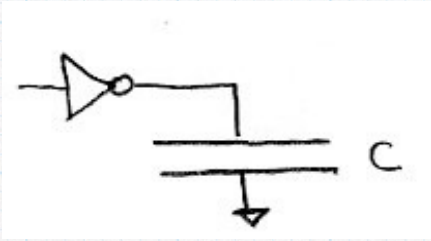
EECS 151/251A Spring 2018 Digital Design and Integrated Circuits

Instructors:
Nick Weaver & John Wawrzynek

Lecture 11

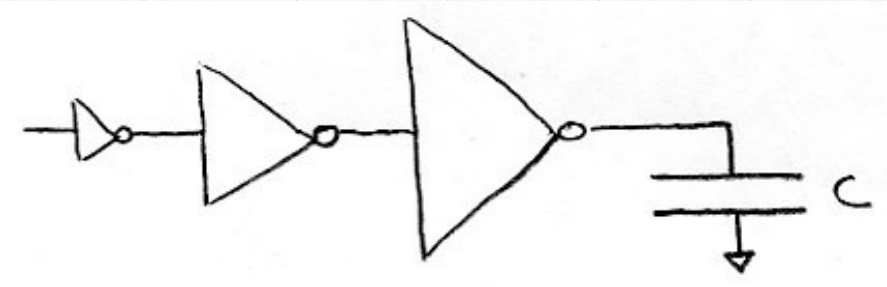
Driving Large Loads

- Large fanout nets: clocks, resets, memory bit lines, off-chip
- Relatively small driver results in long rise time (and thus large gate delay)



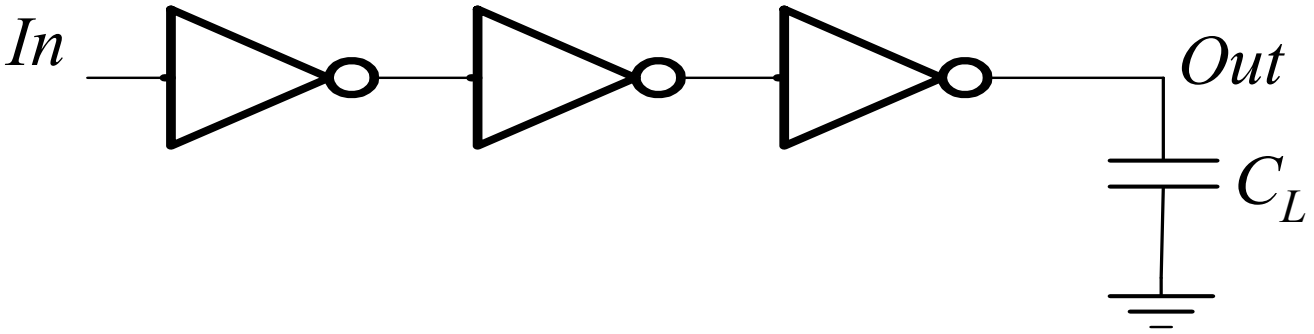
- Strategy:

Staged Buffers



- How to optimally scale drivers?
- Optimal trade-off between delay per stage and total number of stages?

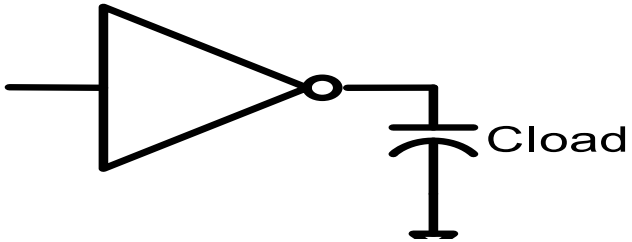
Inverter Chain



- For some given C_L:
  - How many stages are needed to minimize delay?
  - How to size the inverters?
- Anyone want to guess the solution?

Careful about Optimization Problems

- Get fastest delay if build one **very** big inverter
  - So big that delay is set only by self-loading

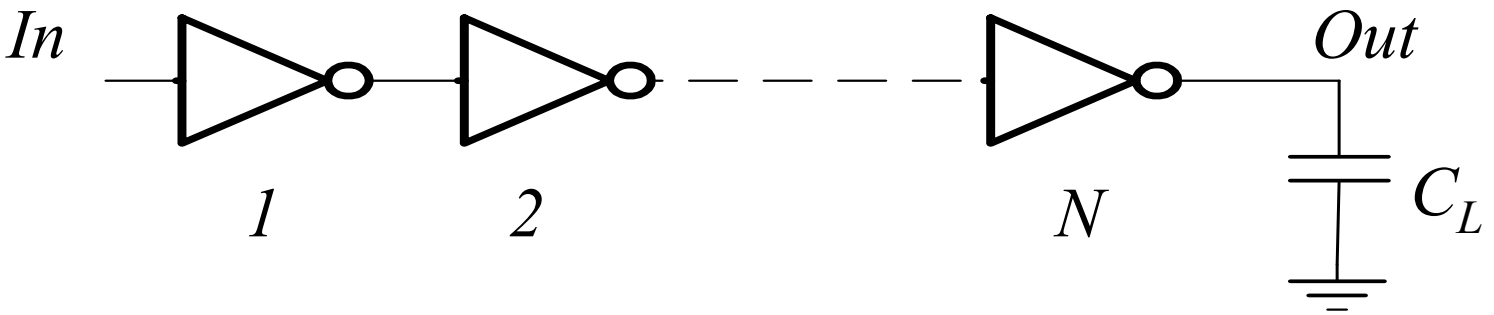


- Likely not the problem you're interested in
  - Someone has to drive this inverter…

Delay Optimization Problem #1

- You are given:
  - A fixed number of inverters
  - The size of the first inverter
  - The size of the load that needs to be driven
- Your goal:
  - Minimize the delay of the inverter chain
- Need model for inverter delay vs. size

Apply to Inverter Chain



$$t_p = t_{p1} + t_{p2} + \ldots + t_{pN}$$

$$t_{pj} = t_{inv}\left(\gamma + \frac{C_{in,j+1}}{C_{in,j}}\right)$$

$$t_p = \sum_{j=1}^{N} t_{p,j} = t_{inv}\sum_{i=1}^{N}\left(\gamma + \frac{C_{in,j+1}}{C_{in,j}}\right), \quad C_{in,N+1} = C_L$$

Optimal Sizing for Given N

- Delay equation has N-1 unknowns, $C_{in,2} \ldots C_{in,N}$
- **To minimize the delay,** find N-1 partial derivatives:

$$t_p = \ldots + t_{inv}\frac{C_{in,j}}{C_{in,j-1}} + t_{inv}\frac{C_{in,j+1}}{C_{in,j}} + \ldots$$

$$\frac{dt_p}{dC_{in,j}} = t_{inv}\frac{1}{C_{in,j-1}} - t_{inv}\frac{C_{in,j+1}}{C_{in,j}^2} = 0$$

Optimal Sizing for Given N (cont'd)

- Result: every stage has **equal fanout (f)**:

$$\frac{C_{in,j}}{C_{in,j-1}} = \frac{C_{in,j+1}}{C_{in,j}}$$

- Size of each stage is geometric mean of two neighbors:

$$C_{in,j} = \sqrt{C_{in,j-1} C_{in,j+1}}$$

- Equal fanout → every stage will have same delay

Optimum Delay and Number of Stages

- When each stage has same fanout f:

$$f^N = F = C_L / C_{in,1}$$

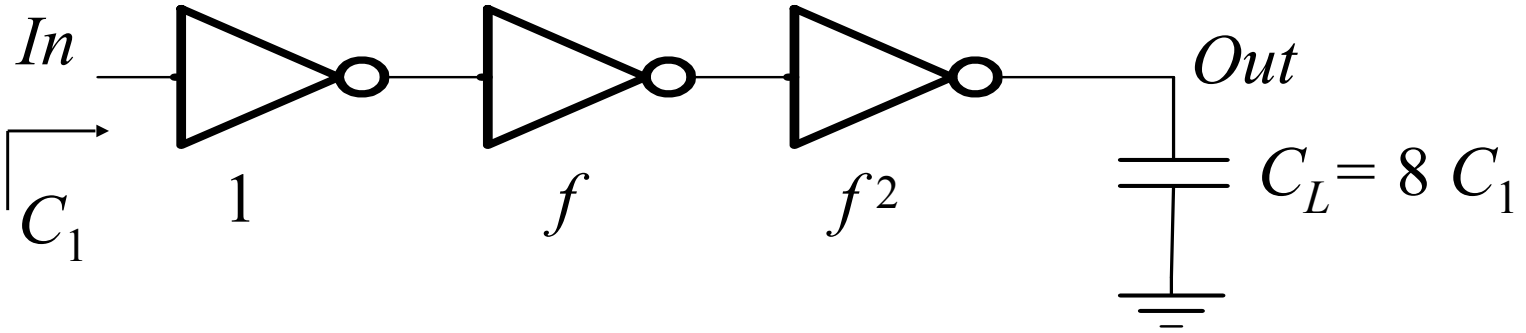
- Fanout of each stage:

$$f = \sqrt[N]{F}$$

- Minimum path delay:

$$t_p = N t_{inv} \left(\gamma + \sqrt[N]{F} \right)$$

Example



C_L/C_1 has to be evenly distributed across $N = 3$ stages:

Delay Optimization Problem #2

- You are given:
  - The size of the first inverter
  - The size of the load that needs to be driven
- Your goal:
  - Minimize delay by finding optimal number and sizes of gates
- So, need to find N that minimizes:

$$t_p = Nt_{inv}\left(\gamma + \sqrt[N]{C_L/C_{in}}\right)$$

Untangling the Optimization Problem

- Rewrite N in terms of fanout/stage f:

$$f^N = C_L/C_{in} \rightarrow \boxed{N = \frac{\ln\left(C_L/C_{in}\right)}{\ln f}}$$

$$t_p = Nt_{inv}\left(\left(C_L/C_{in}\right)^{1/N} + \gamma\right) = t_{inv}\ln\left(C_L/C_{in}\right)\left(\frac{f+\gamma}{\ln f}\right)$$

$$\frac{\partial t_p}{\partial f} = t_{inv}\ln\left(C_L/C_{in}\right)\cdot\frac{\ln f - 1 - \gamma/f}{\ln^2 f} = 0$$

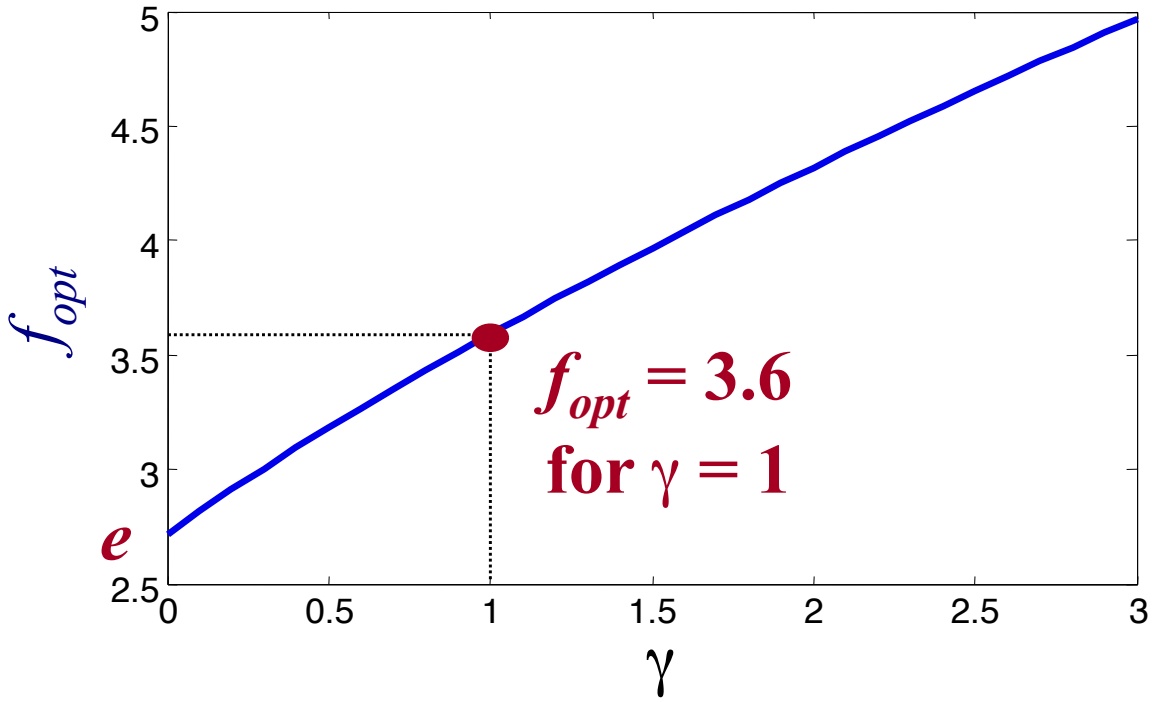
$$\boxed{f = \exp\left(1+\gamma/f\right)} \quad \text{(no explicit solution)}$$

For $\gamma = 0, f = \mathrm{e}, N = \ln\left(C_L/C_{in}\right)$

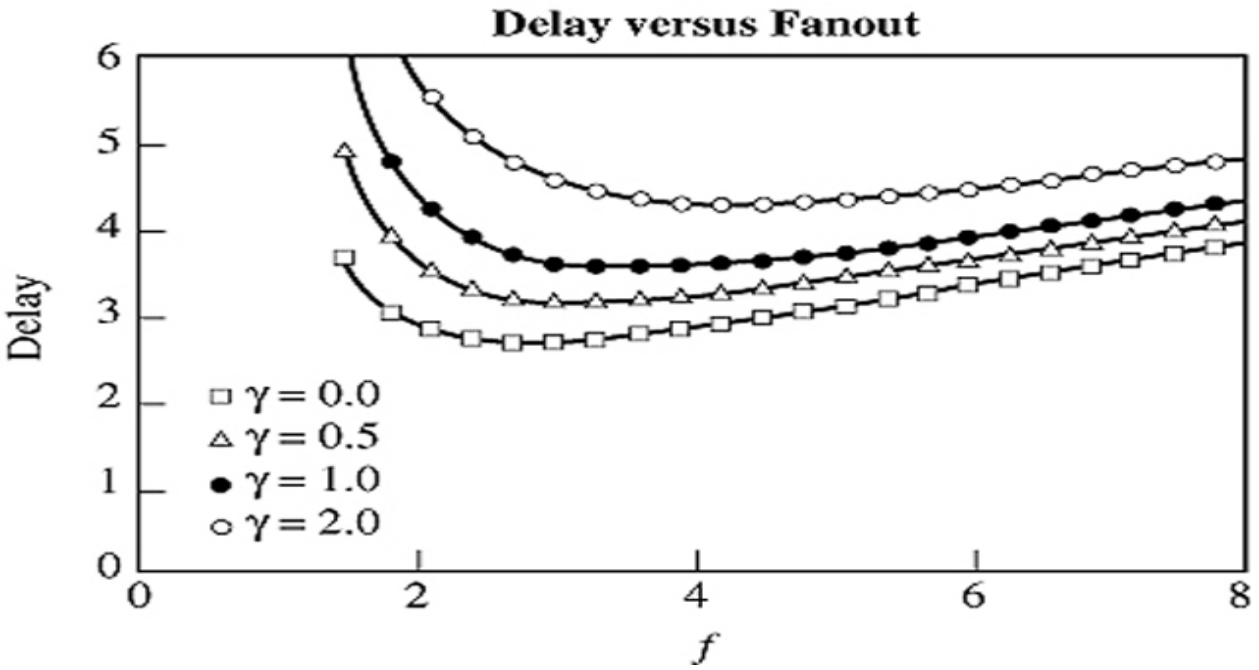
Optimum Effective Fanout f

- Optimum f for given process defined by γ

$$f = \exp(1 + \gamma / f)$$



In Practice: Plot of Total Delay



[Hodges, p.281]

- Why the shape?
- Curves very flat for $f > 2$
  - Simplest/most common choice: $f = 4$

Normalized Delay As a Function of F

$$t_p = Nt_{inv}\left(\gamma + \sqrt[N]{F}\right), \quad F = C_L / C_{in}$$

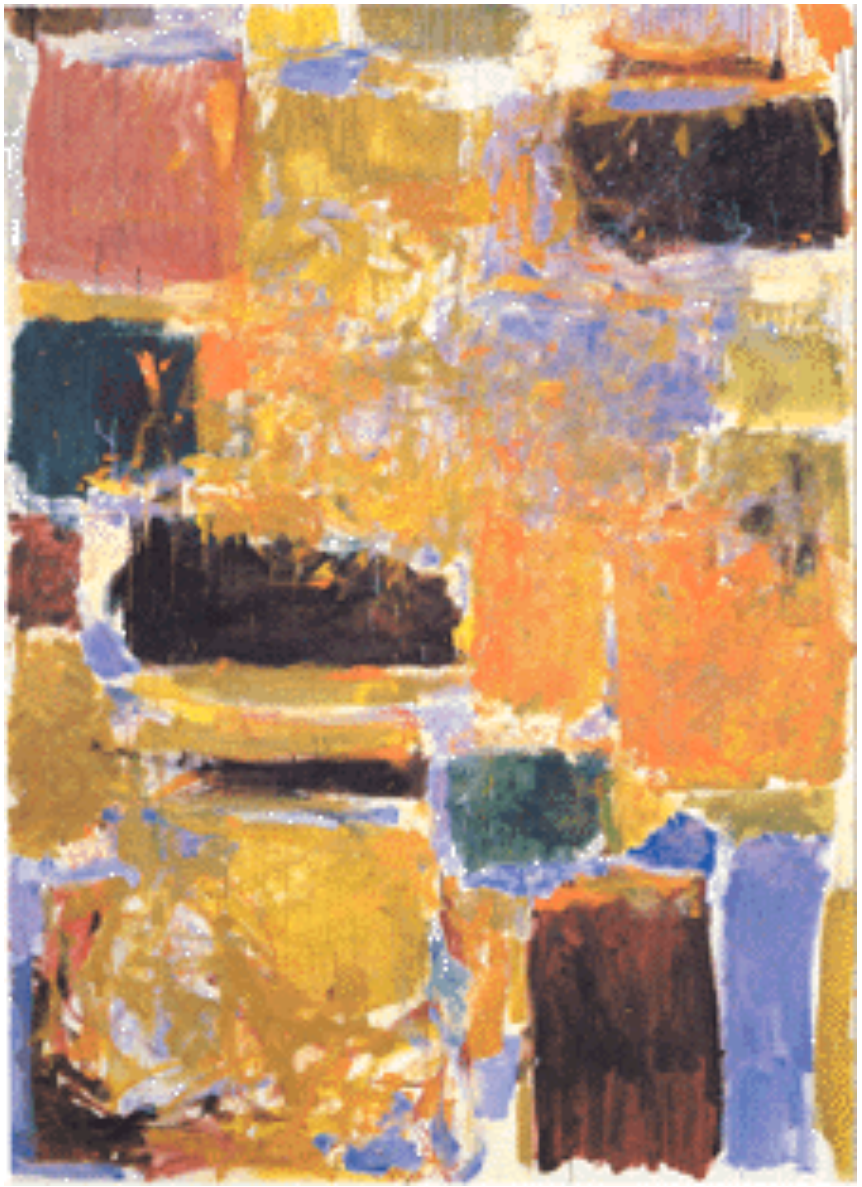
F	Unbuffered	Two Stage	Inverter Chain
10	11	8.3	8.3
100	101	22	16.5
1000	1001	65	24.8
10,000	10,001	202	33.1

$(\gamma = 1)$

[Rabaey: page 210]

Conclusion

- Timing Optimization: You start with a target on clock period. What control do you have?
- Biggest effect is RTL manipulation.
  - i.e., how much logic to put in each pipeline stage.
  - We will be talking later about how to manipulate RTL for better timing results.
- In most cases, the tools will do a good job at logic/circuit level:
  - Logic level manipulation
  - Transistor sizing
  - Buffer insertion
  - But some cases may be difficult and you may need to help
- The tools will need some help at the floorpan and layout



Energy-delay trade-offs

Power, Energy, and Power Density

- Energy: The total energy consumed for an operation
  - Measured in joules
    - Alternately in power*time (e.g. watt-hours for batteries)
- Power:
  - The amount of energy per unit time used
- Power-density:
  - The amount of energy per unit time per unit area

When Optimizing for "Power"...

- Colloquially, we talk about power optimization...
- But really we want to optimize for ***energy*** consumption or ***power density***
  - Energy consumption -> battery life
  - Power density -> ability to cool the chip
- Cooling is often steady-state
  - We can temporarily use more power but eventually the chip will either slow/ shut down or blow up

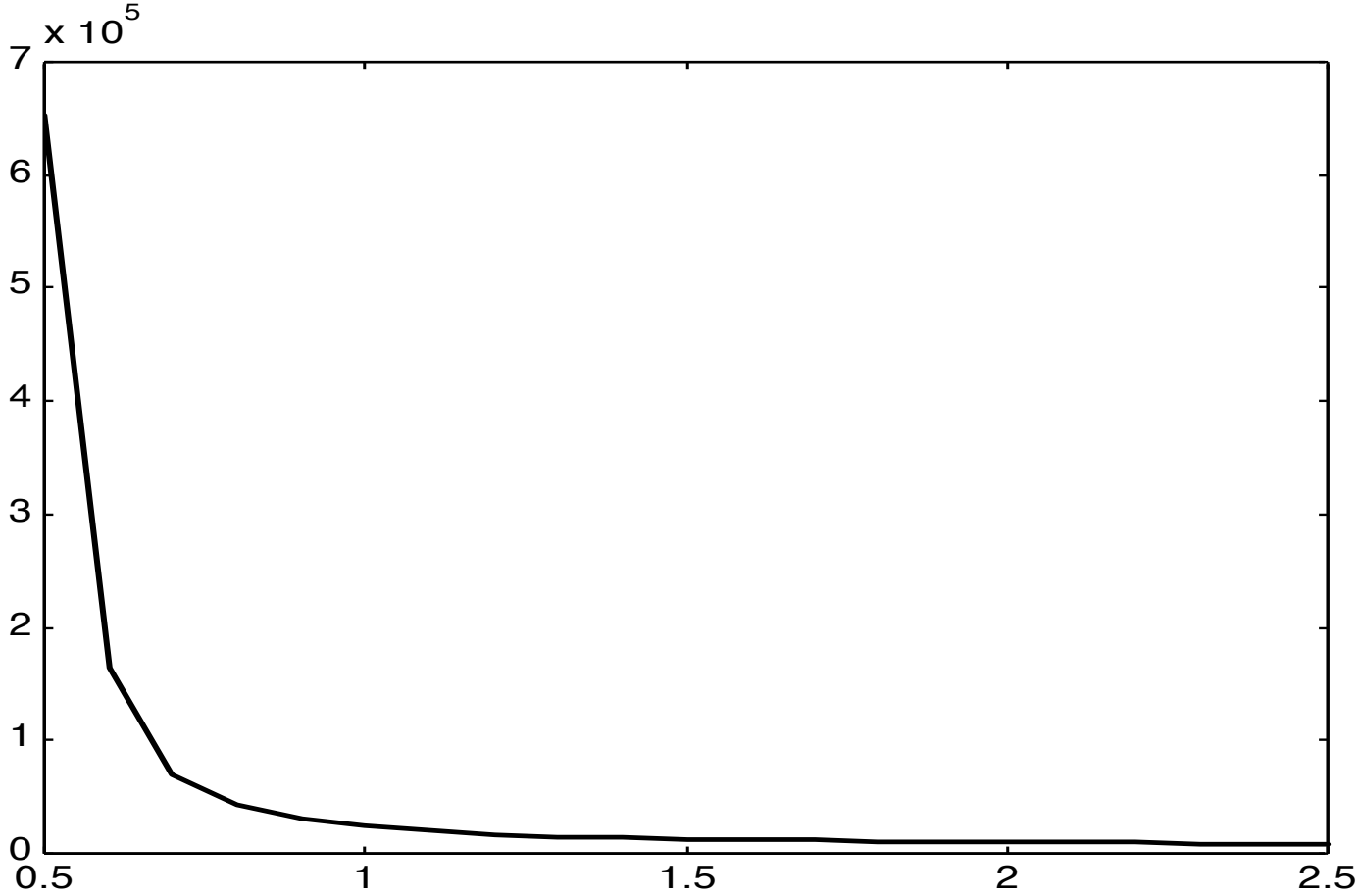
Where Does Power Go in CMOS?

- Switching power
  - Charging/discharging capacitors
  - Proportional to C V^2 F
    - Switching ***energy*** is only proportional to C V^2
    - Lowering voltage slows transistors
- Short-circuit power
  - Both pull-up and pull-down on during transition
- Leakage power
  - Transistors are imperfect switches
- Static currents
  - Biasing currents, in e.g. analog, memory

The Knobs of the Circuit Designer

- Architectural choices
- Logic Structure
- Width of Transistors
- Supply Voltage
- (Threshold Voltage)

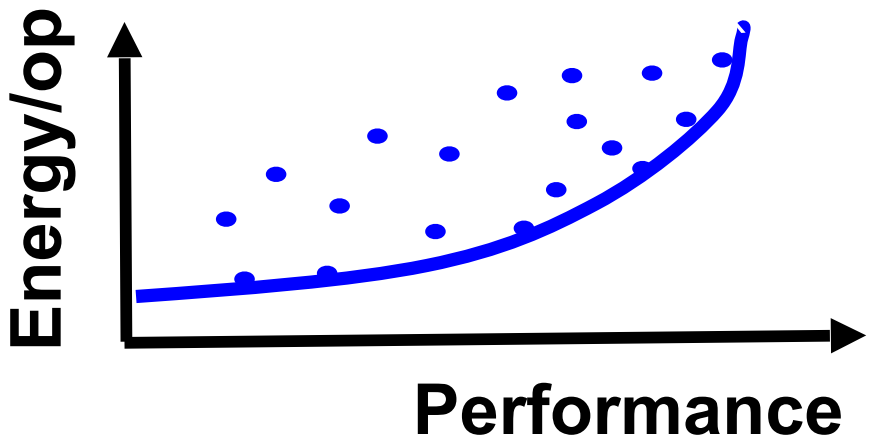
Transistor Resistance as a Function of Voltage



Principles for Power Reduction

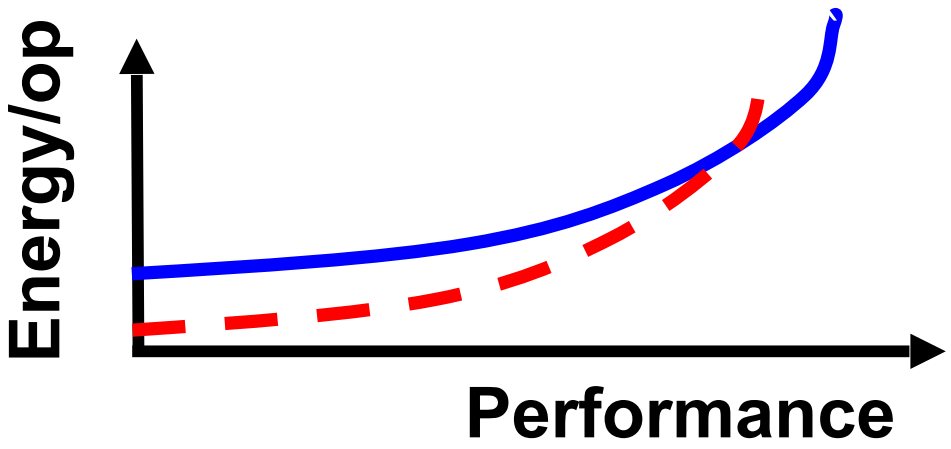
- Most important idea: reduce waste
- Examples:
  - Don't switch capacitors you don't need to
    - Clock gating, glitch elimination, logic re-structuring
  - Don't run circuits faster than needed
    - Power α V_{DD}^2 – can save a lot by reducing supply for circuits that don't need to be as fast
    - Parallelism falls into this category
- Let's say we do a good job of that – then what?

Energy – Performance Space



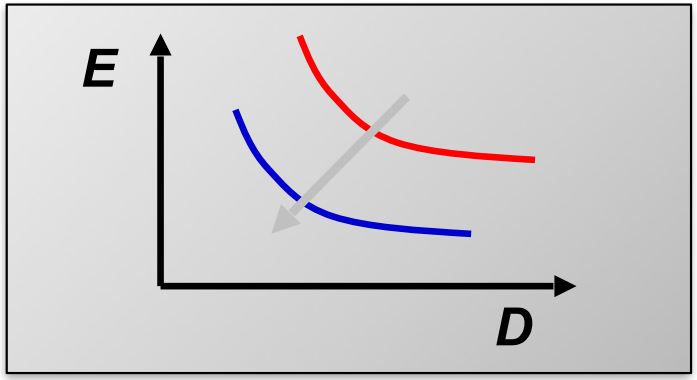
- Plot all possible designs on a 2-D plane
  - No matter what you do, can never get below/to the right of the solid line
- This line is called “Pareto Optimal Curve”
  - Usually (always) follows law of diminishing returns

Optimization Perspective

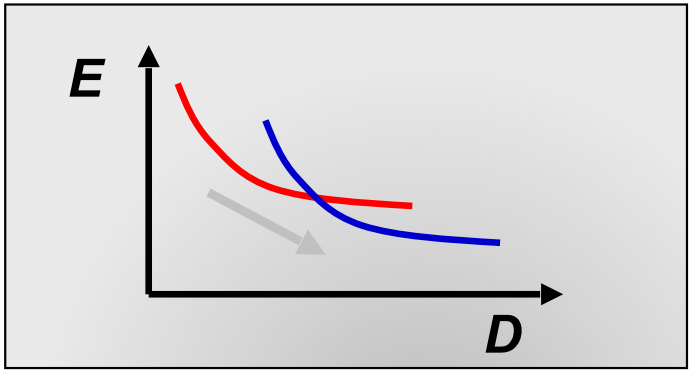


- Instead of metrics like EDP (Energy-Delay Product), this curve often provides information more directly
  - Ex1: What is minimum energy for XX performance?
  - Ex2: Over what range of performance is a new technique (dotted line) actually beneficial?

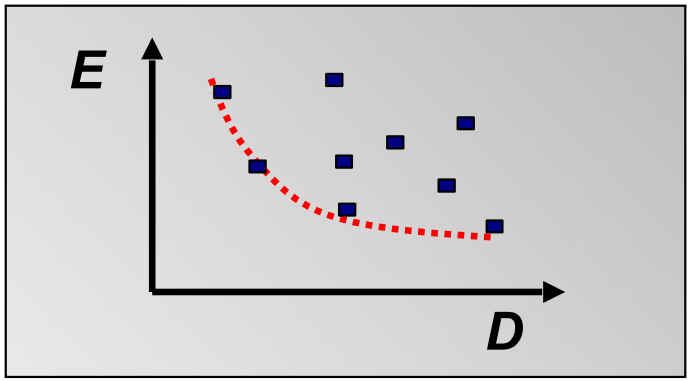
Expanding the Playing Field



Removing inefficiencies (1)

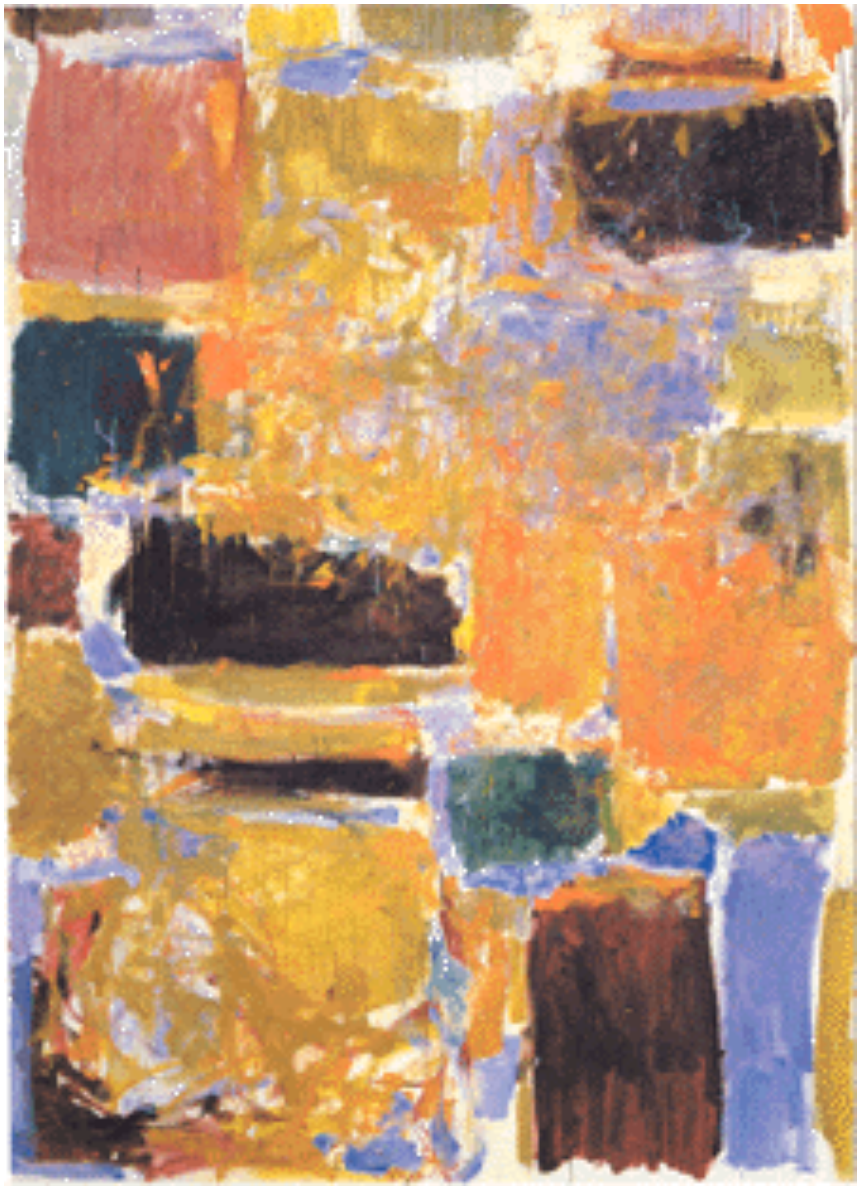


Alternative topologies (2)



Discrete options (3)

Architecture and system transformations and optimizations reshape the E-D curves

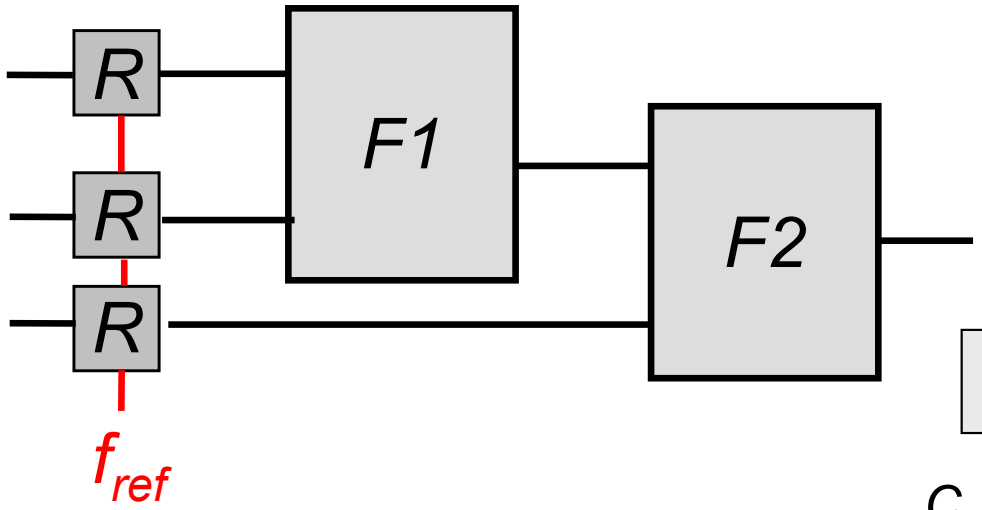


Voltage scaling: Parallelism and Pipelining

Reducing the Supply Voltage (while maintaining performance)

Concurrency:
trading-off clock frequency versus area to reduce power

Consider the following reference design



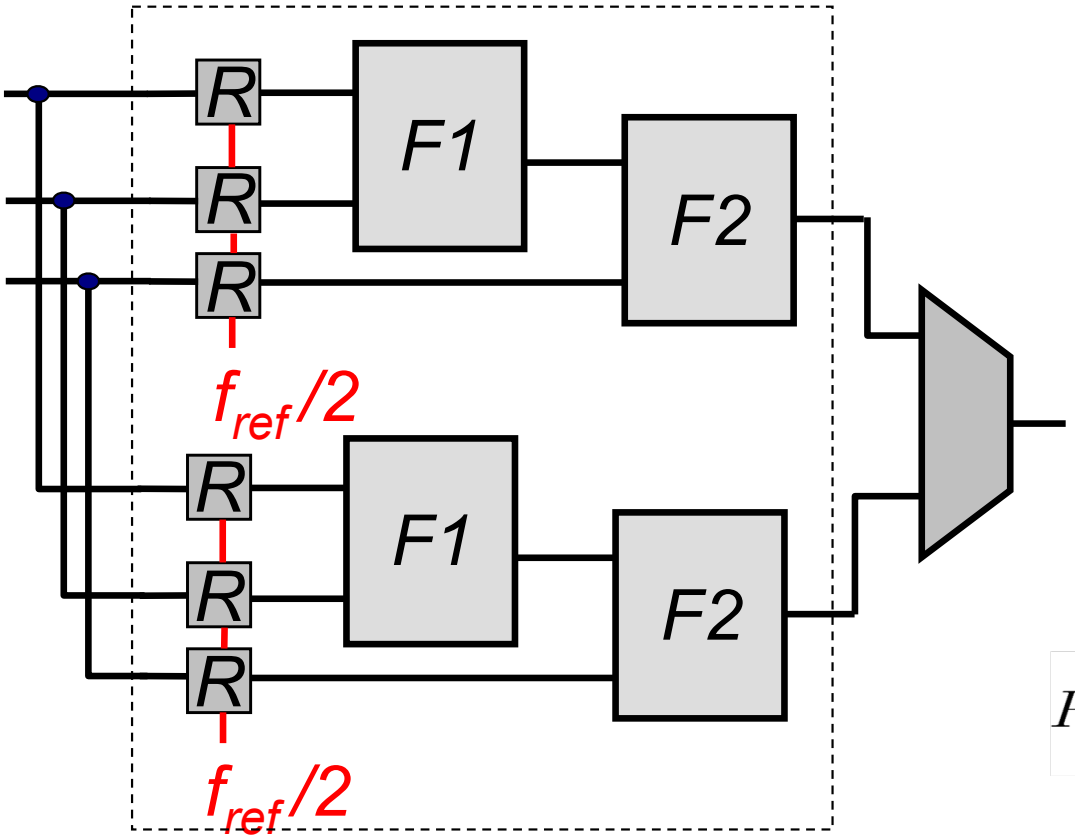
R: register,
F1,F2: combinational logic blocks (adders, ALUs, etc)

$$P_{ref} = C_{ref} \cdot V_{dd,ref}^2 \cdot f_{ref}$$

C_{ref}: average switching capacitance

[A. Chandrakasan, JSSC'92]

A Parallel Implementation



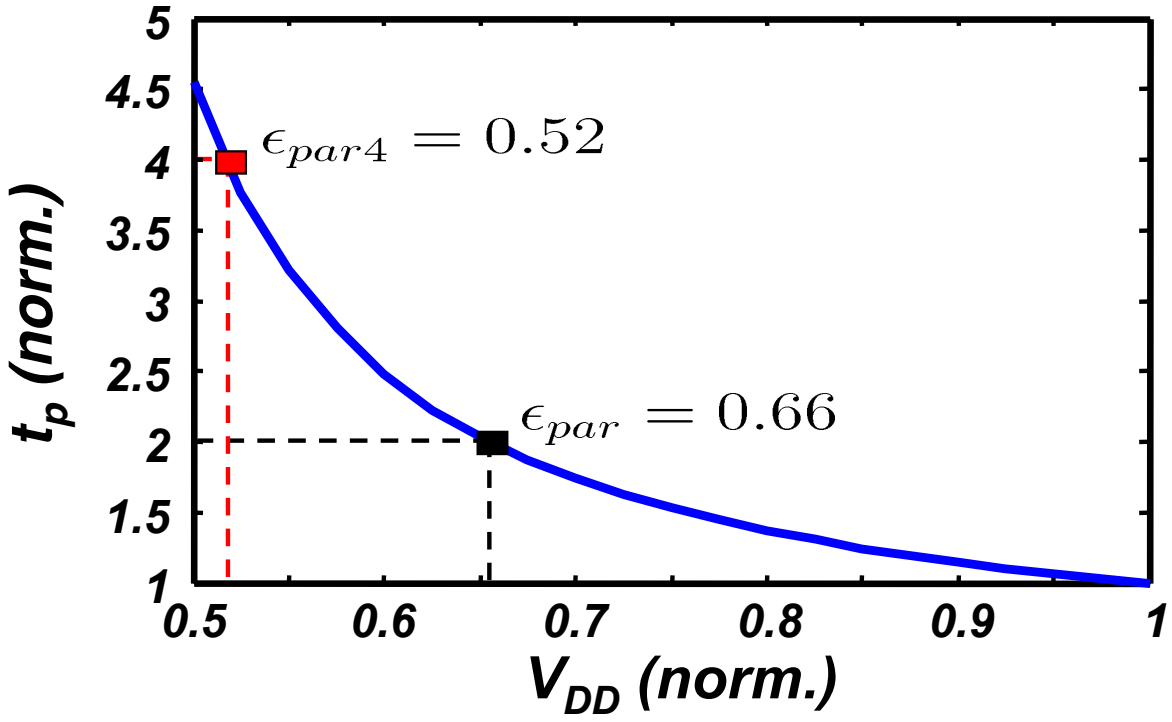
$$f_{par} = f_{ref}/2$$
$$C_{par} = (2 + ov_{par}) \cdot C_{ref}$$
$$V_{dd,par} = \epsilon_{par} \cdot V_{dd,ref}$$

Almost cancels

$$P_{par} = \epsilon_{par}^2 \cdot \left(\frac{2 + ov_{par}}{2}\right) \cdot P_{ref}$$

Running slower reduces required supply voltage
Yields quadratic reduction in power

Example: 90nm Technology

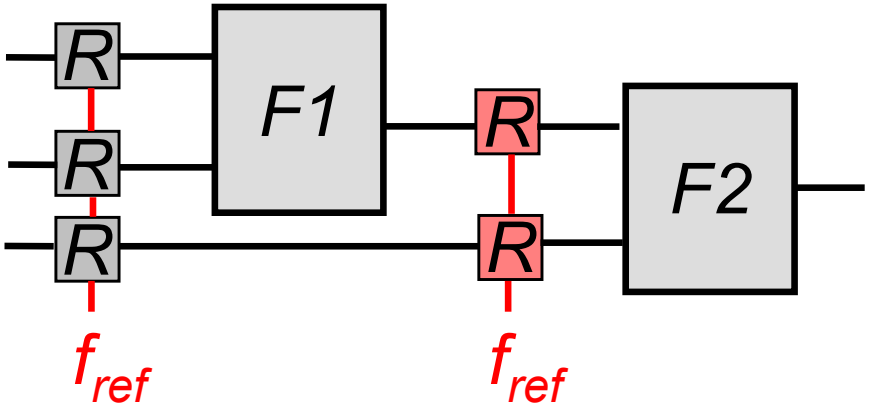


Assuming $ov_{par} = 7.5\%$

$$P_{par} = 0.66^2 \times \frac{2.075}{2} P_{ref} = 0.45 P_{ref}$$

$$P_{par4} = 0.52^2 \times \frac{4.15}{4} P_{ref} = 0.28 Pref$$

A Pipelined Implementation



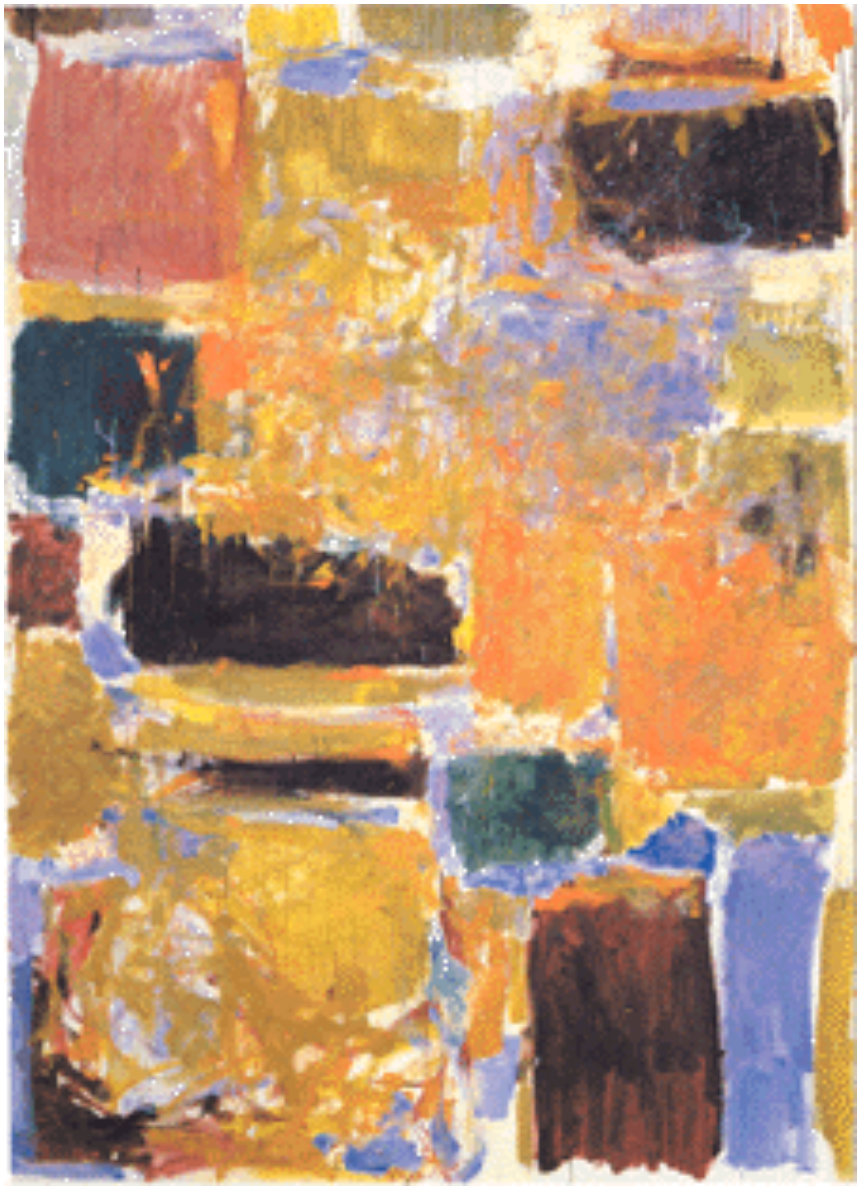
$$f_{pipe} = f_{ref}$$
$$C_{pipe} = (1 + ov_{pipe}) \cdot C_{ref}$$
$$V_{dd,pipe} = \epsilon_{pipe} \cdot V_{dd,ref}$$

$$P_{pipe} = \epsilon_{pipe}^2 \cdot (1 + ov_{pipe}) \cdot P_{ref}$$

Shallower logic reduces required supply voltage
(this example assumes equal V_{dd} for par / pipe designs)

Assuming
ov_{pipe} = 10%

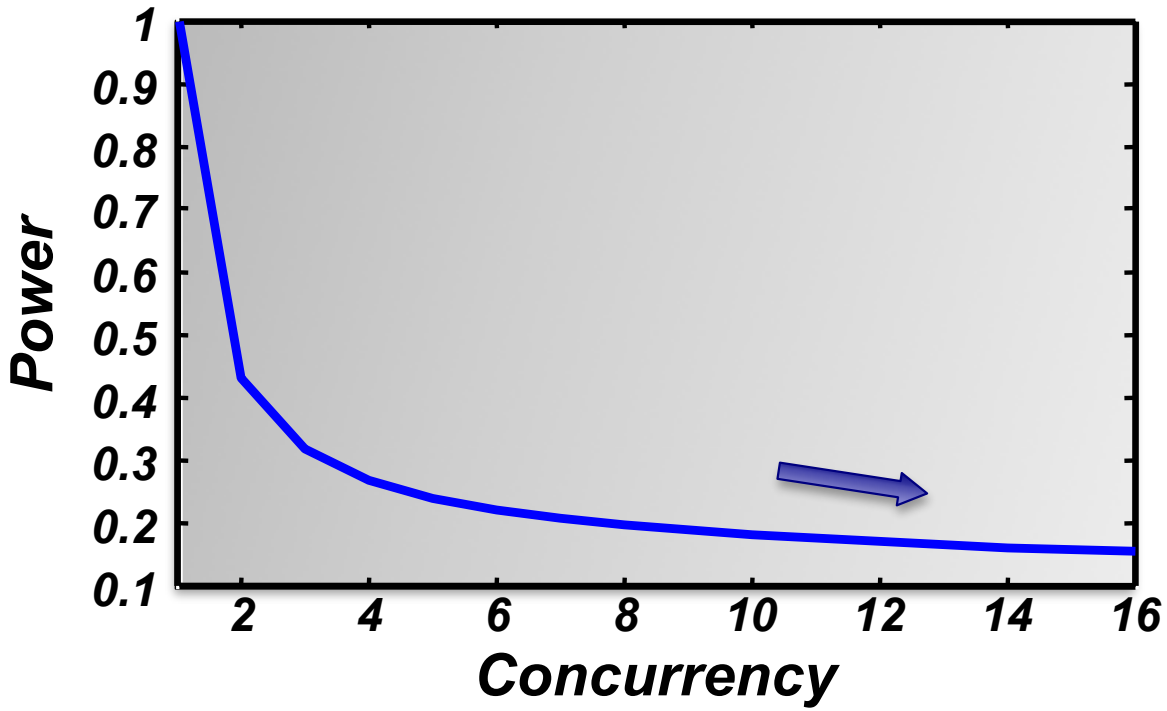
$$P_{pipe} = 0.66^2 \cdot 1.1 \cdot P_{ref} = 0.48 P_{ref}$$
$$P_{pipe4} = 0.52^2 \cdot 1.1 P_{ref} = 0.29 P_{ref}$$



Leveraging Concurrency

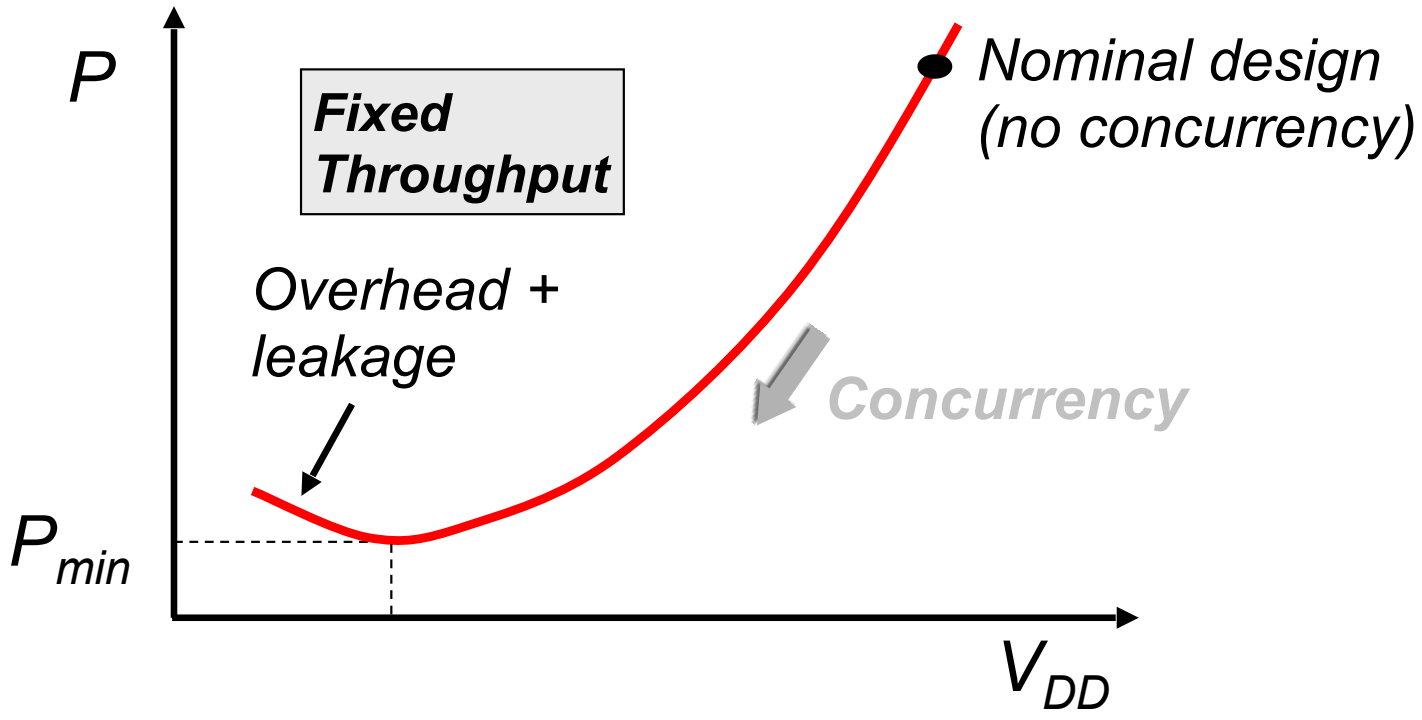
Increasing use of Concurrency Saturates

- Can combine parallelism and pipelining to drive V_{DD} down
- But, close to process threshold overhead of excessive concurrency starts to dominate



Assuming constant % overhead

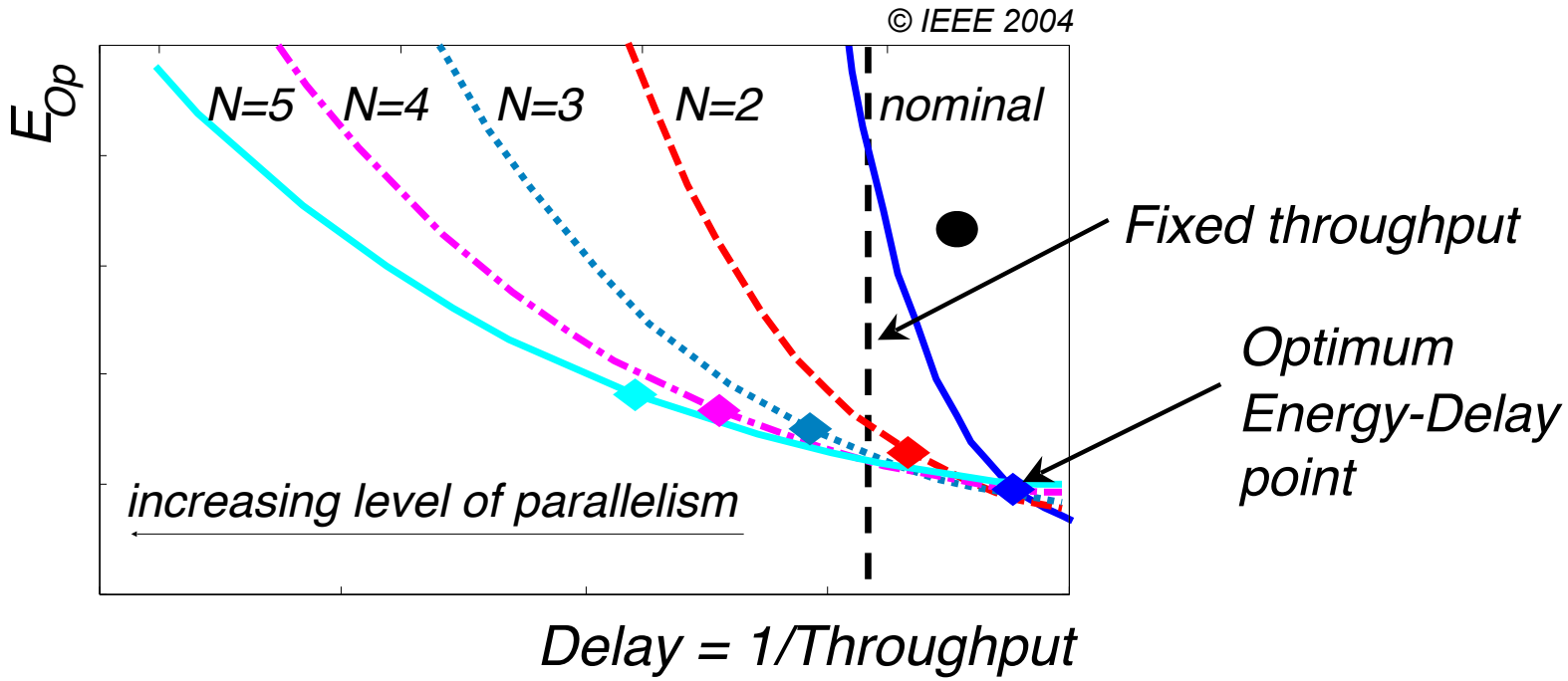
Reducing V_{TH}



Only option: Reduce V_{TH} as well!

But: Must consider Leakage …

Mapping into the Energy-Delay Space



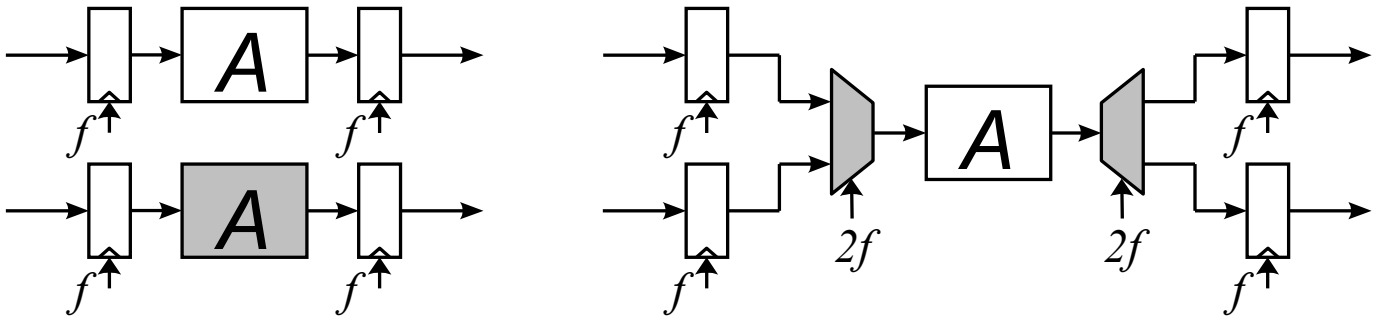
- For each level of performance, optimum amount of concurrency
- Concurrency only energy-optimal if requested throughput larger than optimal operation point of nominal function

[Ref: D. Markovic, JSSC'04]

What if the Required Throughput is Below Minimum?

(that is, at no concurrency)

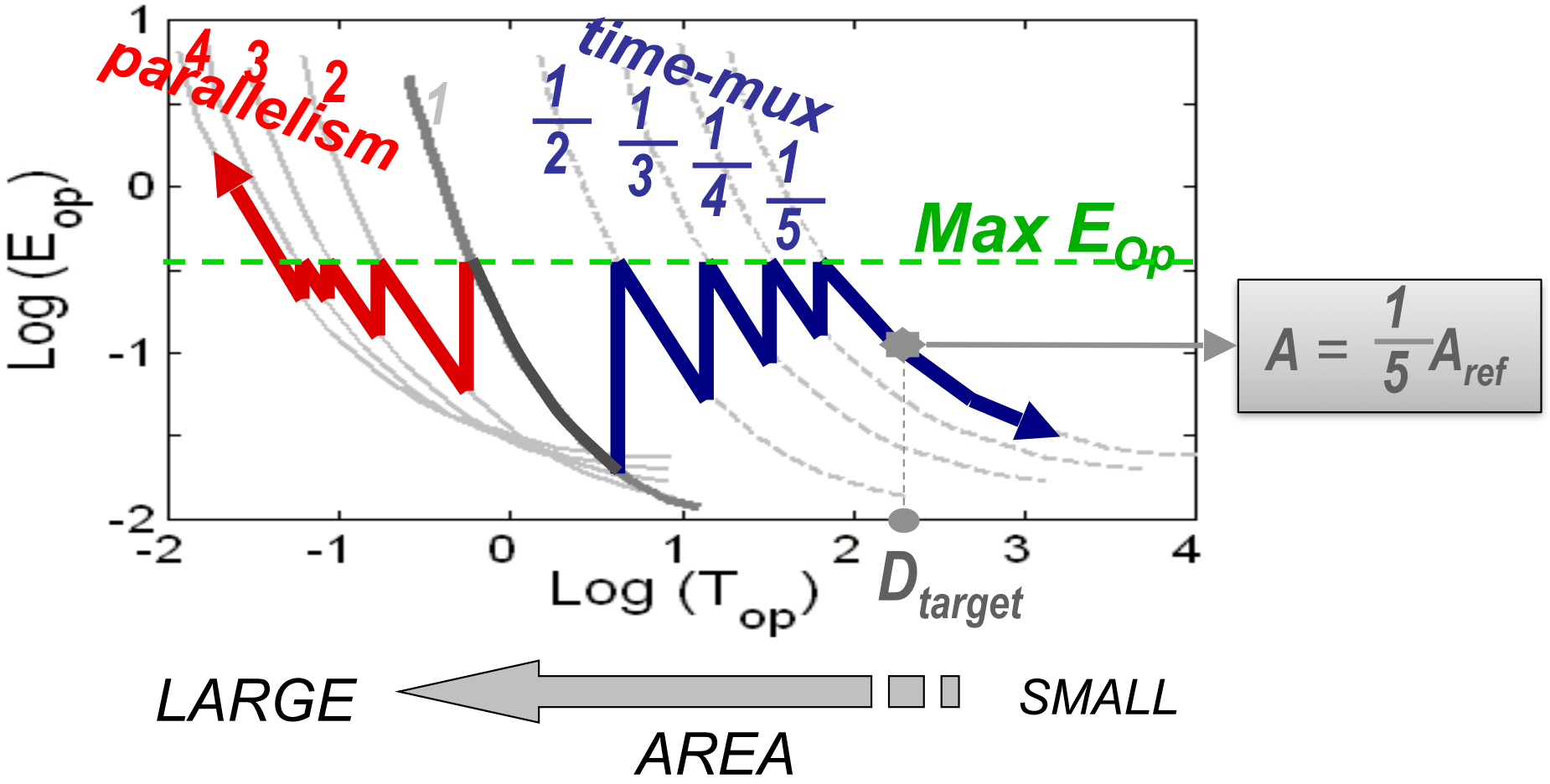
Introduce Time-Multiplexing!



Absorb unused time slack by increasing clock frequency (and voltage …)
Again comes with some area and capacitance overhead!

Concurrency and Multiplexing Combined

Data for 64-b ALU



Some Energy-Inspired Design Guidelines

- **For maximum performance under energy constraint**
  - Maximize use of concurrency at the cost of area
- **For minimum energy under performance constraint**
  - Maximize use of concurrency at the cost of area
- **For minimum area under performance constraint**
  - Least amount of concurrency that meets performance goals

Clock Gating and FPGAs

- We can't shut down unused blocks of our logic...
  - Although the FPGA designers do optimize for reducing leakage...
- The FPGA clocks are distributed using dedicated clock buffers
  - These buffers have multiple inputs: 2 separate clocks and a clock enable
- Can use this to "power off" large design blocks by turning off the clock

More On Xilinx Clock Buffers...

- Have 12 dedicated clock lines
  - Driven by global buffers called BUFG
- Two interesting additions:
  - Has ***two*** clock inputs:
    Allows switching between the two clocks
  - Each clock also can have a global enable:
    Allows turning off large blocks of logic by turning off the clock

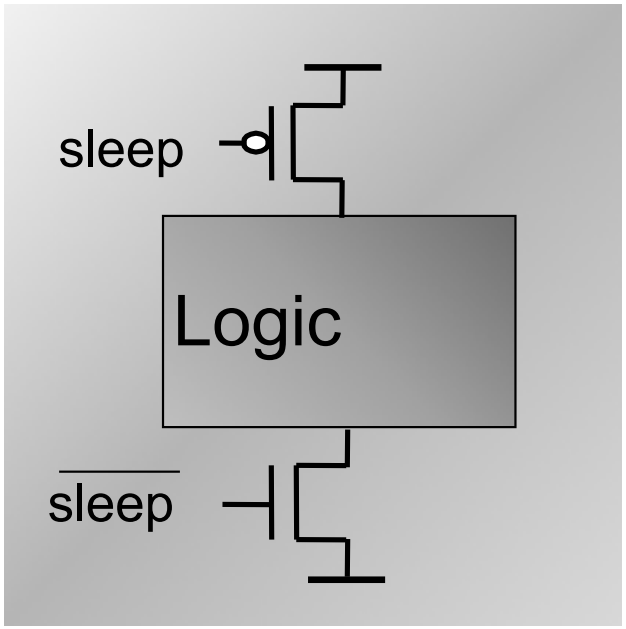
The Leakage Challenge – Power in Standby

- With clock-gating employed in most designs, leakage power has become the dominant standby power source
- With no activity in module, leakage power should be minimized as well
  - Constant ratio between dynamic and static power desirable
- Challenge – how to disable unit most effectively given that no ideal switches are available

Standby Static Power Reduction Approaches

- Multi-V_{th} design
- Power gating
- Body biasing
- Supply voltage ramping

Power Gating



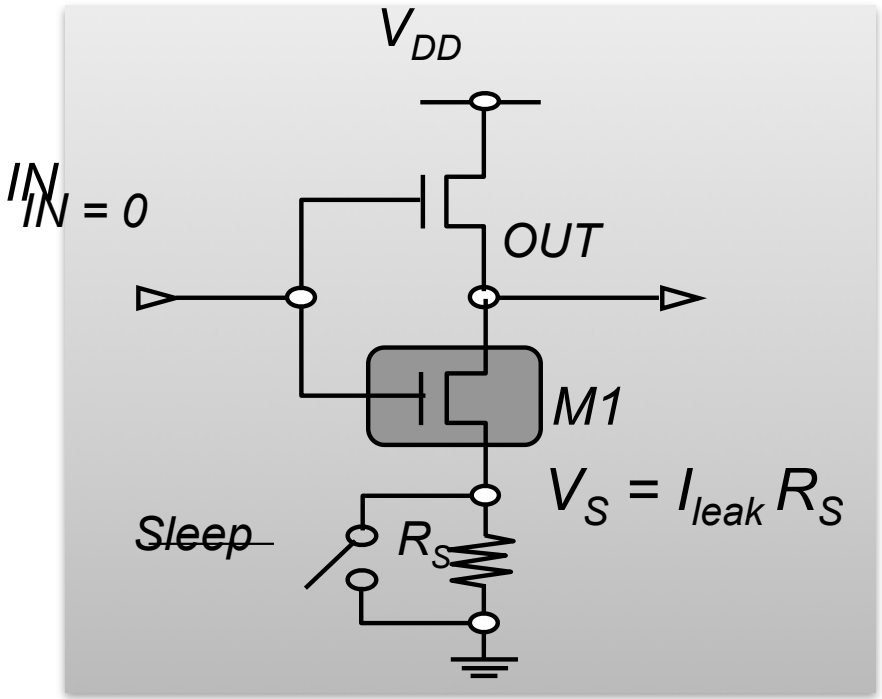
Disconnect module from supply rail(s) during standby

- Footer or header transistor, or both
- Most effective when high V_T transistors are available
- Easily introduced in standard design flows
- But … Impact on performance

Very often called “MTCMOS” (when using high- and low- threshold devices)

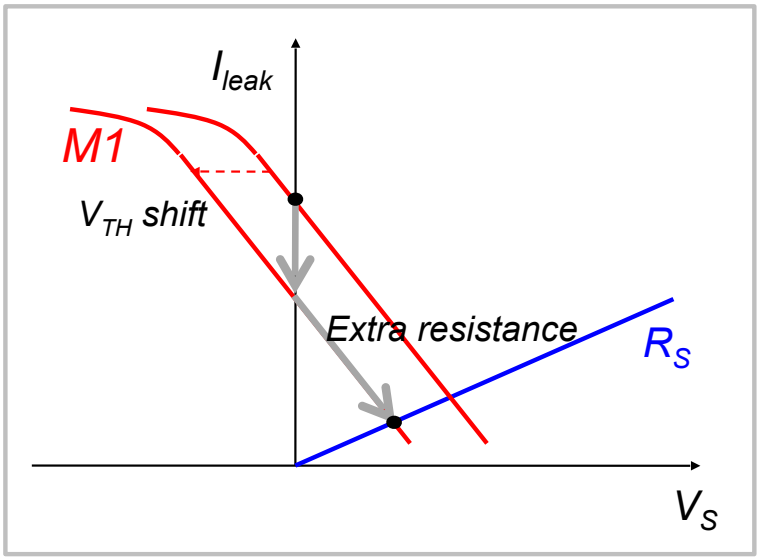
[Ref: T. Sakata, VLSI’93; S. Mutoh, ASIC’93]

Power Gating — Concept



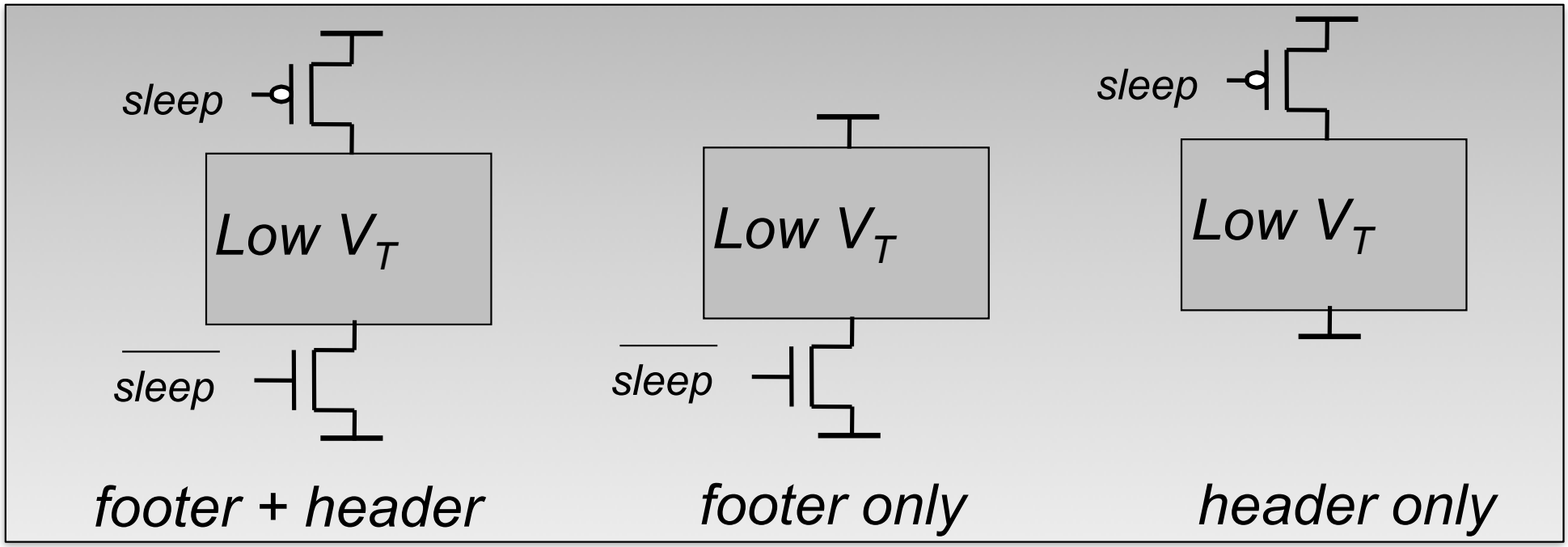
Leakage current reduces because

- Increased resistance in leakage path
- Stacking effect introduces source biasing



(similar effect at PMOS side)

Power Gating Options



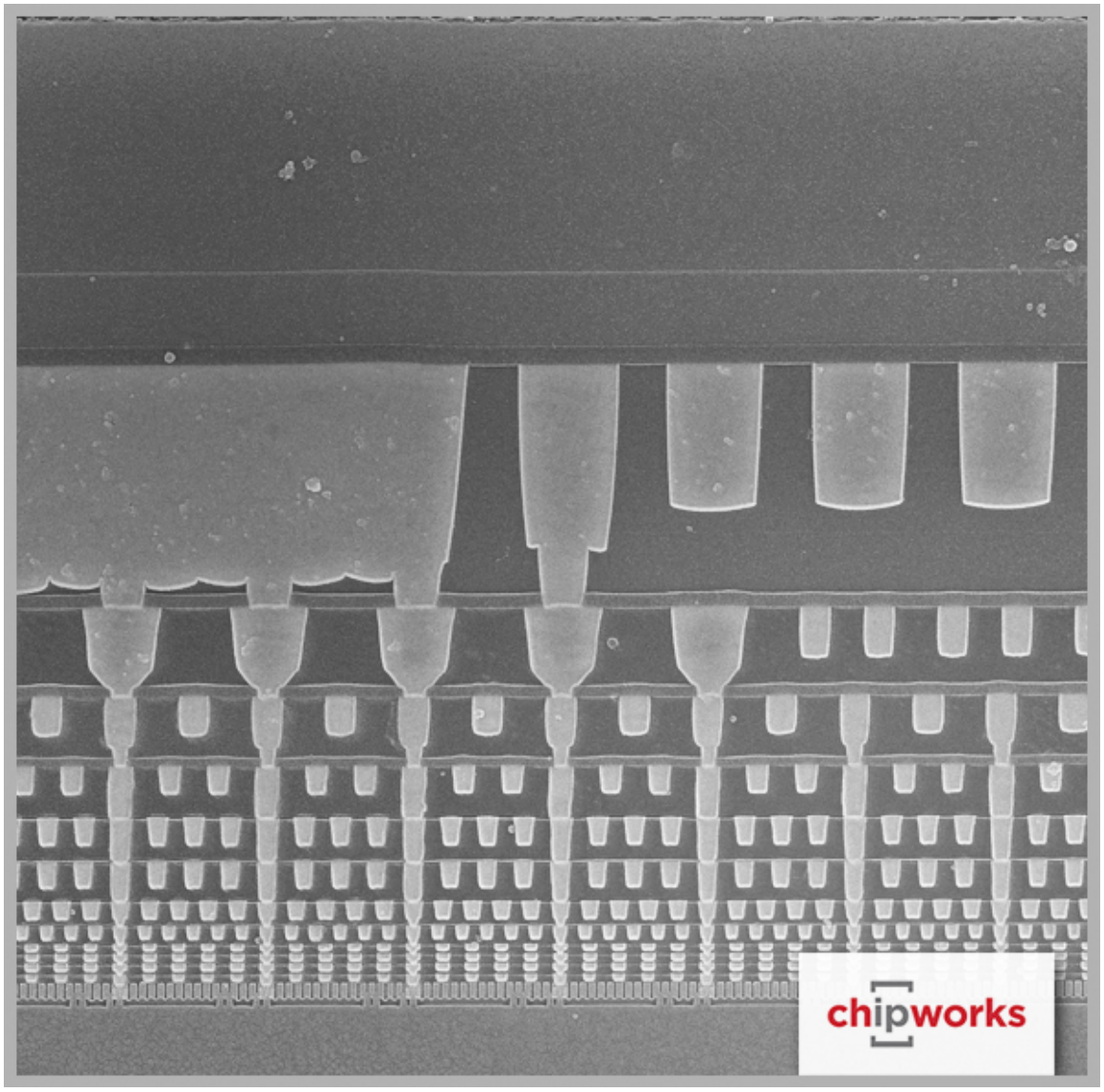
- NMOS sleeper transistor more area efficient than PMOS
- Leakage reduction more effective (under all input patterns) when both footer and header transistors are present

"Hurry Up And Wait"

- Often end up in a position where you ***can't*** reduce the supply voltage
  - FPGAs
  - Constrained by some other voltage in an ASIC
- So slowing down doesn't help your ***energy*** consumption
  - But we can pause and use less energy that way if we can shut down swaths of logic

Wires

The importance of wires

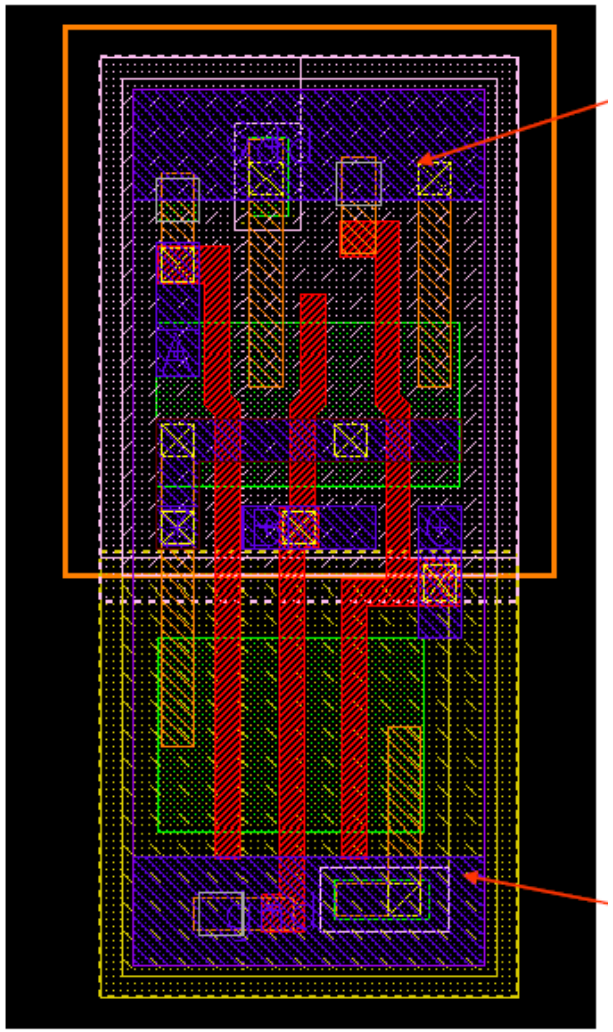


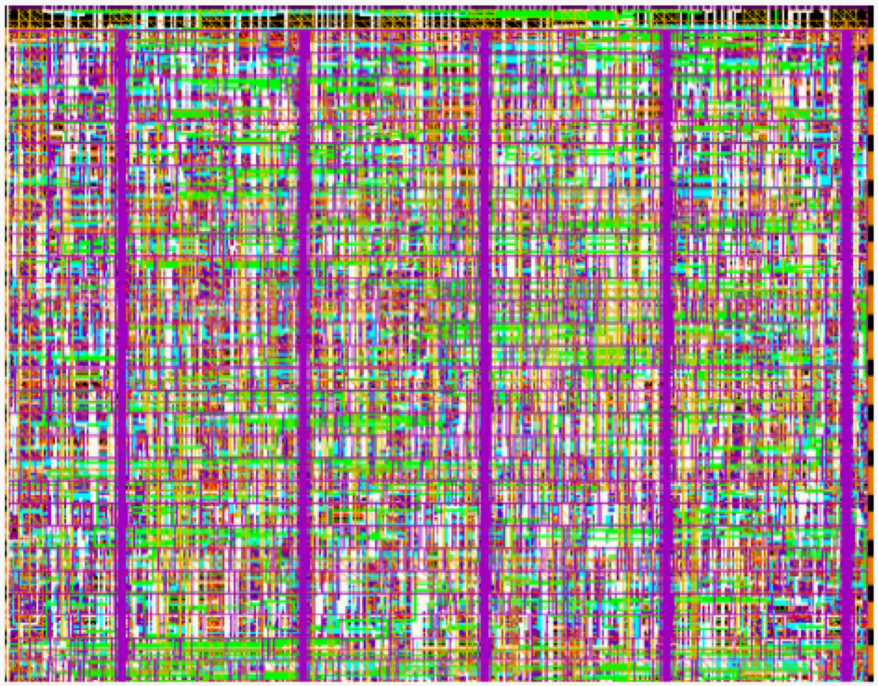
Intel 14nm process

Side view of wiring layers

How Chips are Made (Whiteboard)

Wires dominate in modern designs

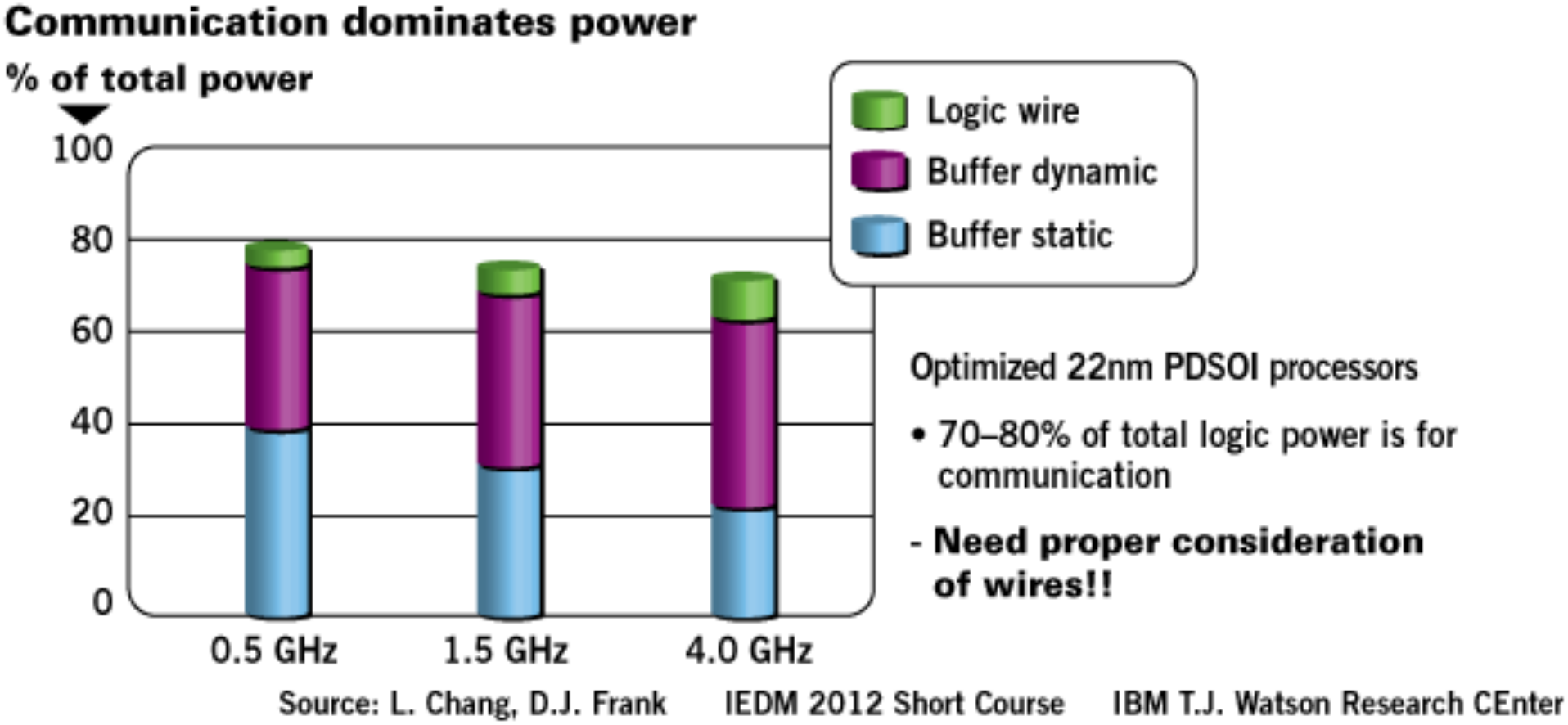




Modern sub-100nm processes "Transistors are free things that fit under wires"

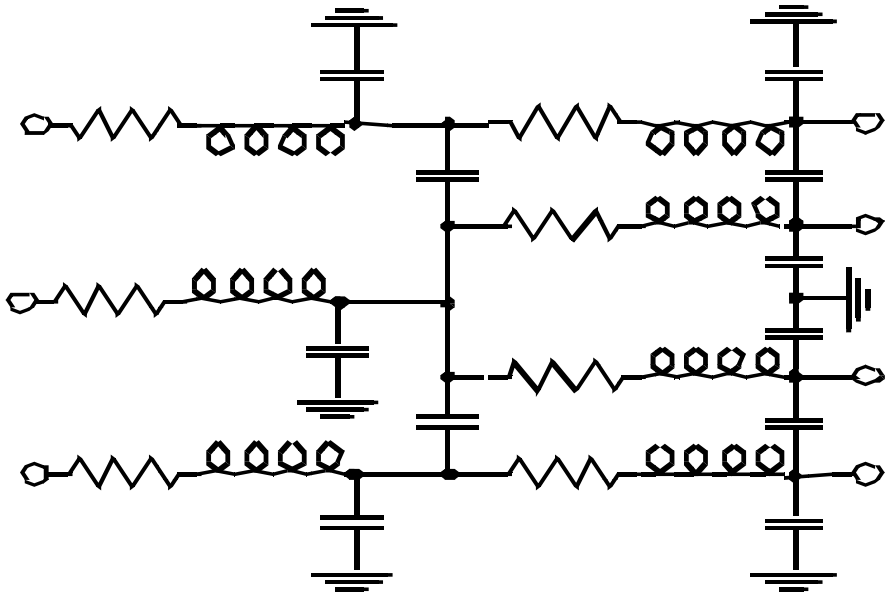
- Cells sized by the number of routing tracks they can pass

Wires dominate energy as well

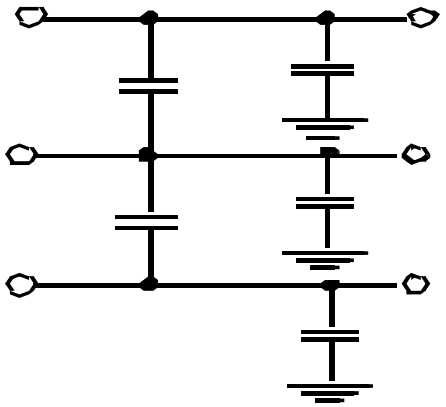


- Most of the energy goes to driving the wires

Wire Models



All-inclusive model

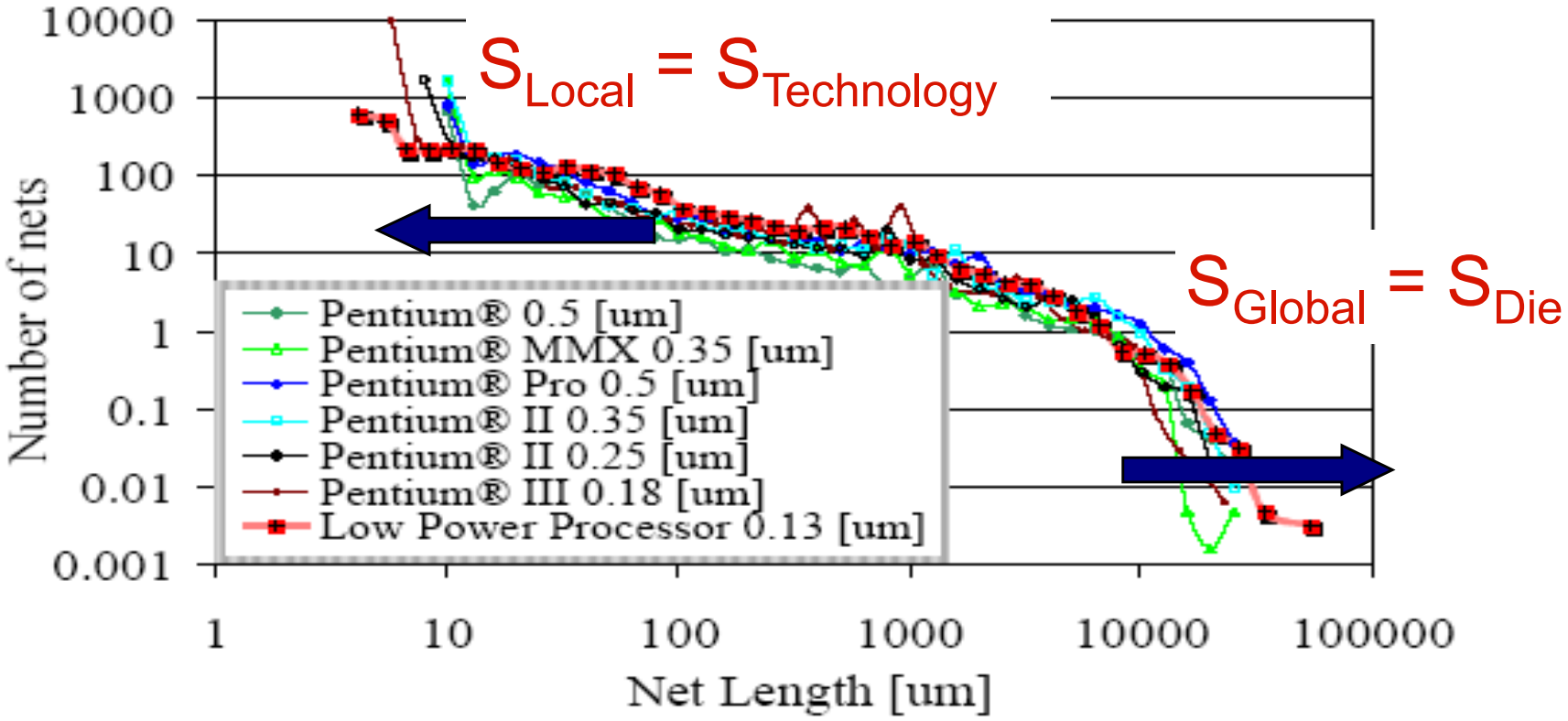


Capacitance-only

Impact of Interconnect Parasitics

- Interconnect and its parasitics can affect all of the metrics we care about
  - Cost, reliability, performance, power consumption
- Parasitics associated with interconnect:
  - Capacitance
  - Resistance
  - Inductance

Interconnect Length Distribution



From Magen et al., "Interconnect Power Dissipation in a Microprocessor"

Wires Onto the Board...

- Chips are only ***part*** of a digital system
  - You have all the components to create a circuit board
- A circuit board is effectively a wiring fabric
  - Composed of layers of signal traces interposed with planes and insulation
  - Layers connected together by vias.