

---

Lecture 18:  
Power,  
Low Power Design

# Overview

---

## Reading

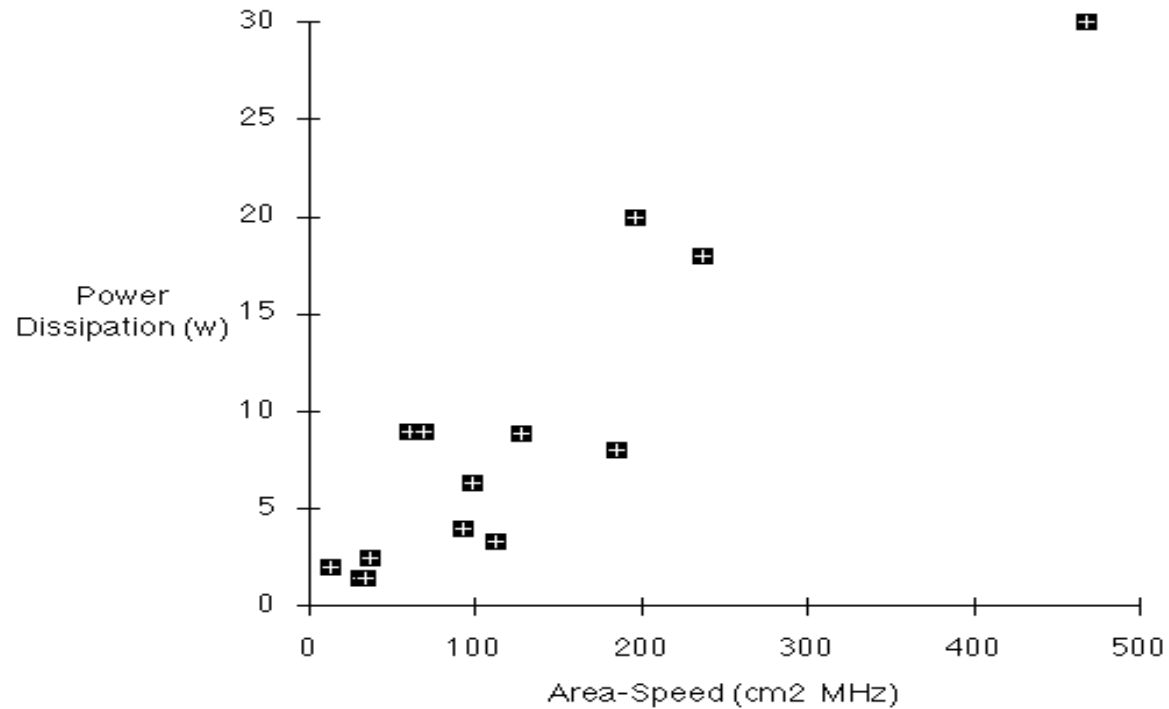
W&E 4.7.1 - 4.7.3 Power Dissipation

## Introduction

In addition to performance, power is becoming a significant concern for designers. CMOS was originally a low power technology, but it is not low power any more. CMOS chips can dissipate 50Watts. This power comes from charging and discharging capacitors and is fundamental to all circuits that drive wires. This lecture will look at power dissipation in CMOS circuits, and discuss various proposed methods for reducing the power. The key areas are technology, and algorithm. Both can make a huge difference in power.

# Power

Trend in CMOS power dissipation is a little frightening



Power seems to be proportional to area and frequency

# Power Issues

---

Three reasons that power is important:

1. Hard to get large current into a chip (Amps of current)

50W at 2.5V is 20Amps

2. Cheap device

Must use plastic package, in a low-cost box

No fans, so thermal resistance is high ( $35^{\circ}\text{C}/\text{Watt}$ )

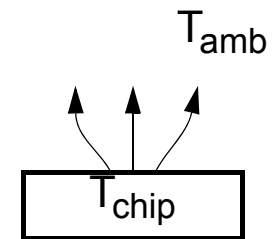
Power must be under 2W

3. For a portable system

Need to carry the energy (Power \* Time) you use

Energy is heavy (20 Wh / lb)

Lower power means less battery weight



# Power

---

Power is current times voltage –  $iV$

There are two types of current that need to be considered

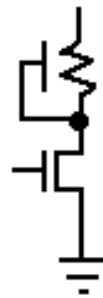
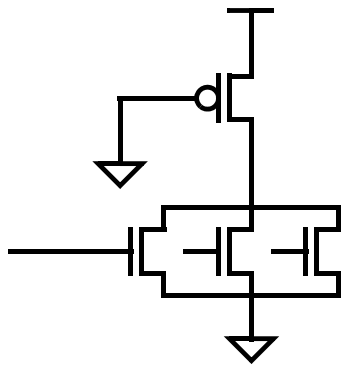
- Static
  - Current caused by a resistor between the supply
  - nMOS style circuits
  - Current easy to estimate,  $V/R$
- Dynamic
  - Current used to charge and discharge capacitors
  - Current depends on how often the capacitor changes state
  - Dominant current in CMOS chips (sometimes only current)

# Static Power

---

Current flows whenever the output is low

- Assume that the output is low 50% of the time.



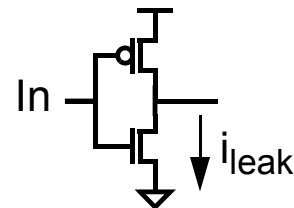
In a  $1\mu$  technology, the current is  $170\mu\text{A}/\mu$  of width for the pMOS. This means that the current for a 4:2 device would be around  $340\mu\text{A}$ , and the power would be  $1.7\text{mWatts}$  (on average) for the gate. 10K gates -- 17Watts

- Sometimes do this in CMOS to get large fanin gates.

Also occurs in analog like circuits

Memory, PLL,  $V_{bb}$  generators, etc.

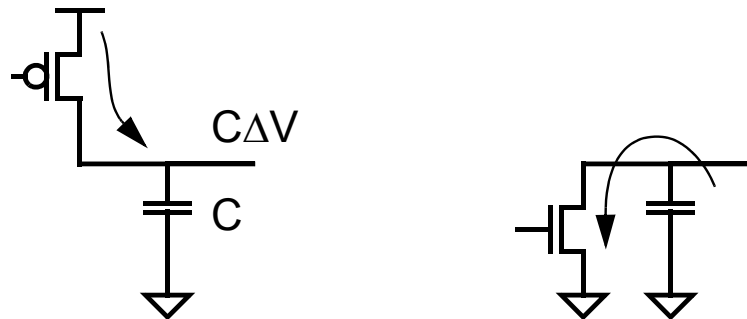
Leakage is another source of static power



# Dynamic Power

---

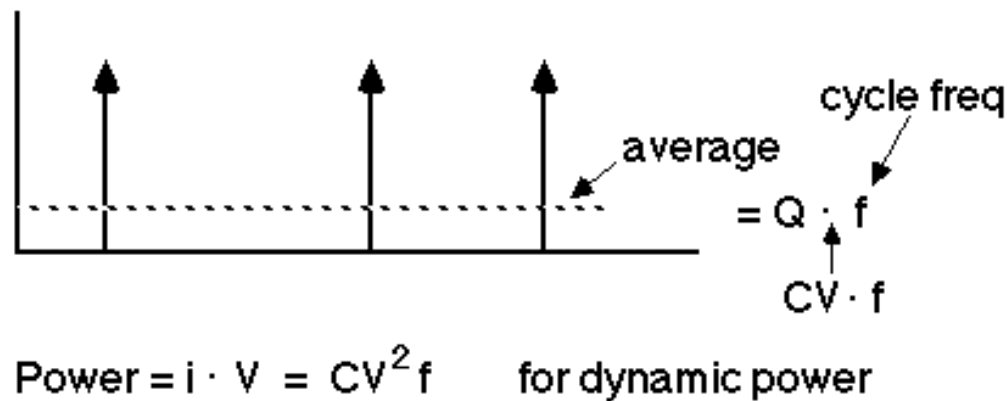
Takes current from the power supply to charge up the capacitor. When the capacitor is discharged the current does not get put back to the supply. The amplitude of the current spike when charging the capacitor depends on the resistance of the switch ( $i = V/R$ ). But the total charge required does not depend on the switch, since must be equal to  $C\Delta V$ .



# Dynamic Power

---

The average current that the 'capacitor' needs is just equal to the charge required to switch the capacitor time the number of times per second the capacitor switches. The latter term is just the frequency that the line is toggling at.



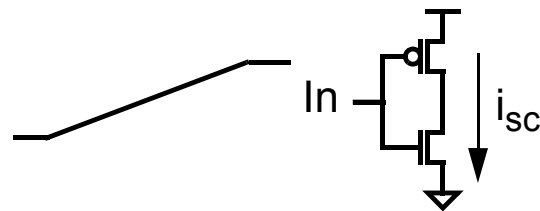
This is a fundamental power that is needed to switch capacitors (without inductors).



# Dynamic (Short Circuit) Power

---

Short Circuit current occurs when the input of a gate is in transition and both the P-channel and N-channel devices are conducting at the same time. (see W&E 4.7.3 for derivation)



In general the percentage of the total dynamic power due to the short circuit current is much smaller than that used to charge and discharge the capacitive loads. Very slow rise and fall times on the inputs could however make this current significant, and must be considered for gates at the end of long wires with large RC delays. In a well designed circuit, this current is small. To first order it can be ignored.

# Dynamic Power

---

Total Power

$$P = \sum_n \frac{1}{2} C_n \Delta V V_{dd} \alpha_n F$$

'n' ranges over all nodes in the circuit

$\alpha_n$  = average number of times that node n transitions/cycle<sup>1</sup>

Generally the signal swing is equal to V<sub>dd</sub>

$$P = C_{eff} V^2 F$$

Dominate power in CMOS circuits

- Static power is usually pretty small
- If the circuit is idle most of the time, the static power can still be important, since it uses energy whether the circuit is doing anything or not.

---

1. The 1/2 in the formula is because signal must make two transitions (up and down) for a cycle

# Low Power Design

---

Need to:

- Understand the basic trade-offs
- Make low power a design objective
  - Along with design time, and performance
- Don't be stupid

Find smart ways to reduce power

- Slower is usually lower power
  - $CV^2F$ , lower F lower power
- Need some metrics for low power designs
  - Provides a way to compare designs
  - Which design / style is better
  - What should a designer expect

# Metrics - Power

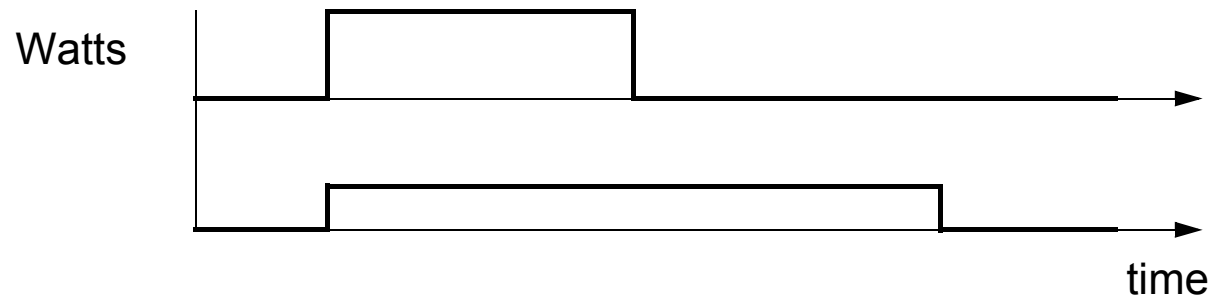
---

Obvious choice

- Sets battery life in hours, packaging limits

Problem

- Dynamic power proportional to  $F$



Often want to do more, not have it take longer

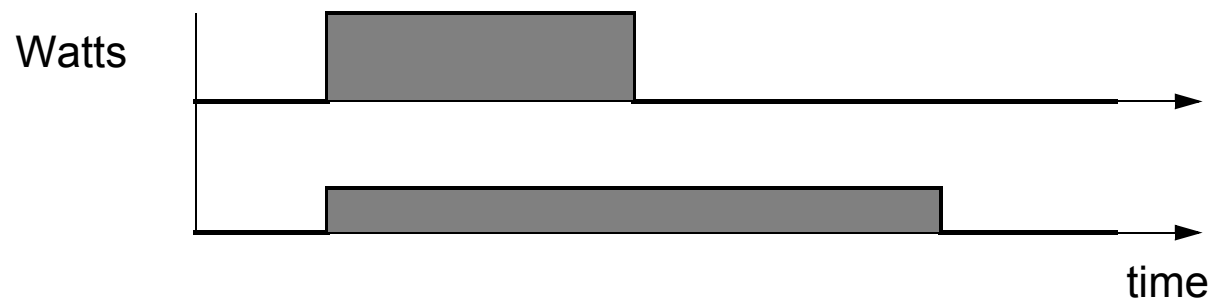
- Comparing the power of two designs can be misleading
- Lower power design could simply be slower

## Metric - Energy / Operation

---

Rather than look at power, look at the total energy needed to complete some operation. Fixes obvious problems with the energy metric, since changing the operating frequency does not change the answer

$$\text{Energy/Op} = \text{Power} * \text{Delay/Op} = kC\Delta VV_{dd}$$



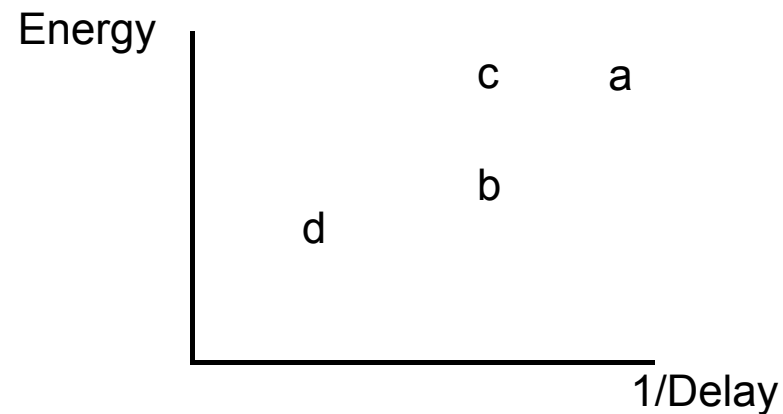
The energy is the area under the curve. While this metric looks promising, the problem is that one can decrease the energy/op by doing stuff that will slow down the chip -- like lowering the supply voltage, or using small transistors.

## Metric - Energy/Op and Delay/Op

---

Since lower energy solutions might simply be lower performance, we need to have a metric that includes both energy and performance.

One solution is to think about a 2-D solution space:



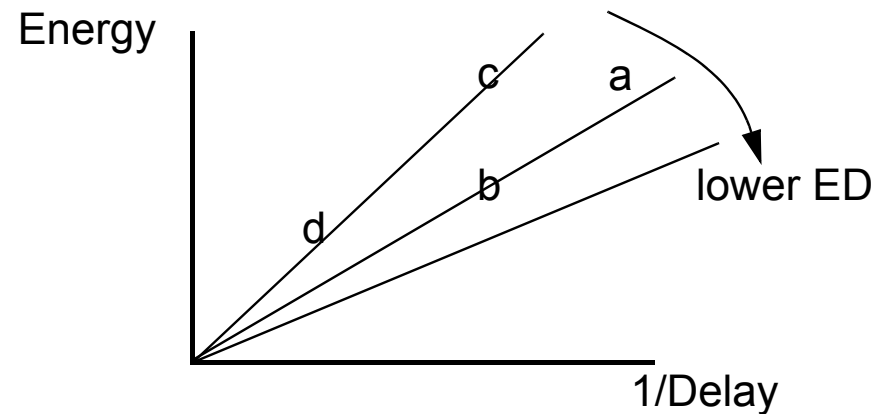
Clearly 'b' is lower power than 'c'. But can we say anything about 'b' and 'd'? While 'd' is slightly lower power, it is also much slower.

# Proposal: Energy/Op \* Delay/Op

---

$$\text{Energy} * \text{delay} = \text{Power} * (\text{Delay}/\text{Op})^2$$

Constant energy-delay products are straight-lines in the graph:



Implies one can trade lower energy/op for increased delay

- Try to show you why this is true

# Voltage Scaling

---

Changing power supply voltage:

- Has a large effect on power, since  $P = CV^2F$
- Affects performance too

$$t_d = \frac{CV}{k(V - V_{th})^\alpha}$$

Energy \* Delay is:

$$ED = \gamma \frac{V^3}{(V - V_{th})^\alpha}$$

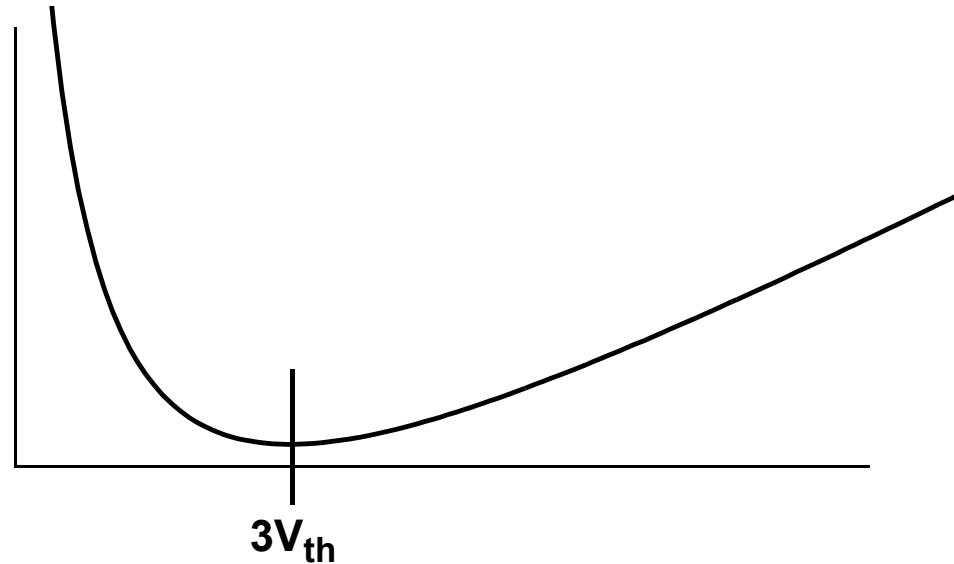
( $\alpha$  is between 1-2 and models velocity saturation)



# Voltage Scaling

---

For quadratic device it is often drawn as:

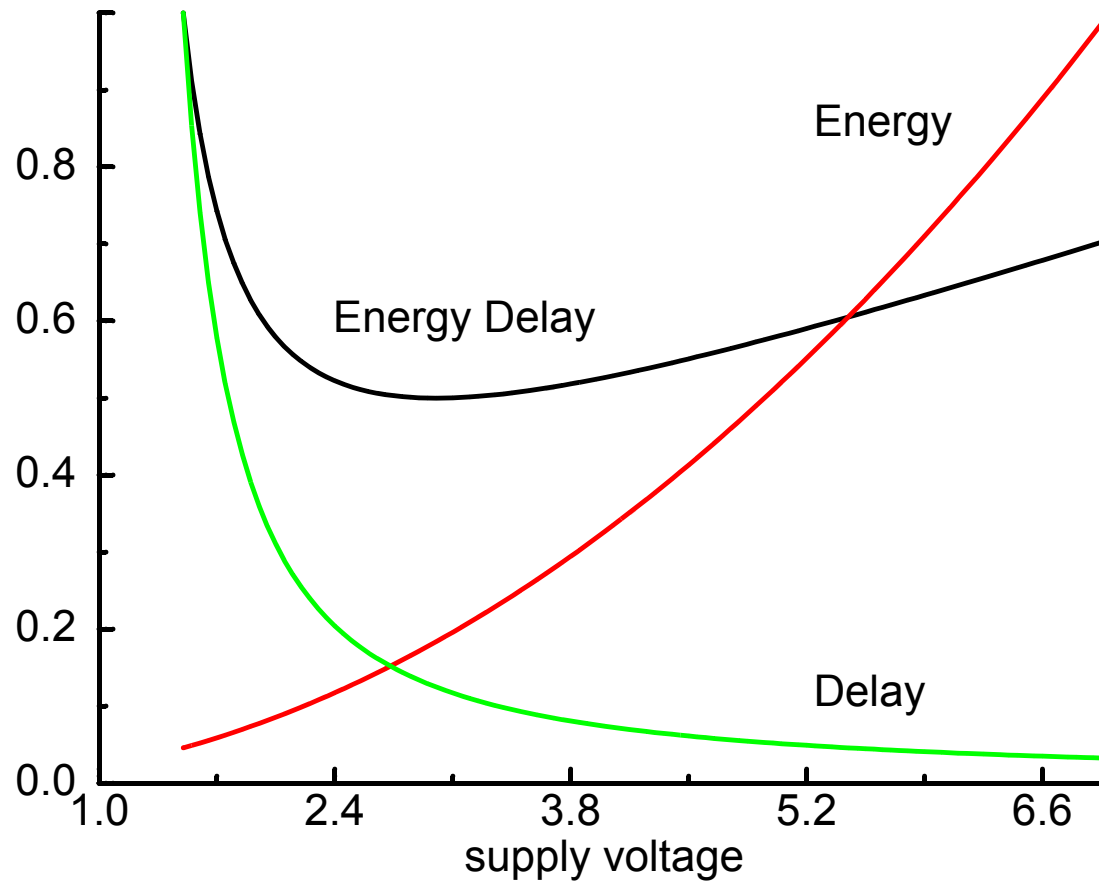


Looks like this is an important optimal point

- But you should be cautious:  
No labels on the graph

# Energy-Delay for Quadratic Devices

---



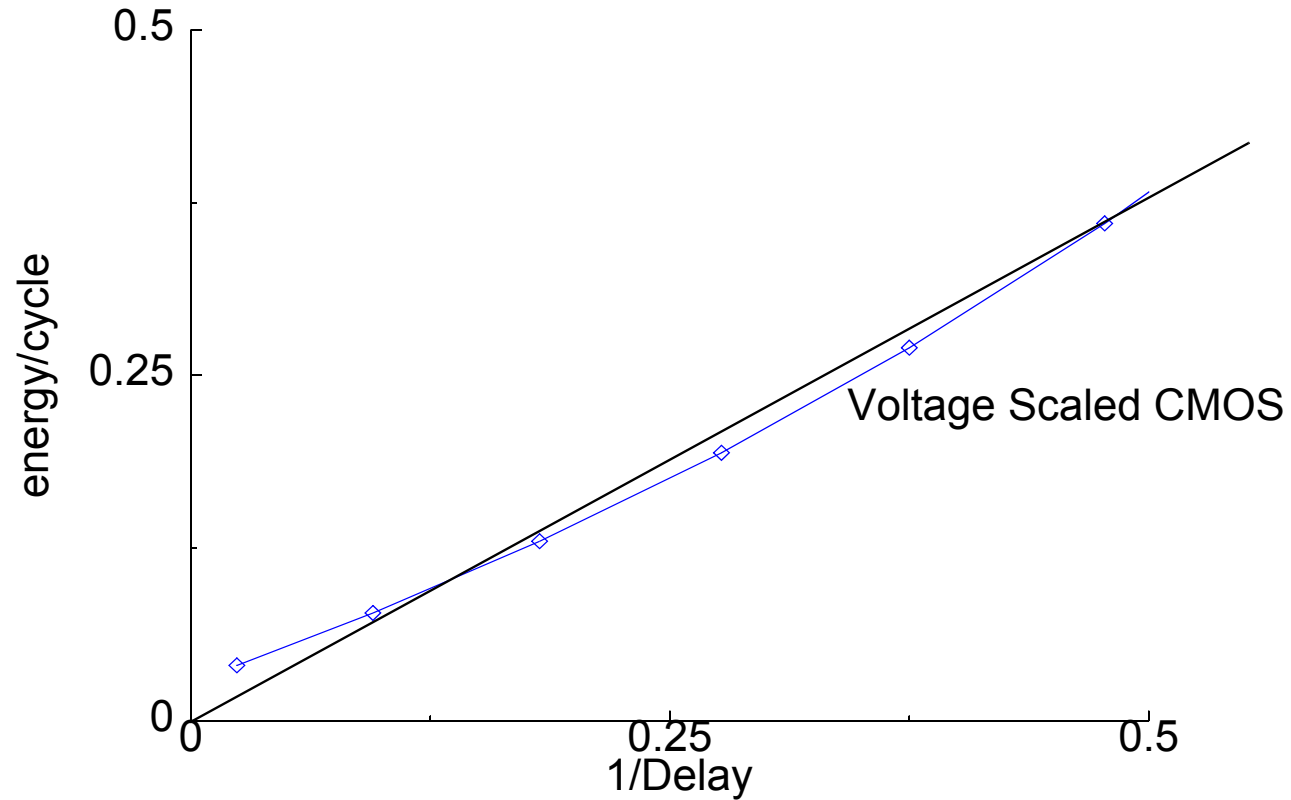
Energy-delay is pretty flat.

---

## Another View of Voltage Scaling

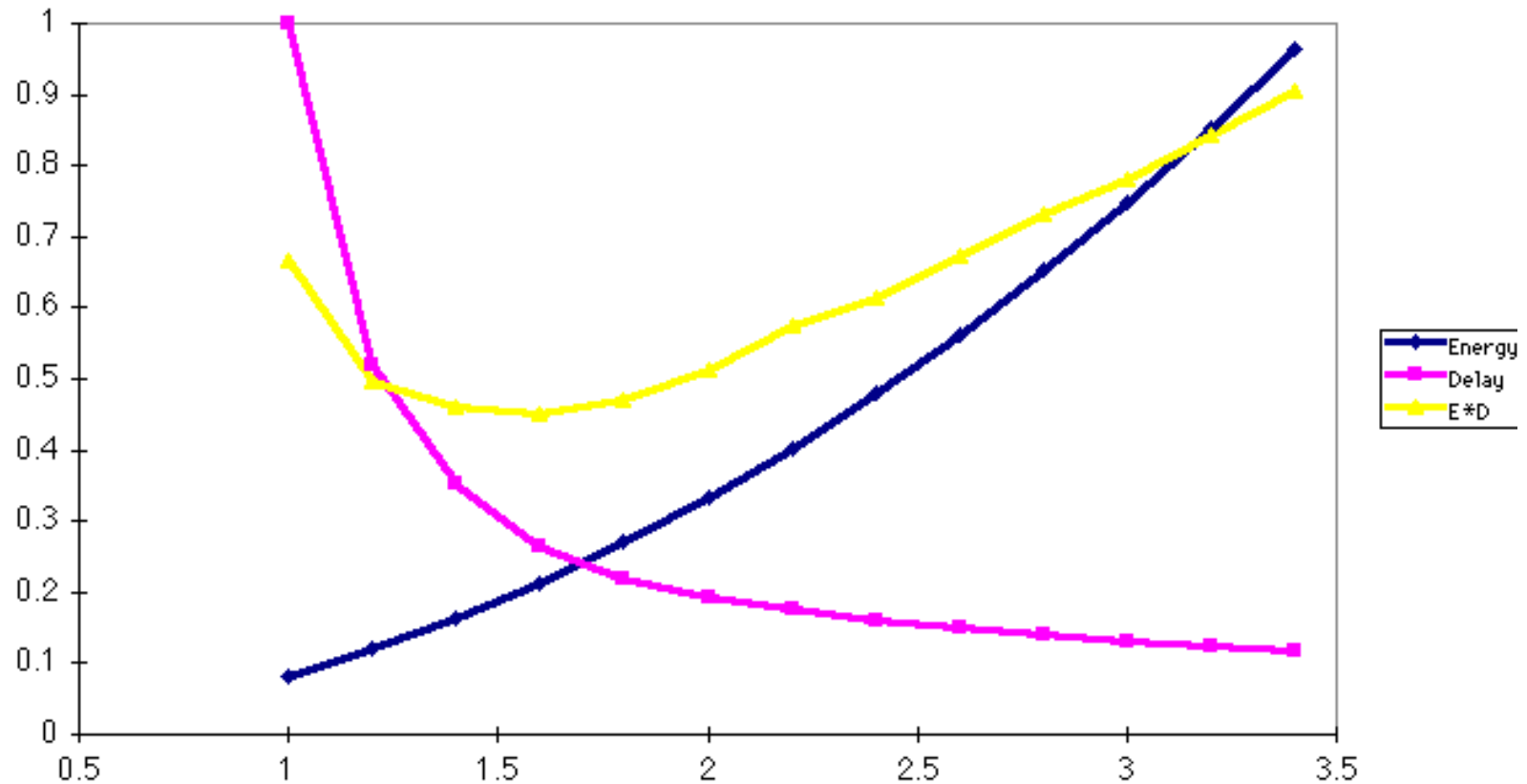
---

Voltage scaling is pretty straight on the Energy 1/Delay plot



Plots energy of quadratic inverter, versus its performance (1/delay)

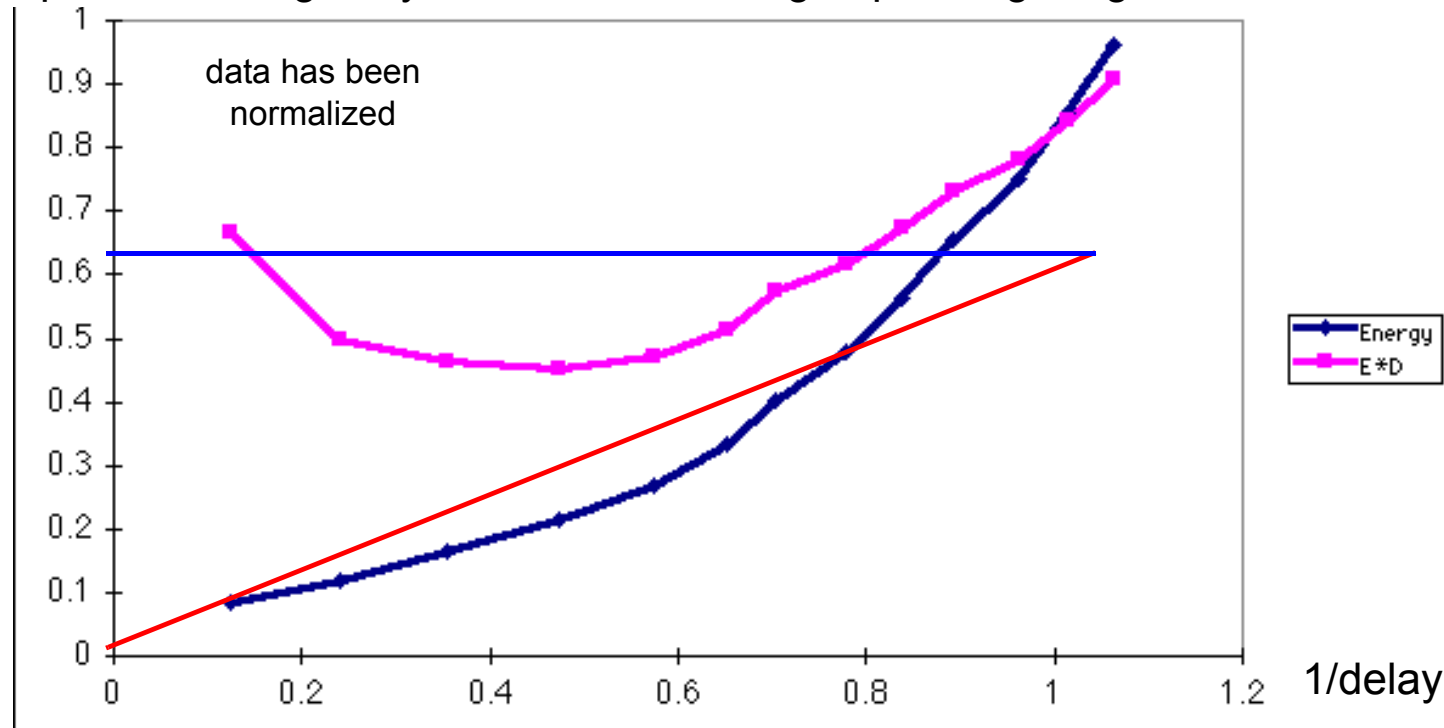
# Energy Delay for 'Real' Devices



Data from a real 0.35 $\mu$  technology.

# Voltage Scaling of a Real Device

E\*D product changes by +/- 30% over voltage operating range



With modern technologies, max operating voltages is not very energy efficient

- Reason modern processors are using aggressively scaled supplies

# Transistor Sizing

---

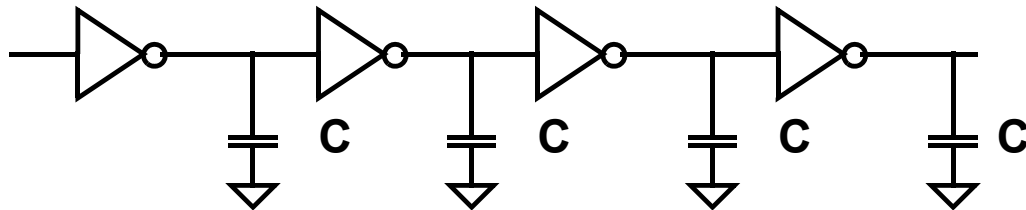
Using smaller transistors is another method of reducing the power

- Power decreases since the low capacitance decreases
- Delay increases since the driving resistance increases

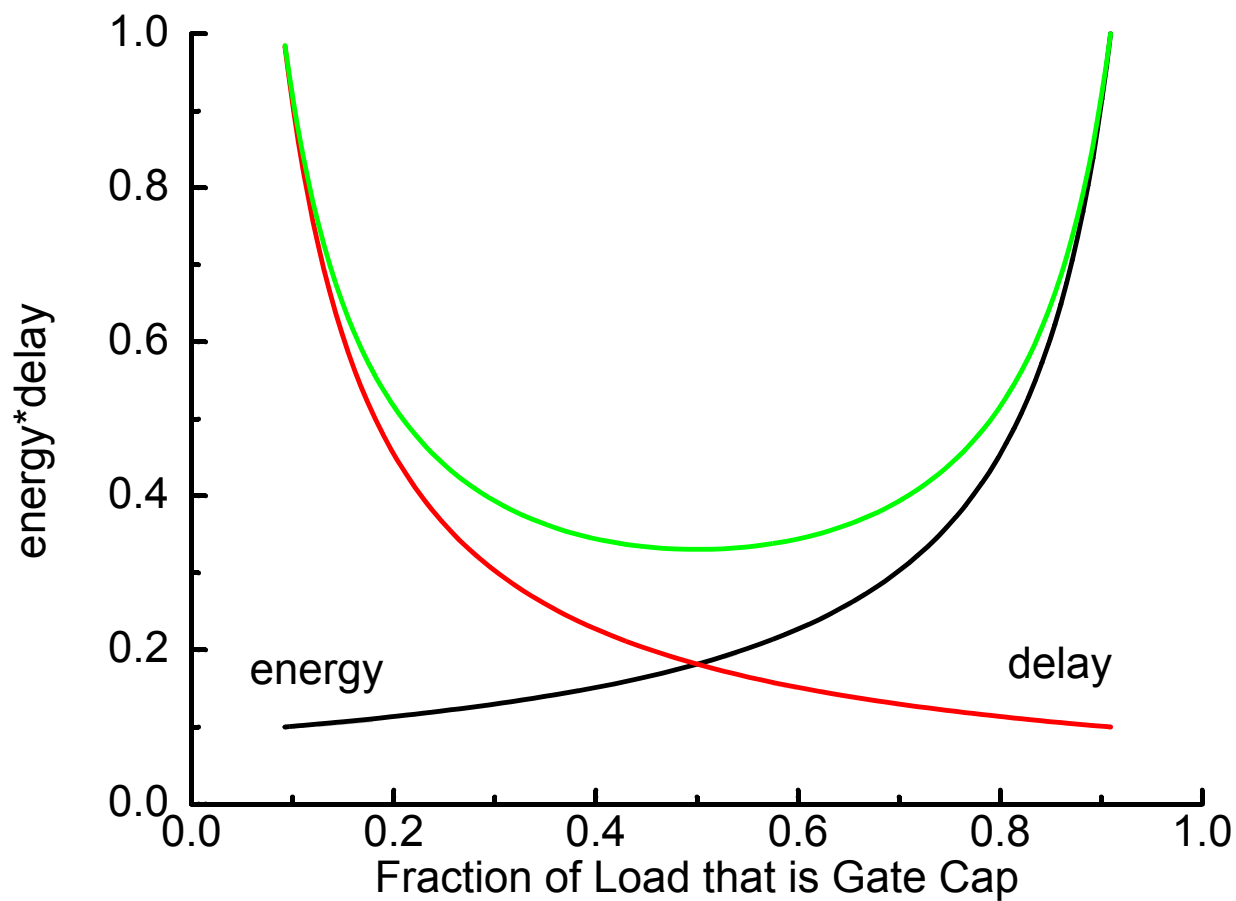
Modeling is a little difficult, since not all path are critical

- Making transistors on non-critical paths smaller is good -- lower power and no delay change

Simple example of critical path changes



## Sizing Energy Delay



Over a reasonable set of load ratios, the energy-delay is pretty constant

# Energy Delay

---

Good metric

- Models intrinsic trade-off of technology
- Flat for a factor of 10 in voltage scaling
- Flat for a factor of 4 in transistor sizing

Implies

- Optimized Power is proportional to  $(1/\text{Delay})^2$
- 1/2 Speed => Energy/Op = 1/2, Ops/sec = 1/2
- Slower peak performance will be lower Energy/Op  
MIPS/Watt should be smaller for slower processors
- Performance is expensive!



# Lower Energy Delay

---

Some techniques:

- Better fabrication technology
  - Helps both performance and energy
  - Mostly need to normalize it out
- Dynamic Power Reduction
  - Selective Activation
  - Swing Reduction
- Problem reformulation
  - This is where the greatest leverage is
  - Redefine / code problem to reduce energy/delay

# Technology

---

Two kinds of technology scaling:

- Constant voltage (what we used to do)
- Scaled voltage (what we seem to be doing now)

Constant Voltage:

- $L = \alpha L$ ;  $V = V$   
C falls as  $\alpha^1$
- Energy/Op =  $k_1 CV^2$  falls as  $\alpha$
- Delay =  $CV/i$   
falls as  $\alpha$  to  $\alpha^2$  depending on whether the transistors are velocity saturated
- Energy delay falls as  $\alpha^2$  to  $\alpha^3$

---

1.  $C/\mu\text{m}$  remains the same, but the wires get shorter. This is true even with fringing capacitance

# Ideal Scaling

---

Ideal scaling:

- $L = \alpha L$ ;  $V = \alpha V$
- Energy/Op =  $k_1 CV^2$   
falls as  $\alpha^3$
- Delay =  $CV/i$   
falls as  $\alpha$
- Energy delay falls as  $\alpha^4$

Better than constant voltage scaling ...

## Extra factor of $\alpha$

---

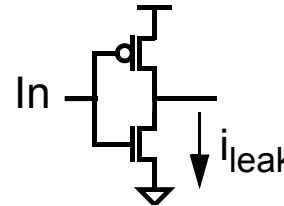
Optimal Energy-Delay product

$$= k V_{th}$$

Scaling  $V_{th}$  reduces dynamic power

Minimum  $V_{th}$  set by static leakage power

- When you turn a transistor off, there is still some leakage
- Leakage current is exponential on  $(V_{th} - V_{gs})$
- Since minimum of  $V_{gs} = 0$ , need some  $V_{th}$  to turn devices off



$$I_{ds} \sim \frac{W}{L} I_s \times \exp\left(\frac{V_{gs} - V_{th}}{av_T}\right)$$

# Dynamic Power Reduction

---

Try to prevent useless transitions

- Use selective activation

If block is not being used, turn off the clock

This will force the latches to hold their values

Most of the signals will not change

- Try to reduce transitions on large capacitance nodes

Prevent glitches when possible

But this is mostly small changes, a factor or two if you are lucky

# Big Wins

---

The major way to reduce power is to rethink about the problem at the high level. Like most things, this is where the leverage is. There are two general techniques that will help with power:

Use parallelism

- Improve delay, and thus energy \* delay
- Use voltage scaling / transistor sizing  
    convert excess speed to low power

Reformulate the problem

- Reduce required computation
- Improves energy/op and delay/op

# Parallelism

---

If the problem can be solved in a parallel fashion



- $\text{Energy/Op} = 1 + \Delta_e$   
Have a small amount of overhead in distributing the operands
- $\text{Delay/Op} = 1/n + \Delta_d$   
Again small overhead in making the solution parallel
- $\text{Energy delay} = 1/n + \text{overhead}$

# Problem Reformulation

---

Assume operation needs 100 instructions

- Energy = 100  $E_{inst}$
- Delay = 100  $T_{inst}$

If another algorithm only needs 50 instructions

- Energy delay = 1/4 Old solution
- Could use a slower, lower power processor

Works for hardware too

- Remove multiplies, need for FP, etc.
- Reduce the energy needed for operation, and time required



# Low Power Design

---

Good design has always meant careful trade-offs

- Used to worry about performance, area, and design time
- Need to consider energy too

Energy delay metric is helpful in making these trade-offs

- Reminds about the trade-off between performance and power
- High speed techniques good for low power

Parallelism very helpful

Leverage is at the top and bottom

- Better technologies have much better energy-delay products

Same performance at  $\alpha^2$ - $\alpha^4$  the power

- Key is to think about the system-level problem

Here one can make order of magnitude changes

