Power and Energy

Dennard Scaling

- Power = aCFV^2

- Approximation for power usage by transistors

- Dennard noted that as transistors shrink (by ~30%)
  - Capacitance ~ Area – 50%
  - Voltage by 30% (V = field * length)
  - Delay by 30% (time = length / velocity)

- So area and power stay in proportion

Dennard Scaling Ends!

- What went wrong?
  - Power = Dynamic + Static
  - Dennard's scaling law was only dynamic power
  - So transistors constantly "leak"

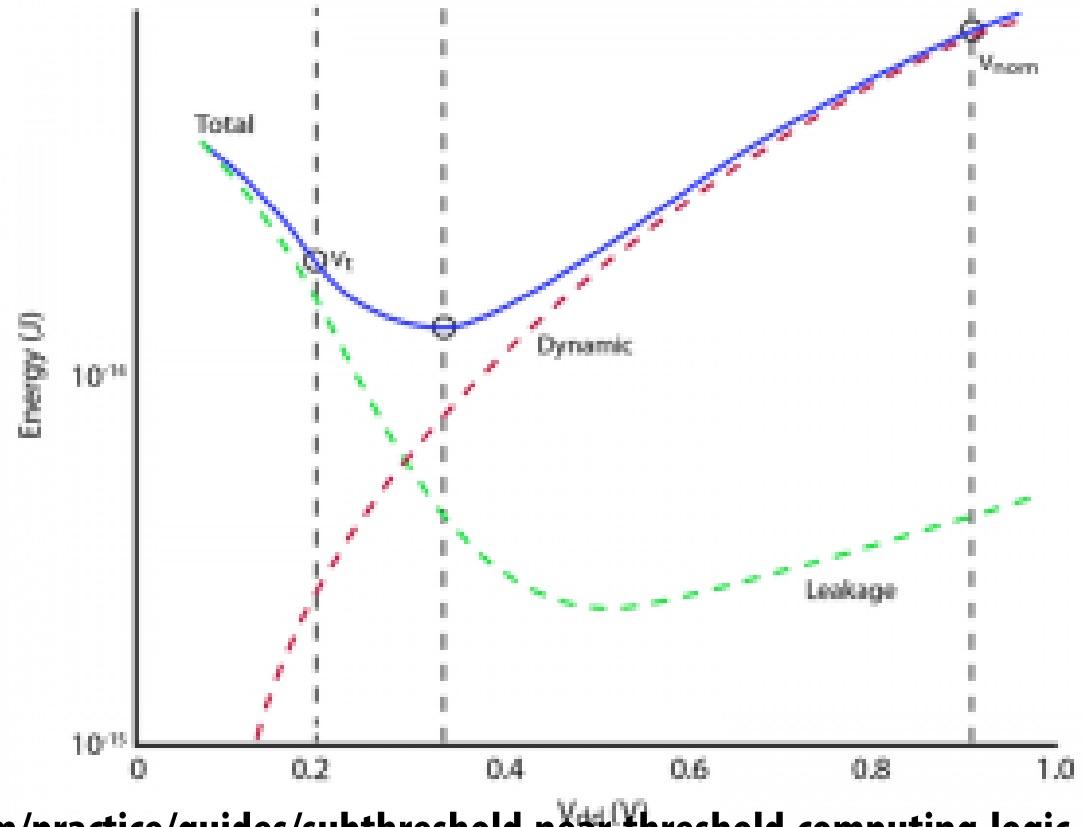
- Suddenly, architects cannot power all transistors nor continue to increase frequency

Less power

- Clock gating
  - OR clock signal with disable, so clock holds constant
- Power gating
  - Additional logic for the on-chip power wires
- Dark silicon?
  - Could we have circuits in which can never power everything?

Near Threshold Computing

- Transistors normally run at Vnom
  - Around 0.3, the transistor power is minimalized
  - But Vdd has to be >0.2 (Vt) on a normal transistor



https://www.techdesignforums.com/practice/guides/subthreshold-near-threshold-computing-logic-ntv/

Approaching the Threshold

- As supply voltage drops
  - Transistor frequency has increased variation
  - Transistors may become unreliable

- For example, an adder may have some errors
- A cache or memory could
  - Lower refresh rate
  - Least significant floating point bits are dropped

Approximate Computing

- Can a program benefit from computing near the threshold?
  - Yes!
  - Many algorithms are already approximations
  - Increasing the error tolerance can permit
    - Faster running time (less iterations)
    - Using error-prone storage or ALUs

- One example:
  - K-means clustering could introduce 5% error to save 50x energy

Intermittent Computing

- We want to deploy many devices into the wild
  - Simple sensors, etc
- Devices cannot be part of the electrical grid
  - Derive power from ambient conditions (light, sound, RF, etc)
- Device stores power into capacitor
  - When capacitor is full, start computing
  - When capacitor is nearly empty, save results to flash, NVM, etc

Making computation stop

- How do you design a processor that will be interruptible?

- How do you modify computation so that it can resume?