Slide Set 13

Power

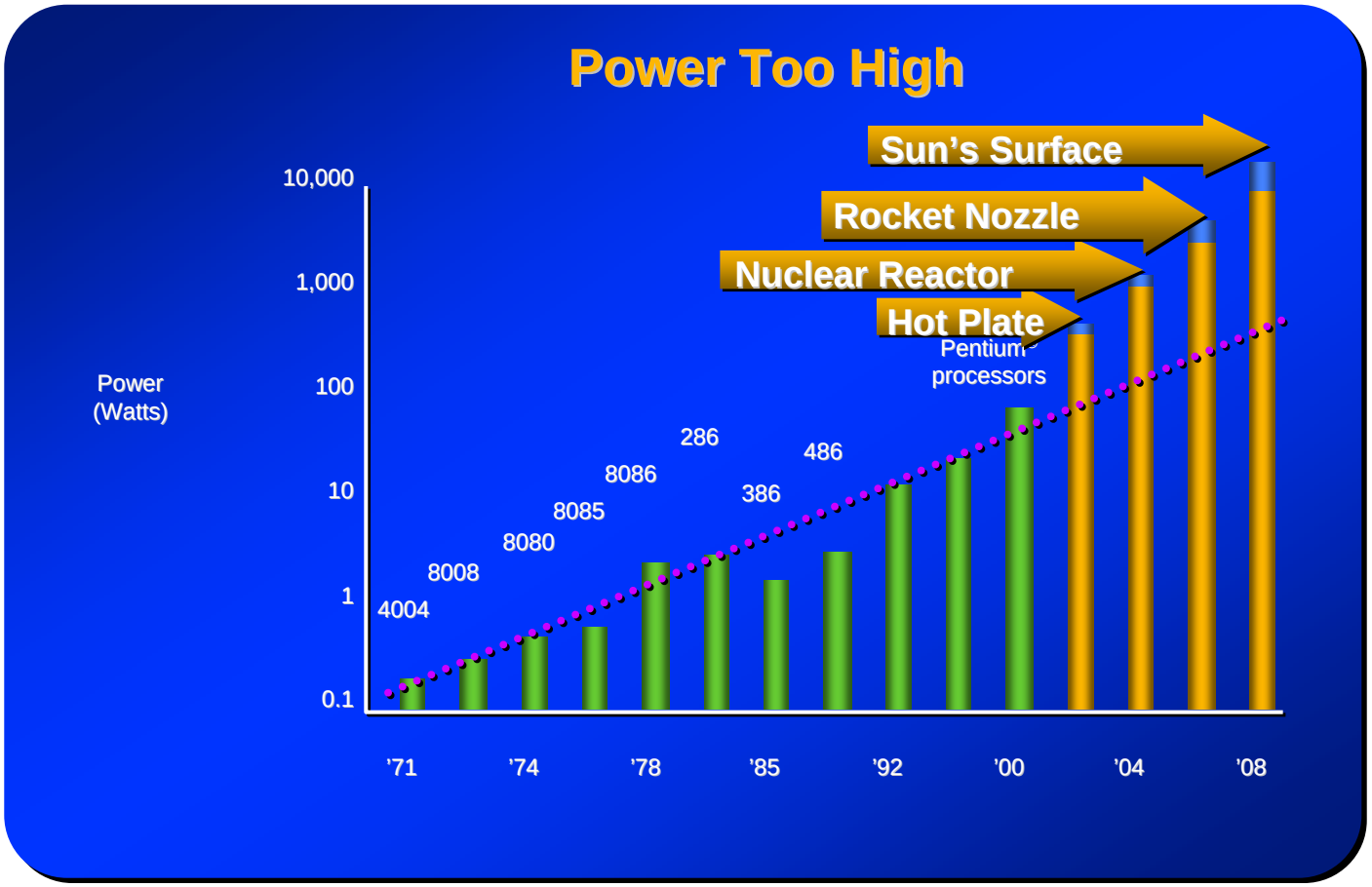
Steve Wilton
Dept. of ECE
University of British Columbia
stevew@ece.ubc.ca

Overview

Wolf 3.6, 4.6, 5.5, 8.5

So far we have talked about how to optimize circuits for area and delay, but we have neglected one important design goal: power. In the past, power has always been a secondary concern, but not today. CMOS was originally a low power technology, but it is not low power any more. CMOS chips can dissipate >100Watts. This power comes mostly 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.

Who Cares about Power?



Source: Intel

Who Cares about Power?

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. Cheaper device
   - Must use plastic package, in a low-cost box
   - Power must be under 2W

3. Important in portable systems
   - Need to carry the energy (Power * Time) you use
   - Energy is heavy (20 Wh / lb)
   - Lower power means less battery weight

This Slide Set:

1. What is Power
2. How to Estimate Dynamic Power of a Circuit
3. Design for Low Power

What is Power?

What is Power?

Power is current times voltage – I x V

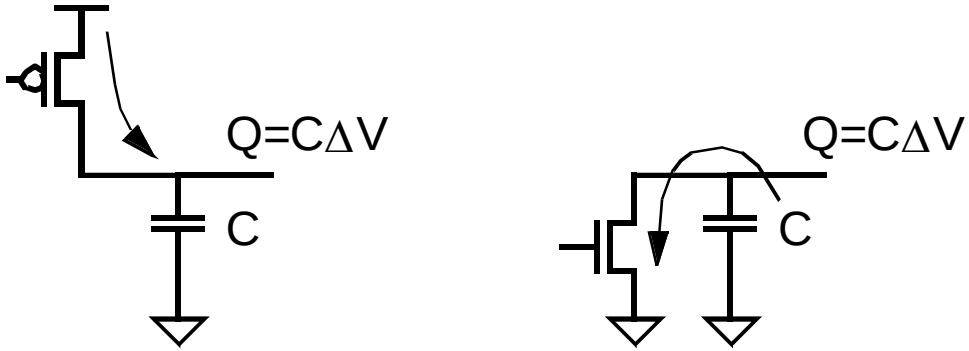
There are two types of current that need to be considered

- Static (current flowing from Vdd to Gnd)
  - In CMOS, this was relatively small in the past and due primarily to junction leakage current
  - Pseudo-NMOS style circuits will dissipate static power when output is low
  - Regular static CMOS now has leakage currents in the form of subthreshold current which is getting larger

- Dynamic (charging current)
  - Current used to charge and discharge capacitors
  - Current depends on how often the capacitor changes state
  - Dominant current in CMOS chips

Dynamic (Switching) Power

It takes current from the power supply to charge up the capacitor (parasitic capacitance). When the capacitor is discharged the current does not get put back to the supply. It goes to ground, which dissipates power. 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 it must be equal to CΔV.



Therefore, **total dynamic power depends on amount of capacitance that is switched**.

Dynamic (Switching) Power

Dynamic Power for one transition:

$$P_{dynamic} = I * V = C * V^2$$

If the frequency is f, then each wire might change as much as f times per second:

$$P_{dynamic} = f * C * V^2$$

But, of course, not each wire changes every clock cycle

$$P_{dynamic} = \alpha * f * C * V^2$$

"Activity" of a wire.

Definition of α

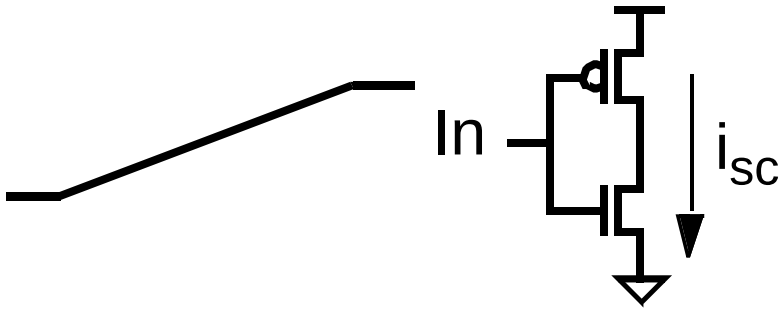
α_n =average number of expected *full* transitions of node n each cycle

A full transition is considered as a switch to high followed by a switch to low.

This “expected number” is almost always less than 1, and is, in fact, almost always less than 0.5

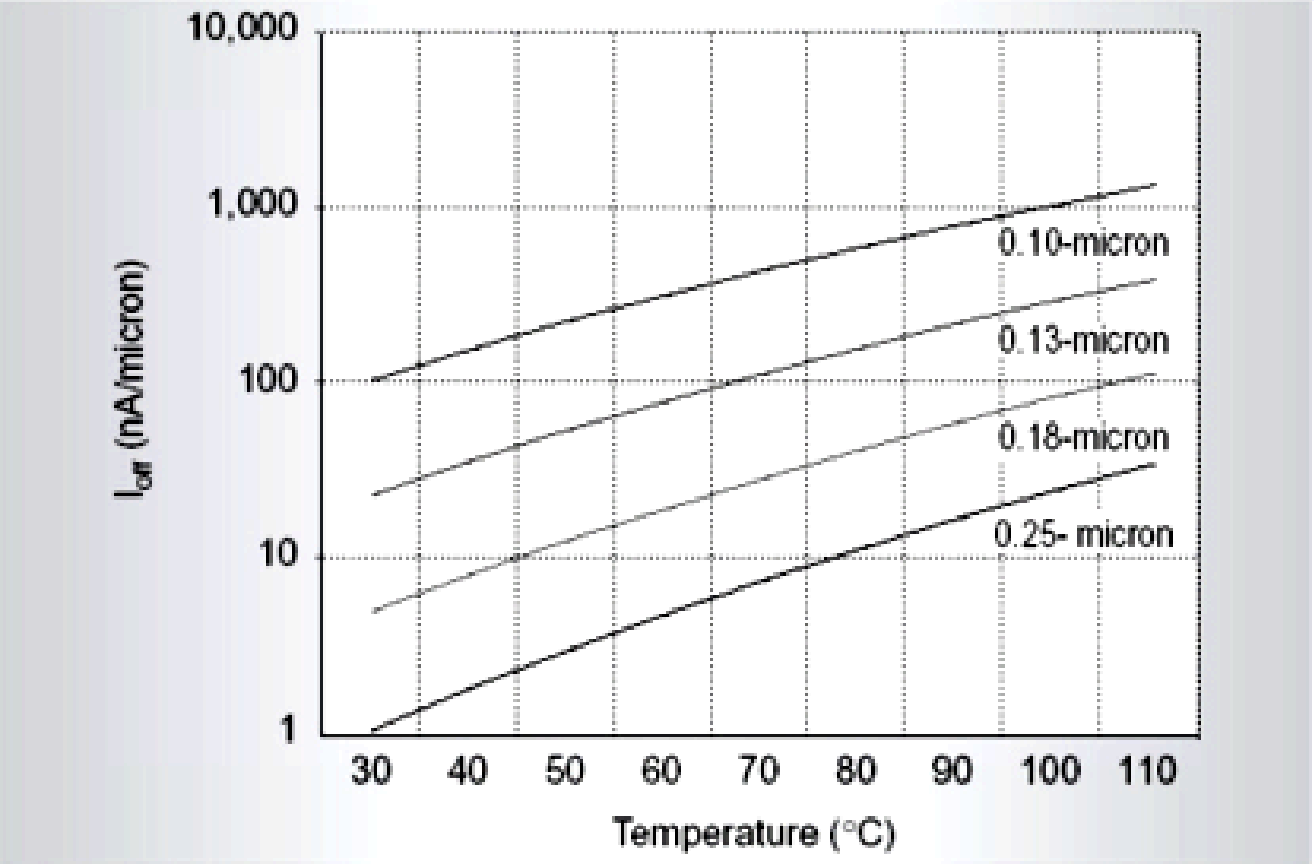
Short-Circuit Power

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.



In general the percentage of the total power due to the short circuit current is 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.

Leakage is becoming more important…



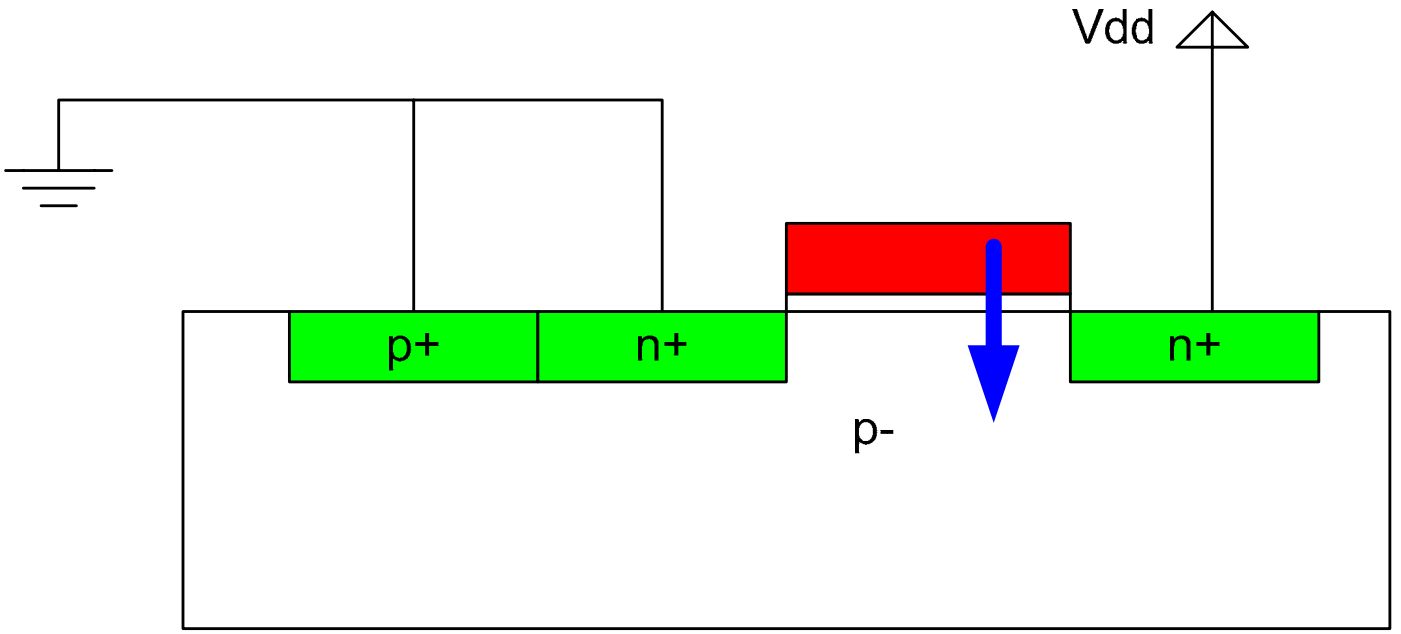
Increases 5 times each generation

Temperature – exponential dependence

Source: Avi Timor

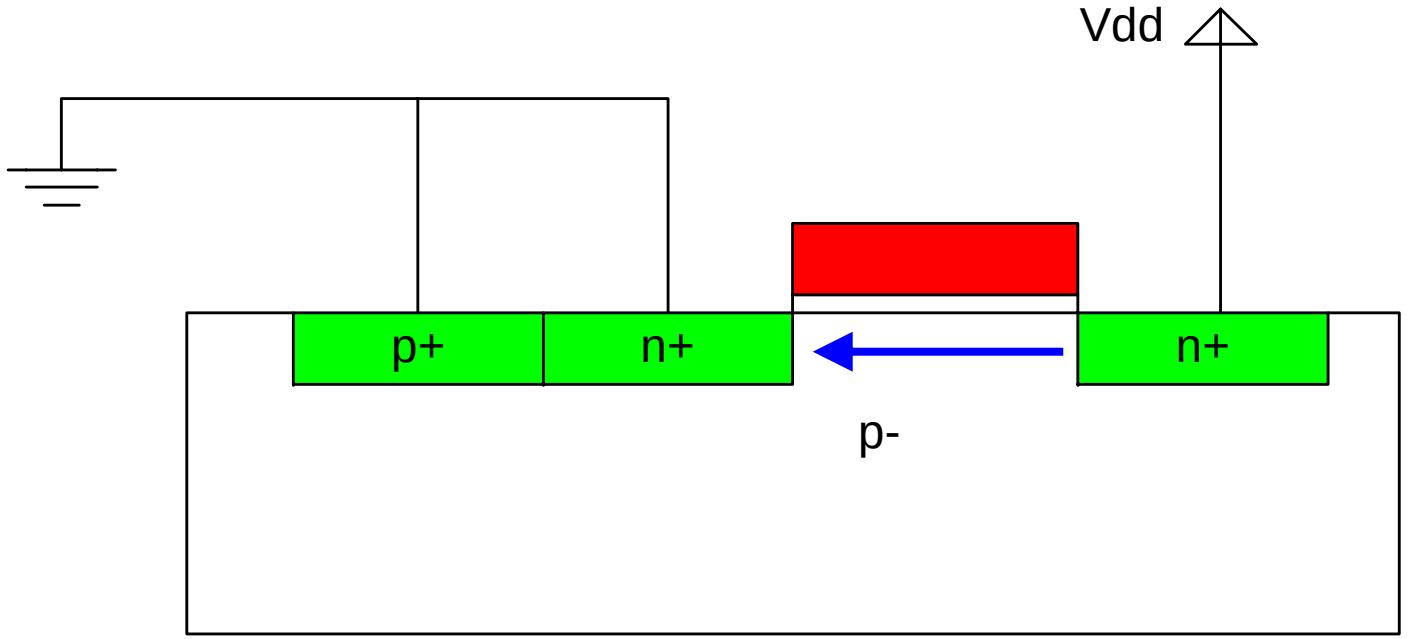
CMOS Leakage Current (Static Power)

Source 1: Gate Oxide Leakage



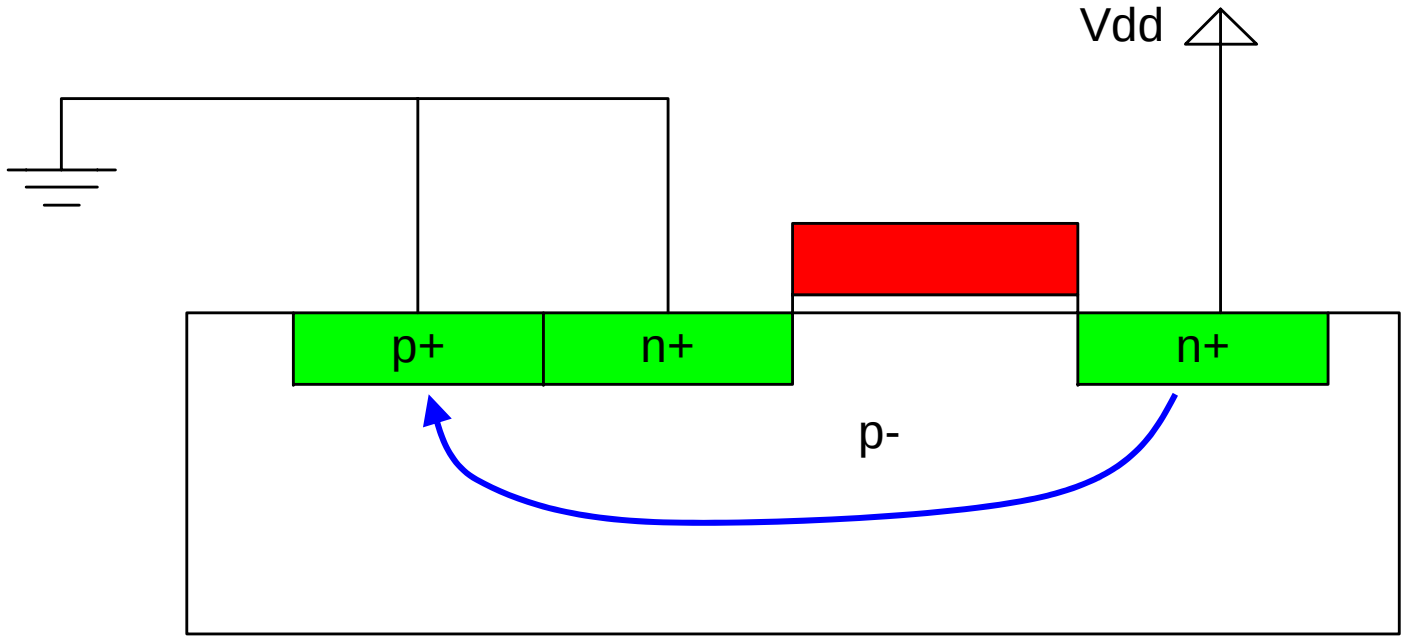
CMOS Leakage Current (Static Power)

Source 2: Subthreshold Channel Leakage:



CMOS Leakage Current (Static Power)

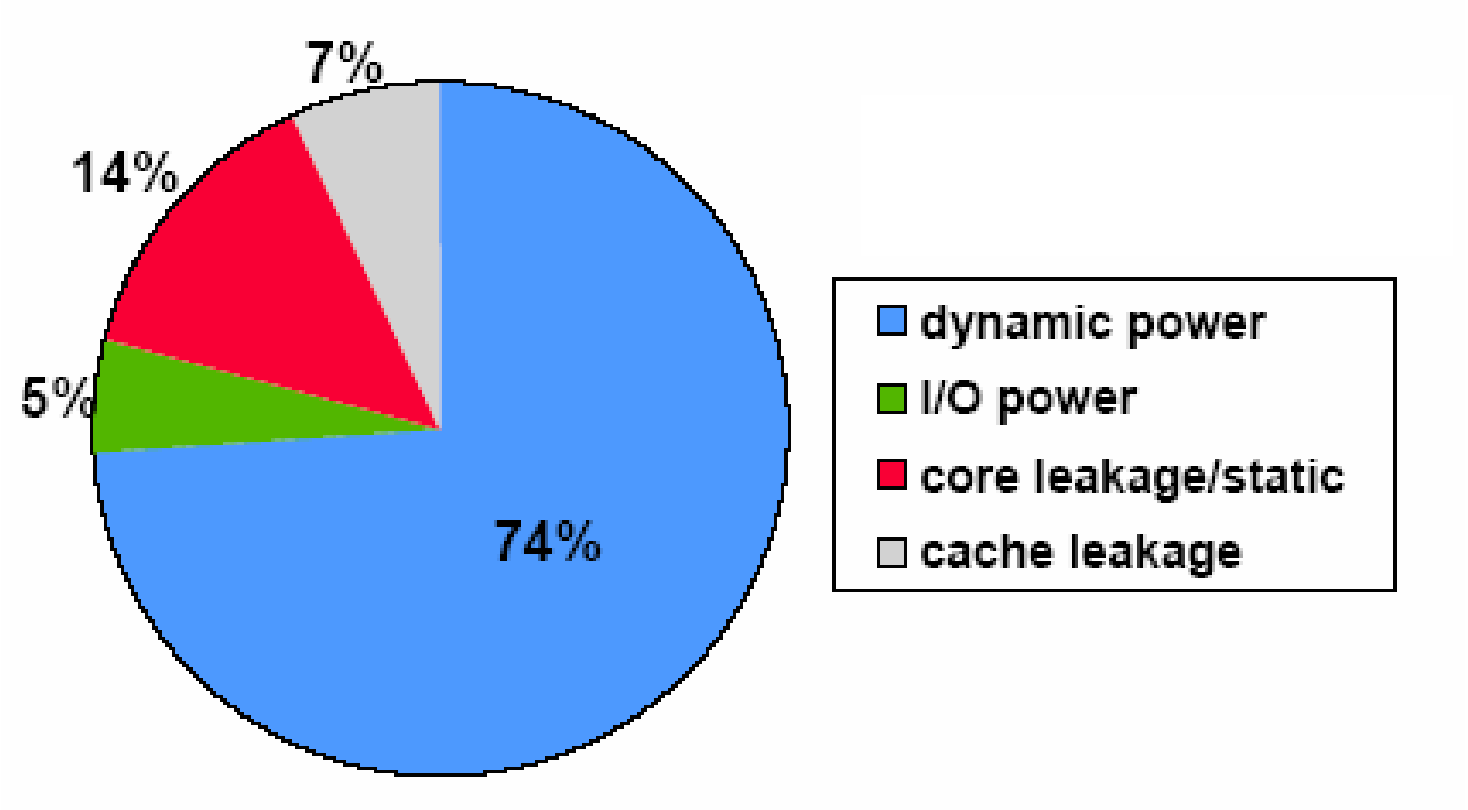
Source 3: Reverse-Biased PN-junction:



There are other sources of leakage too

Some people compare storing charge to “keeping water in a paper bag”

Intel Itanium-2 Power Breakdown



Source: J. Stinson et al “A 1.5GHz Third Generation Itanium(R) Processor “, ISSC 2003

Summary: Total Power

Total Power is given by

$$P = C_{eff} V_{dd}^2 \alpha_n F + V_{dd} \times I_{sc} + V_{dd} \times I_{leak}$$

α_n =average number of times that node n transitions/cycle

– Recall that a full transition is considered as a switch to high followed by a switch to low.

- Generally the dynamic power dominates power in CMOS (>75%)
- Static power is usually pretty small (<10%)
- Short-circuit power is non-negligible, but < dynamic power (<25%)
- 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.

Estimating Dynamic Power for a Circuit

Overview of Estimating Dynamic Power

$$P_{dynamic} = \text{sum over all nodes}(\alpha_{\mathbf{n}} * f * C_{\mathbf{n}} * Vdd^2)$$

We will work out the dynamic power dissipated by each node separately, and then sum all these to get the overall power

Note:

1. f and Vdd are known.
2. We can calculate $C_{\mathbf{n}}$ (parasitic capacitance at node n) using methods from earlier in the course
3. The only thing we don't know how to do is to calculate $\alpha_{\mathbf{n}}$ for each node

How do we estimate the value of α

α_n =average number of expected *full* transitions of node n each cycle

Recipe for finding α_n for each node:

1. For each node, calculate $\mathbf{p_n}$, the *probability that each node is high* over a long period of time.
2. For each node, use $\mathbf{p_n}$ to calculate $\boldsymbol{\alpha_n}$

Let's look at each of these steps separately.

Estimating Activities: Step 1: Probabilities

Step 1: For each node, calculate p_n, the probability that each node is high over a long period of time.

How do we do this?

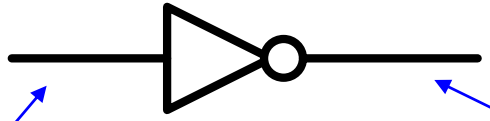
Make an assumption regarding p for each input to your circuit

- If you don't know any better, assume $\mathbf{p_n}$ =0.5 for each input

For each gate, starting with those connected to the inputs, calculate the $\mathbf{p_n}$ of the gate output using $\mathbf{p_n}$ of the gate inputs. This gives $\mathbf{p_n}$ for all the nodes in the circuit.

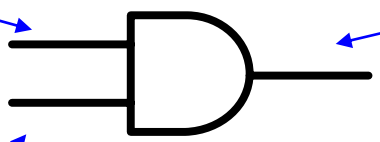
Estimating Activities: Step 1: Probabilities

Simplest Case: Inverter



Pn=0.4 **Pn=0.6**

AND Gate



Pn=0.3 **Pn=0.12** **Pn=0.4**

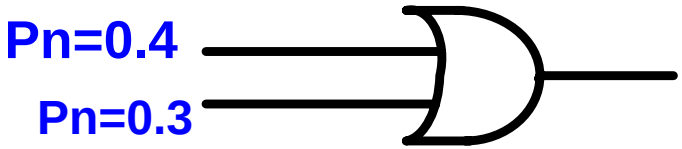
00	0
01	0
10	0
11	1

There's a 40% x 30% chance that both inputs are one.

Therefore, there is a 12% chance that the output is one.

Estimating Activities: Step 1: Probabilities

OR Gate:



00	0
01	1
10	1
11	1

Probability we are here: ((1-0.4)*0.3)=0.18

Probability we are here: (0.4*(1-0.3))=0.28

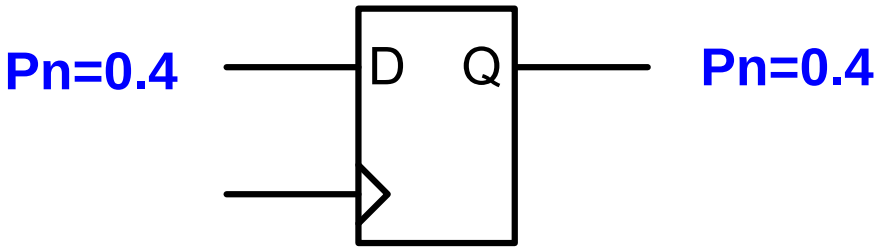
Probability we are here: (0.4*0.3)=0.12

So, probability the output is 1 is 0.18+0.28+0.12 = 0.58

For ***any*** logic gate, you can use the truth table this way to work out the output probabilities (given the input probabilities).

Estimating Activities: Step 1: Probabilities

What about an edge-triggered flip-flop?



Take home exercise: what about a level-sensitive latch?

How do we estimate the value of α

Back to where we were before.

Recipe for finding α for each node:

1. For each node, calculate $\mathbf{p_n}$, the *probability that each node is high* over a long period of time.
2. For each node, use $\mathbf{p_n}$ to calculate α_n

We just learned how to do Step 1

How do we do step #2 ?

Step 2: Estimating value of α given p

Observation: A node switches when:

It was 0, and switches to 1

or it was 1, and switches to 0

Probability it is one = p_n

Probability is was one = p_n

Probability it is zero = $(1-p_n)$

Probability it was zero = $(1-p_n)$

Prob (it was 0 **and** then becomes 1) = $(1-p_n) * p_n$

Prob (it was 1 **and** then becomes 0) = $p_n * (1-p_n)$

Step 2: Estimating value of α given p

From the previous slide:

Prob (it was 0 **and** then becomes 1) = $(1-p_n) * p_n$
Prob (it was 1 **and** then becomes 0) = $p_n * (1-p_n)$

Prob that it changes each cycle = $2*p_n*(1-p_n)$

Over a long time, the expected number of changes each cycle = $2*p_n*(1-p_n)$

Recall: α_n = expected number of full transitions per cycle

Therefore: $\alpha_n = p_n*(1-p_n)$

Summary of dynamic power estimation

1. Assume all inputs have $p_n = 0.5$
2. Starting from the inputs, work out p_n for every node in the circuit
   - Use the truth table method we discussed
3. For each nodes, $\alpha_n = p_n*(1-p_n)$
   - One exception: clock signals have $\alpha_n = 1$
4. For each node, Power = $\alpha * f * C * V^2$
5. For the entire circuit, Power = sum of the power dissipated by each node

Design for Low Power

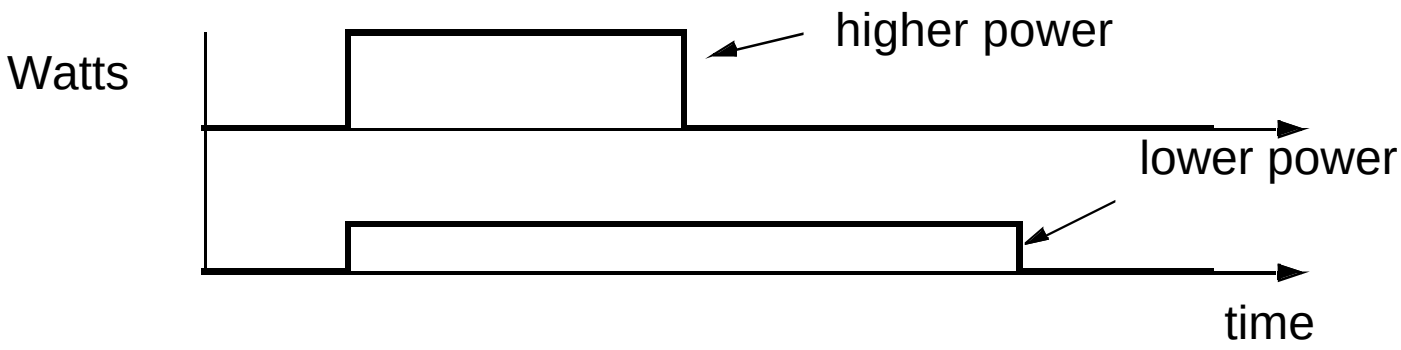
Metric 1: Power

Low Power is an obvious choice as a design goal

- Sets battery life in hours, packaging limits, etc.

But is it really what we want to minimize?

- Comparing the power of two designs can be misleading
- Lower power design could simply be slower
- Dynamic power proportional to frequency



- Lowering the frequency is not the right way to reduce power
- But lowering the activity of an operation is a good way to reduce power

Metric 2: Energy/Operation

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

$$\text{Energy/Op} = \text{Power} * \text{Delay/Op} = C V_{dd}^2 \alpha_n F * \text{Delay/Op}$$

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

Metric 3: Power * delay^2

Some people attempt to minimize:

Power * $(\text{Delay})^2$

This gives more priority to speed, which is often most important

Which is the right metric to optimize for?

- Depends on the application you are designing for
  - If battery life is important: minimize energy (metric 2)
  - If you are worried about heat: minimize power (metric 1)
  - If speed is more important than power, minimize metric 3

Very effective way to reduce power: reduce Vdd

Dynamic Energy:

$$\text{Power} = C\, V_{dd}^2\, \alpha_n F$$

So, **power is proportional to the SQUARE of Vdd**.

Static power is also strongly affected by Vdd.

But, reducing Vdd increases delay:

$$\text{Idrain} = \text{constant} * W/L * (Vdd - Vt)^2$$

$$\text{delay} = \frac{Cload(Vdd)}{\text{Idrain}} = \frac{Cload*(Vdd)}{\text{constant} * W/L * (Vdd-Vt)^2}$$

So very roughly, **delay is inversely proportional to Vdd**

Reducing Vdd

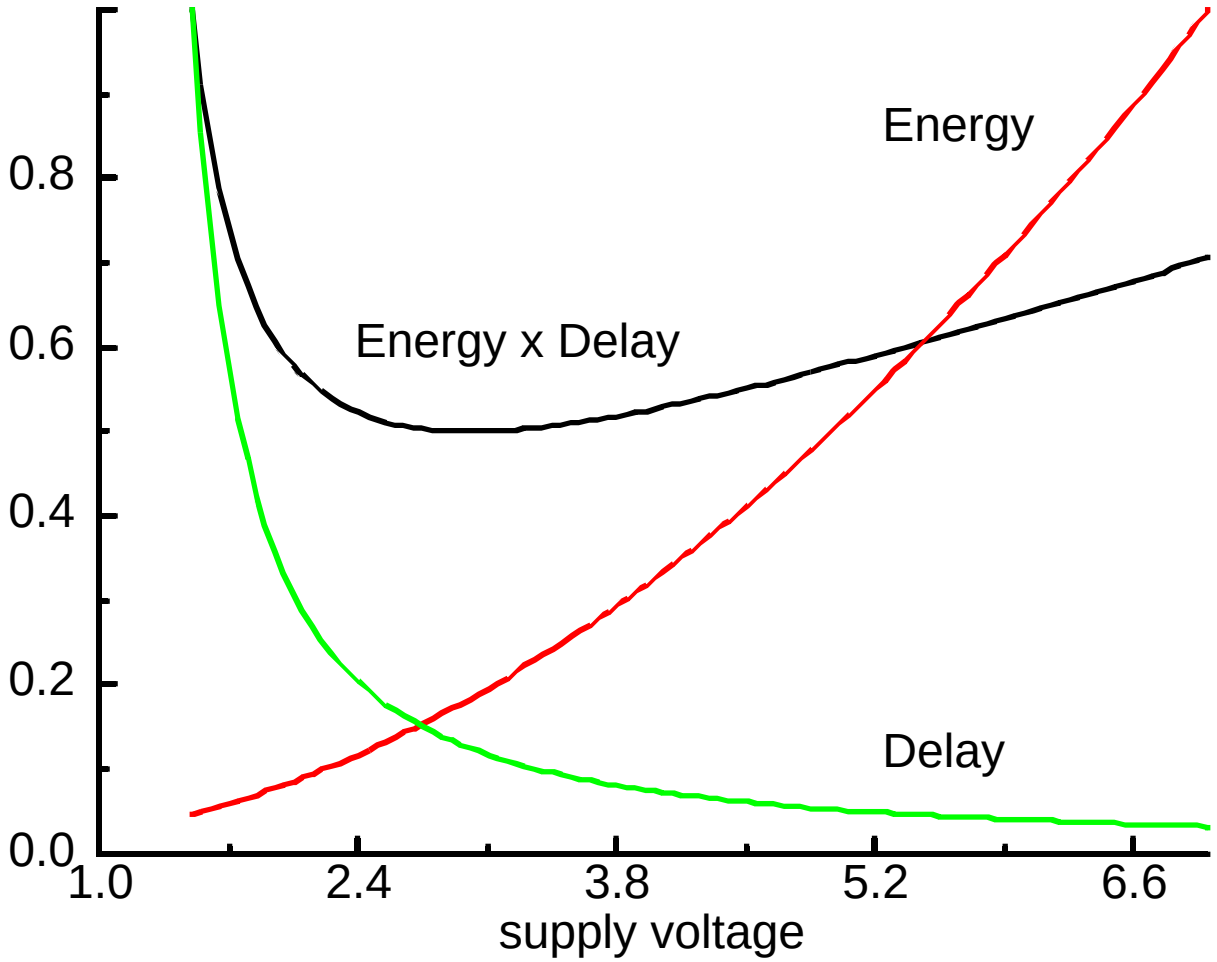
From the last slide:

- Power is proportional to the SQUARE of Vdd
- Delay is inversely proportional to Vdd

So, energy is very roughly inversely proportional to Vdd

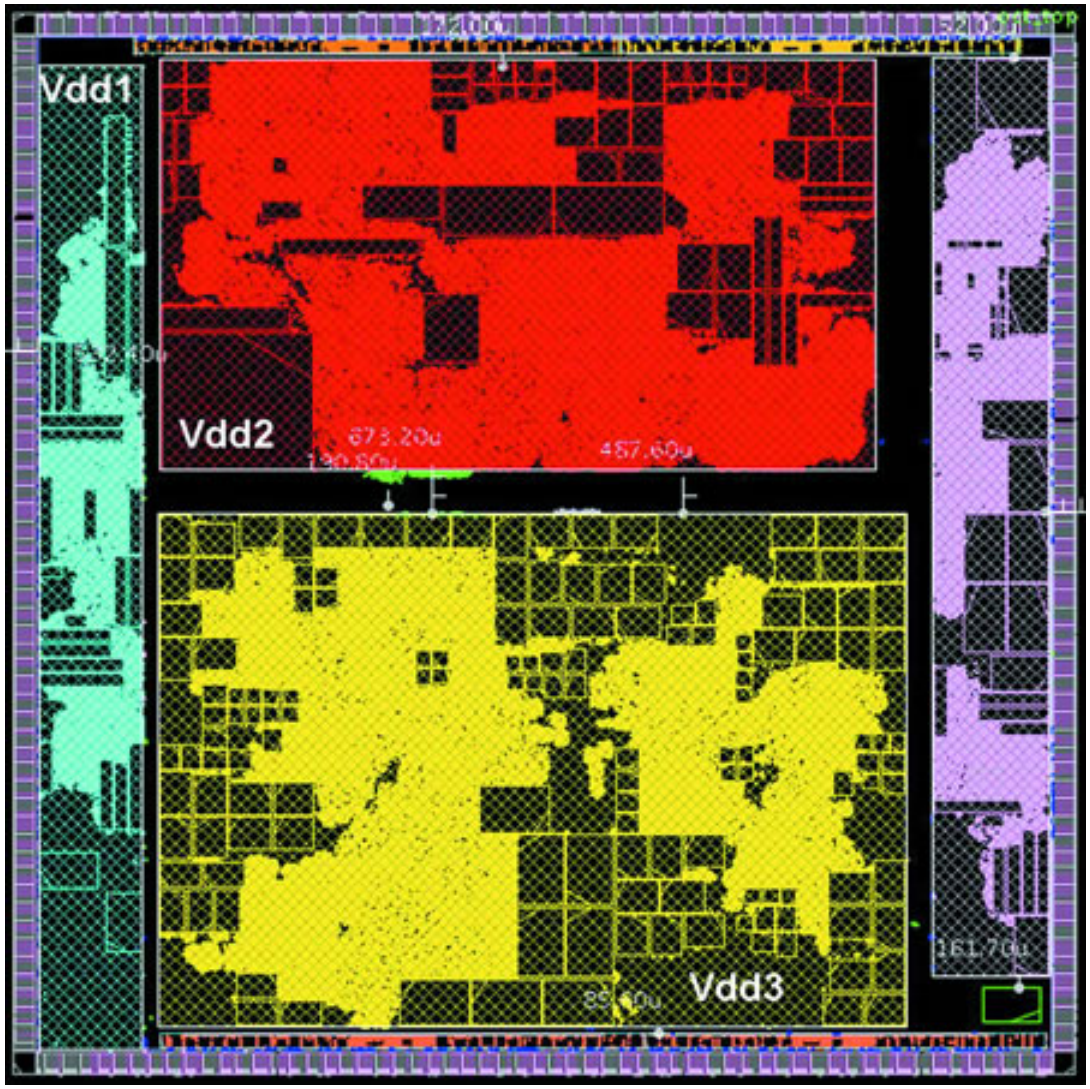
If we are wiling to give up some performance, we can reduce total energy.

Energy vs. Delay



Voltage Islands

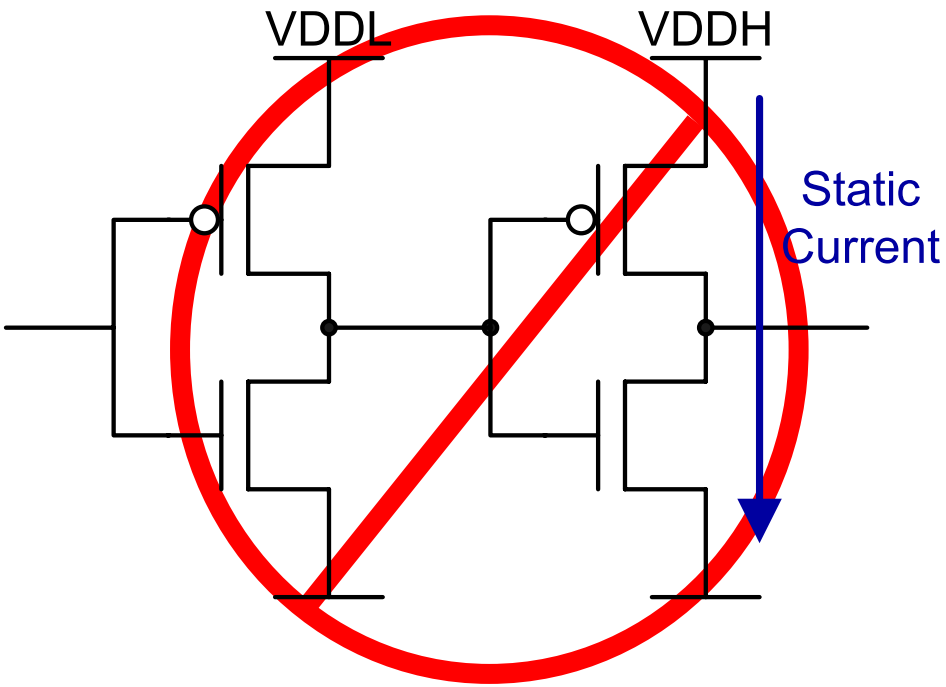
- Time-critical parts of the chip: high voltage
- Other parts of the chip: low voltage
- Can have many voltage islands, each with a different voltage
- Need level converter to communicate between islands



To do this, you need level converters…

Cannot directly connect V_{DDL} and V_{DDH} cells

- Output of V_{DDL} gate can't be raised higher than V_{DDL}
- When connected to V_{DDH} gate, PMOS will never be completely cut-off → Static Current



Another way to reduce power: two different Vt's

Vt is the voltage at which a transistor starts to conduct:

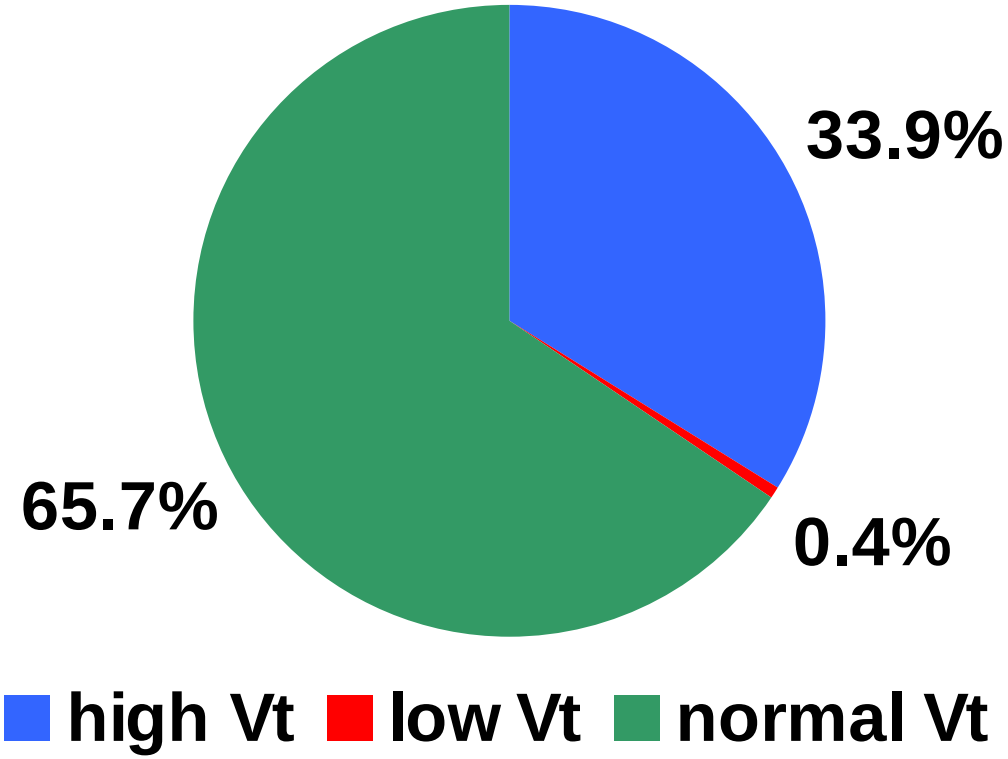
For high speed, a low Vt is better

But, a low Vt will have more leakage

Idea: have "high Vt" transistors and "low Vt" transistors

- Use the low Vt transistors for speed critical paths
- Use the high Vt transistors for everything else

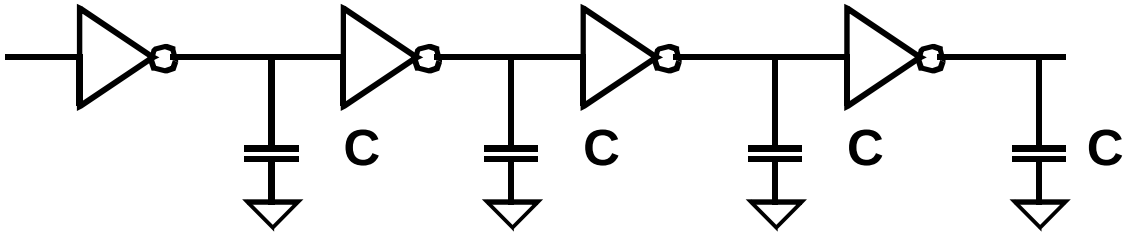
POWER5 Device Width



Source: DAC'2004: Clabes et al.

Reducing Power: Transistor Sizing

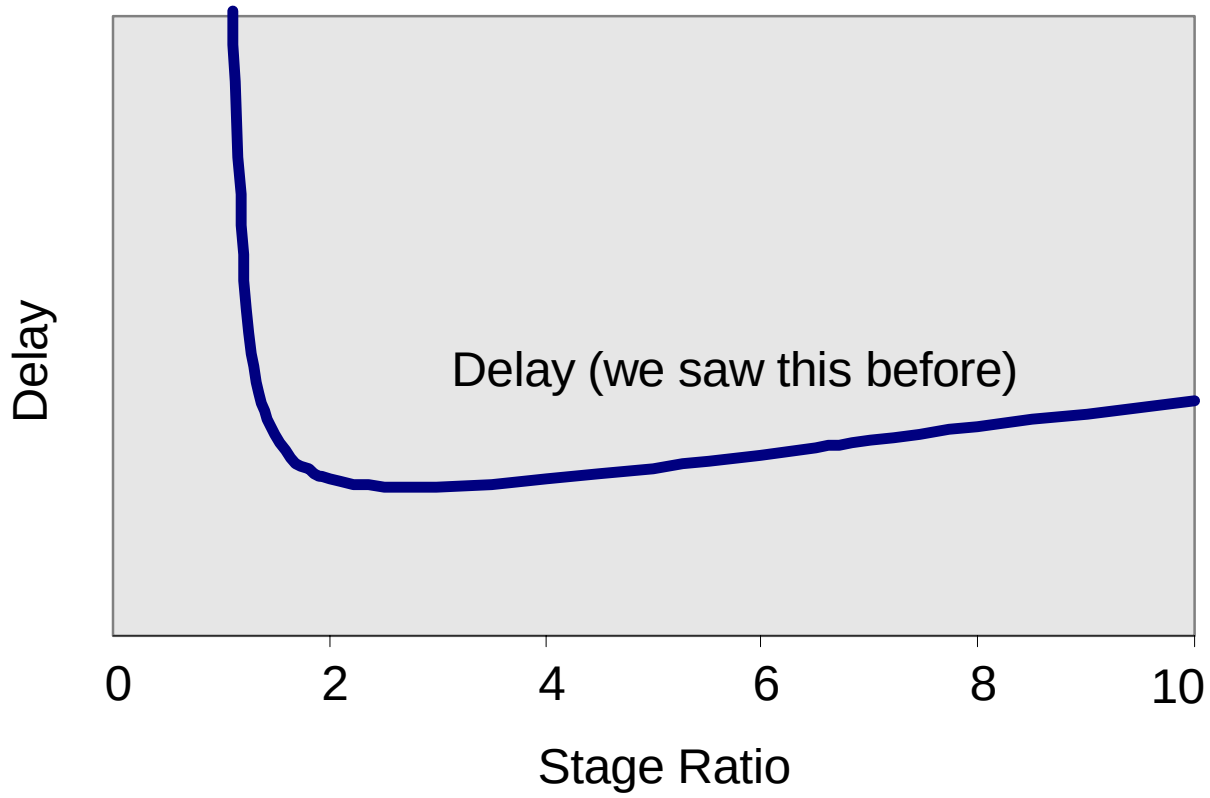
Recall, we learned that there is an optimum stage ratio (for delay)



What about Power?

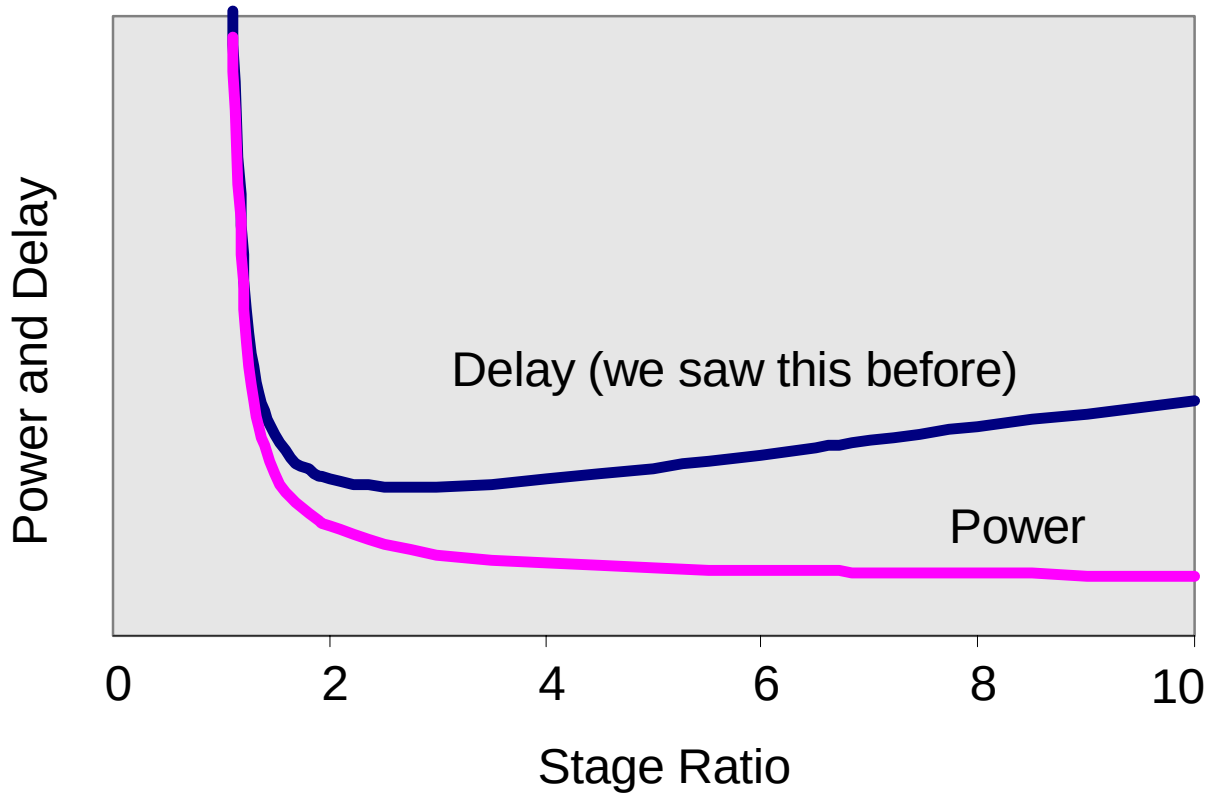
Stage Ratio Results

Consider a string of progressively larger inverters:



Stage Ratio Results

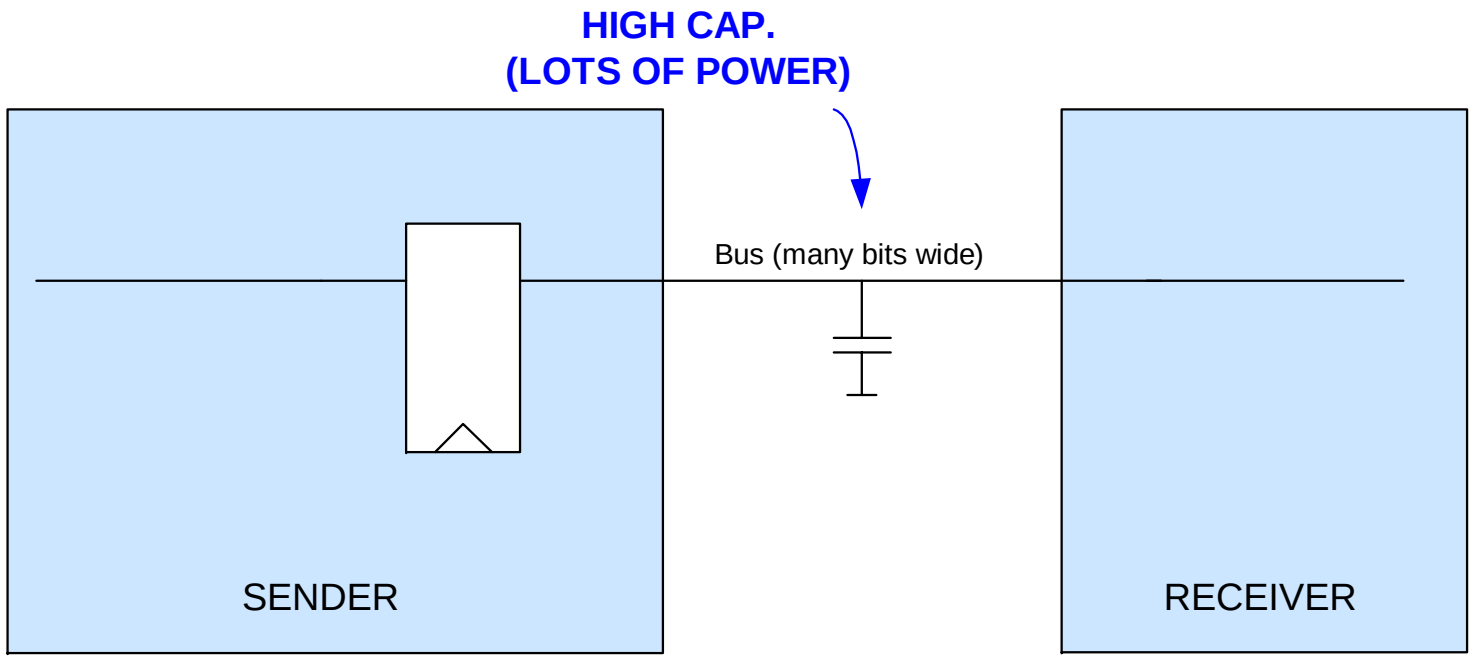
Consider a string of progressively larger inverters:



Fewer Stages -> Fewer Transistors -> Less Capacitance -> Lower Power

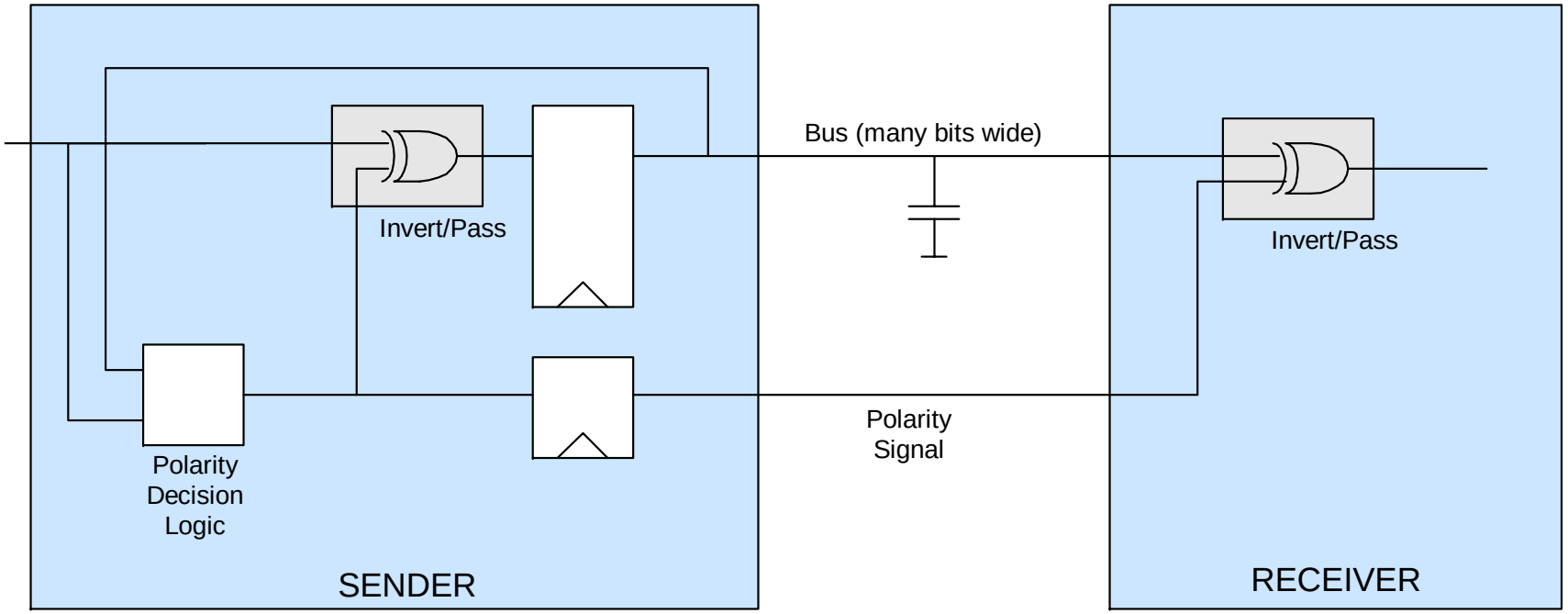
Logic-Level Techniques

Consider sending data over a long distance:



Logic-Level Techniques

Bus Invert Encoding:

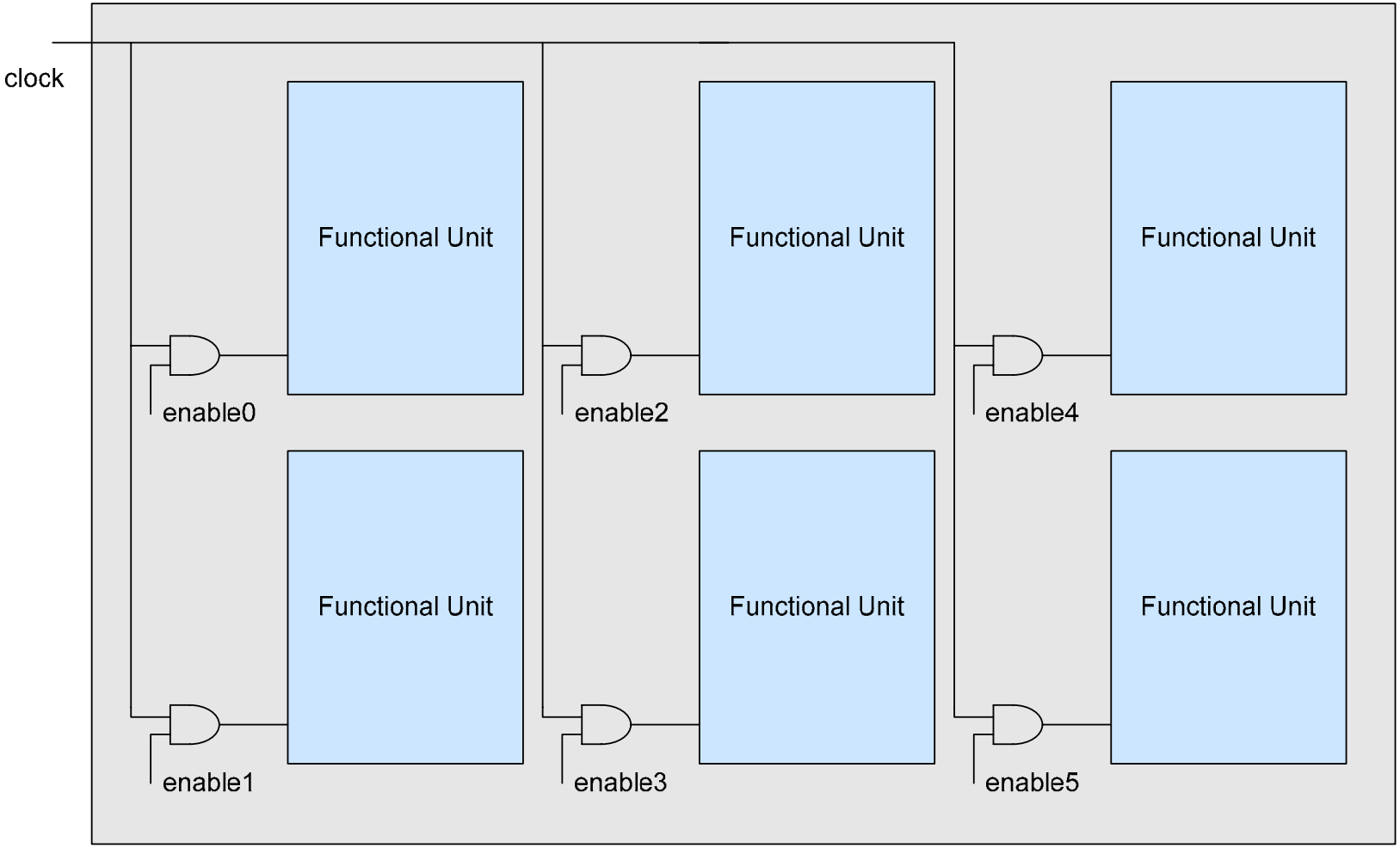


If it would use less power to send inverse of bus, do so.

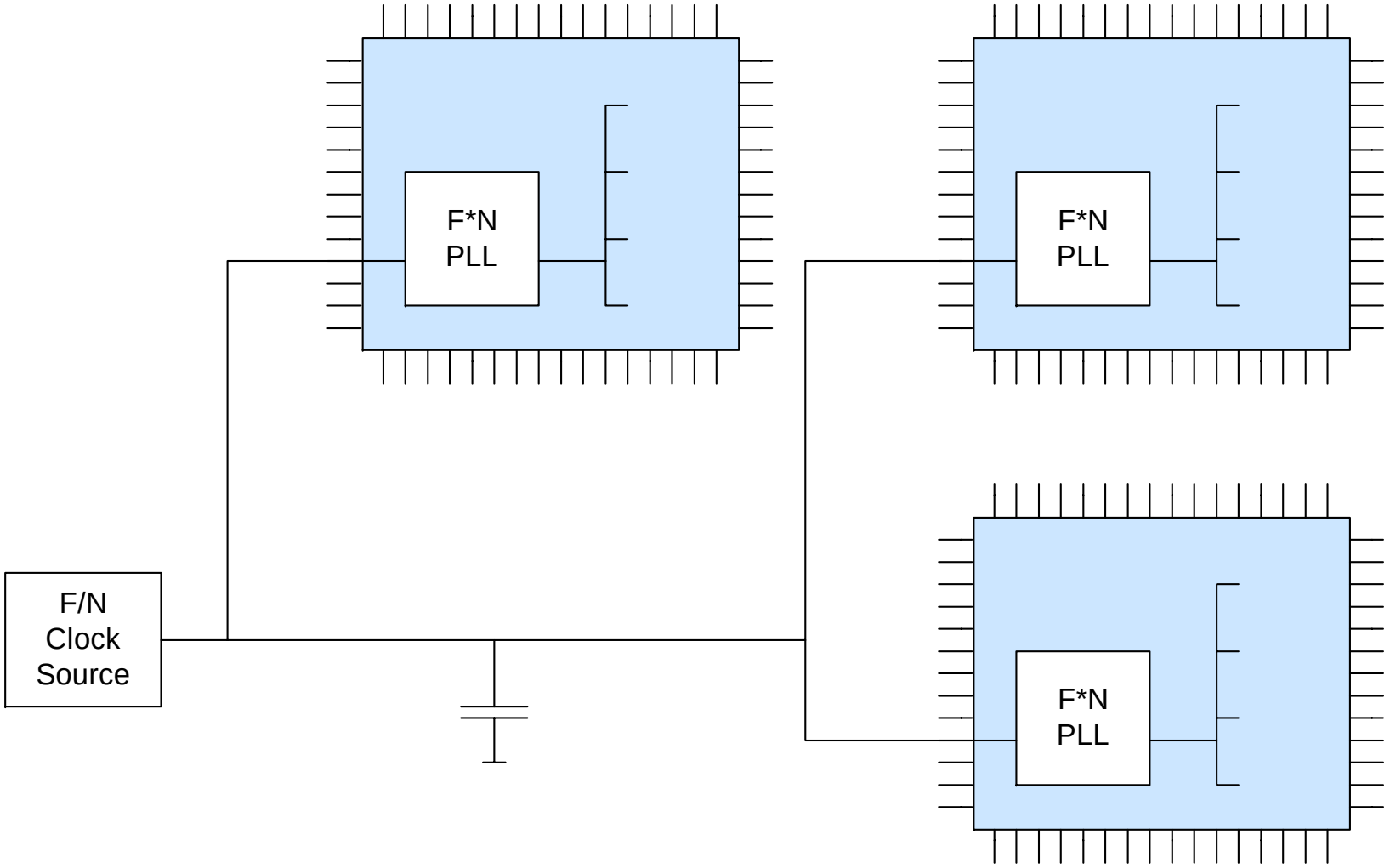
Example: if we just sent 0000, and we are now to send 0111, send 1000 instead

Higher-Level Techniques: Clock Gating

Produce a qualified clock for each functional unit:

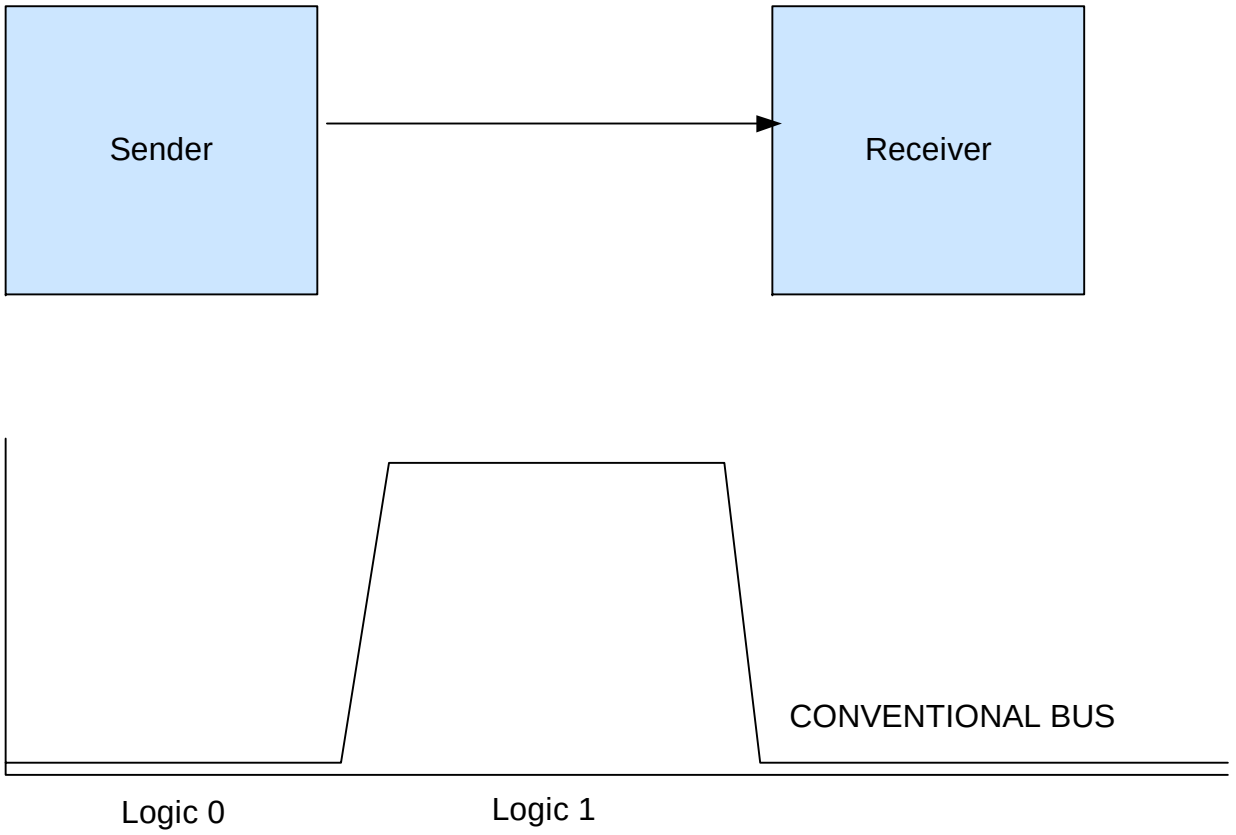


Higher-Level Techniques: Clock Division



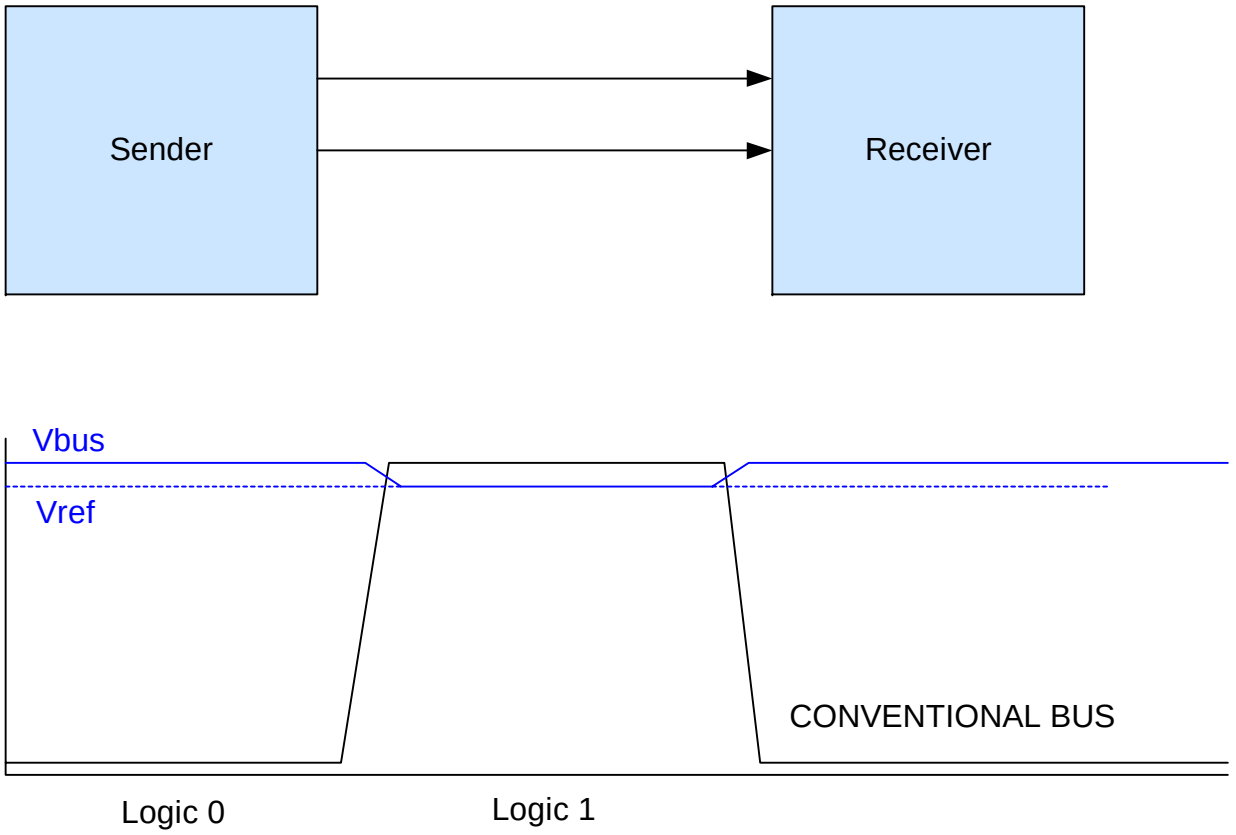
Low Swing (differential) Bus

Normal Way:



Low Swing (differential) Bus

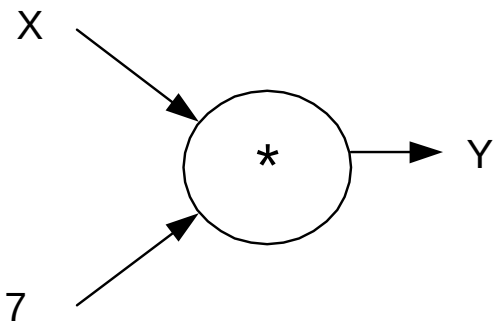
Normal Way:



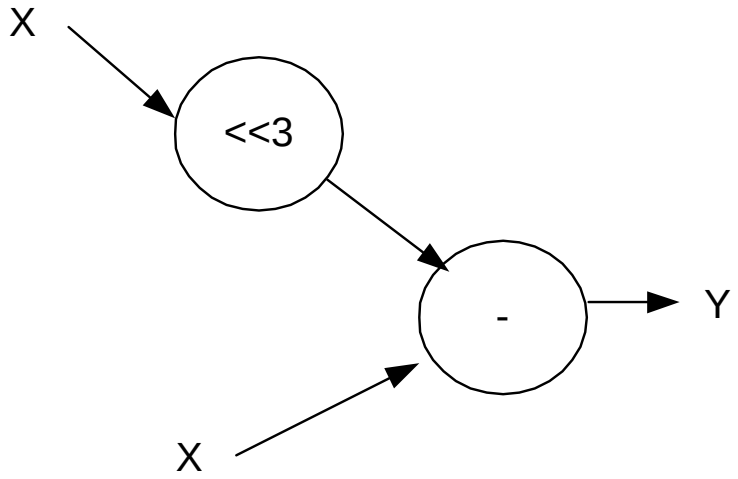
Algorithmic-Level Optimizations

This is where the real pay-off is:

Example:



vs



Y = 7X

Y=(X<<3)-X

Big Wins:

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

Reformulate the problem

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

Shut things off when not in use

- Use gated clocks to control flip-flop power
- Use voltage scaling / transistor sizing
  - convert excess speed to low power

Use parallelism

- Improve delay, and thus energy * delay