ECE 571 – Advanced Microprocessor-Based Design Lecture 5

Vince Weaver

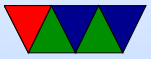
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

2 February 2016

Announcements

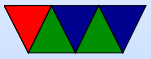
- HW#2 was posted
- HW#1 Mostly graded



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
```

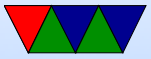


- Valgrind, 1m10.189s == roughly 20 times slower?

```
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
```

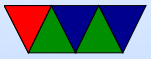


- Skid instructions – mov is more likely than sub?

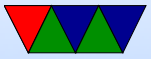
```
perf annotate:
  1.14  5f0:   mov    (%r10),%edx
  0.56         lea    (%rdx,%r13,1),%eax
  0.80         movzbl (%r15,%rax,1),%eax
  3.29         sub    %r9d,%eax
```

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
```



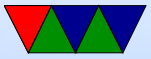
Power and Energy



Definitions and Units

People often say Power when they mean Energy

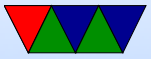
- Energy – Joules, kWH (3.6MJ), Therm (105.5MJ), 1 Ton TNT (4.2GJ), eV (1.6×10^{-19} J), BTU (1055 J), horsepower-hour (2.68 MJ), calorie (4.184 J)
- Power – Energy/Time – Watts (1 J/s), Horsepower (746W), Ton of Refrigeration (12,000 Btu/h)
- Volt-Amps (for A/C) – same units as Watts, but not same thing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



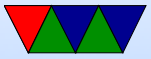
Power and Energy in a Computer System

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

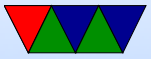
- Old, but hard to find thorough breakdowns like this
- Thinkpad Laptop, 1.3GHz Pentium M, 256M, 14" display
- Oscilloscope with voltage probe and clamp-on current probe.
- Measured V and Current. P=IIR. V=IR P=IV, subtractive for things without wires



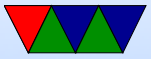
- Total System Power 14-30W
  - Hard Drive 0.5-2W (Flash/SSD less)
  - LCD 1W (slightly more black than white)
  - Backlight Inverter (this is before LED) 1-4W depending on brightness
  - CPU 2-15W (with scaling)
  - GPU 1-5W
  - Memory 0.45 - 1.5W
  - Power Supply Loss - 0.65W
  - Wireless 0.1 - 3W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



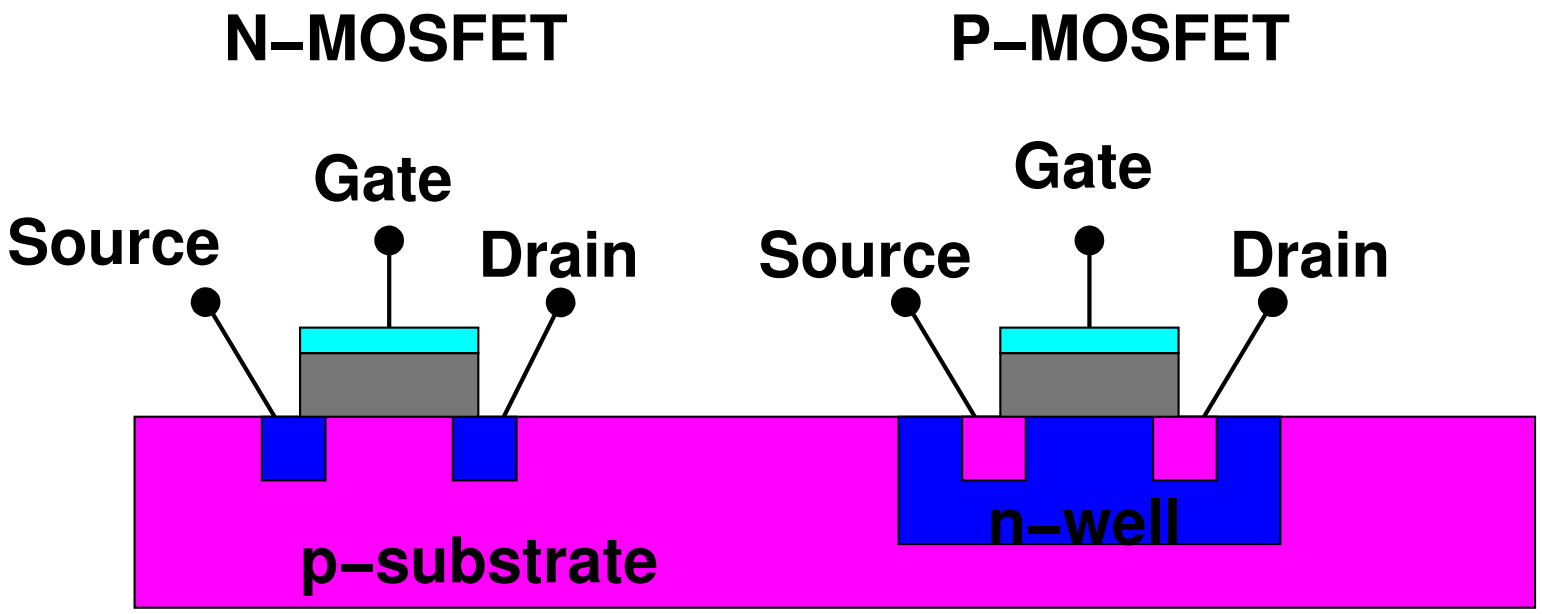
- Not in paper but USB 2.0 – 5V, can draw 5 units of 100mA each, 2.5W)

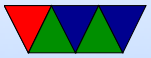


CPU Power and Energy



CMOS Transistors



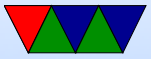


CMOS Dynamic Power

- $P = C\Delta V V_{dd} \alpha f$
  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 $P \approx \frac{1}{2} C V_{dd}^2 f$

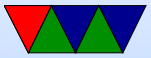
- Some pass-through loss (V momentarily shorted)



CMOS Dynamic Power Reduction

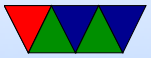
How can you reduce Dynamic Power?

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



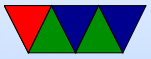
CMOS Static Power

- 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: $P_{static} = I_{leakage} V_{dd}$



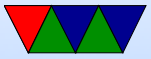
Leakage Mitigation

- 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



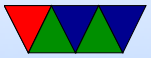
Total Energy

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



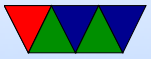
Delay

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



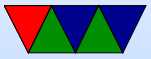
Thermal Issues

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



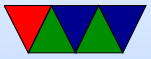
Metrics to Optimize

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



Power Optimization

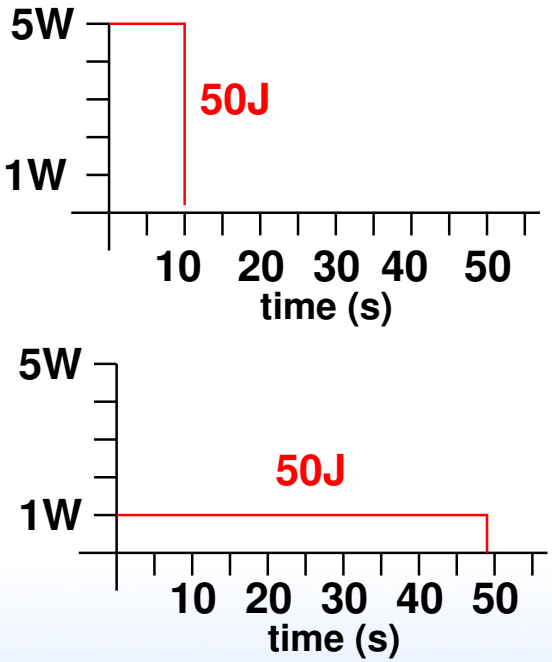
- Does not take into account time. Lowering power does no good if it increases runtime.

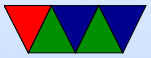


Energy Optimization

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

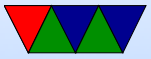
  Which is better?



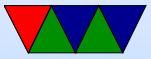


Energy Delay – Watt/t*t

- 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

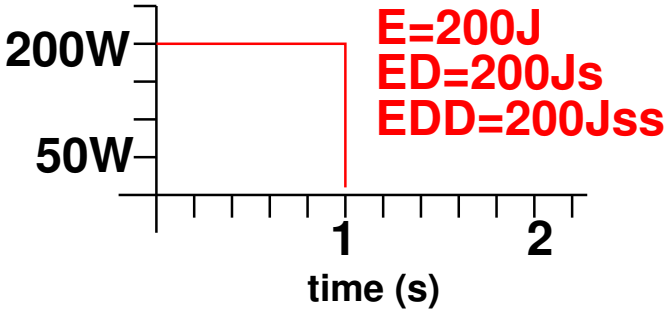


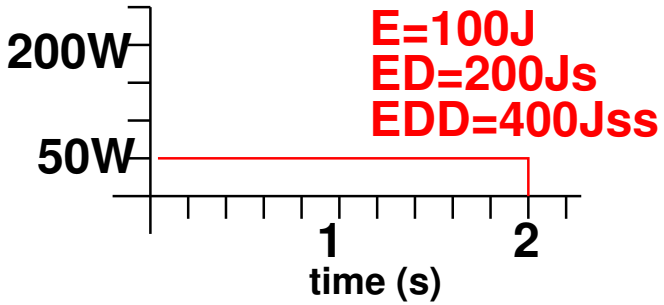
- Technology Scaling – reduces V and power.
- Transition Reduction – better logic design, have fewer transitions
  Get rid of clocks? Asynchronous? Clock-gating?

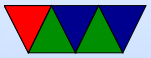


ED Optimization

Which is better?

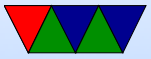






Energy Delay Squared– E*t*t

- 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



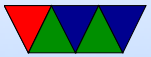
Energy Delay / Energy Delay Squared

Lower is better.

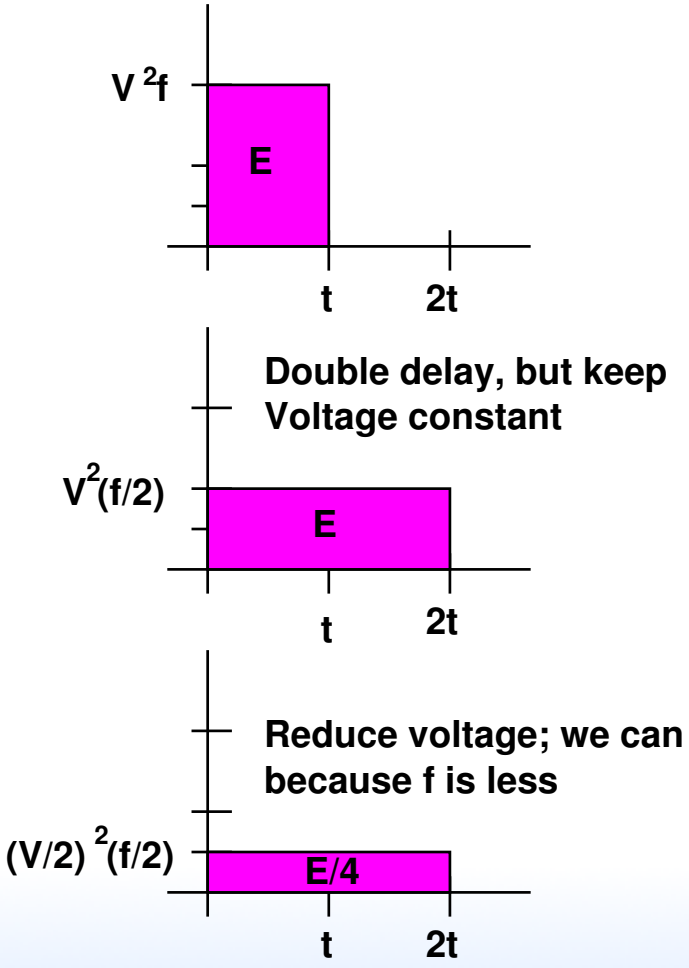
Energy	Delay	ED	ED^2
5J	2s	$10Js$	$20Js^2$
5J	3s	$15Js$	$45Js^2$

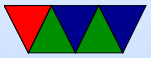
Same ED, Different ED^2

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

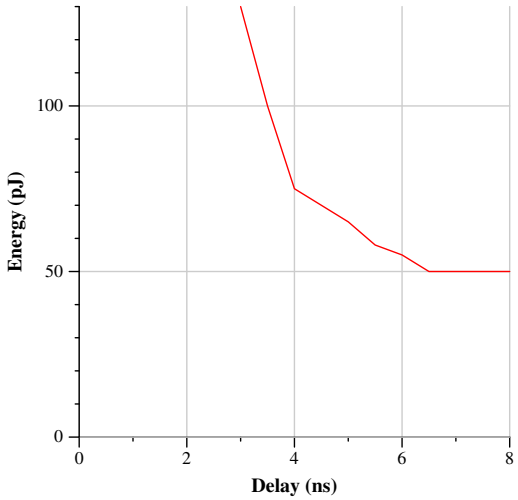


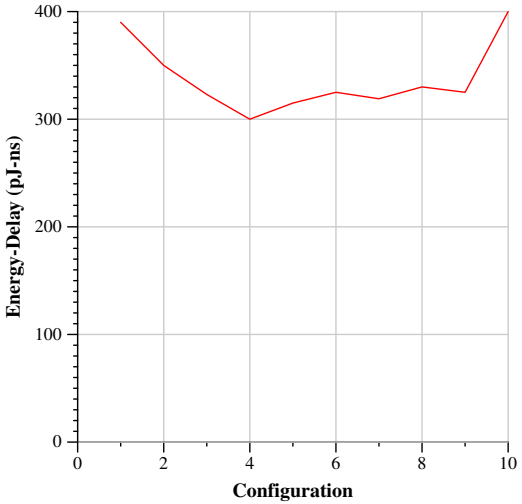
Energy Example

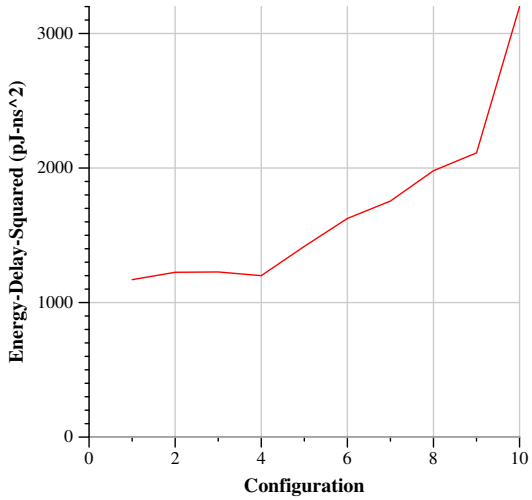




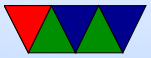
Energy-Delay Product Redux





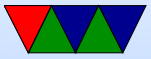


Roughly based on data from "Energy-Delay Tradeoffs in CMOS Multipliers" by Brown et al.



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]