

18-447 Lecture 12: Energy and Power

James C. Hoe

Department of ECE

Carnegie Mellon University

Housekeeping

- Your goal today
 - a working understanding of energy and power
 - appreciate their significance in comp arch today
- Notices
 - Lab 2, **due this week**
 - Lab 3 posted but starts after break
 - HW 3, **due **Wed** 3/13** (Handout #8)
 - Midterm 1, **Wed 3/13, covers up to Lec ~~13~~12**
- Readings
 - Design challenges of technology scaling, Borkar, 1999.
 - Synthesis Lectures (advanced optional): Power-Efficient Comp Arch: Recent Advances, 2014

First some intuitions

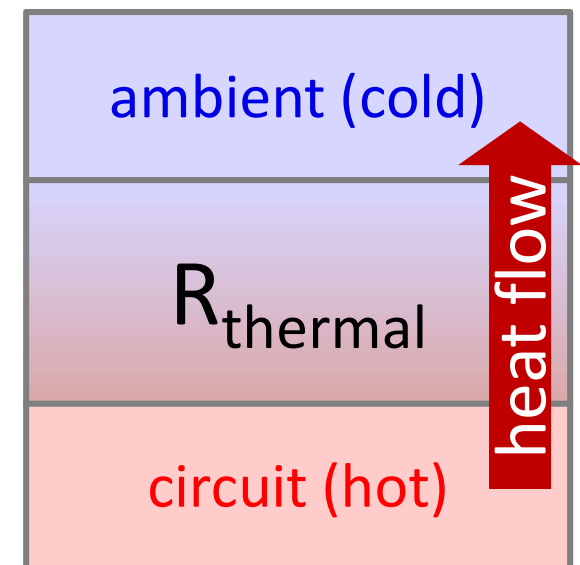
Energy and Power

- CMOS logic transitions involve charging and discharging of parasitic capacitances
- Energy (Joule) dissipated as resistive heat when “charges” flow from VDD to GND
 - take a certain amount of energy per operation (e.g., addition, reg read/write, (dis)charge a node)
 - to the first order, energy \propto amount of compute
- Power (Watt=Joule/s) is rate of energy dissipation
 - more op/sec then more Joules/sec
 - to the first order, power \propto performance

Power concerns usually more about heat removal

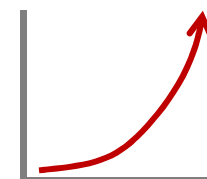
Heat and Thermal Resistance

- Resistive heat in the circuit must be removed in steadystate (www.youtube.com/watch?v=BSGcnRanYMM)
- Can summarize everything between **circuit** and **ambient** by characteristic $R_{\text{thermal}} = K/W$
 - convey power W in heat across temperature difference $K = T_{\text{circuit}} - T_{\text{ambient}}$
- To dissipate more power in circuit
 1. let T_{circuit} get hotter (to a point)
 2. turn-up AC \Rightarrow lower T_{ambient}
 3. better cooling \Rightarrow lower R
- Economics/market driven choices



Work and Perf. from Joules and Watt

- Fastest without energy/power awareness won't be fastest once constrained
 - power bounds performance directly
 - energy bounds work directly; want for lower J/op bounds performance indirectly



recall that $\text{power} \propto \text{perf}^{\alpha}$ ($\alpha > 1$)

- Consider in context
 - mobile device: limited energy source, hard-to-cool form factor
 - desktop: cooler size, noise, complexity, cost
 - data-center: electric bill, cooling capacity and cost

Ultimately driven by desirability and economics

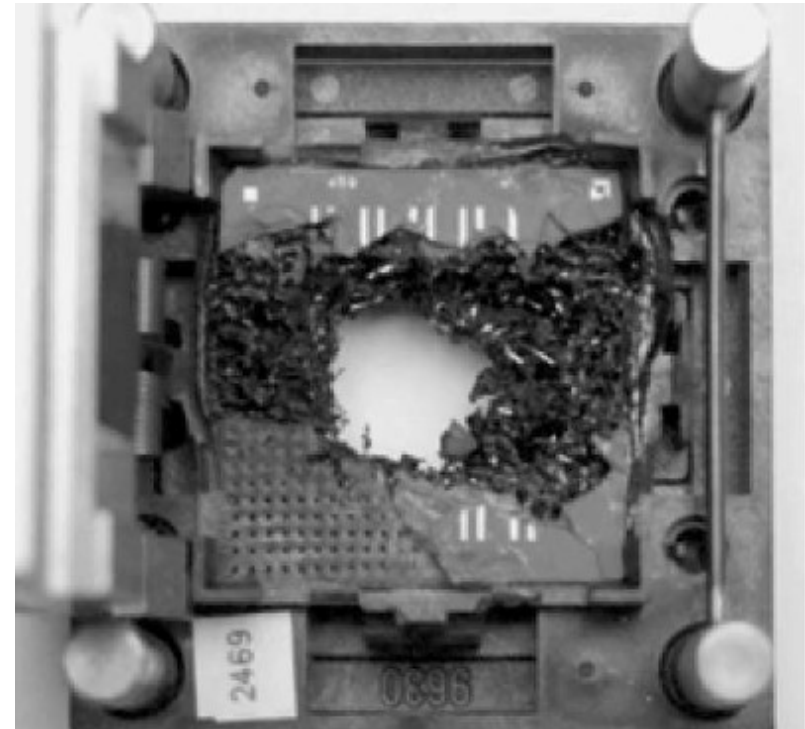
Cooler transistors also faster transistors



[image from Wikipedia, "Overclocking"]

Hot Transistors Leak More, Get Hotter

- Beyond a threshold, stopping the clock cannot arrest positive-feedback runaway
- Modern processors have temp sensors to slow the clock before entering runaway



*Thermal runaway in integrated circuits,
[Vassighi and Sachdev, 2006]*

Some (first-order) nitty-gritty

Work and Runtime

- **Work**
 - scalar quantity for “amount of work” associated with a task
 - e.g., number of instructions to compute a SHA256 hash
- **$T = \text{Work} / k_{perf}$**
 - runtime to perform a task
 - k_{perf} is a scalar constant for the rate in which work is performed, e.g., “instructions per second”

Energy and Power

- $E_{switch} = k_{switch} \cdot Work$
 - “switching” energy associated with task
 - k_{switch} is a scalar constant for “energy per unit work”
- $E_{static} = k_{static} \cdot T = k_{static} \cdot Work / k_{perf}$
 - “leakage” energy just to keep the chip powered on
 - k_{static} is the so called “leakage power”

Faster execution means lower leakage energy???
- $E_{total} = E_{switch} + E_{static} = (k_{switch} + k_{static}/k_{perf}) \cdot Work$
- $P_{total} = E_{total} / T = k_{switch} \cdot k_{perf} + k_{static}$

In Short

- $T = \textit{Work} / k_{perf}$

less work finishes faster

- $E = E_{switch} + E_{static} = (k_{switch} + k_{static}/k_{perf}) \cdot \textit{Work}$

less work use less energy

- $P = P_{switch} + P_{static} = k_{switch} \cdot k_{perf} + k_{static}$

power independent of amount of work

- Reality check

- **Work** not a simple scalar, inst mix, dependencies ...

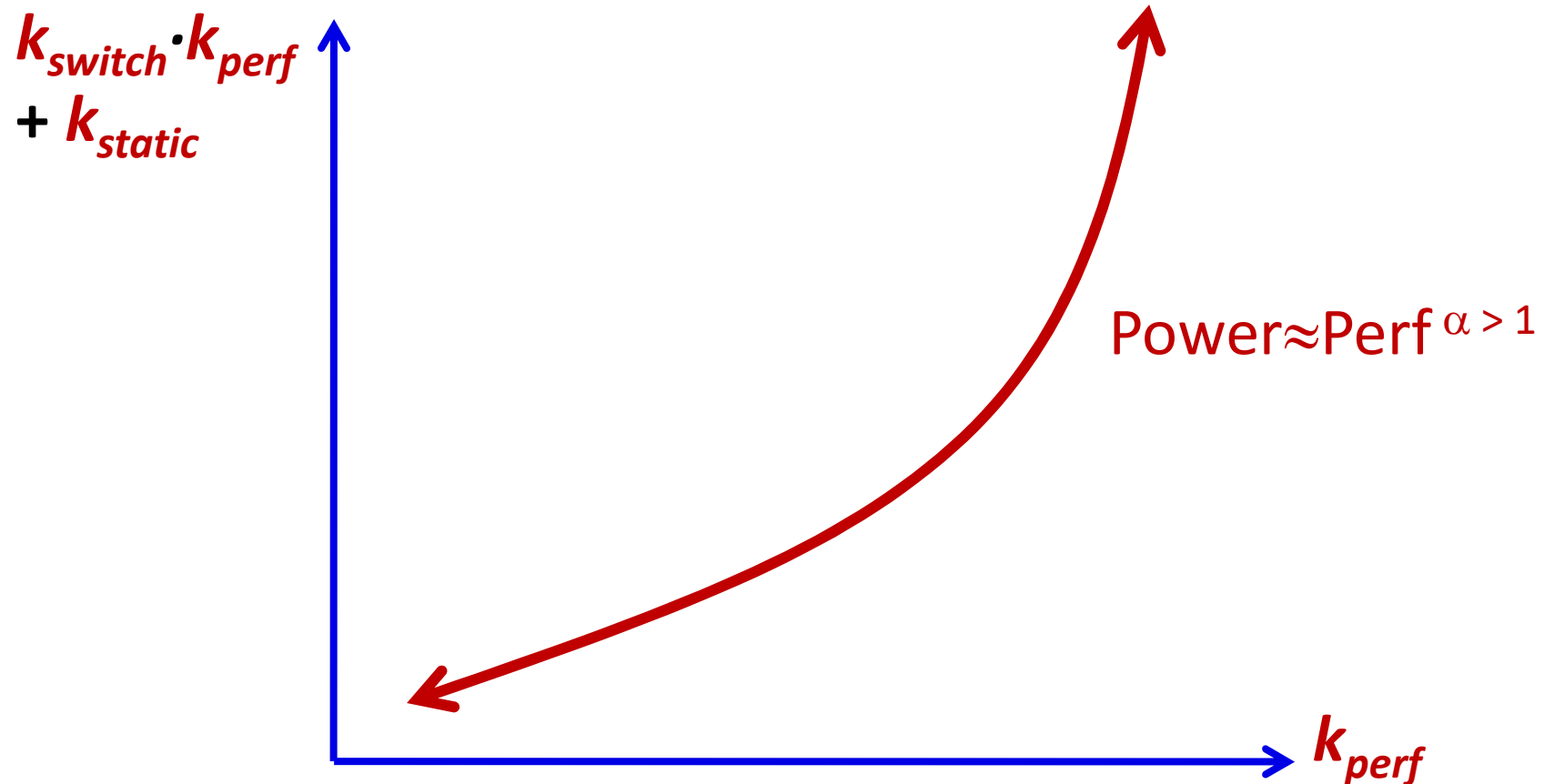
- **k's** are neither scalar nor constant

k_{perf} : inst/sec

k_{switch} : J/inst

k_{static} : J/sec

k_{switch} , k_{static} , k_{perf} not independent



To increase k_{perf} : increase die area increases k_{switch} and k_{static}
faster transistors increases k_{static}

Why so important now?

Ideal Technology Scaling

- Planned scaling occurs in discrete “nodes” where each is $\sim 0.7x$ of the previous in linear dimension
- Take the same design, reducing linear dimensions by $0.7x$ (aka “gate shrink”) leads to ****ideally****
 - die area = $0.5x$
 - delay = $0.7x$; frequency = $1.43x$
 - capacitance = $0.7x$
 - V_{dd} = $0.7x$ (constant field) or $1x$ (constant voltage)
 - power = $0.5x$ (const. field) or $1x$ (const. voltage)
- Take the same area, then
 - transistor count = $2x$, transistor speed = $1.43x$
 - power = $1x$ (const field) or $2x$ (const voltage)

Moore's Law → Performance

- According to scaling theory

@constant complexity (“gate-shrink”):

1x transistors at 1.43x frequency

⇒ 1.43x performance at 0.5x power

@max complexity (“reticle limited”):

2x transistors at 1.43x frequency

⇒ 2.8x performance at constant power

- Historical (until 2005’ish), for high-perf CPUs

expected – ~2x transistors

higher – ~2x frequency (note: faster than scaling predicts)

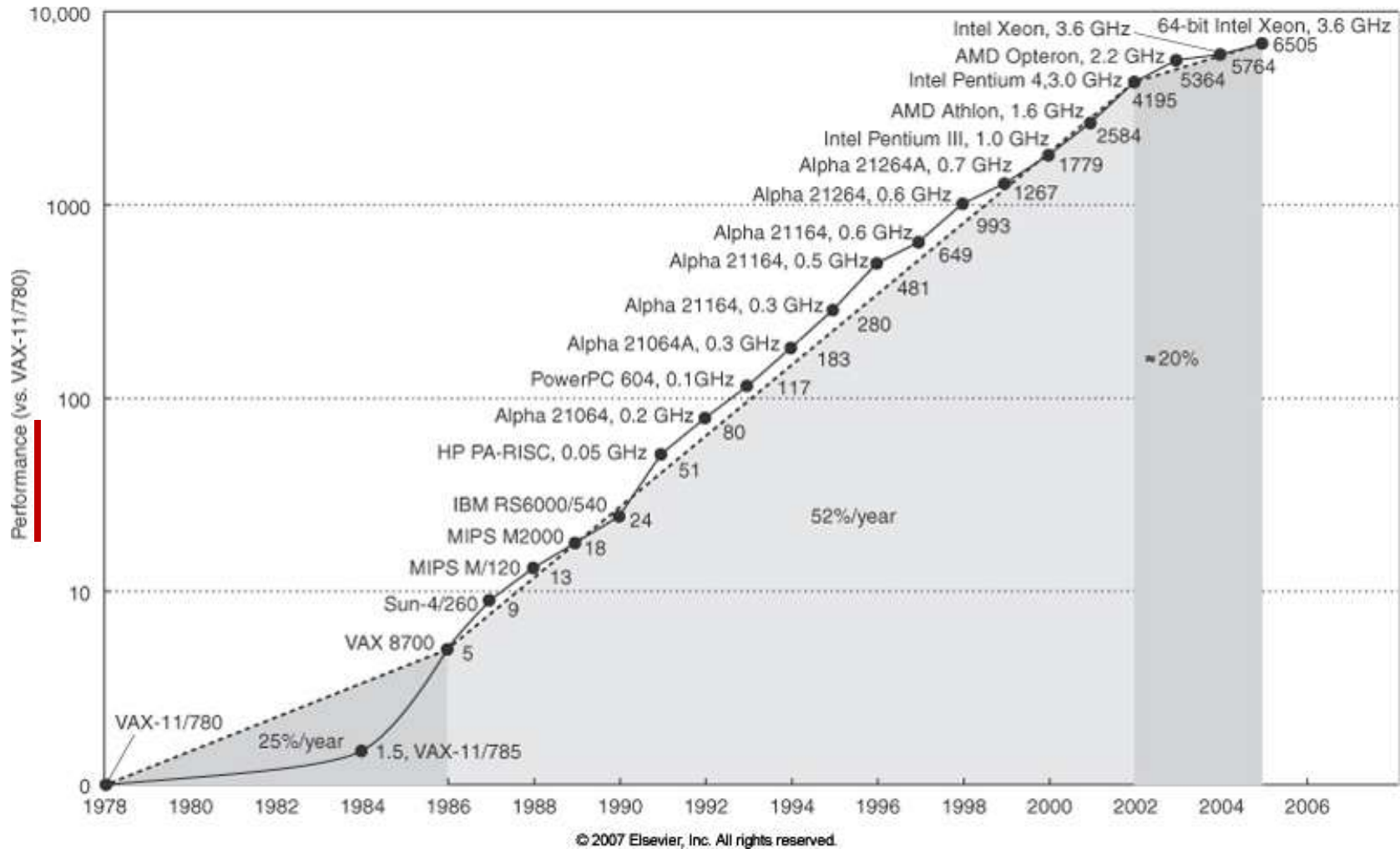
– all together, ~2x performance at ~2x power

lower

higher

Why so far off?

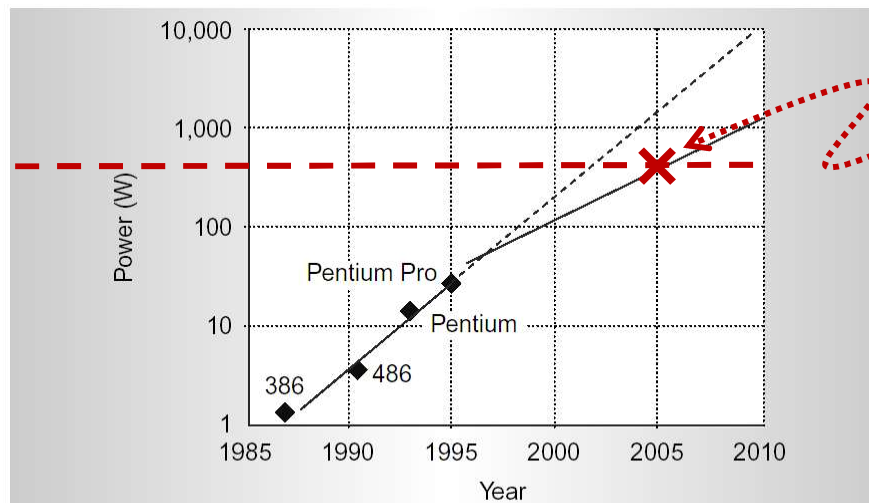
The Other Moore's Law



Performance (In)efficiency

- To hit “expected” performance target
 - push frequency harder by deepening pipelines
 - used the 2x transistors to build more complicated microarchitectures so fast/deep pipelines don't stall (i.e., caches, BP, superscalar, out-of-order)
- The consequence of performance inefficiency is

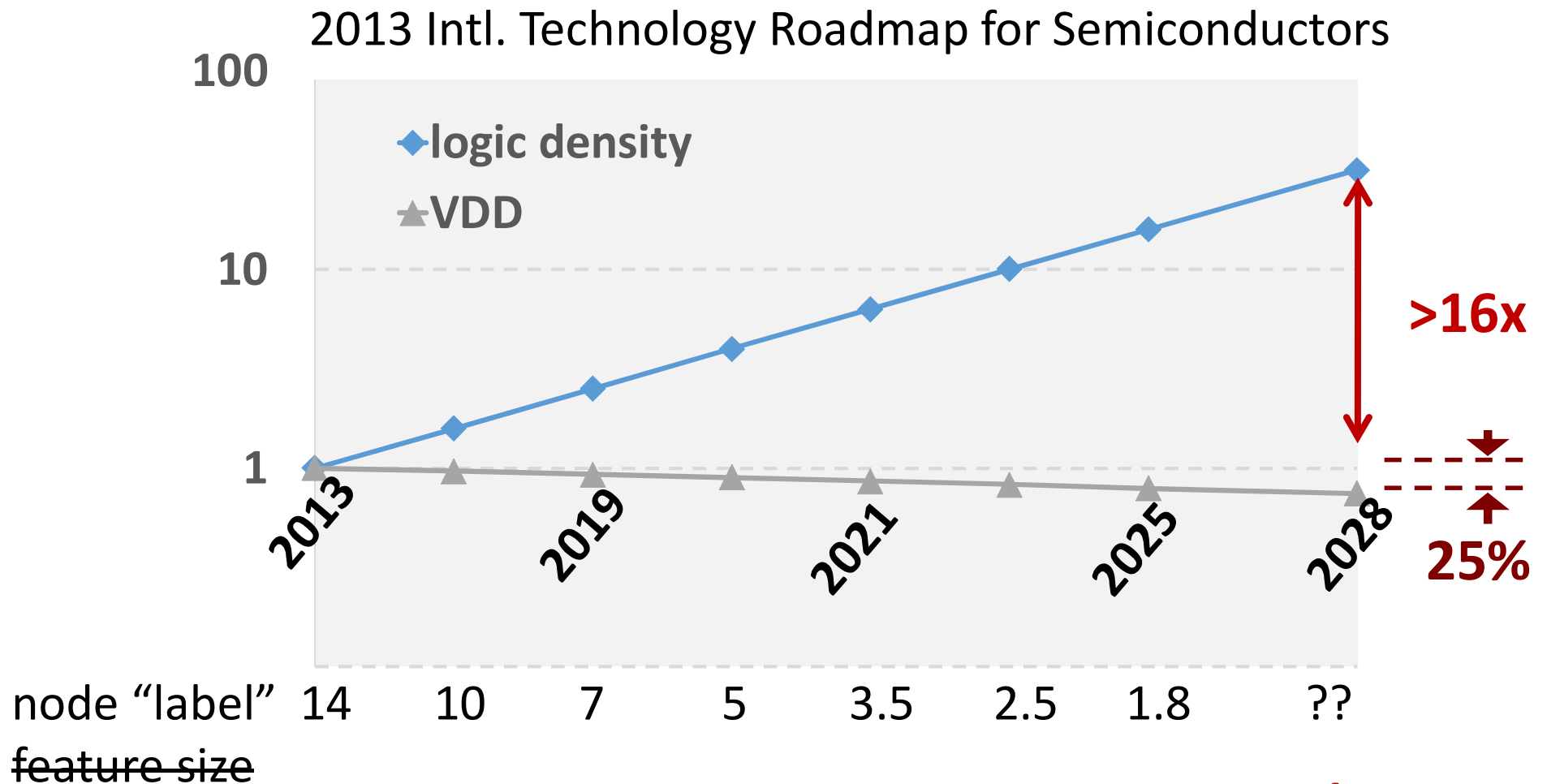
limit of
economical
cooling [ITRS]



2005, Intel
P4 Tehas 150W

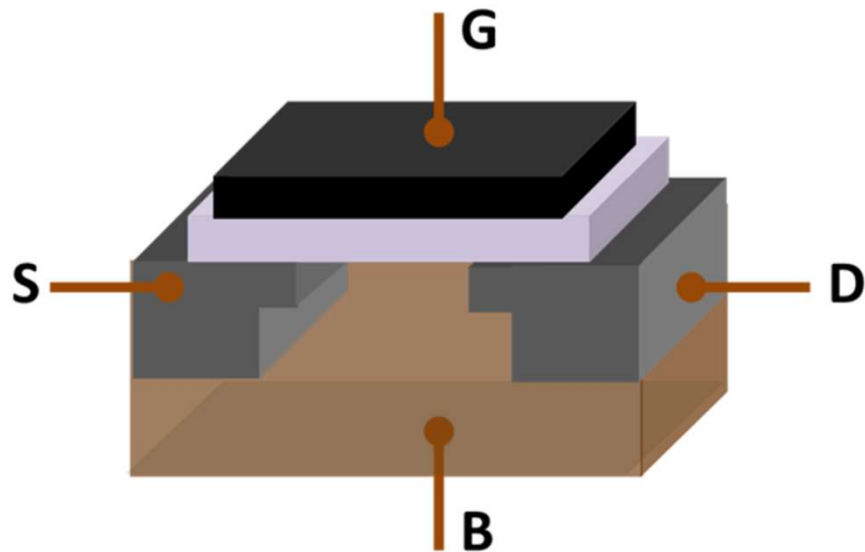
Figure 8. Power dissipation projections. [Borkar, IEEE Micro, July 1999]

Moore's Law without Dennard Scaling

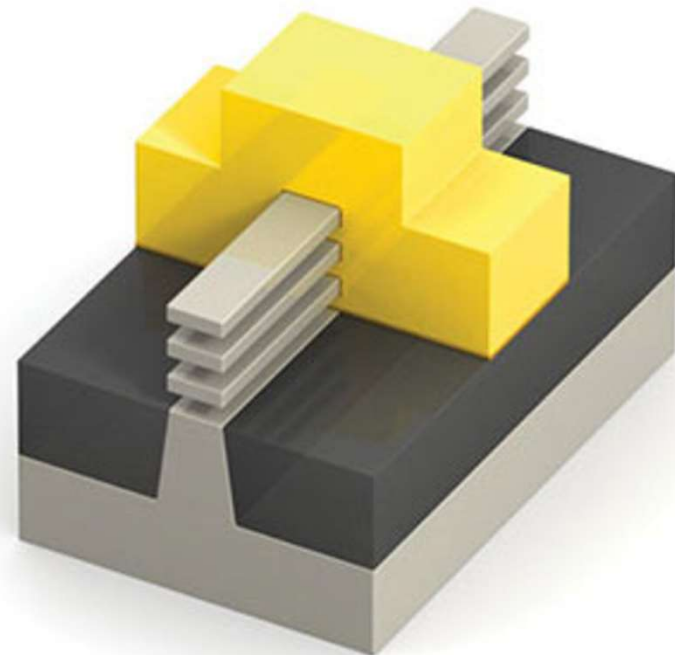


Under fixed power ceiling, more ops/second only achievable if less Joules/op?

What Moore's Law has come down to



[Wikipedia, MOSFET]



[IEEE Spectrum, "The Nanosheet Transistor is the Next (and Maybe Last) Step in Moore's Law"]

Frequency and Voltage Scaling: run slower at lower energy-per-op

Frequency and Voltage Scaling

- Switching energy per transition is
 $\frac{1}{2}CV^2$ (modeling parasitic capacitance)
- Switching power at f transitions-per-sec is
 $\frac{1}{2}CV^2f$
- To reduce power, slow down the clock
- If clock is slower (f'), reduce supply voltage (V') too since transistors don't need to be as fast
 - reduced switching energy, $\frac{1}{2}CV^2 \rightarrow \frac{1}{2}CV'^2$
 - lower V' also reduced leakage current/power

Frequency Scaling (by itself)

- If $Work / k_{perf} < T_{bound}$, we can derate performance by frequency scaling by a factor S_{freq}

$$(Work/k_{perf})/T_{bound} < S_{freq} < 1$$

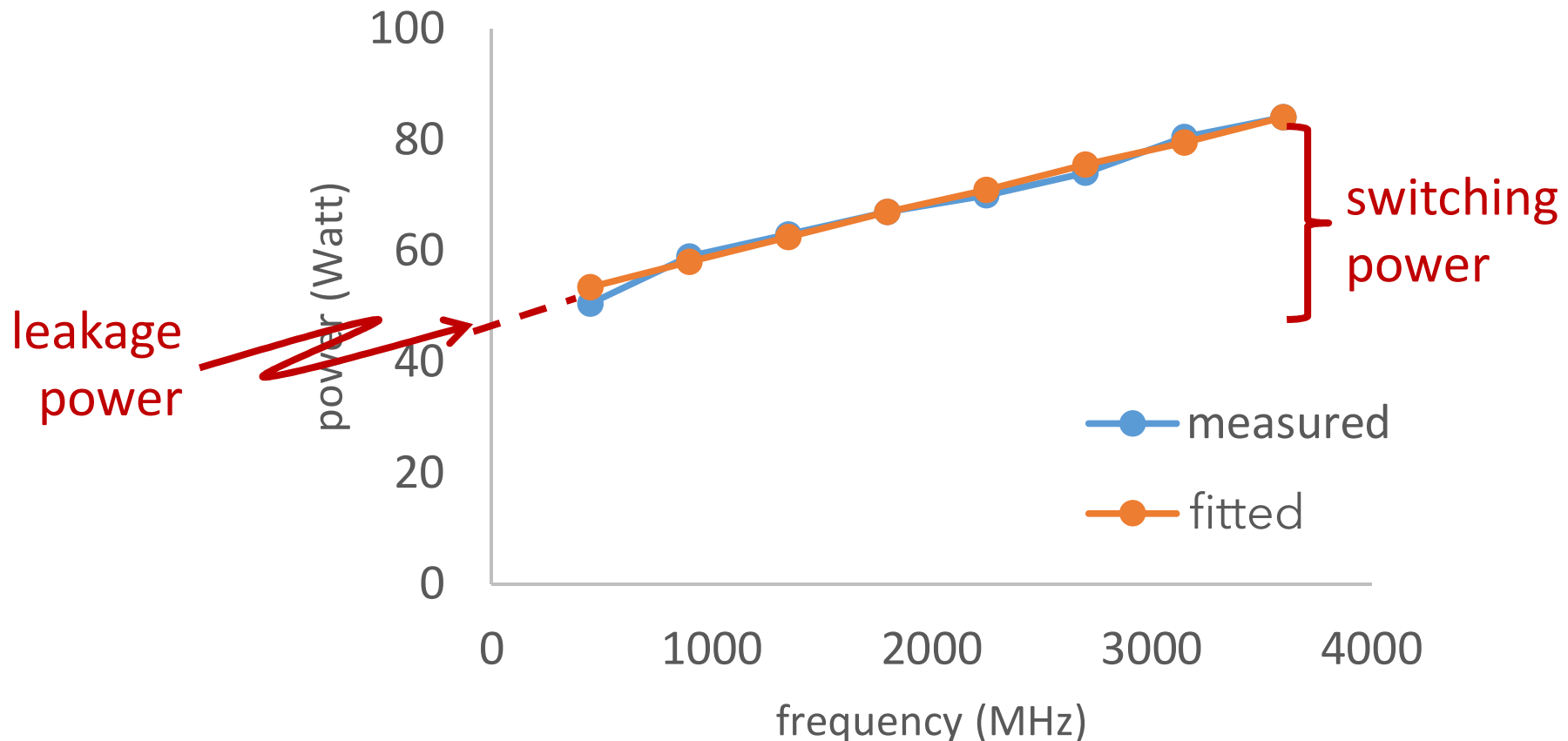
s.t. $k_{perf}' = k_{perf} S_{freq}$

- $T' = Work / (k_{perf} S_{freq})$
 - $1/S_{freq}$ longer runtime
- $P' = k_{switch} \cdot k_{perf} S_{freq} + k_{static}$
 - lower (switching) power due to longer runtime
- $E' = (k_{switch} + k_{static} / (k_{perf} S_{freq})) \cdot Work$
 - higher (leakage) energy due to longer runtime

Not such a good idea

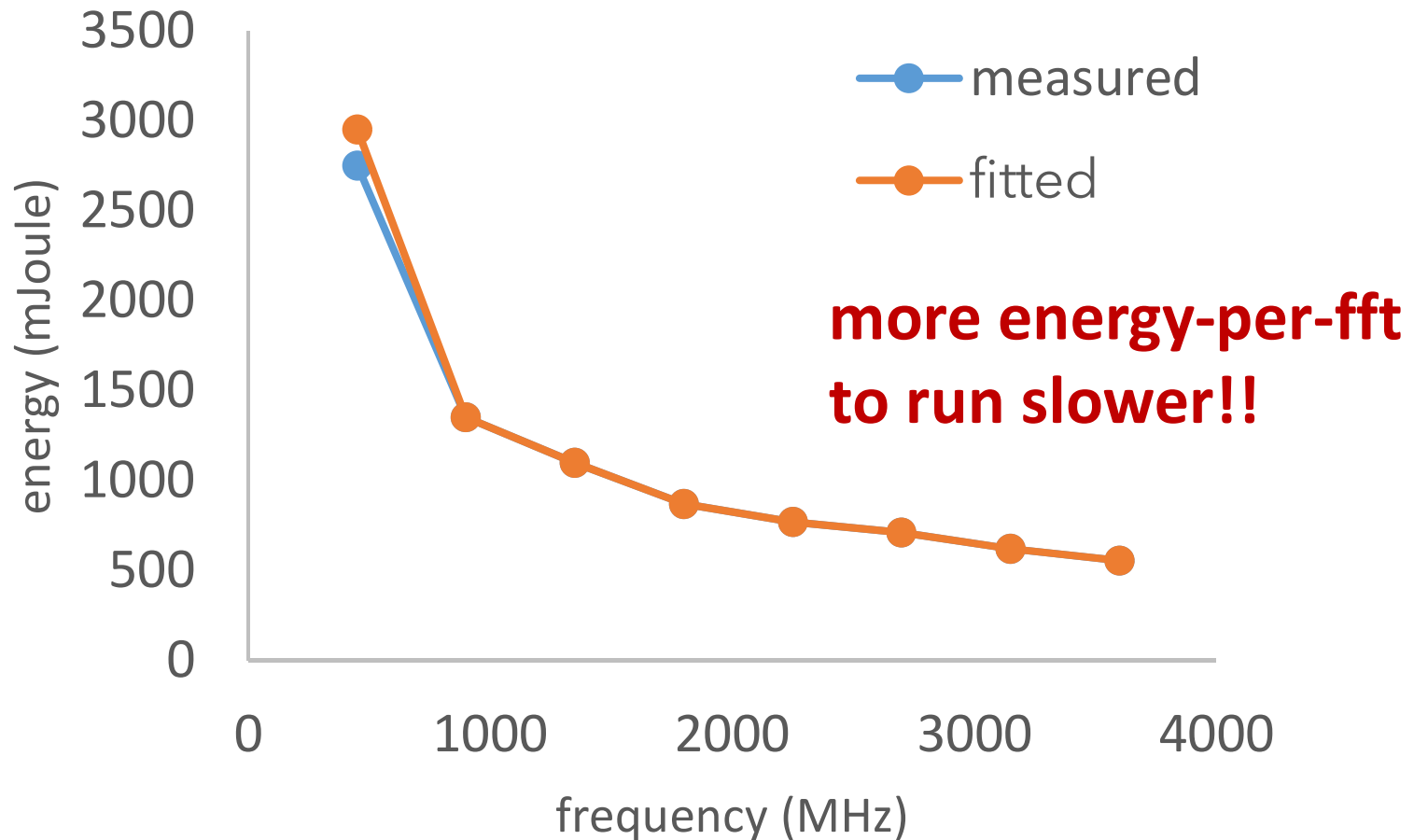
Intel P4 660 Frequency Scaling: FFT_{64K}

circa 2005, 90nm



18-447-S24-L12-S24, Jan $k_{perf}=145$ FFT64K/sec; $k_{switching}=0.24$ J/FFT64K; $k_{static}=49.4$ J/sec

Intel P4 660 Frequency Scaling: FFT_{64K}

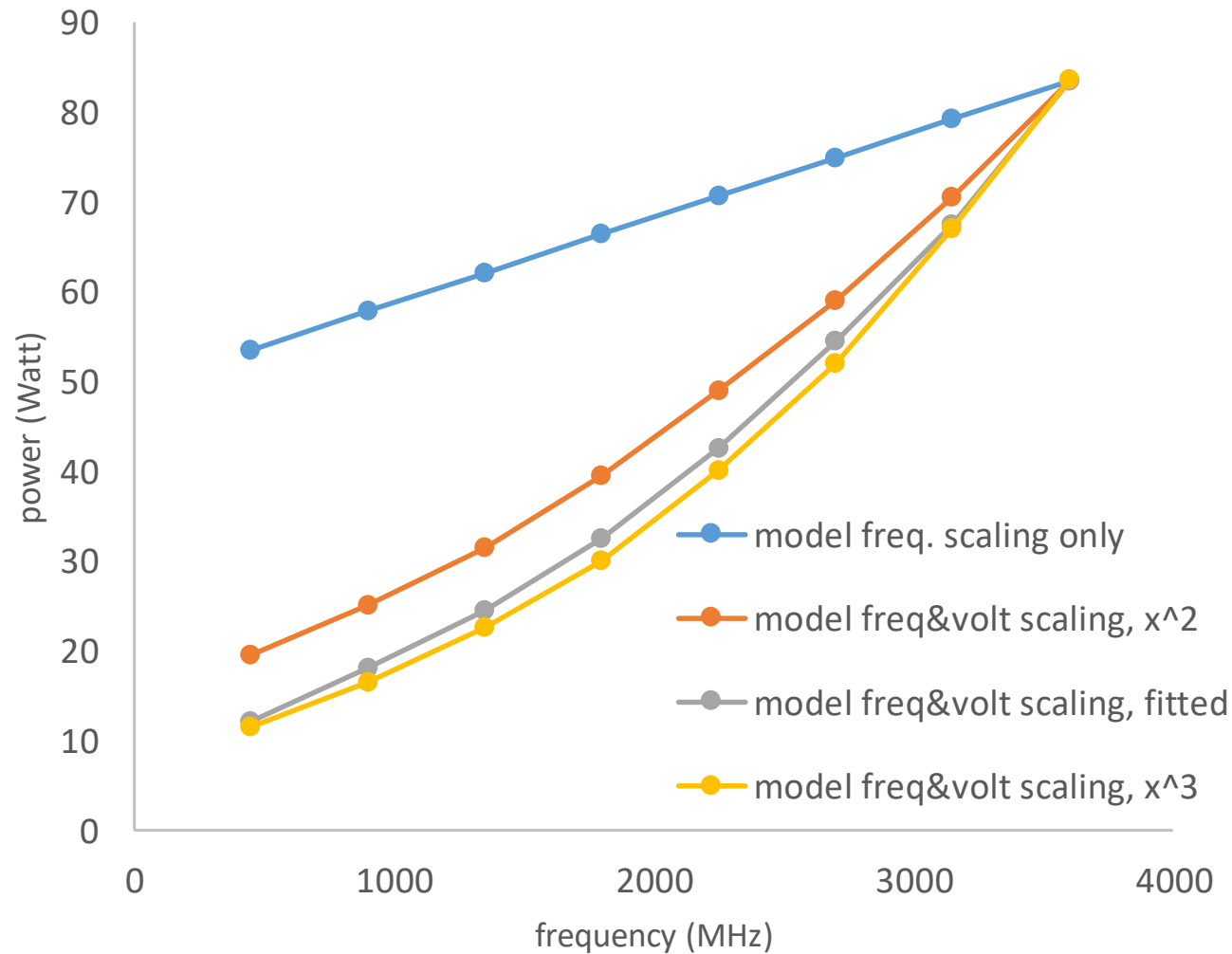


$$k_{perf} = 145 \text{ FFT64K/sec}; k_{switching} = 0.24 \text{ J/FFT64K}; k_{static} = 49.4 \text{ J/sec}$$

Frequency + Voltage Scaling

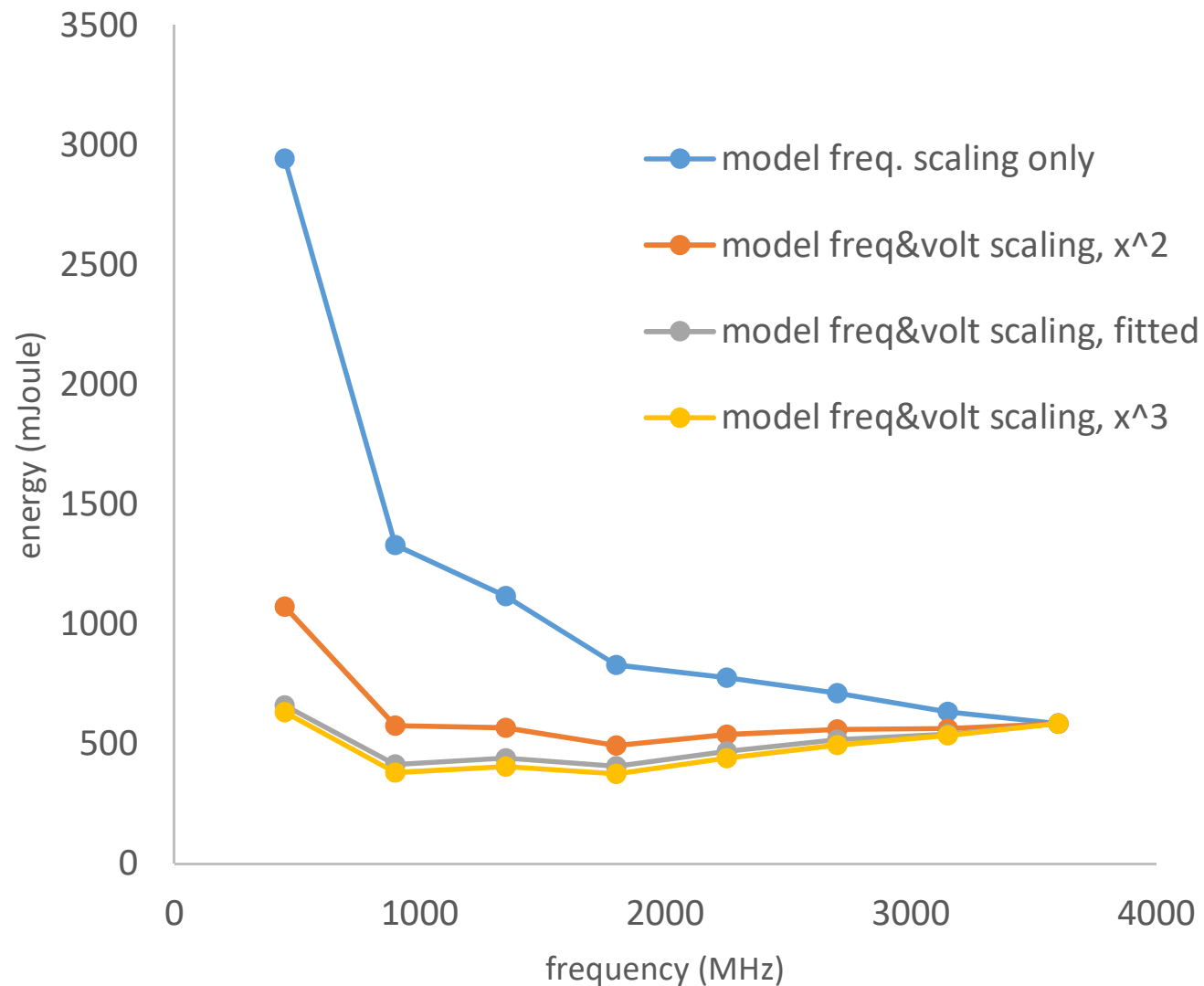
- Frequency scaling by S_{freq} allows supply voltage to be scaled by a corresponding factor $S_{voltage}$
- $E \propto V^2$ thus $k_{switch}'' = k_{switch} \cdot S_{voltage}^2$
- $k_{static}'' = k_{static} \cdot S_{voltage}^{2 \sim 3}$ \Leftarrow very gross approximation of something complicated
- $T'' = Work / (k_{perf} \cdot S_{freq})$
 - $1/S_{freq}$ longer runtime
- $E'' = (k_{switch} \cdot S_{voltage}^2 + k_{static} \cdot S_{voltage}^3 / k_{perf} \cdot S_{freq}) \cdot Work$
- $P'' = k_{switch} \cdot S_{voltage}^2 \cdot k_{perf} \cdot S_{freq} + k_{static} \cdot S_{voltage}^3$
 - superlinear reduction in power and energy to performance degradation

Intel P4 660 F+V Scaling: FFT_{64K}



circa 2005, 90nm

Intel P4 660 F+V Scaling: FFT_{64K}

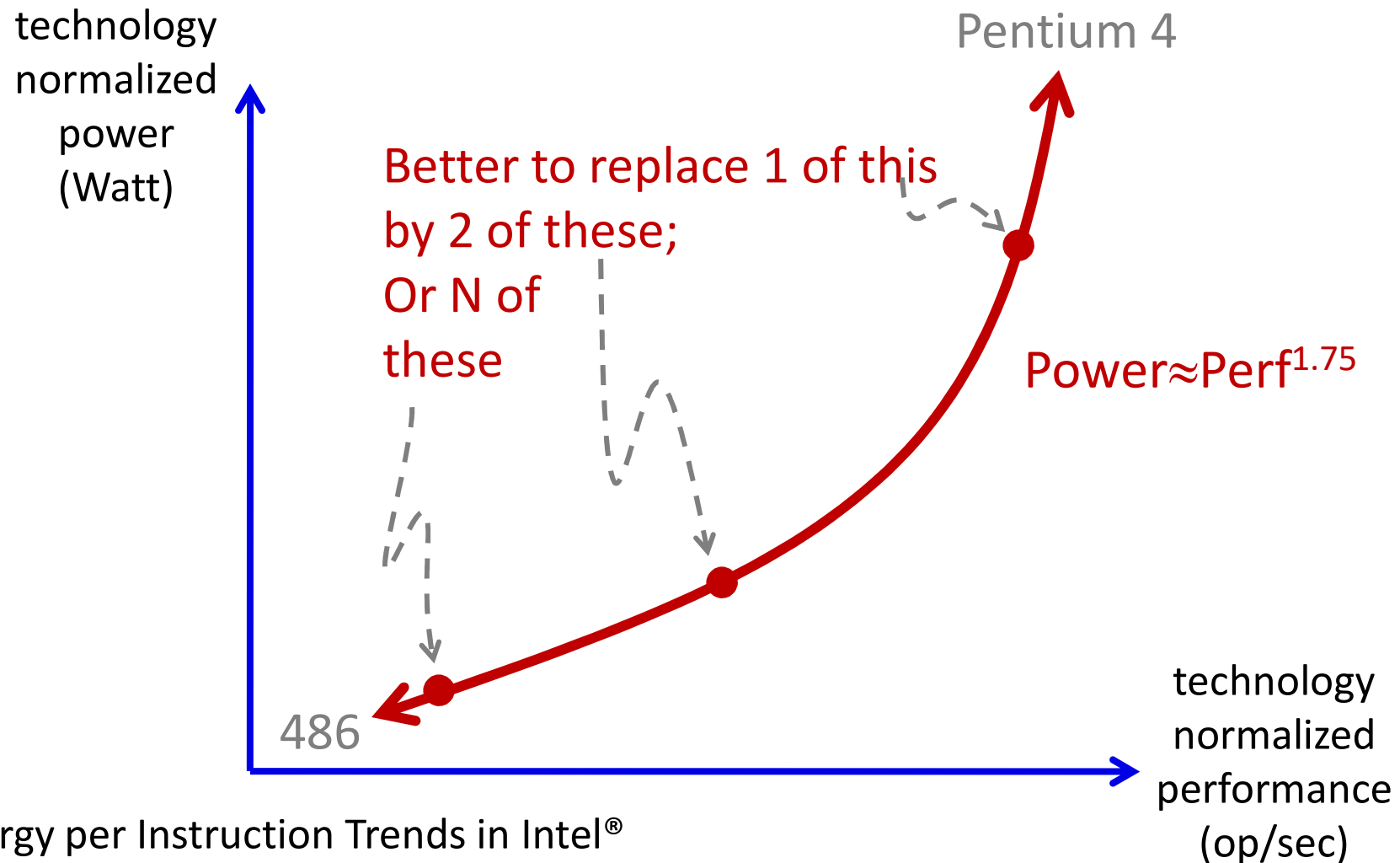


circa 2005, 90nm

Parallelization:

run faster at lower energy-per-op
by
running slower at lower energy-per-op

Cost of Performance in Power



[Energy per Instruction Trends in Intel®
Microprocessors, Grochowski et al., 2006]

Parallelization

- Ideal parallelization over N CPUs (to go fast)
 - $T = \text{Work} / (k_{\text{perf}} \cdot N)$
 - $E = (k_{\text{switch}} + k_{\text{static}} / k_{\text{perf}}) \cdot \text{Work}$
 N -times static power, but N -times faster runtime
 - $P = N (k_{\text{switch}} \cdot k_{\text{perf}} + k_{\text{static}})$
- Alternatively, forfeit speedup for power and energy reduction by $s_{\text{freq}} = 1/N$ (assume $s_{\text{voltage}} \approx s_{\text{freq}}$ below)
 - $T = \text{Work} / k_{\text{perf}}$
 - $E'' = (k_{\text{switch}} / N^2 + k_{\text{static}} / (k_{\text{perf}} N)) \cdot \text{Work}$
 - $P'' = k_{\text{switch}} \cdot k_{\text{perf}} / N^2 + k_{\text{static}} / N$
- Also works with using N slower-simpler CPUs

not true!

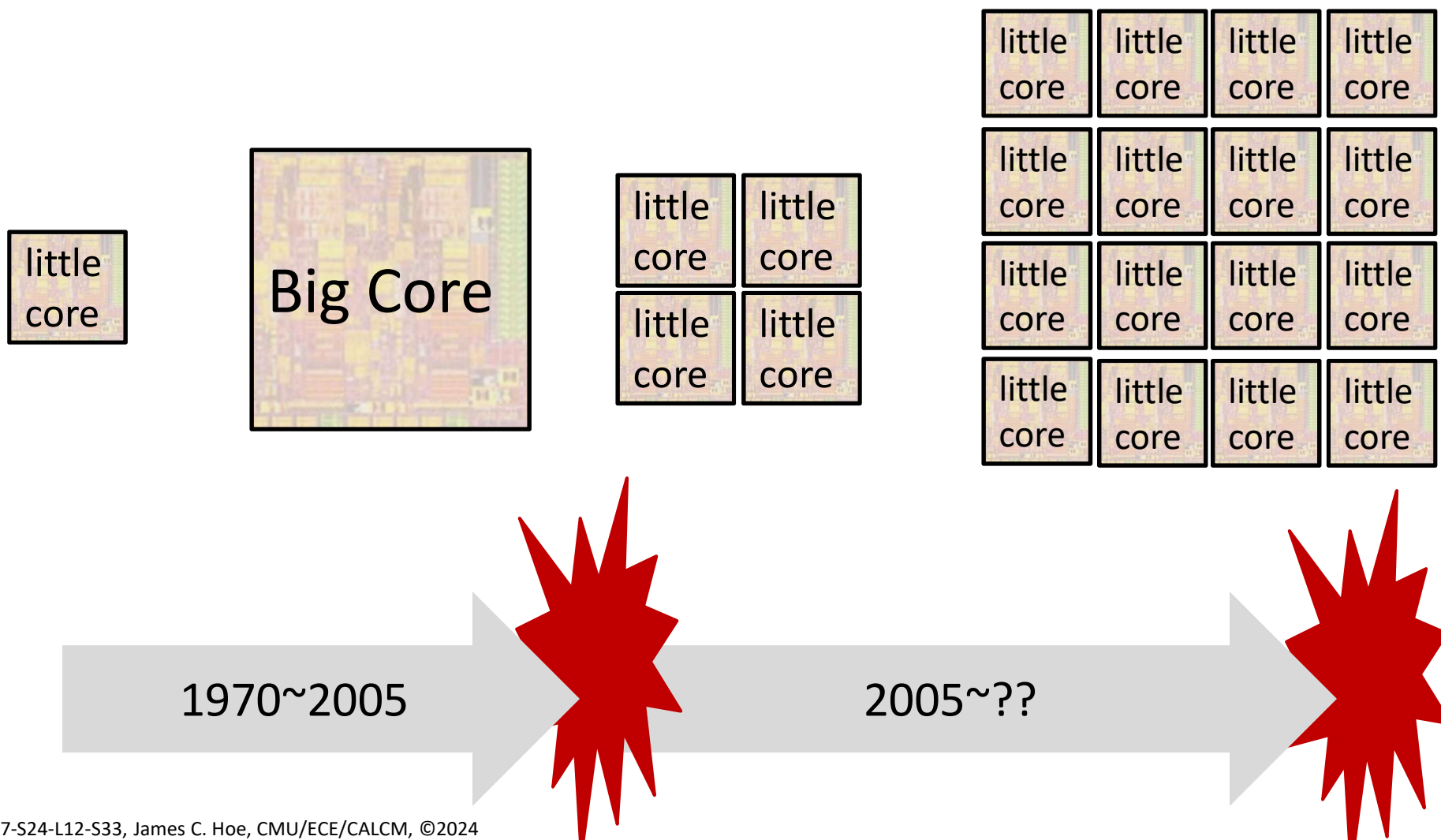


power \propto perf $^{\alpha}$ ($\alpha > 1$)

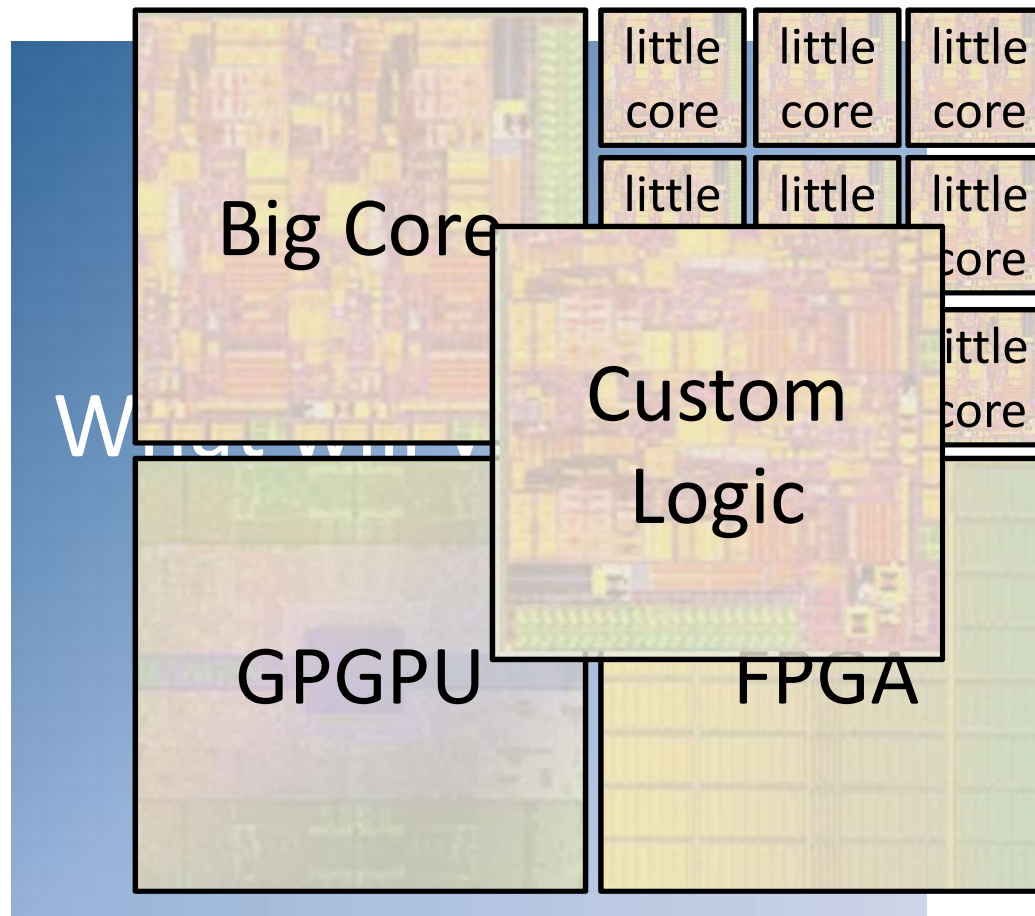
So what is the problem?

- “Easy” to pack more cores on a die to stay on Moore’s law for “aggregate” or “throughput” performance
 - How to use them?
 - life is good if your N units of work are N independent programs \Rightarrow just run them
 - what if your N units of work are N operations of the same program? \Rightarrow rewrite as parallel program
 - what if your N units of work are N sequentially dependent operations of the same program? \Rightarrow ??
- How many cores can you use up meaningfully?**

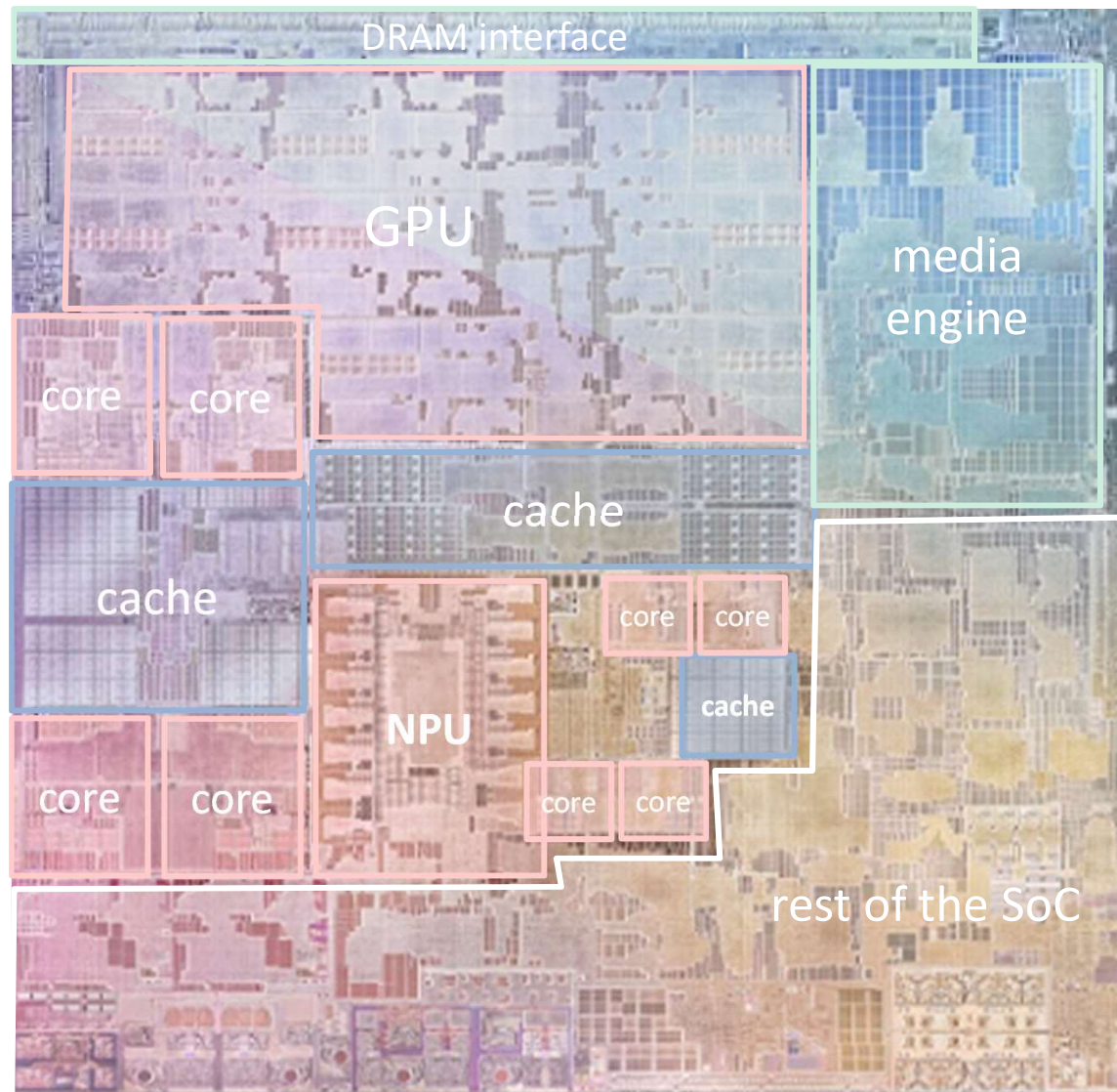
Moore's Law Scaling with Cores



Remember: it is all about **Perf/Watt** and **Ops/Joules**



Heterogenous System-on-Chip



[raw M1 die photo from apple.com]