), 1 Ton TNT (4.2GJ), eV (1.6×10^{-19} J), BTU (1055 J), horsepower-hour (2.68 MJ), calorie (4.184 J)
- Power – Energy/Time – Watts (1 J/s), Horsepower (746W), Ton of Refrigeration (12,000 Btu/h)
- Volt-Amps (for A/C) – same units as Watts, but not same thing
- Charge – mAh (batteries) – need V to convert to Energy



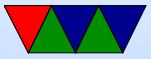
Power and Energy in a Computer System

Power Consumption Breakdown on a Modern Laptop, A. Mahersi and V. Vardhan, PACS'04.

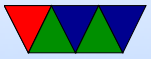
- Old, but hard to find thorough breakdowns like this
- Thinkpad Laptop, 1.3GHz Pentium M, 256M, 14" display
- Oscilloscope with voltage probe and clamp-on current probe.
- Measured V and Current. P=IIR. V=IR P=IV, subtractive for things without wires



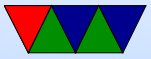
- Total System Power 14-30W
  - Hard Drive 0.5-2W (Flash/SSD less)
  - LCD 1W (slightly more black than white)
  - Backlight Inverter (this is before LED) 1-4W depending on brightness
  - CPU 2-15W (with scaling)
  - GPU 1-5W
  - Memory 0.45 - 1.5W
  - Power Supply Loss - 0.65W
  - Wireless 0.1 - 3W (wifi on cellphones)
  - CDROM 3-5W



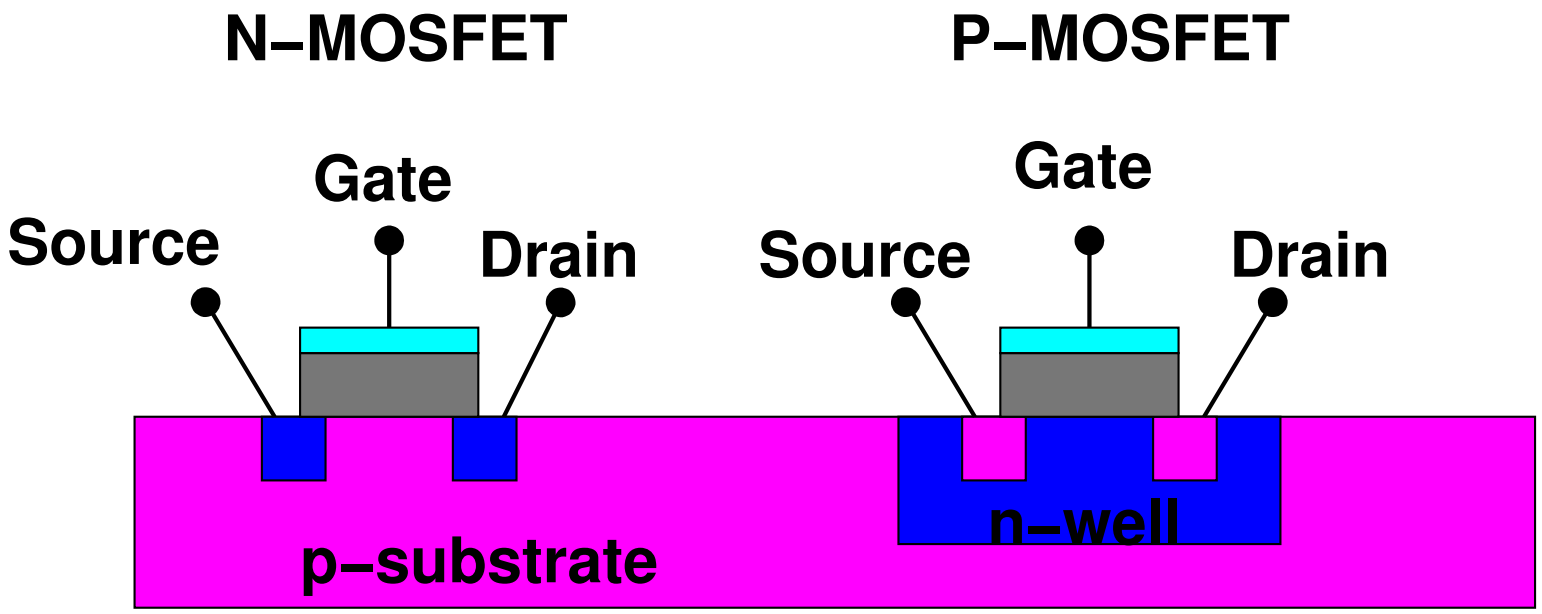
- Not in paper but USB 2.0 – 5V, can draw 5 units of 100mA each, 2.5W)

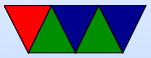


CPU Power and Energy



CMOS Transistors



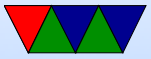


CMOS Dynamic Power

- $P = C\Delta V V_{dd} \alpha f$
  Charging and discharging capacitors big factor ($C\Delta V V_{dd}$) from V_{dd} to ground
  α is activity factor, transitions per clock cycle
  F is frequency

- α often approximated as $\frac{1}{2}$, $\Delta V V_{dd}$ as V_{dd}^2 leading to $P \approx \frac{1}{2} C V_{dd}^2 f$

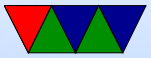
- Some pass-through loss (V momentarily shorted)



CMOS Dynamic Power Reduction

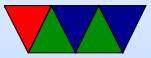
How can you reduce Dynamic Power?

- Reduce C – scaling
- Reduce V_{dd} – eventually hit transistor limit
- Reduce α (design level)
- Reduce f – makes processor slower



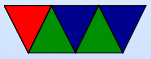
CMOS Static Power

- Leakage Current – bigger issue as scaling smaller. Forecast at one point to be 20-50% of all chip power before mitigations were taken.
- Various kinds of leakage (Substrate, Gate, etc)
- Linear with Voltage: $P_{static} = I_{leakage} V_{dd}$



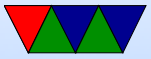
Leakage Mitigation

- SOI – Silicon on Insulator (AMD, IBM but not Intel)
- High-k dielectric – instead of SO2 use some other material for gate oxide (Hafnium)
- Transistor sizing – make only the critical transistors fast; non-critical ones can be made slower and less leakage prone
- Body-biasing
- Sleep transistors



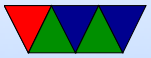
Total Energy

- $E_{tot} = [P_{dyanmic} + P_{static}]t$
- $E_{tot} = [(C_{tot}V_{dd}^2\alpha f) + (N_{tot}I_{leakage}V_{dd})]t$



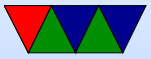
Delay

- $T_d = \frac{C_L V_{dd}}{\mu C_{ox}(\frac{W}{L})(V_{dd}-V_t)}$
- Simplifies to $f_{MAX} \sim \frac{(V_{dd}-V_t)^2}{V_{dd}}$
- If you lower f, you can lower V_{dd}



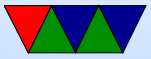
Thermal Issues

- Temperature and Heat Dissipation are closely related to Power
- If thermal issues, need heatsinks, fans, cooling



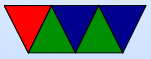
Metrics to Optimize

- Power
- Energy
- MIPS/W, FLOPS/W (don't handle quadratic V well)
- $Energy * Delay$
- $Energy * Delay^2$



Power Optimization

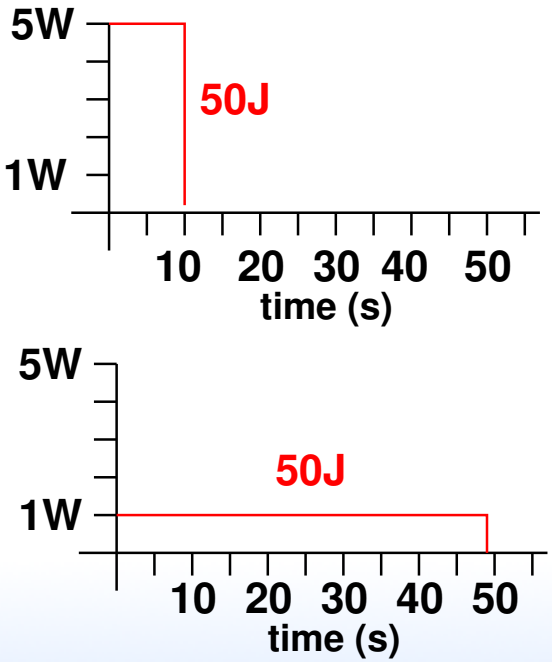
- Does not take into account time. Lowering power does no good if it increases runtime.

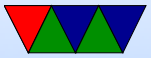


Energy Optimization

- Lowering energy can affect time too, as parts can run slower at lower voltages

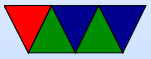
  Which is better?



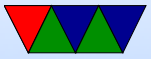


Energy Delay – Watt/t*t

- Horowitz, Indermaur, Gonzalez (Low Power Electronics, 1994)
- Need to account for delay, so that lowering Energy does not made delay (time) worse
- Voltage Scaling – in general scaling low makes transistors slower
- Transistor Sizing – reduces Capacitance, also makes transistors slower

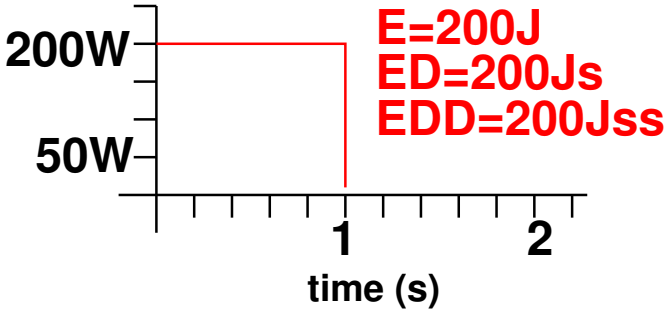


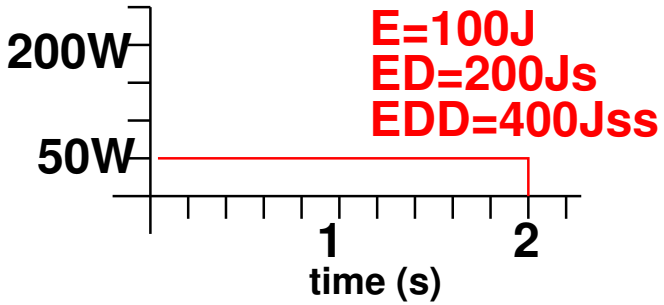
- Technology Scaling – reduces V and power.
- Transition Reduction – better logic design, have fewer transitions
  Get rid of clocks? Asynchronous? Clock-gating?

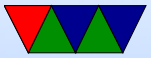


ED Optimization

Which is better?

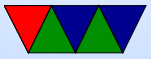






Energy Delay Squared– E*t*t

- Martin, Nyström, Pénzes – Power Aware Computing, 2002
- Independent of Voltage in CMOS
- Et can be misleading
  Ea=2Eb, ta=tB/2
  Reduce voltage by half, Ea=Ea/4, ta=2ta, Ea=Eb/2, ta=tb
- Can have arbitrary large number of delay terms in Energy product, squared seems to be good enough



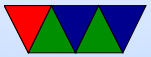
Energy Delay / Energy Delay Squared

Lower is better.

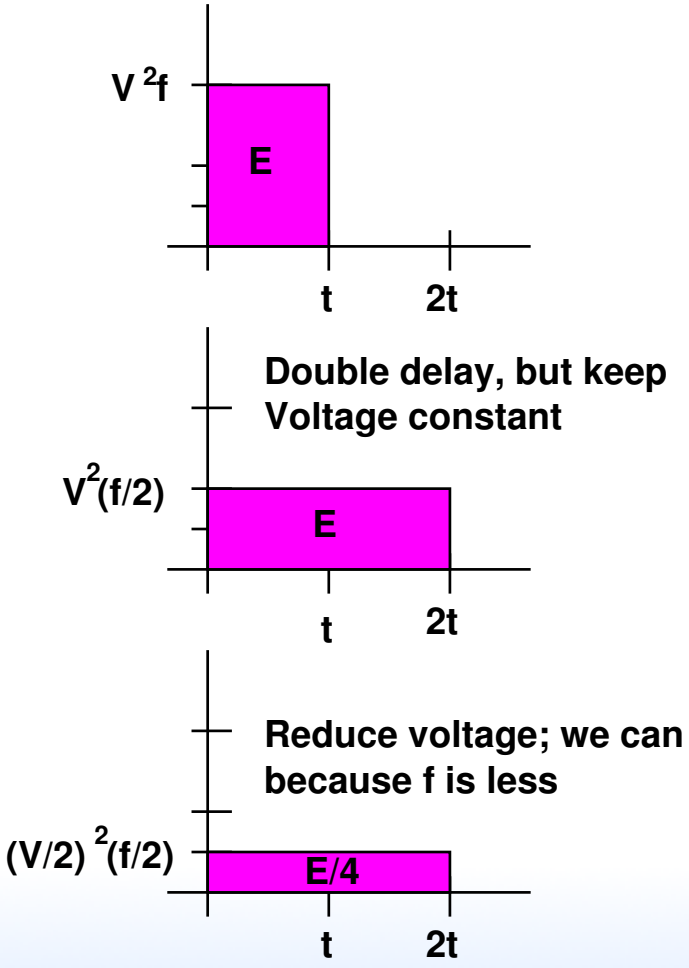
Energy	Delay	ED	ED^2
5J	2s	$10Js$	$20Js^2$
5J	3s	$15Js$	$45Js^2$

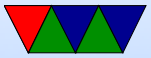
Same ED, Different ED^2

Energy	Delay	ED	ED^2
5J	2s	$10Js$	$20Js^2$
2J	5s	$10Js$	$50Js^2$

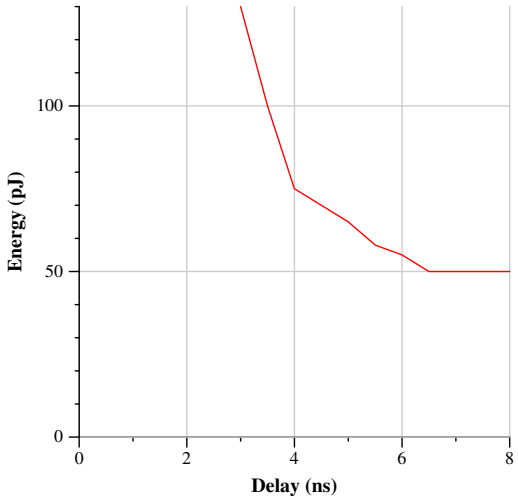


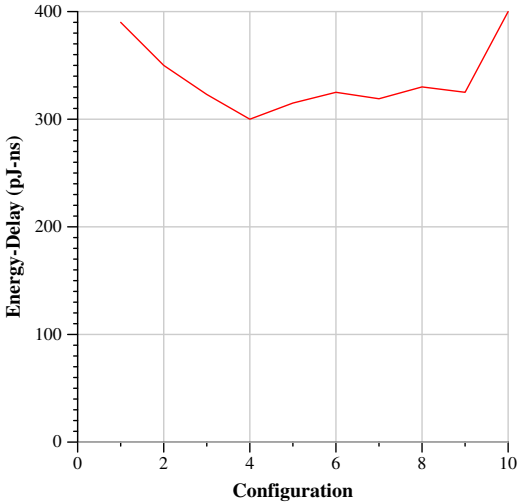
Energy Example

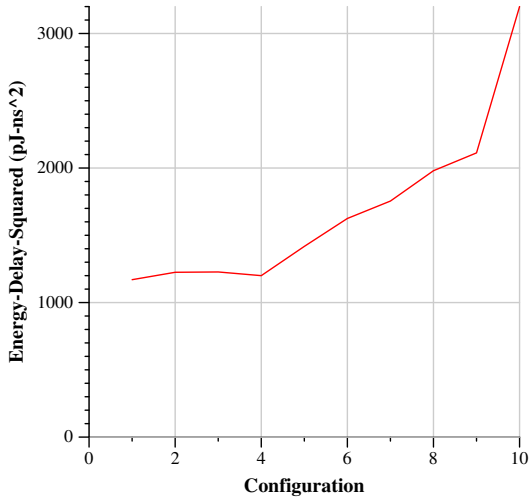




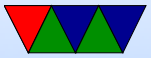
Energy-Delay Product Redux





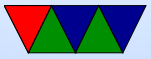


Roughly based on data from "Energy-Delay Tradeoffs in CMOS Multipliers" by Brown et al.



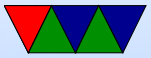
Raw Data

Delay	Energy	ED	ED^2
3	130	390	**1170**
3.5	100	350	1225
3.8	85	323	1227
4	75	**300**	1200
4.5	70	315	1418
5	65	325	1625
5.5	58	319	1755
6	55	330	1980
6.5	**50**	390	2535
8	50	400	3200

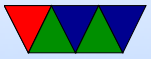


Other Metrics

- $Energy - Delay^n$ – choose appropriate factor
- $Energy - Delay - Area^2$ – takes into account cost (die area) [McPAT]
- Power-Delay – units of Energy – used to measure switching
- Energy Delay Diagram – [SWEEP]

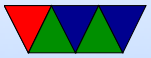


Measuring Power and Energy



Why?

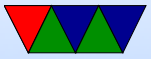
- New, massive, HPC machines use impressive amounts of power
- When you have 100k+ cores, saving a few Joules per core quickly adds up
- To improve power/energy draw, you need some way of measuring it



Energy/Power Measurement is Already Possible

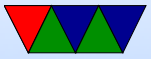
Three common ways of doing this:

- 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



Measuring Power and Energy

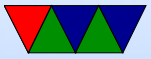
- Sense resistor or Hall Effect sensor gives you the current
- Sense resistor is small resistor. Measure voltage drop. Current V=IR Ohm's Law, so V/R=I
- Voltage drops are often small (why?) so you made need to amplify with instrumentation amplifier
- Then you need to measure with A/D converter
- $P = IV$ and you know the voltage
- How to get Energy from Power?



Power and Energy Concerns

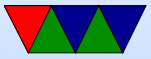
Table 1: ATLAS 300x300 DGEMM (Matrix Multiply)

Machine	Processor	Cores	Frequency	Idle	Load	Time	Total Energy
Raspberry Pi	ARM 1176	1	700MHz	3.0W	3.3W	23.5s	77.6J
Gumstix Overo	Cortex-A8	1	600Mhz	2.6W	2.9W	27.0s	78.3J
Beagleboard	Cortex-A8	1	800MHz	3.6W	4.5W	19.9s	89.5J
Pandaboard	Cortex-A9	2	900MHz	3.2W	4.2W	1.52s	6.38J
Chromebook	Cortex-A15	2	1.7GHz	5.4W	8.1W	1.39s	11.3J



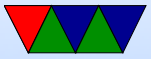
Questions

- Which machine consumes the least amount of energy? (Pandaboard)
- Which machine computes the result fastest? (Chromebook)
- Chromebook is a laptop so also includes display and wi-fi
- Consider a use case with an embedded board taking a picture once every 20 seconds and then performing a



300x300 matrix multiply transform on it. Could all of the boards listed meet this deadline?
No, the Raspberry Pi and Gumstix Overo both take longer than 20s and the Beagleboard is dangerously close.

- Assume a workload where a device takes a picture once a minute then does a 300x300 matrix multiply (as seen in Table 1). The device is idle when not multiplying, but under full load when it is. Over an hour, what is the energy usage of the Chromebook? What is the energy usage of the Gumstix?

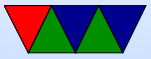


Chromebook per minute: $(1.39s \times 8.1W) + (58.61s \times 5.4W) = 327.75J$

Chromebook per hour: 327.75J * 60 = 19.7kJ

Gumstix per minute: $(27s \times 2.9W) + (33s \times 2.6W) = 164.1J$

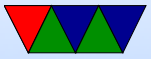
Gumstix per hour: 164.1J * 60 = 9.8kJ



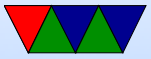
Pandaboard Power Stats

- 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

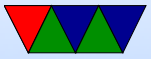


Easy ways to reduce Power Usage

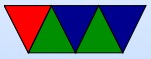


DVFS

- 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?

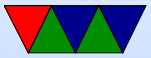


- Often takes time, on order of milliseconds, to switch frequency. Switching voltage can be done with less hassle.



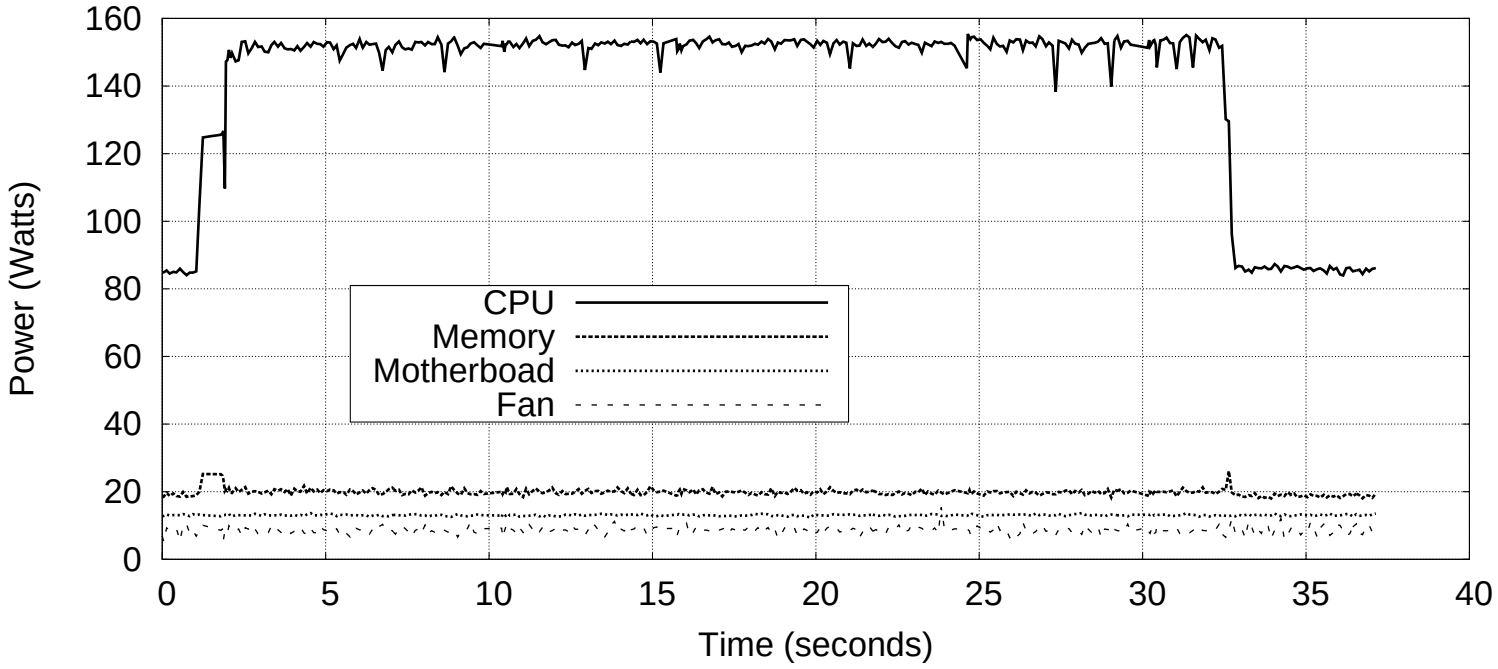
When can we scale CPU down?

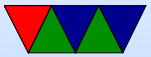
- System idle
- System memory or I/O bound
- Poor multi-threaded code (spinning in spin locks)
- Thermal emergency
- User preference (want fans to run less)



Existing Related Work

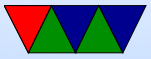
Plasma/dposv results with Virginia Tech's PowerPack



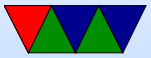


Powerpack

- 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 connecter



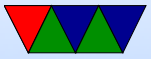
- Memory Power – DIMMs powered by four 5VDC pins
- Motherboard Power – 3.3V pins. Claim NIC contribution is minimal, checked by varying workload
- System fans



Shortcomings of current methods

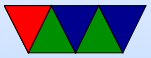
- Each measurement platform has a different interface
- Typically data can only be recorded off-line, to a separate logging machine, and analysis is done after the fact
- Correlating energy/power with other performance metrics can be difficult

- PAPI (Performance API) is a cross-platform, open-source library for gathering performance-related data
- PAPI-C interface makes adding new power measuring



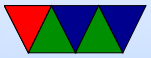
components straightforward

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



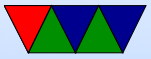
More PAPI benefits

- 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

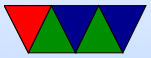


Current PAPI Components

- Support for power components in recent PAPI 5.0 release
- Under development components found in `papi.git`

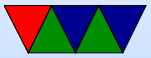


Watt's Up Pro Meter

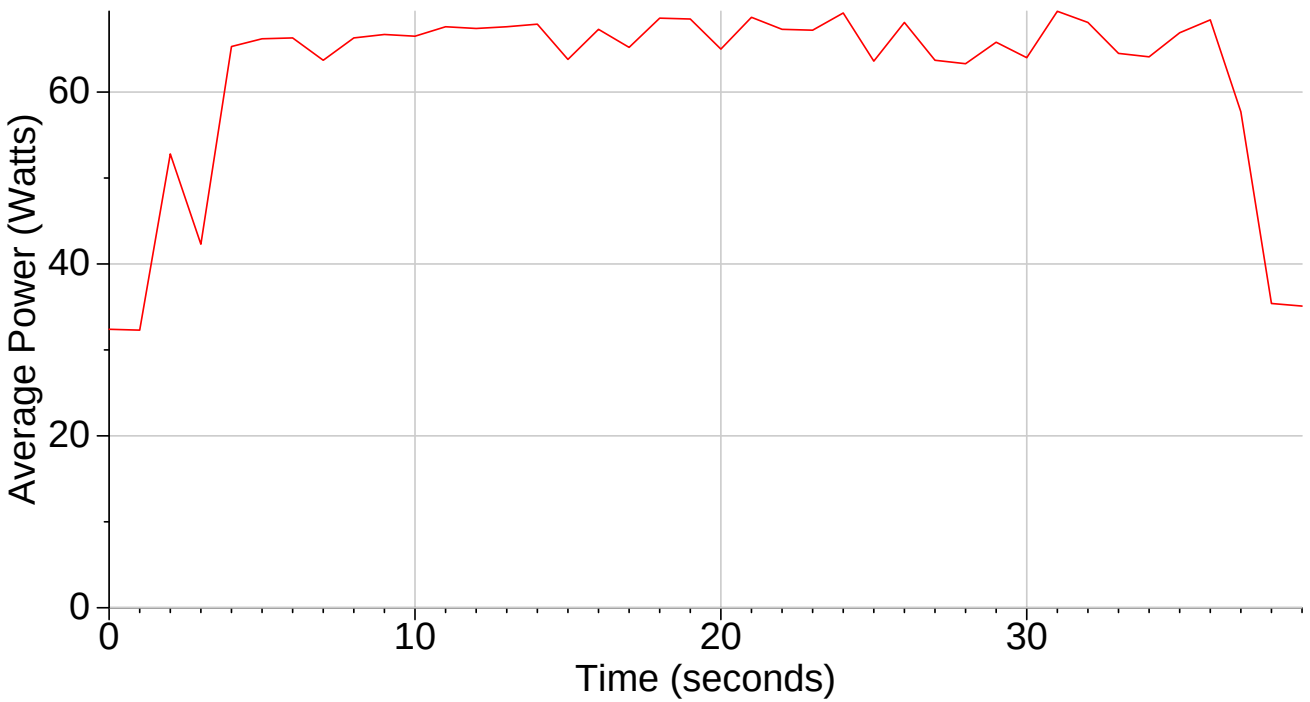


Watt's Up Pro Features

- Can measure 18 different values with 1 second resolution (Watts, Volts, Amps, Watt-hours, etc.)
- Values read over USB
- Joules can be derived from power and time
- Can only measure system-wide

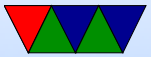


Watt's Up Pro Graph



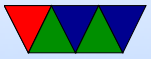
PLASMA Cholesky Factorization N=10,000 threads=2

Measured on Core2 Laptop



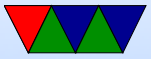
RAPL

- **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



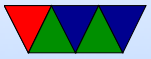
How RAPL Works

- 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)



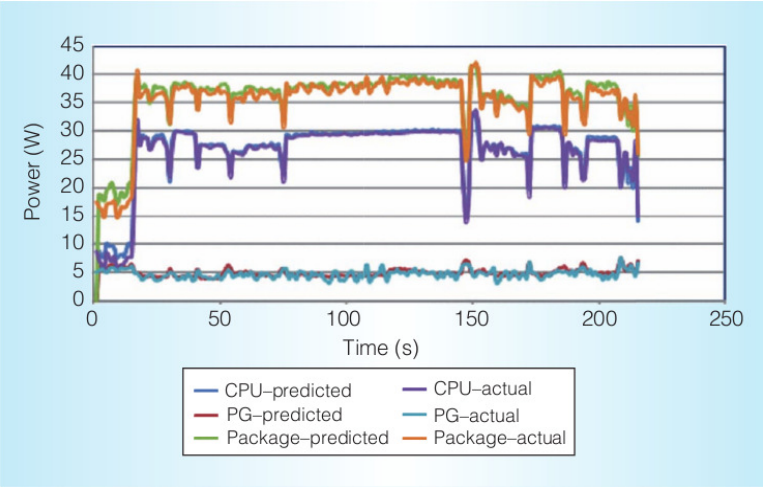
Available RAPL Readings

- `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

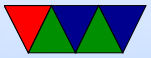


RAPL Measurement Accuracy

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

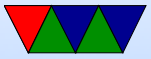


Rotem et al. (IEEE Micro, Mar/Apr 2012)



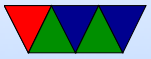
RAPL Accuracy, Continued

- 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.

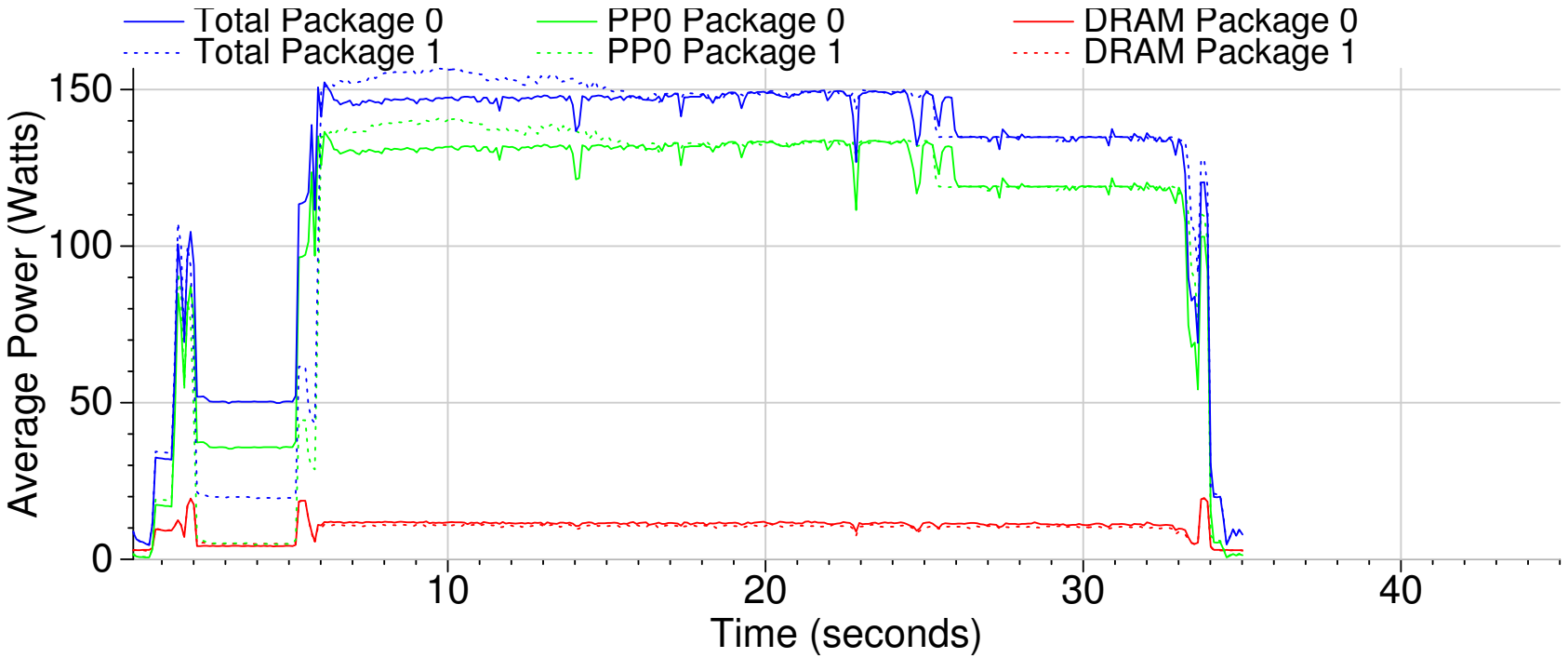


RAPL PAPI Interface

- Access to RAPL data requires reading a CPU MSR register. This requires operating system support
- Linux currently has no driver and likely won't for the near future
- Linux does support an "MSR" driver. Given proper read permissions, MSRs can be accessed via `/dev/cpu/*/msr`
- PAPI uses the "MSR" driver to gather RAPL values

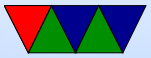


RAPL Power Plot

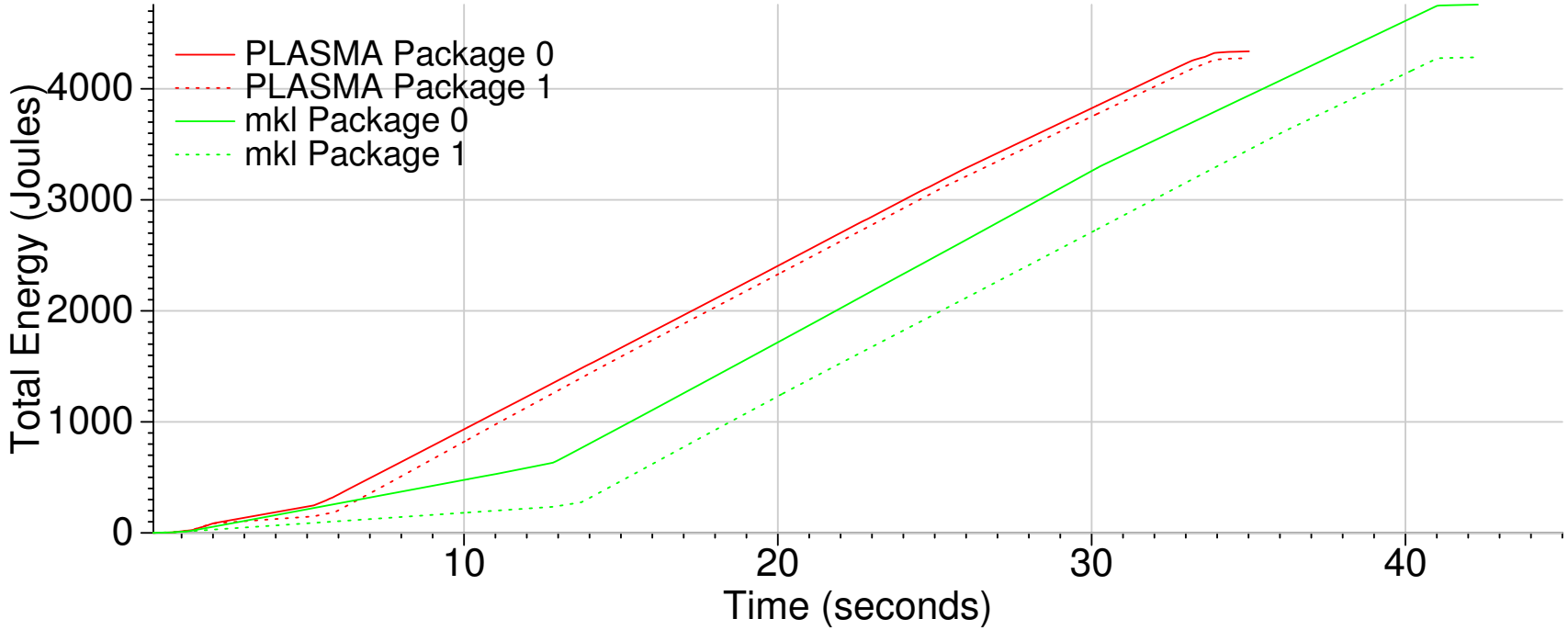


PLASMA Cholesky Factorization N=30,000 threads=16

Measured on SandyBridge EP

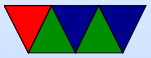


RAPL Energy Plot



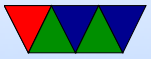
Cholesky Factorization N=30,000 threads=16

Measured on SandyBridge EP

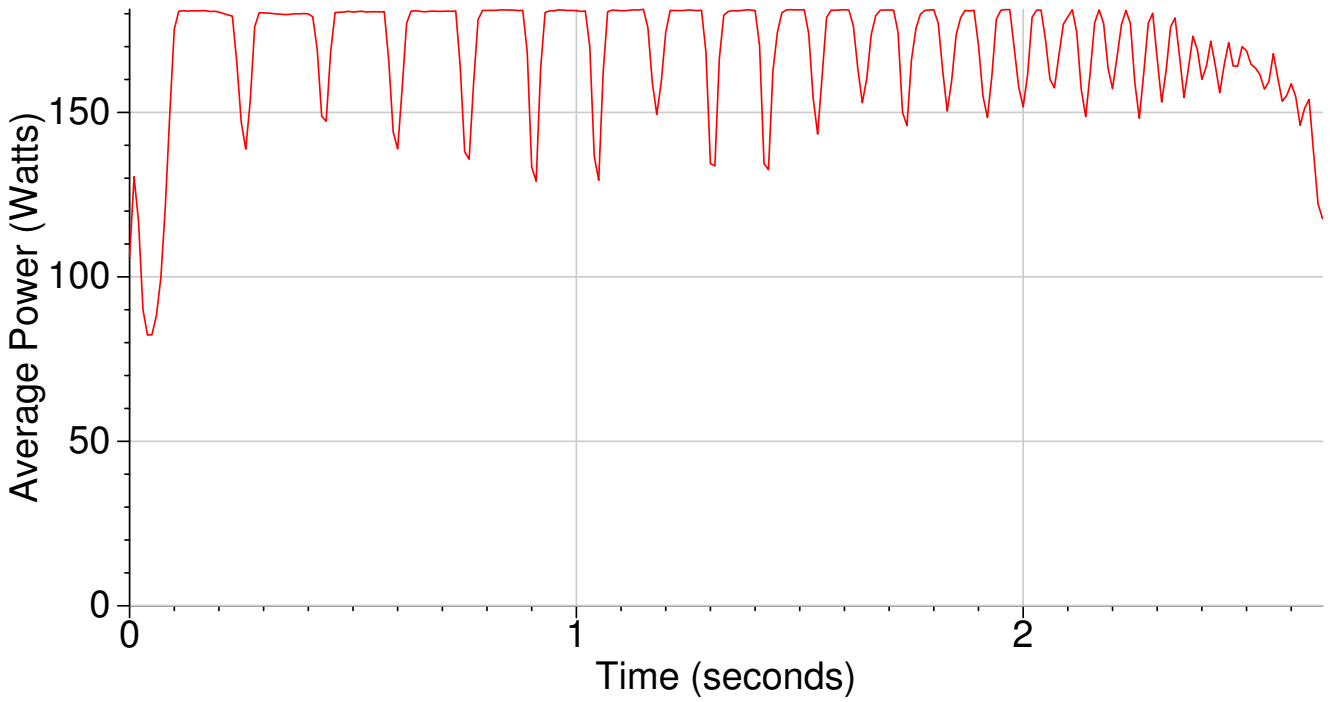


NVML

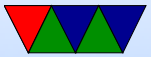
- Recent NVIDIA GPUs support reading power via the NVIDIA Management Library (**NVML**)
- On Fermi C2075 GPUs it has milliwatt resolution within $\pm$5W and is updated at roughly 60Hz
- The power reported is that for the entire board, including GPU and memory



NVML Power Graph

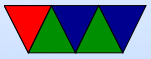


MAGMA LU 10,000, Nvidia Fermi C2075



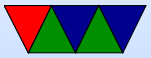
Near-future PAPI Components

These components do not exist yet, but support for them should be straightforward.



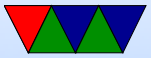
AMD Application Power Management

- Recent AMD Family 15h processors also can report "Current Power In Watts" via the Processor Power in the TDP MSR
- Support for this can be provided similar to RAPL
- We just need an Interlagos system where someone gives us the proper read permissions to `/dev/cpu/*/msr`



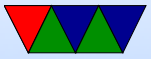
PowerMon 2

- PowerMon 2 is a custom board from RENCI
- Plugs in-line with ATX power supply.
- Reports results over USB
- 8 channels, 1kHz sample rate
- We have hardware; currently not working



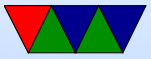
PAPI-based Power Models

- There's a lot of related work on estimating energy/power using performance counters
- PAPI user-defined event infrastructure can be used to create power models using existing events
- Previous work (McKee et al.) shows accuracy to within 10%



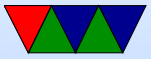
Measuring using PAPI

Measuring Energy/Power with PAPI is done the same as measuring any other event

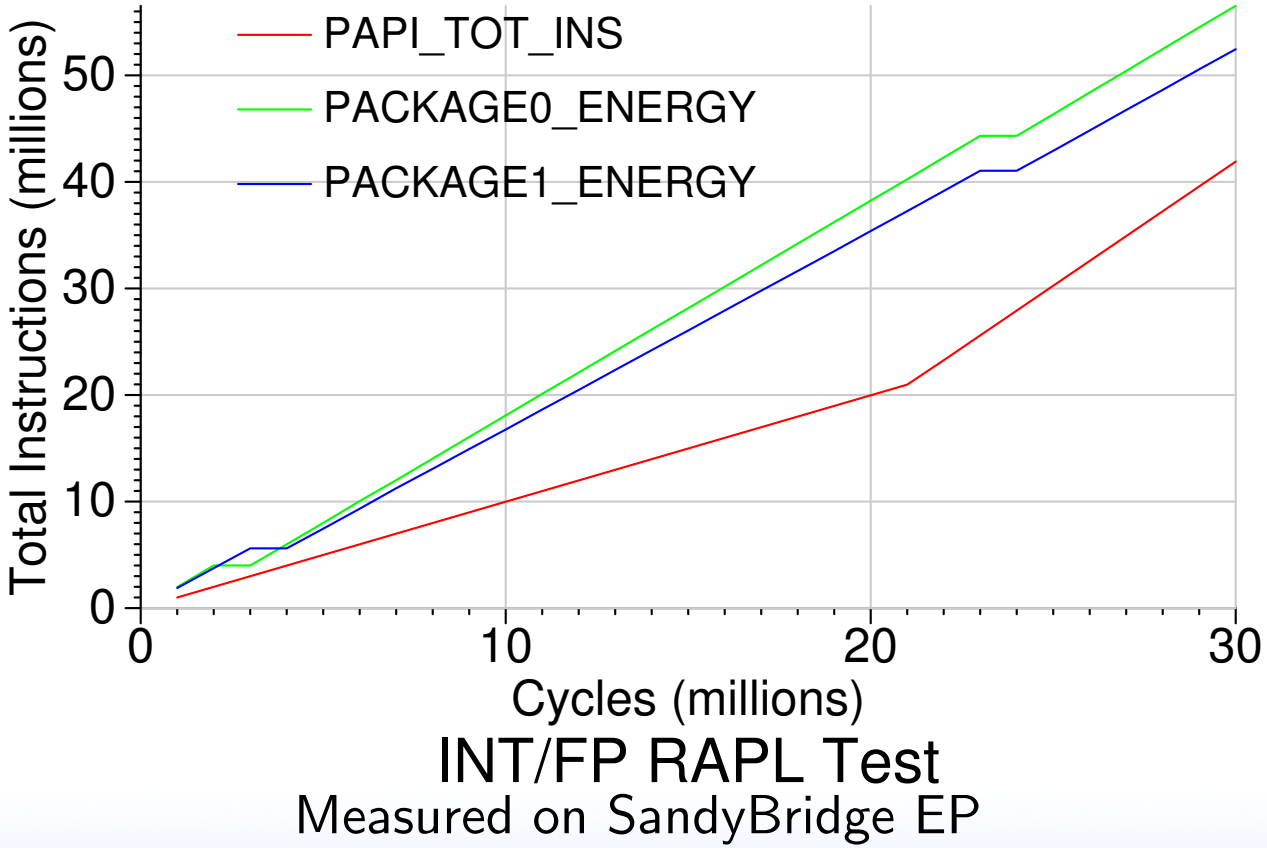


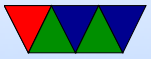
Listing Events

```
>  papi_native_avail
=====================================
 Events in Component: linux-rapl
=====================================
-------------------------------------
| PACKAGE_ENERGY:PACKAGE0
|   Energy used by chip package 0
-------------------------------------
| PACKAGE_ENERGY:PACKAGE1
|   Energy used by chip package 1
-------------------------------------
| DRAM_ENERGY:PACKAGE0
|   Energy used by DRAM on package 0
-------------------------------------
```



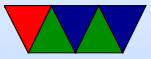
Measuring Multiple Sources





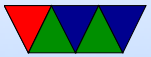
Beagle Board

- 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.



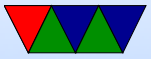
PandaBoard

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



Raspberry Pi

- People using external power meters.



TODO

Notes from RAPL validation? More results from students I hired? More details of trying to build boards to do this?