# ECE 571 – Advanced Microprocessor-Based Design Lecture 9

Vince Weaver

https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

23 September 2024

## Announcements

- HW#3 was posted, RAPL
- Note, the equake benchmark takes a while to run (a few minutes). Don't give up on it.
- Also, there are a lot of people running HW#3 so you might want to be sure no one else is running when you start yours. You can use `w`, or `top`, or `htop`
  In an ideal world I'd have you using `slurm`

## Power and Energy Continued

## Power and Energy in a Computer System

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

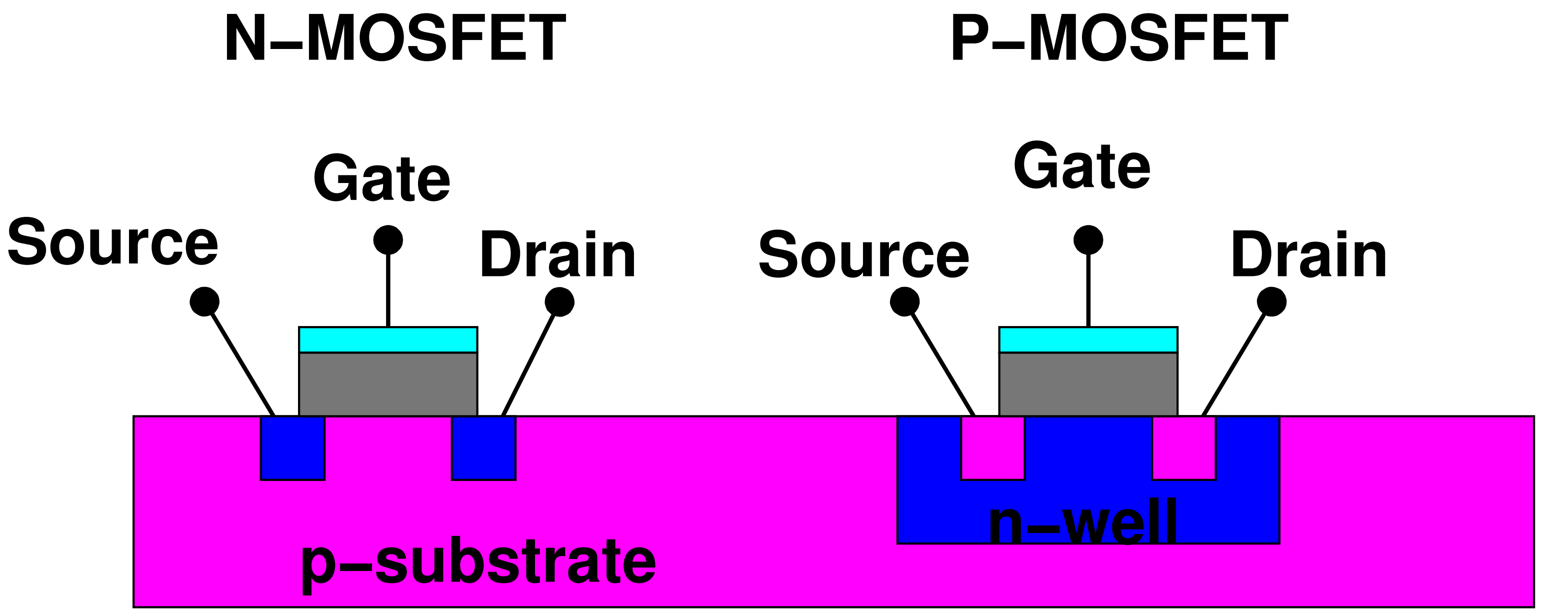
- Old, but hard to find thorough breakdowns like this
- Thinkpad Laptop, 1.3GHz Pentium M, 256M, 14" disp
- Oscilloscope, voltage probe and clamp-on current probe
- Measured V and Current. P=IIR. V=IR P=IV, subtractive for things w/o wires
- Total System Power 14-30W
- Old: no LED backlight, no SDD, etc.

## Modern results are from CUGR/REU student research.

| | Laptop (2004) | Modern | Server? |
|---|---|---|---|
| Hard Drive | 0.5-2W | 5W | |
| LCD | 1W | | |
| Backlight | 1-4W | | |
| CPU | 2-15W | 60+W | |
| GPU | 1-5W | 50+W | |
| Memory | 0.5-1.5W | 1-5W | |
| Power Supply | 0.65W | | |
| Wireless | 0.1 - 3W | | |
| CD-ROM | 3-5W | | |
| USB | (max 2.5W) | | |
| USB keyboard | | 0.04W | |
| USB mouse | | 0.03W | |
| USB flash | | 0.5W | |
| USB wifi | | 0.5W | |

## CPU Power and Energy

## Traditional CMOS Transistors



## Modern CMOS Transistors

TODO: diagram of FinFets and Gate-all-around

## 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
  $\alpha$ is activity factor, transitions per clock cycle
  F is frequency

- $\alpha$ 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 $\alpha$ (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

## Notes on Process Technology (older)

- 8micron (8000nm) 6502
- 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

# Notes on Process Technology (recent)

- 14nm and smaller – ??
  Extreme UV (13.5nm light, hard-vacuum required)?
  Electron beam?
  Intel got stuck here
- 10, 7, 5, 3, 2 – FinFETs, GAAFET
- Intel calls 2nm 20A

## 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/65nm
- Pi2 BCM2836 – 28nm

## 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)
- $Energy * Delay$
- $Energy * Delay^2$

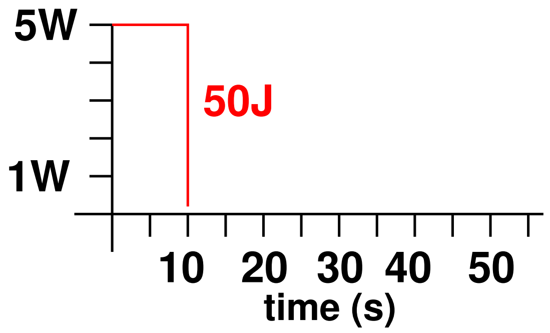
## Power Optimization

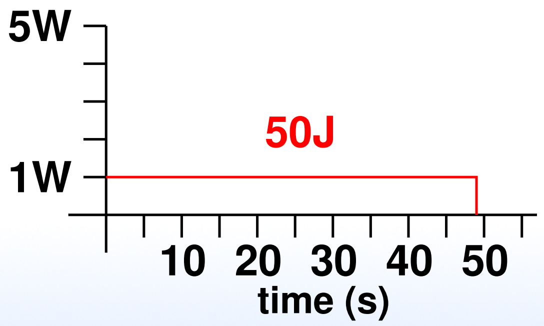
- Does not take into account time. Lowering power does no good if it increases runtime.

## Energy Optimization

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

Which is better?





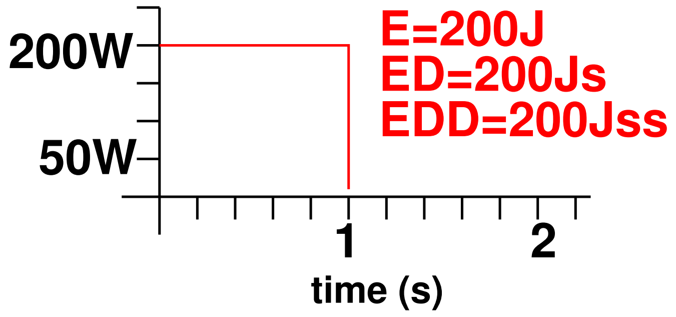
## Energy Delay – Watt/t*t

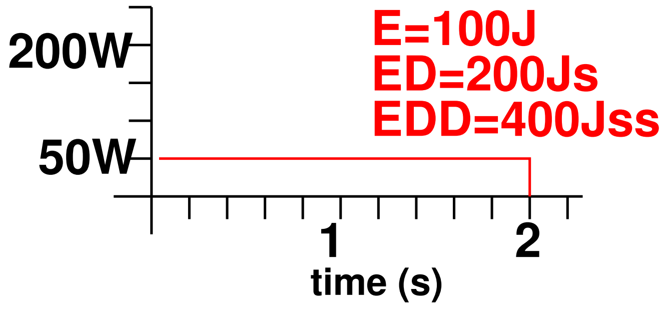
- Horowitz, Indermaur, Gonzalez (Low Power Electronics, 1994)
- Need to account for delay, so that lowering Energy does not made delay (time) worse
- Voltage Scaling – in general scaling low makes transistors slower
- Transistor Sizing – reduces Capacitance, also makes transistors slower

- 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

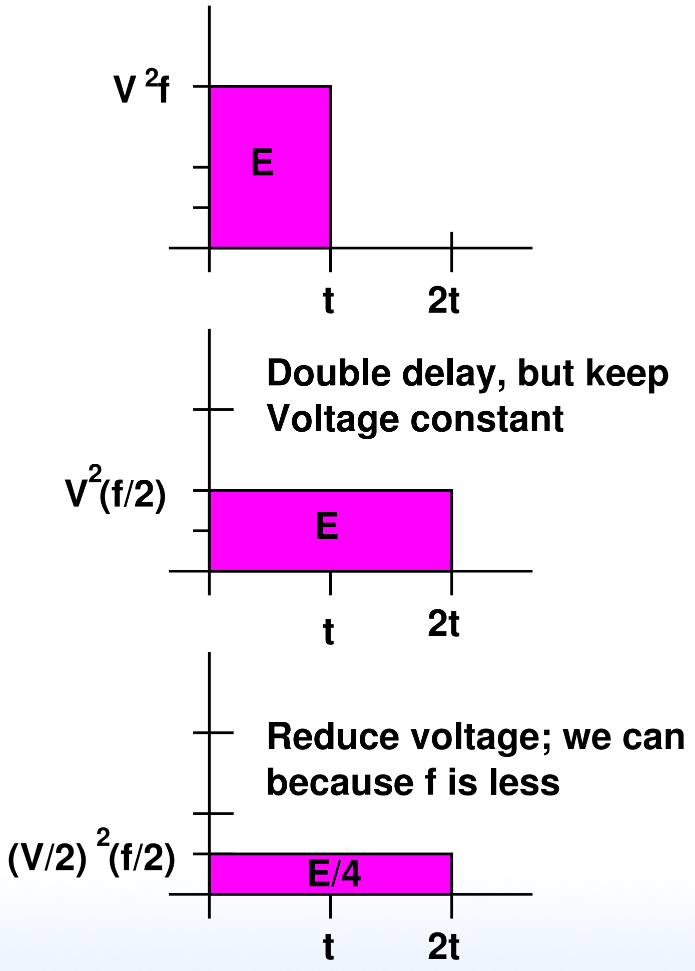
Lower is better.

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

Same $ED$, Different $ED^2$

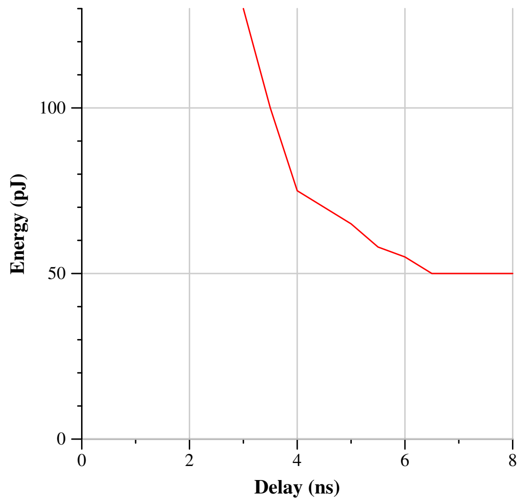
| Energy | Delay | $ED$ | $ED^2$ |
|---|---|---|---|
| 5J | 2s | $10Js$ | $20Js^2$ |
| 2J | 5s | $10Js$ | $50Js^2$ |

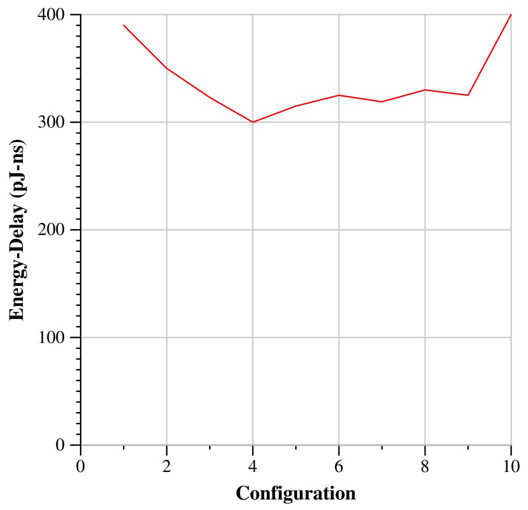
# Energy Example

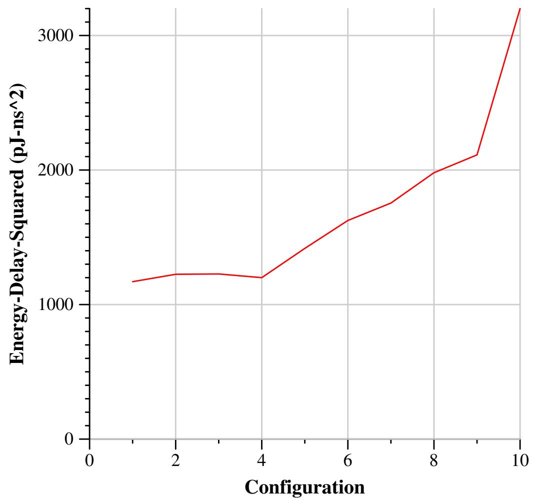


$V^2f$
E
t
2t
Double delay, but keep Voltage constant
$V^2(f/2)$
E
t
2t
Reduce voltage; we can because f is less
$(V/2)^2(f/2)$
E/4
t
2t

## 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]
- Energy-Delay-FIT (reliability?)

## Power and Energy Concerns

Table 1: OpenBLAS HPL N=10000 (Matrix Multiply)

| Machine | Processor | Cores | Freq | Idle Power | Load Power | Time Time | Total Energy |
|---|---|---|---|---|---|---|---|
| Raspberry Pi 2 | Cortex-A7 | 4 | 900MHz | 1.8W | 3.4W | 454s | 1543J |
| Dragonboard | Cortex-A53 | 4 | 1.2GHz | 2.4W | 4.7W | 241s | 1133J |
| Raspberry Pi 3 | Cortex-A53 | 4 | 1.2GHz | 1.8W | 4.3W | 178s | 765J |
| Jetson-TX1 | Cortex-A57 | 4 | 1.9GHz | 2.1W | 13.4W | 47s | 629J |
| Macbook Air | Broadwell | 2 | 1.6GHz | 10.0W | 29.1W | 14s | 407J |

1. Which machine has the lowest under-load power draw?
   Pi 2

2. Which machine consumes the least amount of energy?
   Broadwell Macbook Air

3. Which machine computes the result fastest?
   Broadwell Macbook Air

4. Consider a use case with an embedded board taking a picture once every 60 seconds and then performing a matrix-multiply similar to the one in the benchmark (perhaps for image-recognition purposes). Could all of the boards listed meet this deadline?
   No, only the Jetson and Macbook Air can meet the

deadline

5. Assume a workload where a device takes a picture once a minute then does a large matrix multiply (as seen in Table 1). The device is idle when not multiplying, but under full load when it is.

(a) Over a mine, what is the total energy usage of the Jetson TX-1?
Each Minute = (13s Idle * 2.1W) + (47s Load *13.4W) = 657J

(b) Over a minute, what is the total energy usage of the Macbook Air?

Each Minute $=$ (46s * 10W) $+$ (14*29.1) $=$ 867J

## 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)