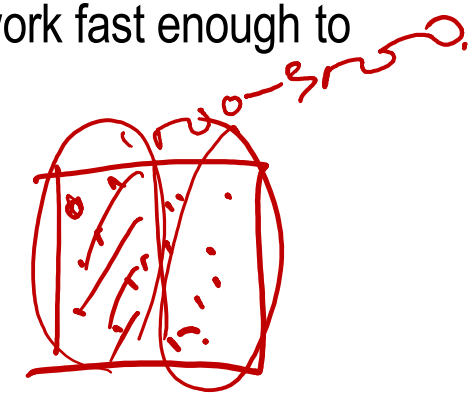
Lecture 10:
Pipeline – Improving Energy-Efficiency

“Creativity is just connecting things.
When you ask creative people how they did something,
they feel a little guilty because they didn’t really do it,
they just saw something.
It seemed obvious to them after a while.”

Steve Jobs

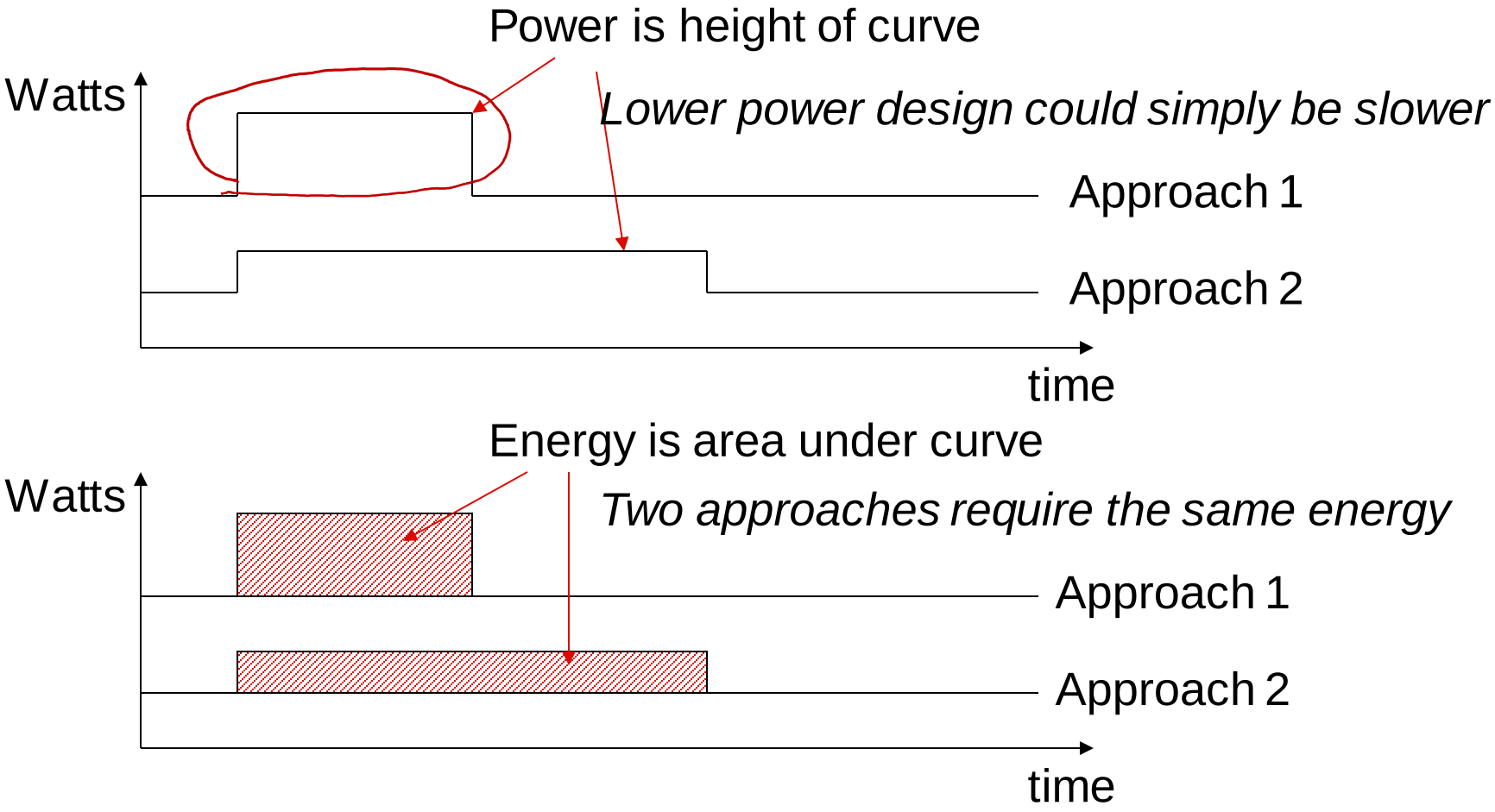
Review: Power and Energy Figures of Merit

- power in Watts
  - ✓ (energy/time) poses constraints, i.e., processor can only work fast enough to max out the power delivery or cooling solution
  - ✓ peak power
    - determines power ground wiring designs
    - sets packaging/cooling limits
    - impacts signal noise margin and reliability analysis



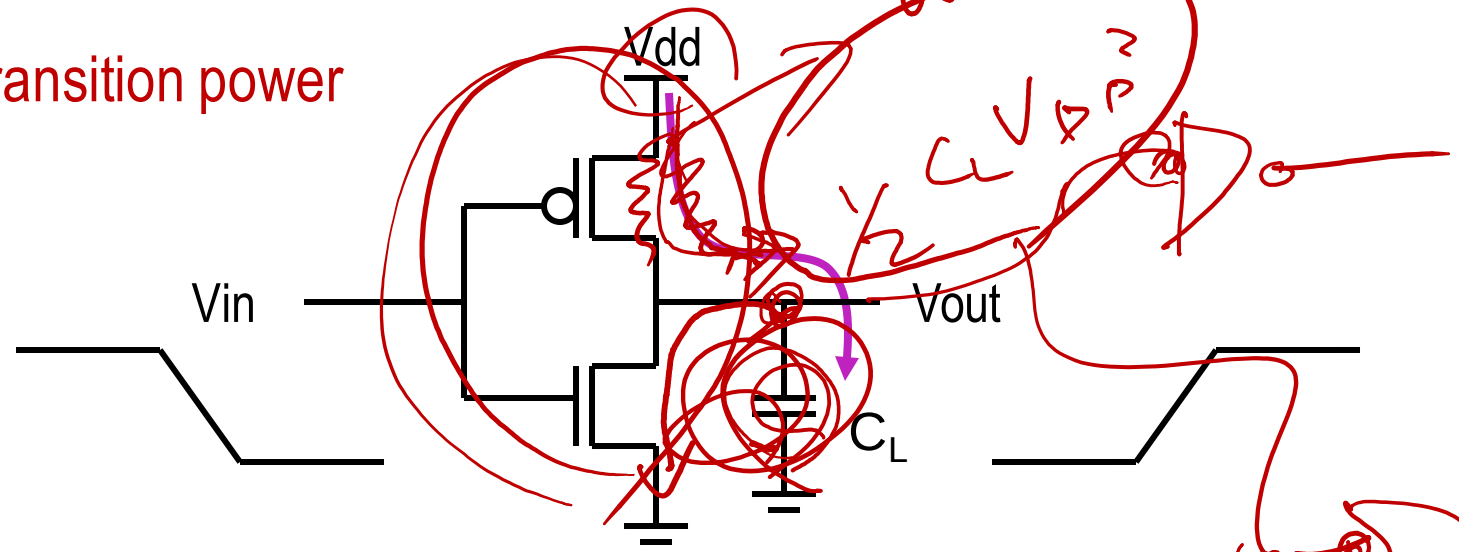
- energy in Joules
  - ✓ ultimate metric, i.e., the true "cost" of performing a fixed task
  - ✓ energy = power × delay
    - Joules = Watts × seconds
    - lower energy number means less power to perform a computation at the same frequency
- example:
  - ✓ if processor A consumes 1.2× power of processor B, but finishes the task in 30% less time, which processor is more energy efficient and by how much?

Review: Power versus Energy



(Dynamic) Power Dissipation

transition power



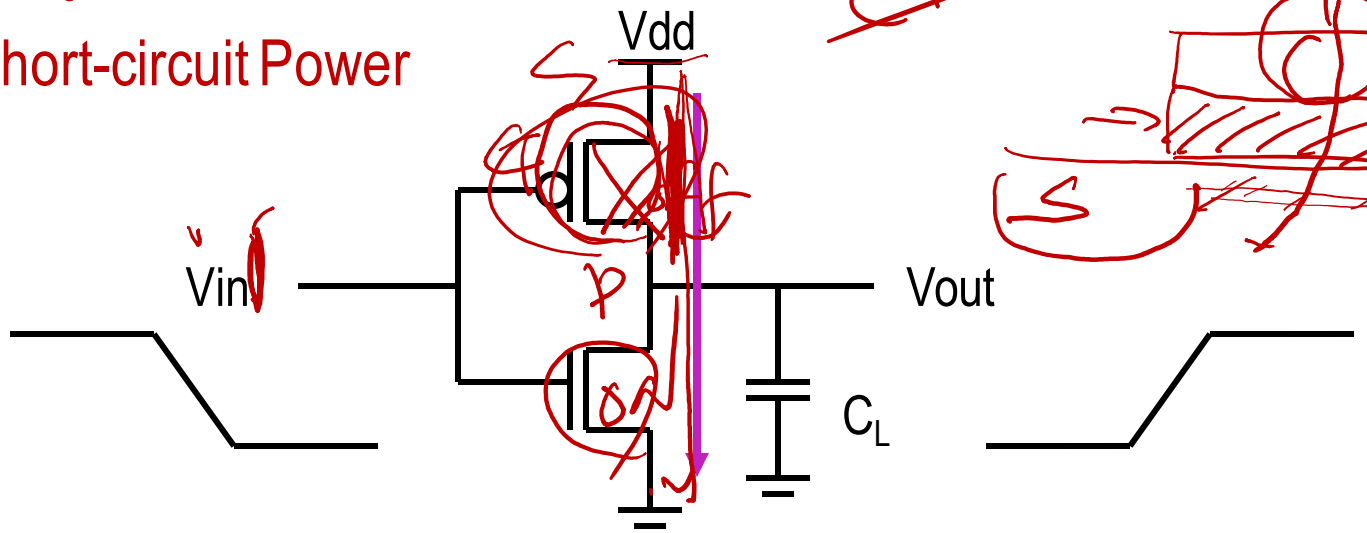
$$\text{energy/transition} = C_L \times V_{DD}^2 \times P_{0/1 \rightarrow 1/0}$$

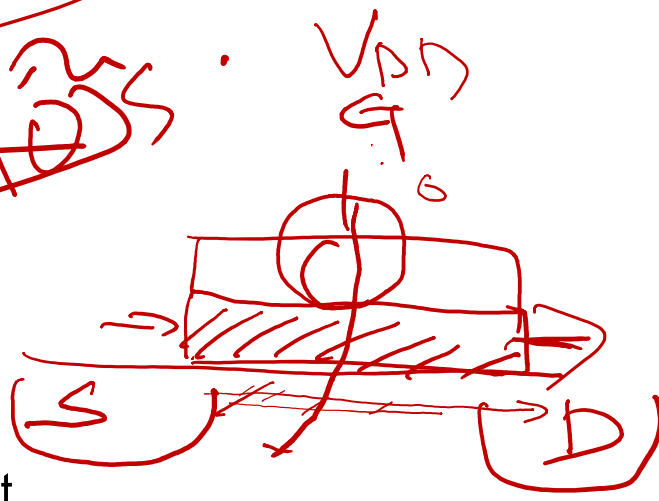
$$\text{power} = C_L \times V_{DD}^2 \times f$$

(Short Circuit) Power Dissipation

Leakage power dissipation

Short-circuit Power





$$\text{Energy/transition} = t_{sc} \times V_{DD} \times I_{peak} \times P_{0/1 \rightarrow 1/0}$$

$$\text{Power} = t_{sc} \times V_{DD} \times I_{peak} \times f$$

10%

CMOS Energy & Power Equations

$$E = C_L V_{DD}^2 P_{0\to1} + t_{sc} V_{DD} I_{peak} P_{0\to1} + V_{DD} I_{leakage}$$

$$f_{0\to1} = P_{0\to1} * f_{clock}$$

$$P = C_L V_{DD}^2 f_{0\to1} + t_{sc} V_{DD} I_{peak} f_{0\to1} + V_{DD} I_{leakage}$$

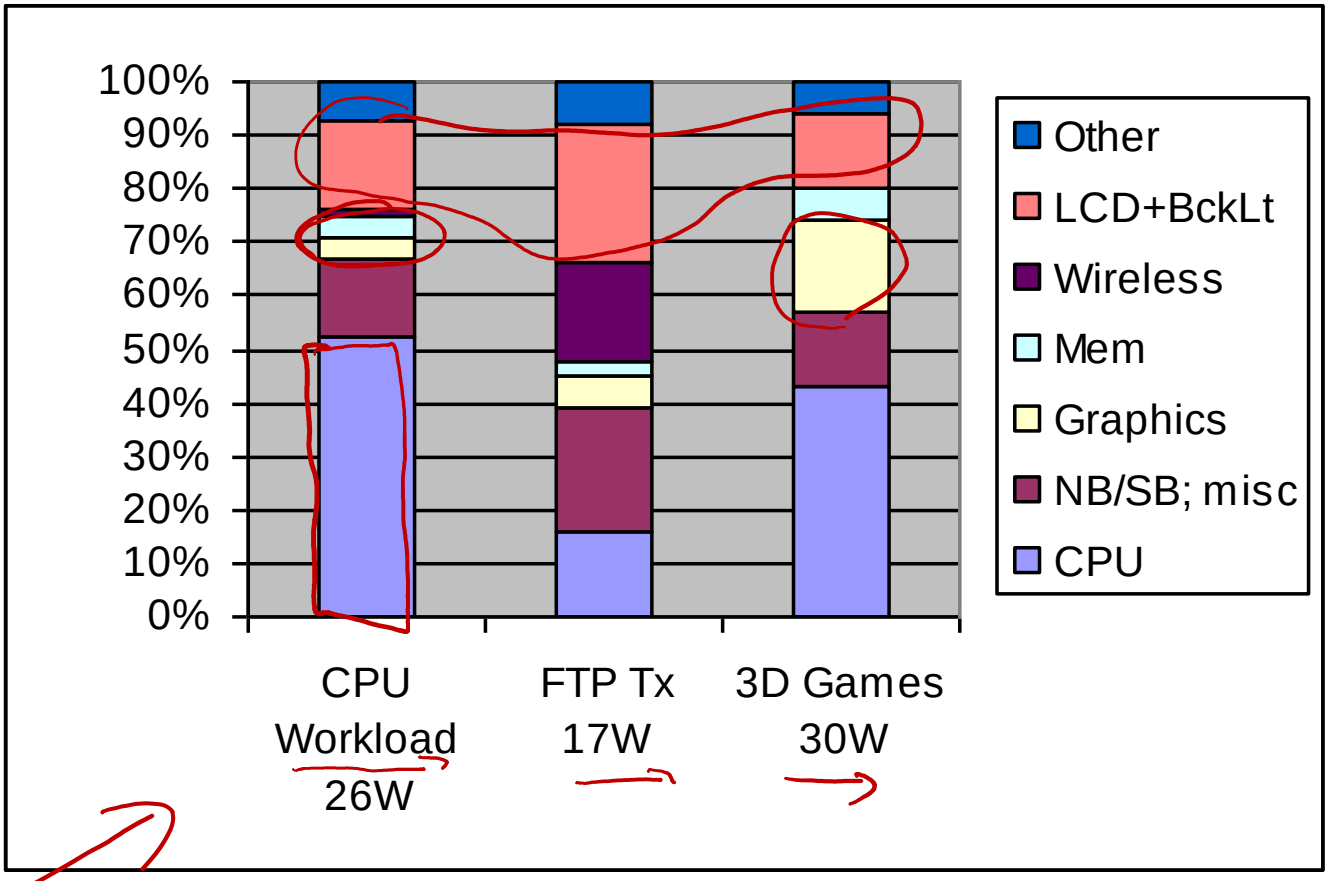
- dynamic power (≈ 40 - 70% today and decreasing relatively)
- short-circuit power (≈ 10 % today and decreasing absolutely)
- leakage power (≈ 20 – 50 % today and increasing)

Low Power Design Techniques

Three main classes of methods to reduce energy:

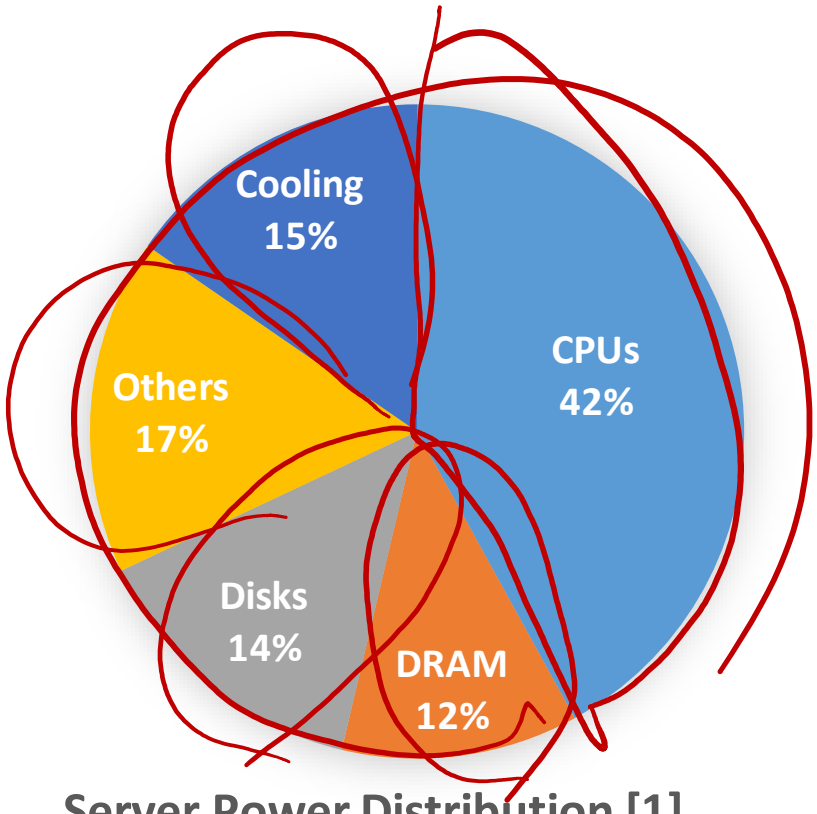
- cheating
  - ✓ reducing the performance of the design
- reducing waste
  - ✓ stop using energy for stuff that does not produce results
  - ✓ stop waiting for stuff that you don't need (parallelism)
- problem reformulation
  - ✓ reduce work (less energy and less delay)

Power Consumption of Modern Laptop



Src: Mahesri et al., U of Illinois, 2004

Power consumption of Server

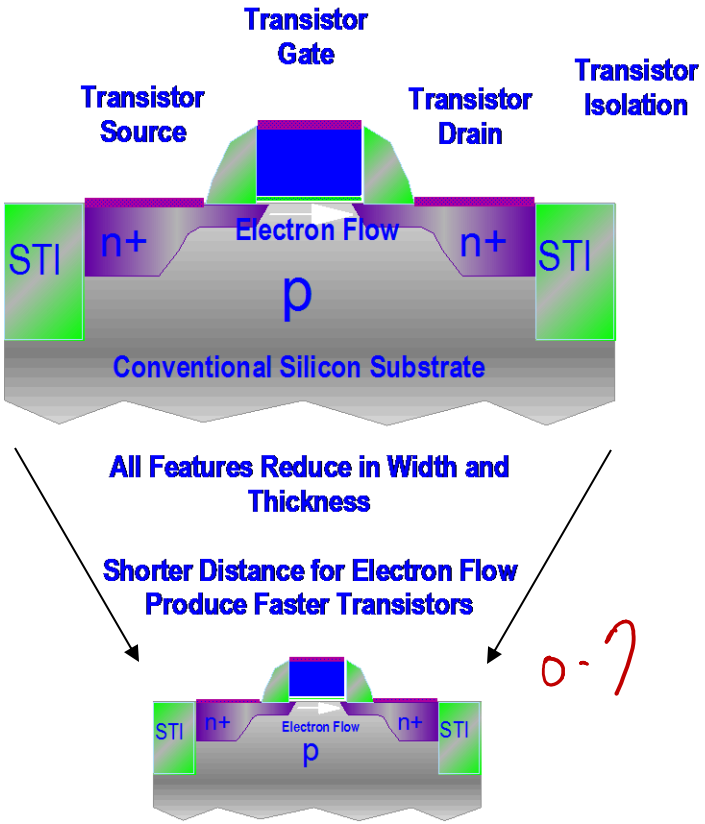


Server Power Distribution [1]

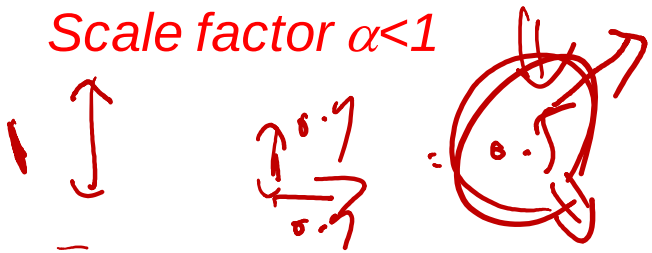
Designing within limits: power & energy

- thermal limits (for most parts self-heating is a substantial thermal issue)
  - ✓ package cost (4-5W limit for cheap plastic package, 50-100W/sq-cm air cooled limit, 5k-7.5kW 19" rack)
  - ✓ device reliability (junction temp > 125C quickly reduces reliability)
  - ✓ performance (25C -> 105C loss of 30% of performance)
- distribution limits
  - ✓ substantial portion of wiring resource, area for power dist.
  - ✓ higher current => lower R, greater dI/dt => more wire, decap
  - ✓ package capable of low impedance distribution
- energy capacity limits
  - ✓ AA battery ~1000mA.hr => limits power, function, or lifetime
- energy cost
  - ✓ energy for IT equipment large fraction of total cost of ownership

Review of Constant Field Scaling

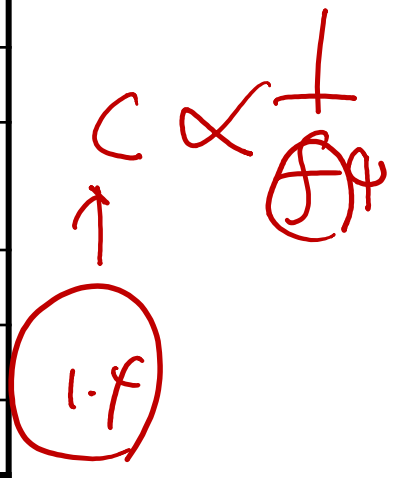


Scale factor α<1



Parameter	Value	Scaled Value
Dimensions	L, W, Tox	αL, αW, αTox
Dopant concentrations	Na, Nd	Na/α, Nd/α
Voltage	V	αV
Field	E	E
Capacitance	C	αC
Current	I	αI
Propagation time (~CV/I)	t	αt
Power (VI)	P	α^2P
Density	d	d/α^2
Power density	P/A	P/A

These are distributions… how do the σs scale?



CMOS Circuit Delay and Frequency

$$P = \tfrac{1}{2} C_{sw} V_{dd} \Delta V f + I_{st} Vdd + I_{static} Vdd$$

digital system frequency determined by:

- **sum of propagation delays across gates in "critical path" --**
  - **each gate delay, includes time to charge/discharge load thru one or more FETs and interconnect delay to distribute the signal to next gate input.**

$$T_d = kCV/I$$
$$= kCV/(V_{dd}-V_t)^{\alpha}$$

Sakuri α-power law model of delay

each technology generation, gate delay reduced about 30% (ITRS '05)

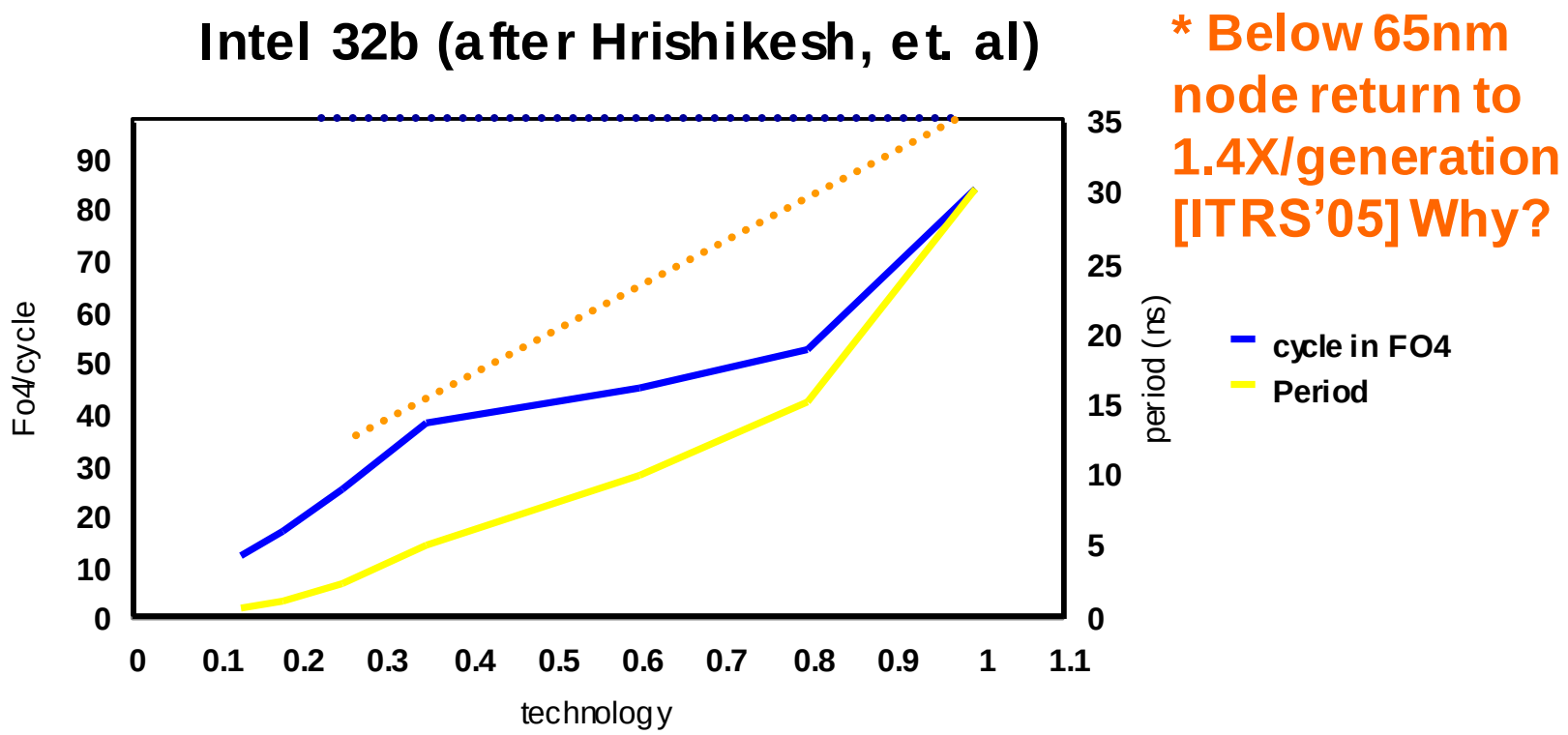
Microprocessor Frequency

$$P = \tfrac{1}{2} C_{sw} V_{dd} \Delta V f + I_{st} V_{dd} + I_{static} V_{dd}$$

In practice the trend is:

Frequency increasing by 2X (delay decreasing by 50%), not the 1.4X (30%) for constant field scaling for 1um to 65nm node (src: ITRS '01).

Why? decreasing logic/stage and increased pipeline depth.



*** Below 65nm node return to 1.4X/generation [ITRS'05] Why?**

Dynamic Energy

Energy dissipated for either output transition consumes:

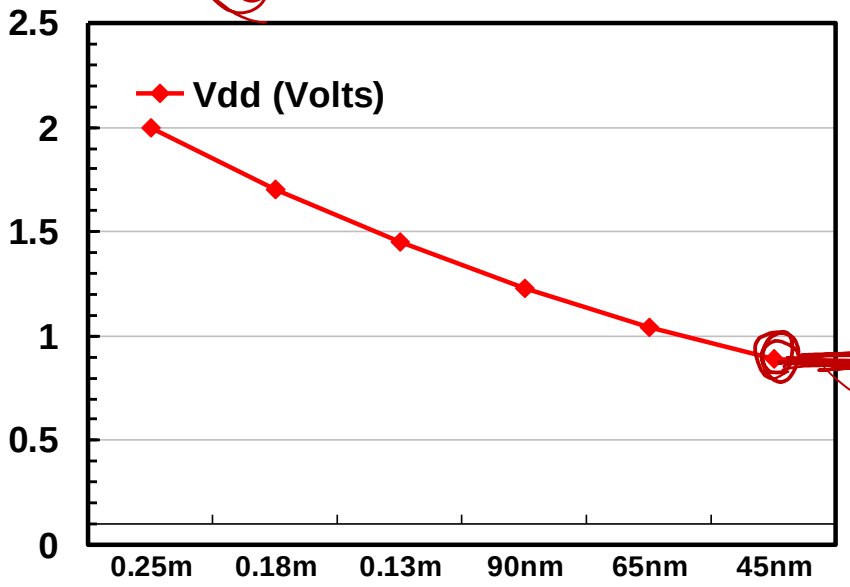
$$\tfrac{1}{2} C_L V_{dd}^2$$

$$P = \tfrac{1}{2} C_{sw} V_{dd} \Delta V_f + I_{st} V_{dd} + I_{static} V_{dd}$$

Gate level energy consumption should improve as α^3 under constant field scaling, but….

Supply Voltage Trend

$$P = \tfrac{1}{2} C_{sw} V_{dd} \Delta V f + I_{st} Vdd + I_{static} Vdd$$



Nominal voltage

Slow decline to 0.7V in 22nm (some think nothing below 0.9V for HP uProcs)

each generation, voltage has decreased 0.85x, not 0.7x for constant field.

Thus, energy/device is decreasing by 50% rather than 65%

Active-Power Reduction Techniques

$$P = \tfrac{1}{2} C_{sw} V_{dd} \Delta V f + I_{st} V_{dd} + I_{static} V_{dd}$$

- active power can be reduced through:
  - ✓ capacitance minimization
    - power/performance in sizing
    - clock-gating
    - glitch suppression
    - hardware accelerators
    - system-on-a-chip integration
  - ✓ voltage minimization
    - (dynamic) voltage-scaling
    - low swing signaling
  - ✓ frequency minimization
    - (dynamic) frequency-scaling

Capacitance minimization

$$P = \tfrac{1}{2} C_{sw} V_{dd} \Delta V f + I_{st} V_{dd} + I_{static} V_{dd}$$

- only the devices (device width) used in the design consume active power!

Functional Clock Gating

$$P = \tfrac{1}{2} C_{sw} V_{dd} \Delta V f + I_{st} V_{dd} + I_{static} V_{dd}$$

- 25-50% of power consumption due to driving flip-flops
- utilization of most latches is low (~10-35%)
- gate off unused flip-flops and associated logic:
  - ✓ unit level clock gating – turn off clocks to FPU, MMX, Shifter, L/S unit, … at clk buffer or splitter
  - ✓ functional clock gating – turn off clocks to individual latch banks – forwarding latch, shift-amount register, overflow logic & latches, …qualify (AND) clock to latch
- asynch is the most aggressive gating – but is it efficient?

Glitch Suppression

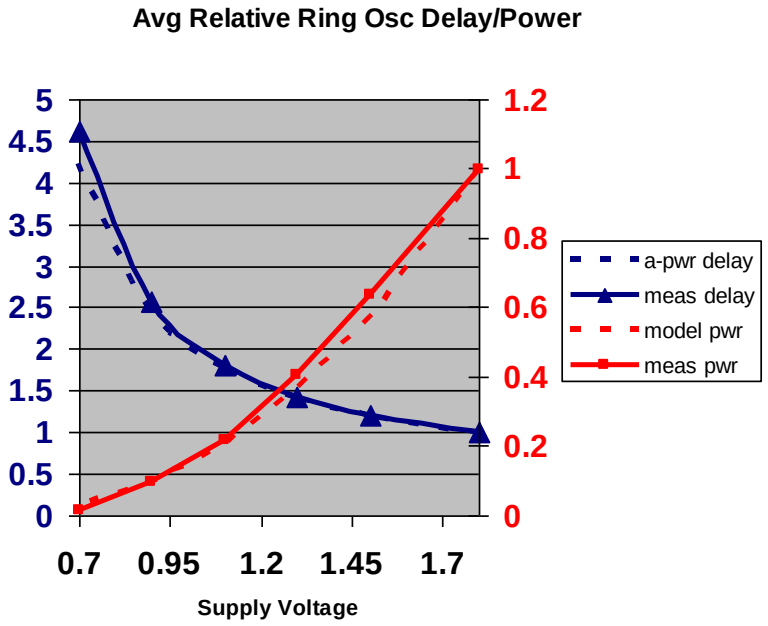
$$P = \tfrac{1}{2} C_{sw} V_{dd} \Delta V f + I_{st} V_{dd} + I_{static} V_{dd}$$

- glitches can represent a sizeable portion of active power, (up to 30% for some circuits in some studies)
- basic mechanisms for avoidance?
  - ✓ use non-glitching logic, e.g. domino
  - ✓ add redundant logic to avoid glitching hazards
    - Increases cap, testability problems
  - ✓ adjust delays in the design to avoid
    - shouldn't EDA tools do this already if it is possible?

Voltage Scaling

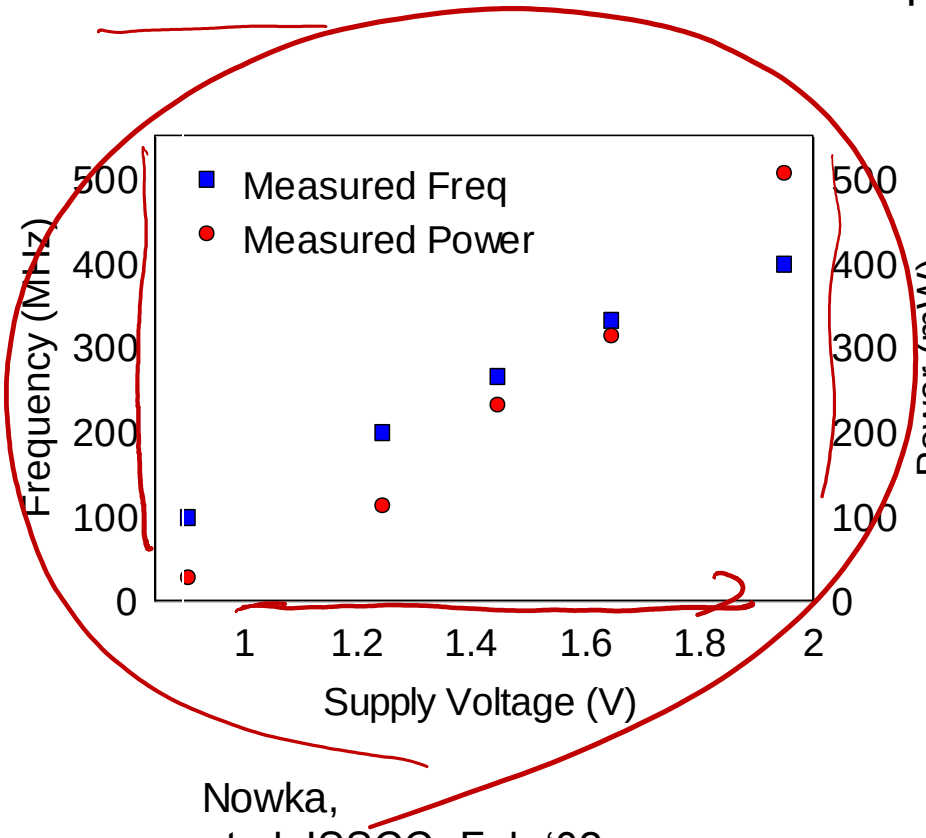
$$P = \tfrac{1}{2} C_{sw} V_{dd} \Delta V f + I_{st} V_{dd} + I_{static} V_{dd}$$

- lowering supply, Vdd and DV, (voltage scaling) is most promising:
  - ✓ frequency ~V, power ~V^3
- challenges
  - ✓ custom CPUs, Analog, PLLs, and I/O drivers don't voltage scale easily
  - ✓ sensitivity to supply voltage varies circuit to circuit – esp SRAM

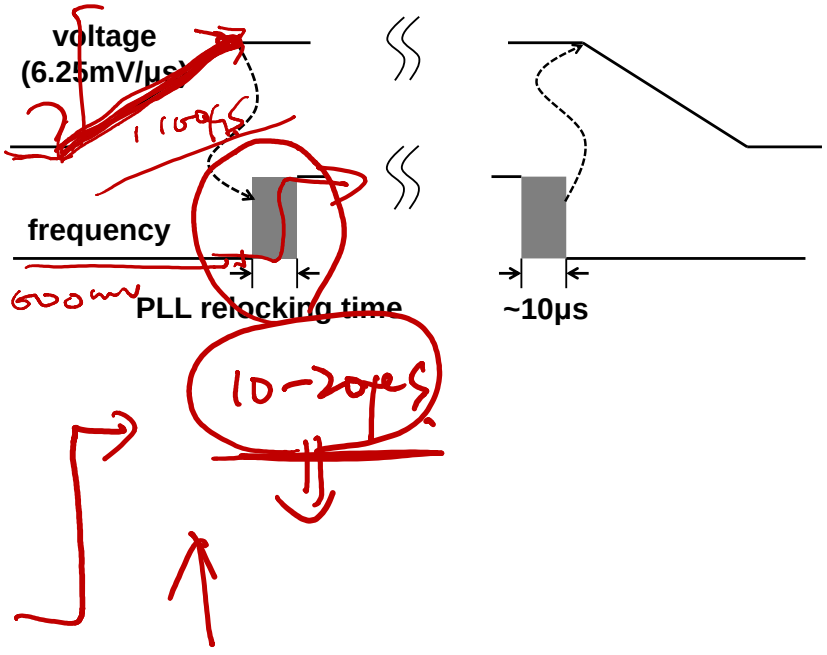


Dynamic Voltage-Scaling

PowerPC 405LP measurements: 18:1 power range over 4:1 frequency range



Nowka,
et.al. ISSCC, Feb '02



Voltage-Frequency-Scaling Measurements

Freq Scaling

Plus DVS

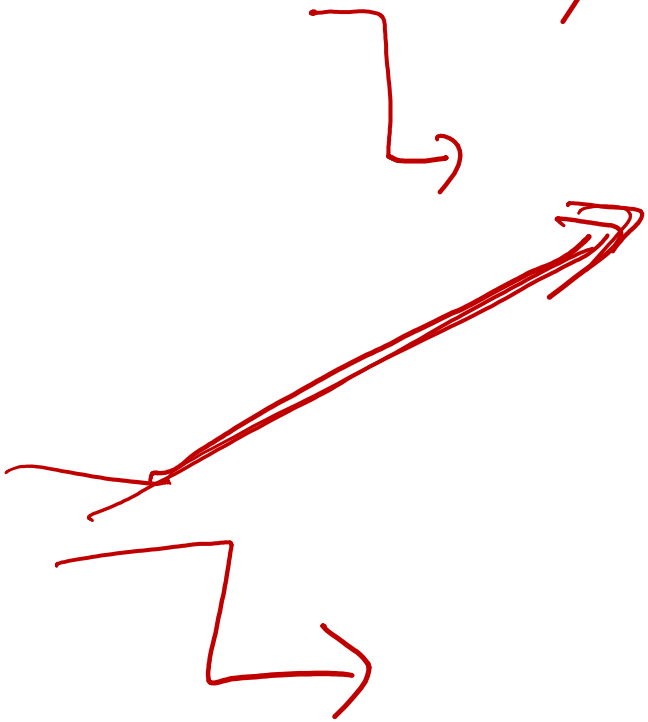
Freq scale ¼ freq, ¼ pwr; DVS ¼ freq, 1/10 pwr

Src: After Nowka, et.al. JSSC, Nov '02

Dynamic Frequency Scaling

$$P = \tfrac{1}{2} C_{sw} V_{dd} \Delta V f + I_{st} V_{dd} + I_{static} V_{dd}$$

- lowering frequency lowers power linearly
  - ✓ DOES NOT improve energy efficiency, just slows down energy consumption
  - ✓ important for avoiding thermal problems



Static Power

$$P = \tfrac{1}{2} C_{sw} V_{dd} \Delta V f + I_{st} V_{dd} + I_{static} V_{dd}$$

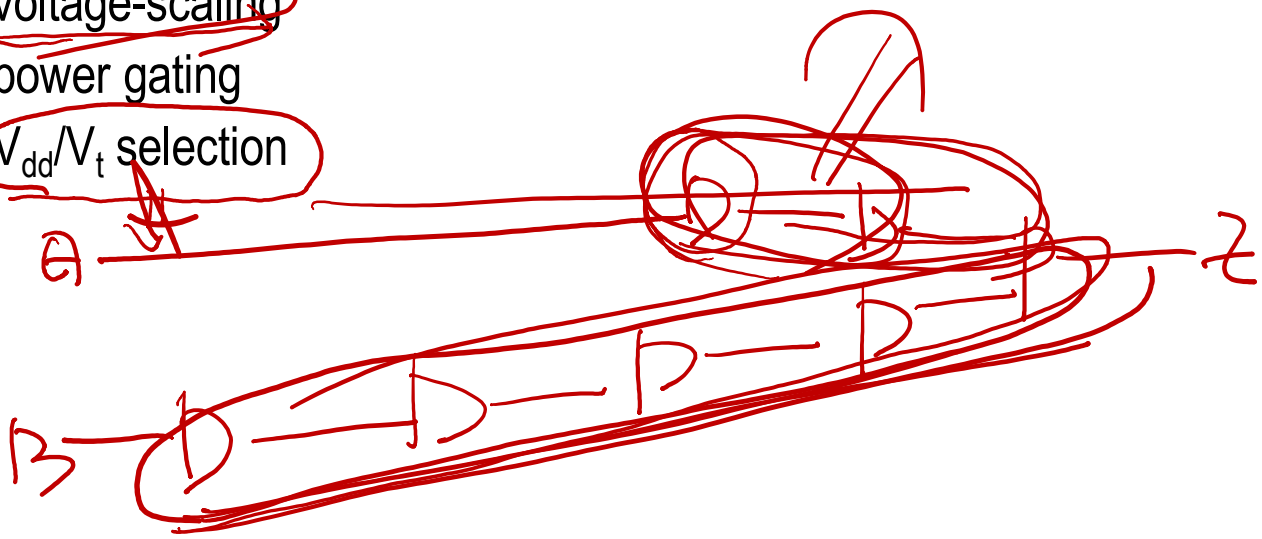
- static energy consumption
  - ✓ even uA per transistor currents can add up – we have billions of transistors
    - subthreshold MOS currents
    - junction currents
    - gate tunneling

~ 40%

Standby-Power Reduction Techniques

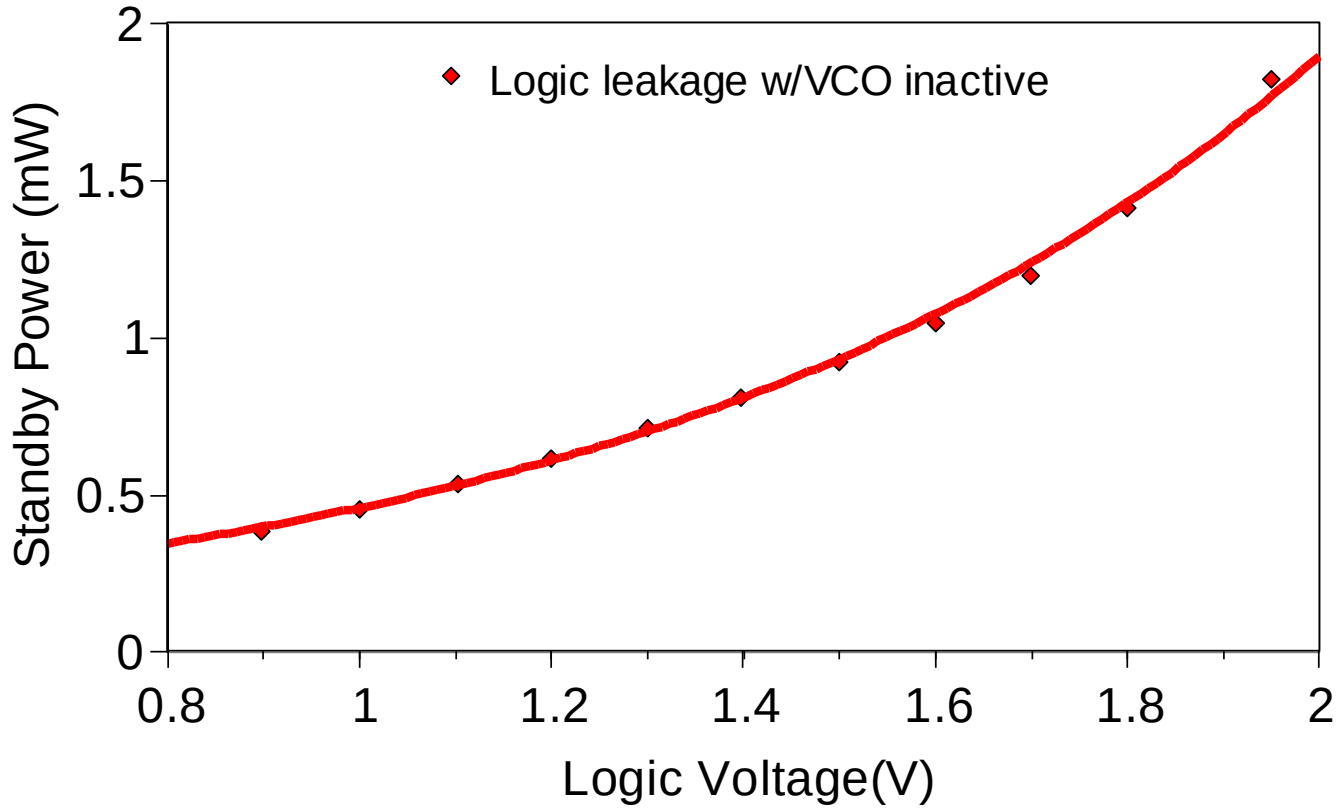
$$P = \tfrac{1}{2} C_{sw} V_{dd} \Delta V f + I_{st} V_{dd} + I_{static}(V_{dd}, t_{ox}, V_t) V_{dd}$$

- standby power can be reduced through:
  - ✓ capacitance minimization
    - leakage current is proportional to the size of transistors
  - ✓ voltage-scaling
  - ✓ power gating
  - ✓ V_{dd}/V_t selection



Voltage Scaling Standby Reduction

- decreasing supply voltage significantly improves standby power

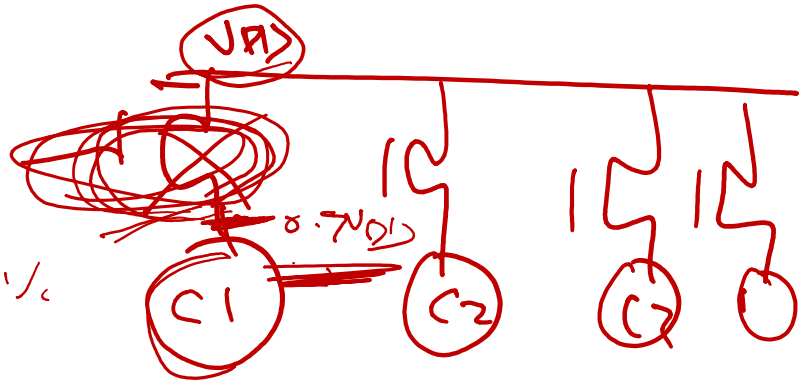


Subthreshold dominated technology

After Nowka, et.al. ISSCC '02

Power Gating

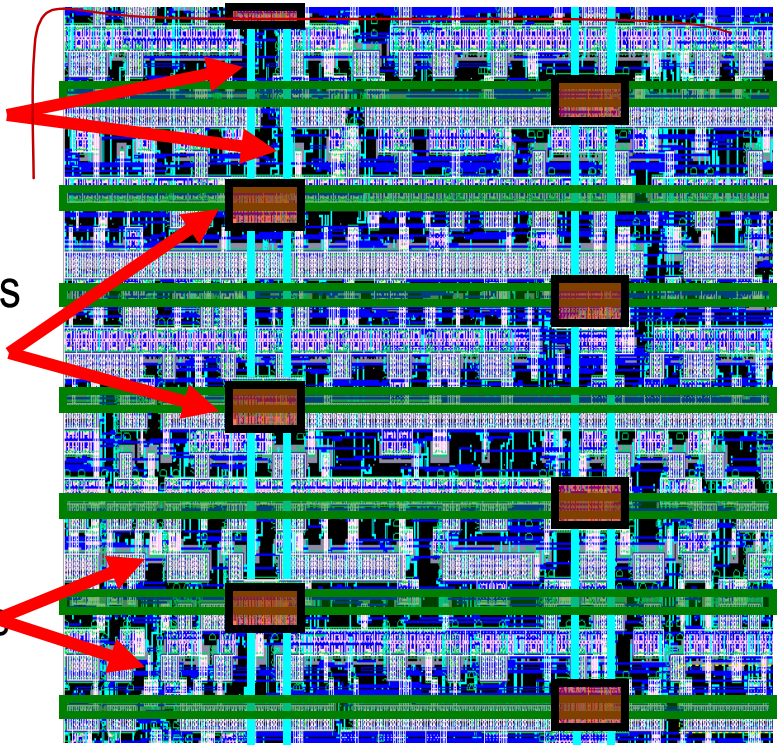
- since transistors still leak when clk is off
  - ✓ reduce leakage power by 250×
  - ✓ cost performance and drop in vdd



power switch control signals

embedded power switches

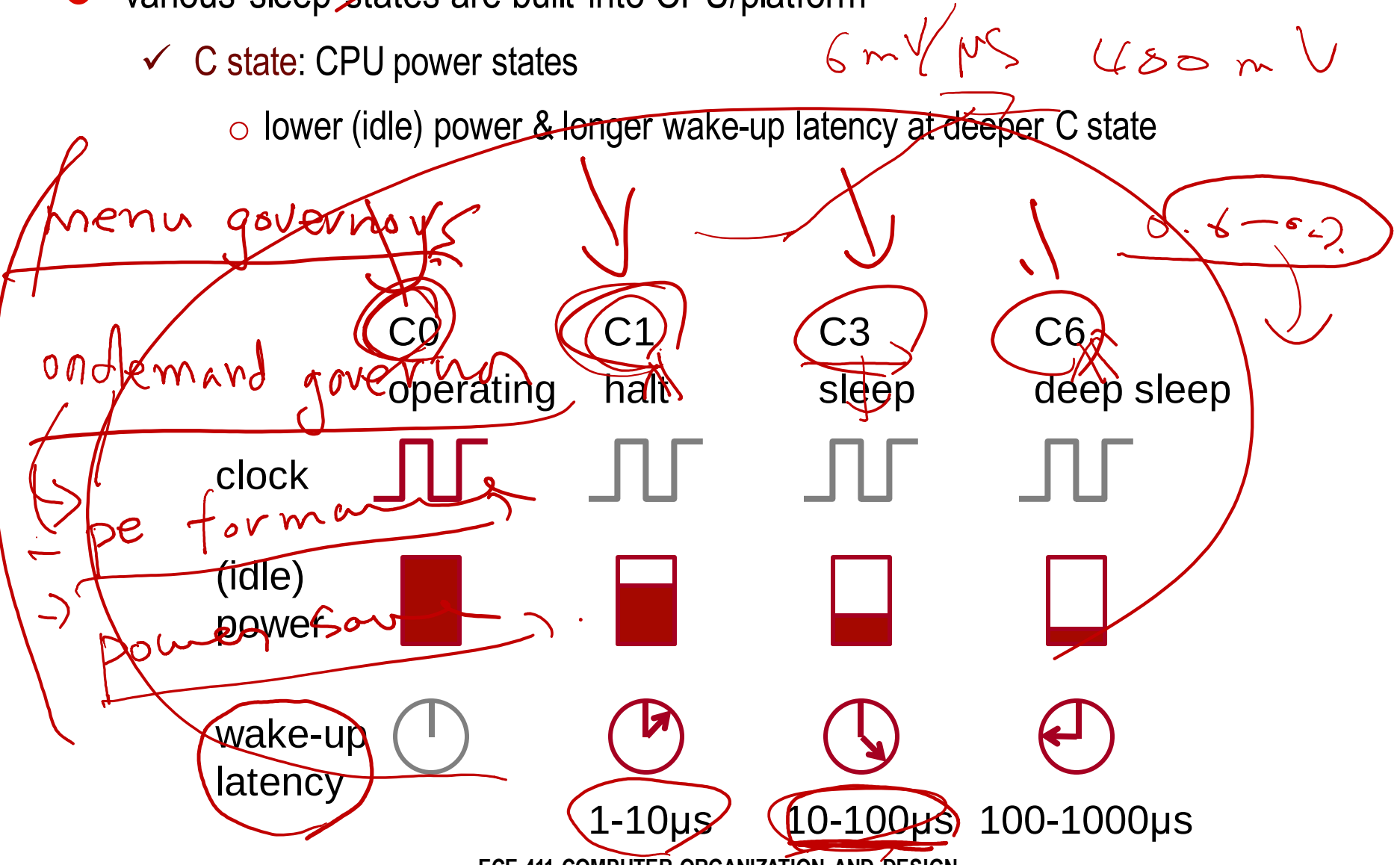
rows of standard cells



Royannez, et al, 90nm Low Leakage SoC Design Techniques for Wireless Applications, ISSCC 2005

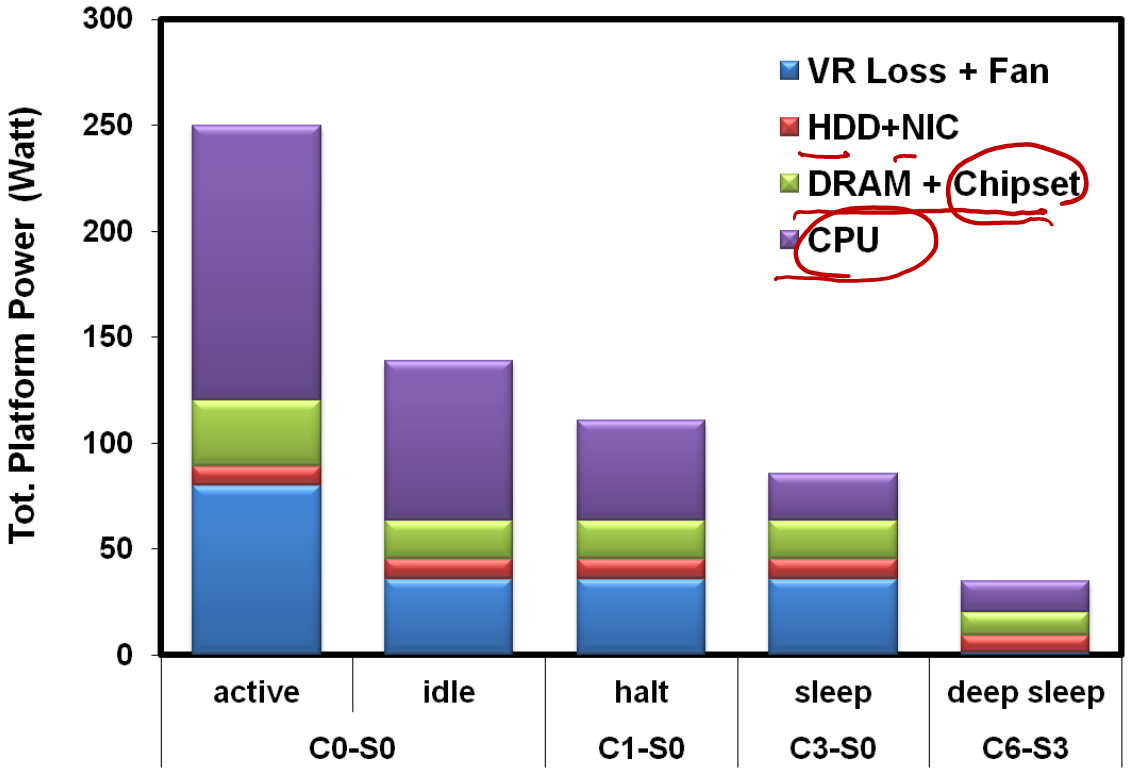
Sleep States

- various sleep states are built into CPU/platform
  - ✓ C state: CPU power states
    - lower (idle) power & longer wake-up latency at deeper C state

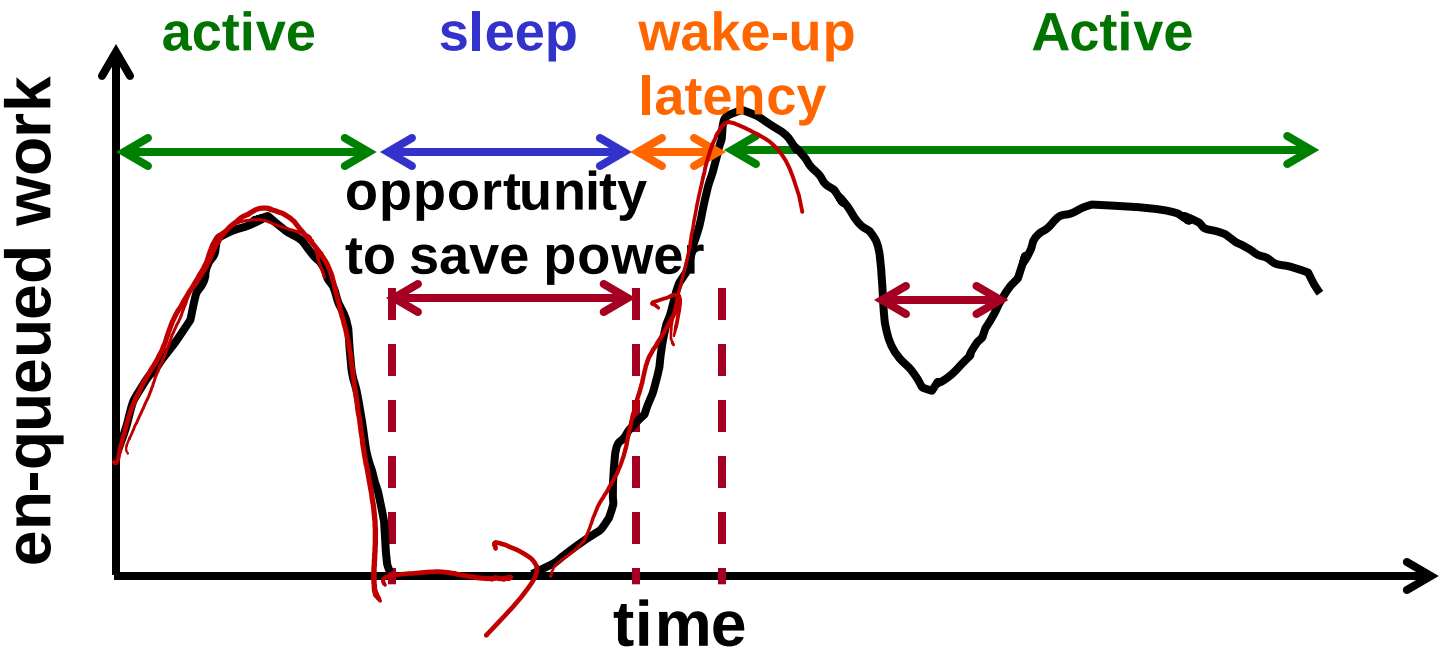


Sleep States

- ✓ S state: platform power state
  - S0: operating state
  - S3: CPU/RAM/HDD/NIC in (deep) sleep state w/ 1-10 s exit latency



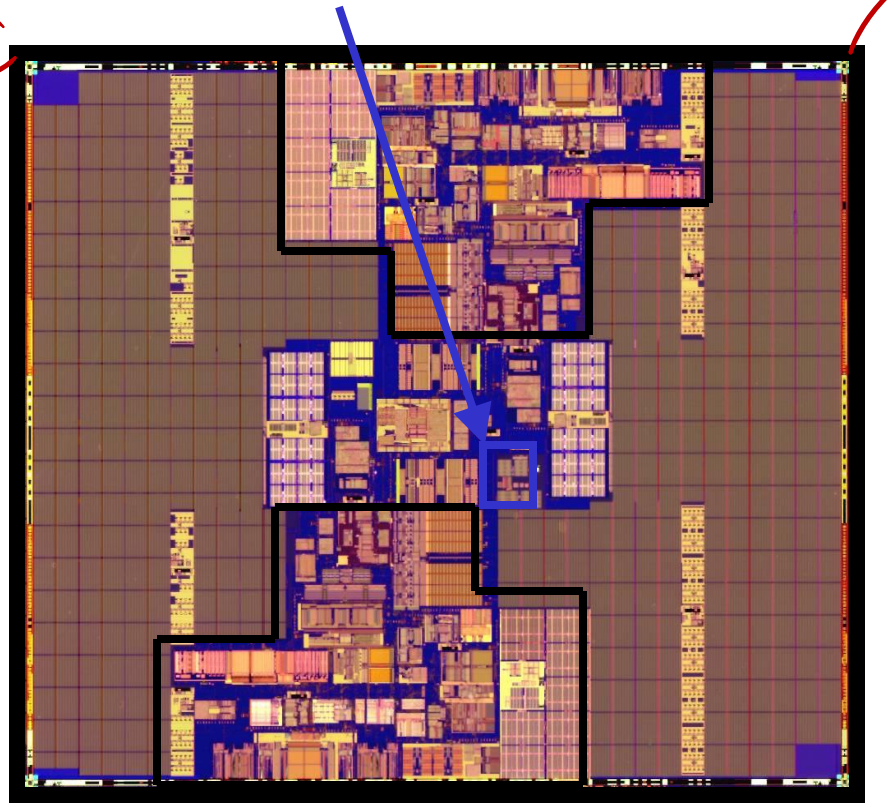
Server Power Management



- less work, slow down or sleep to reduce power consumption

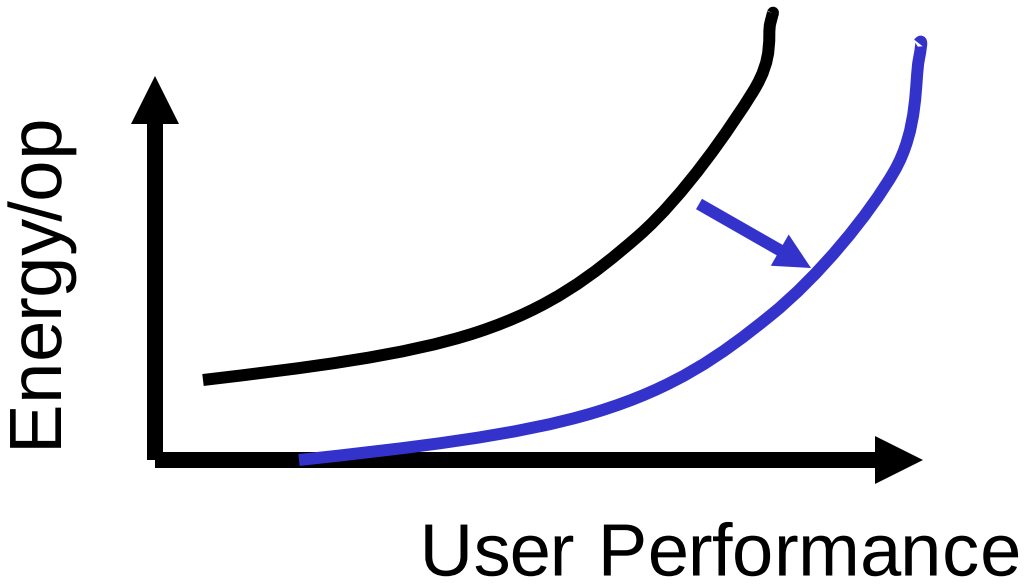
Constant Power Scaling

- Foxton controller on Itanium II
  - ✓ raises VDD and boosts F when most units idle
  - ✓ lowers VDD for parallel code to stay in budget



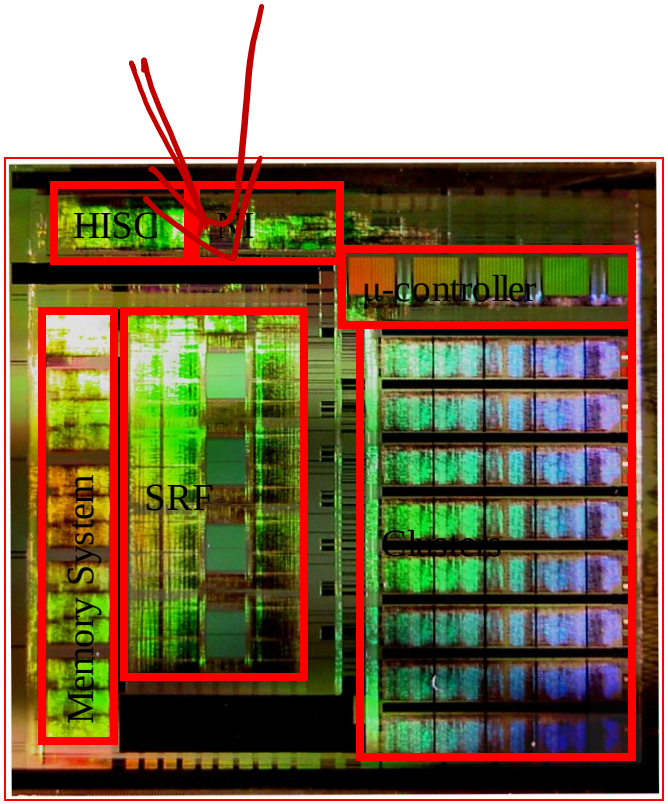
Problem Reformulation

- best way to save energy is to do less work
  - ✓ energy directly reduced by the reduction in work but required time for the function decreases as well, and convert this into extra power gains
  - ✓ shifts the optimal curve down and to the right



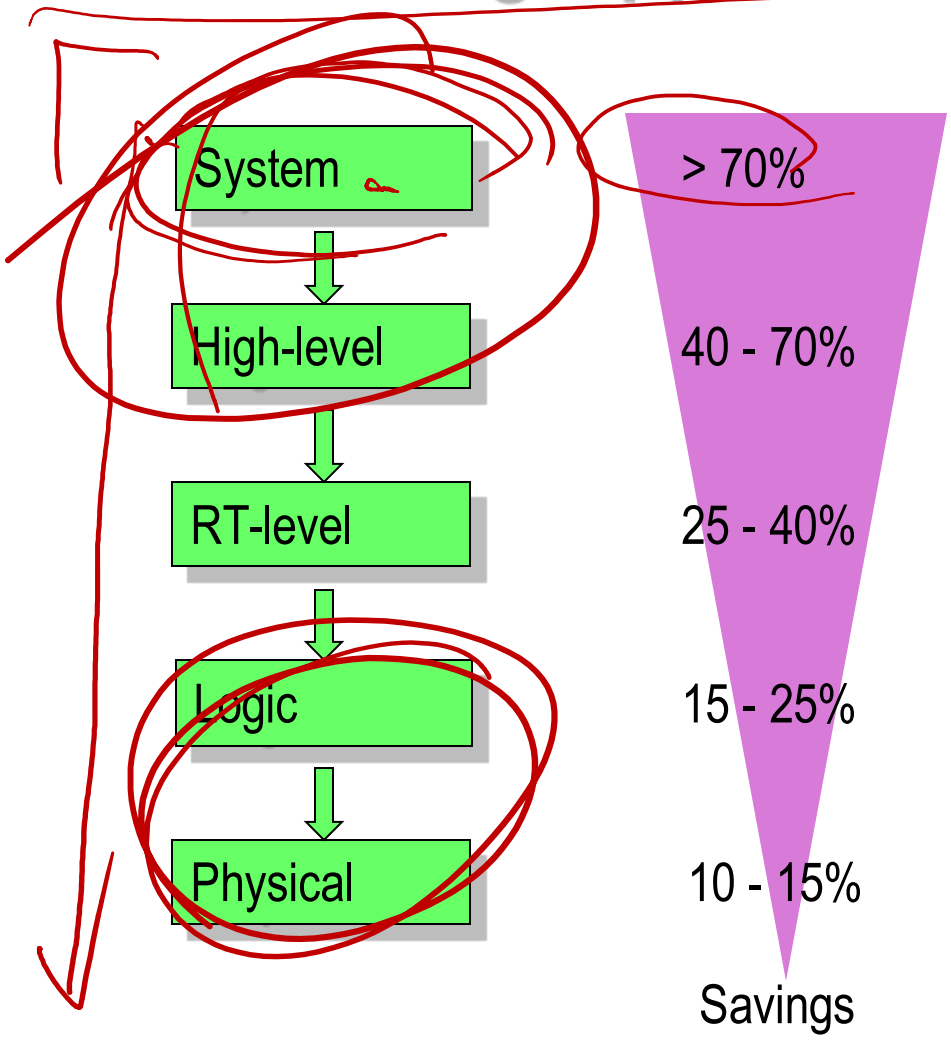
Exploit Specialization

- optimize execution units for different applications
  - ✓ reformulate the hardware to reduce needed work
  - ✓ can improve energy efficiency for a class of applications
- stream / vector processing is a current example
  - ✓ exploit locality, reuse
  - ✓ high compute density



Bill Dally et al, Stanford Imagine

Power Saving Opportunities



HW/SW Co-design, Custom ISA, Algorithm Design, Communication Synthesis
Scheduling, Binding, Pipelining, Behavioral Transformation
Clock gating, Power gating, Precomputation, Operand Isolation, State Assignment, Retiming
Logic Restructuring, Technology Mapping, Rewiring, Pin Ordering & Phase Assignment
Fanout Optimization, Buffering, Transistor Sizing, Placement, Partitioning, Clock Tree Design, Glitch Elimination

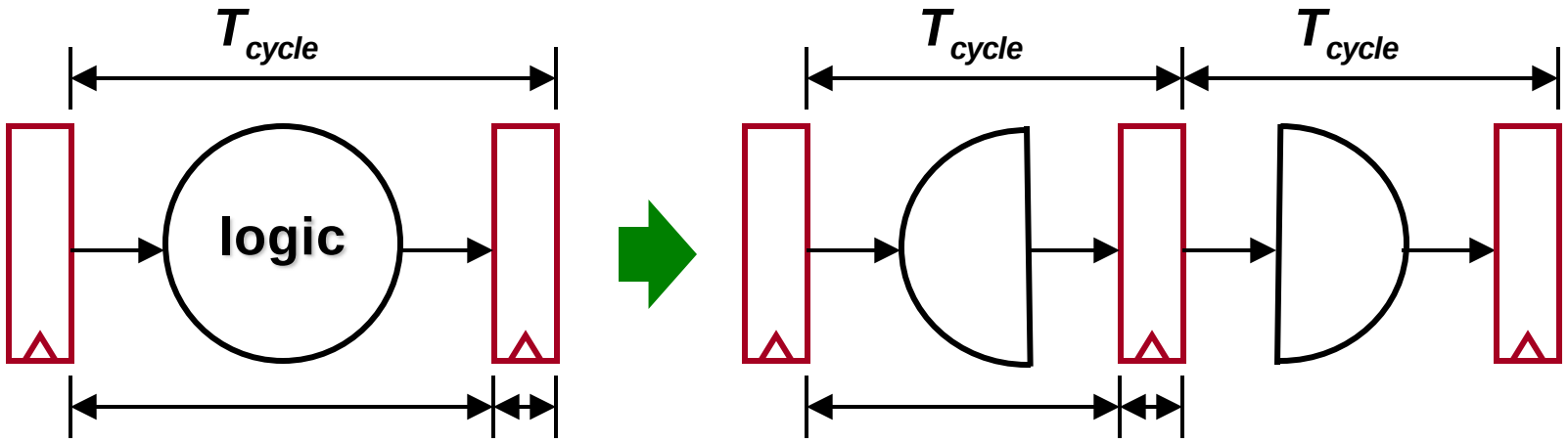
A power-conscious design methodology addresses power at every level of the design hierarchy

Source: Pedram, 1999

Announcement

- today's lecture:
  - ✓ not in the text book
- next lecture: dynamic scheduling
  - ✓ Ch. 3.4 – 3.8 (HP2)
- MP assignment
  - ✓ MP2 due on 2/25 5pm

Pipelining & Energy Saving (1)



$$T_{logic}(V_{DD}) + T_{flop}(V_{DD})$$

$$\frac{T_{logic}(V)}{2} + T_{flop}(V)$$

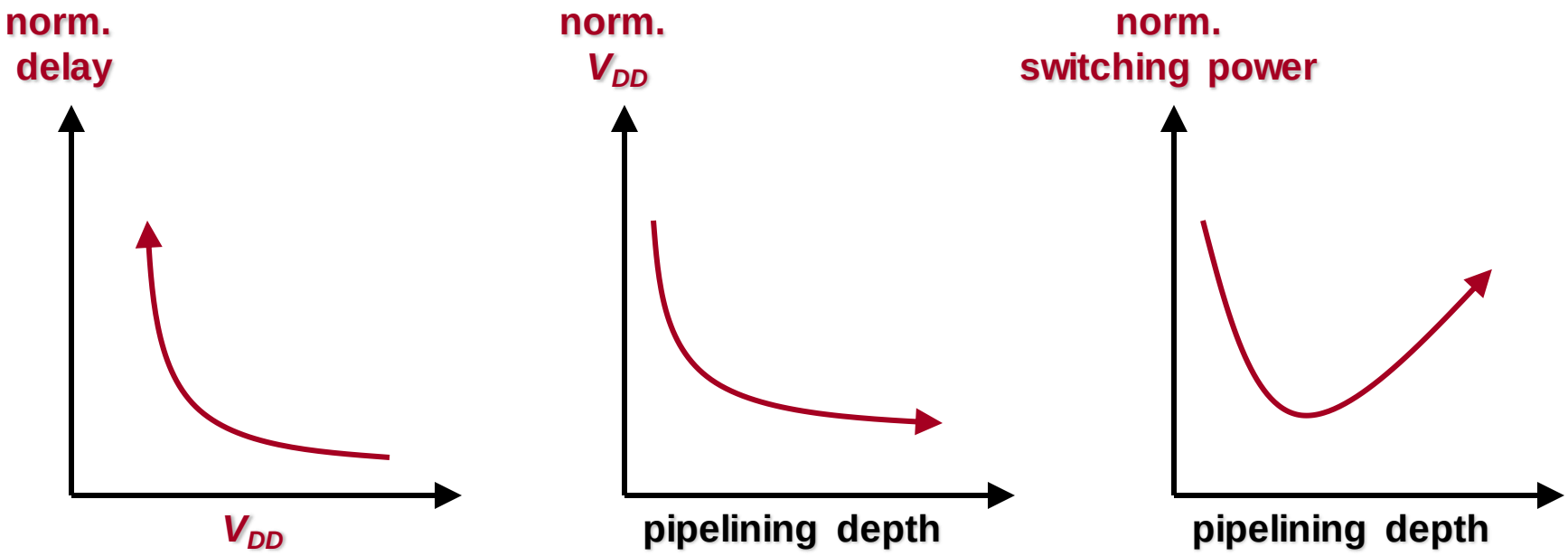
- Finding V such that

pipeline stage multiplication factor

$$T_{logic}(V_{DD}) + T_{flop}(V_{DD}) = \frac{T_{logic}(V)}{N} + T_{flop}(V)$$

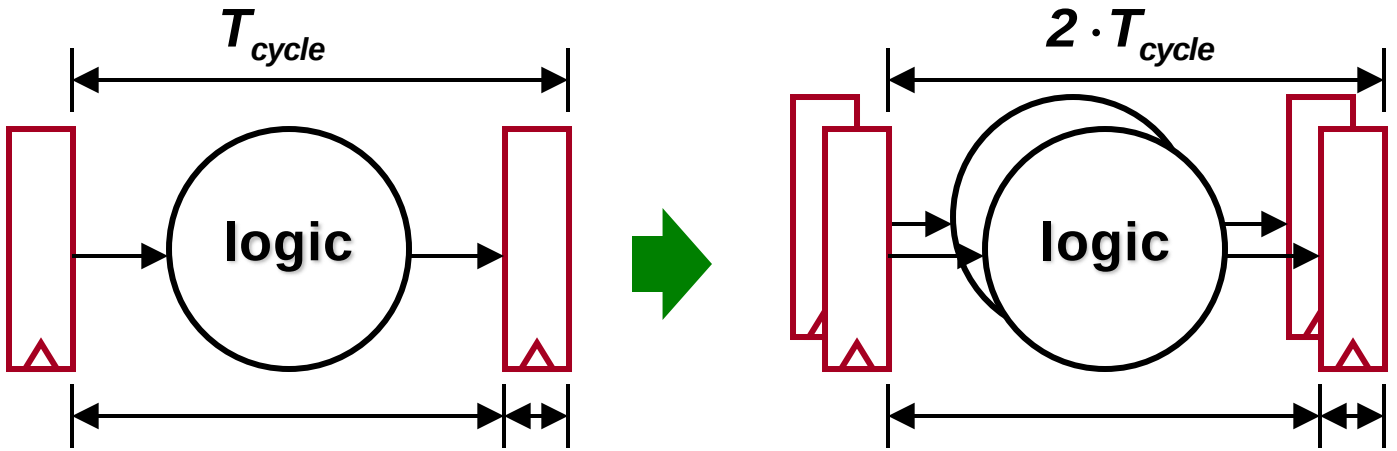
- Increasing pipelining depth reduces V_{DD} resulting in quadratic energy saving while giving the same throughput @ the same frequency

Pipelining & Energy Saving (2)



- Logic delay increases superlinearly w/ reduced V_{DD}
- Increasing pipelining depth increases the timing overhead per stage by flip-flops
- Total power overhead by the increased # of flip-flops increases w/ pipelining depth

Parallel Proc. & Energy Saving (1)



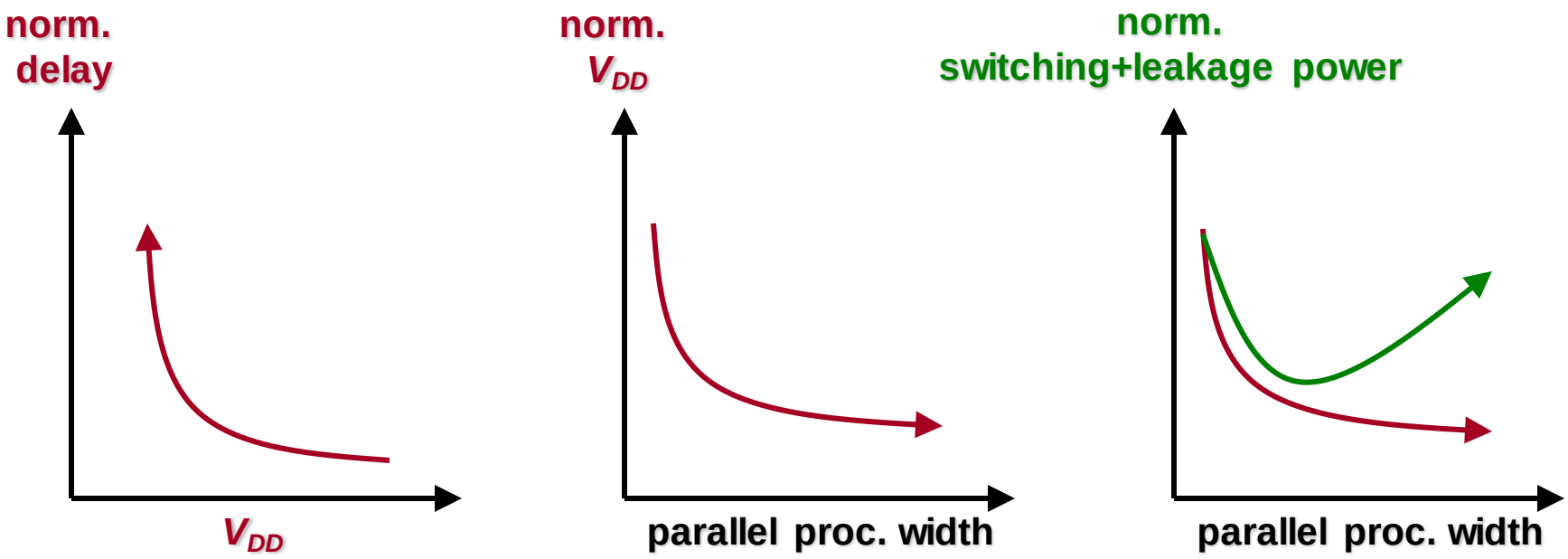
- Finding V such that

parallel processing width

$$T_{logic}(V_{DD}) + T_{flop}(V_{DD}) = \frac{1}{M} \cdot \left(T_{logic}(V) + T_{flop}(V)\right)$$

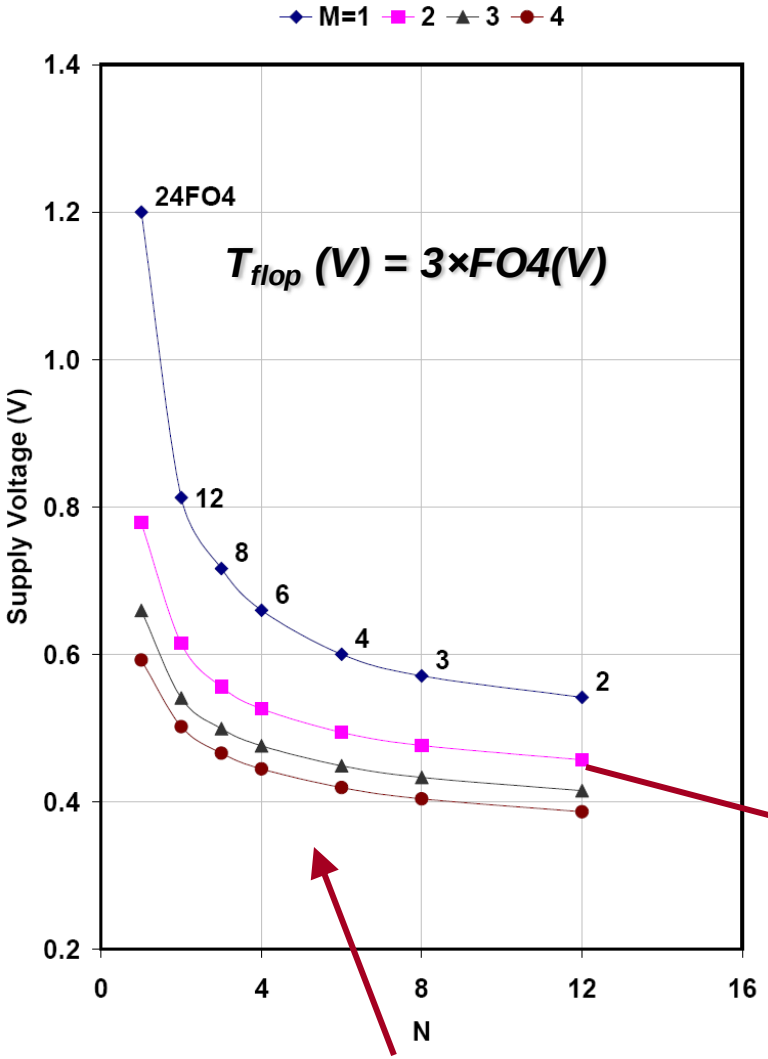
- Doubling # of parallel proc. width reduces V_{DD} resulting in quadratic energy saving while giving the same throughput @ the half frequency

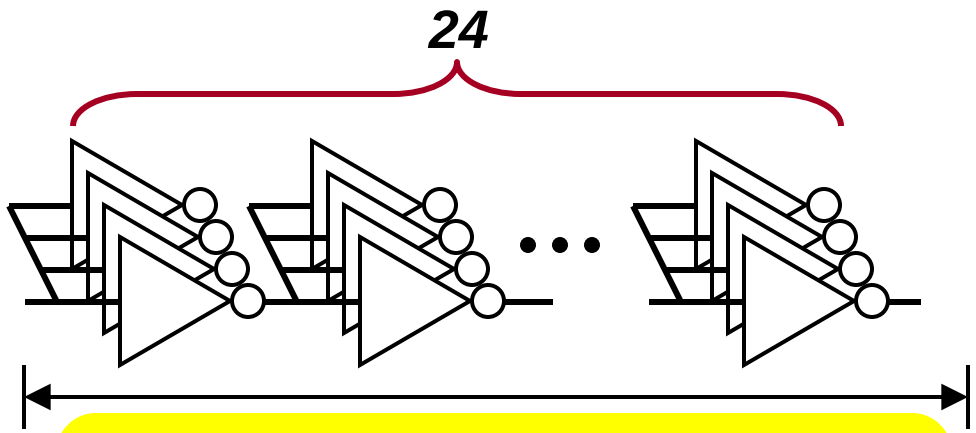
Parallel Proc. & V_{DD} Scaling (2)



- Logic delay increases superlinearly w/ reduced V_{DD}
- Increasing parallel proc. width increases the area (leakage) overheads

Supply Voltage Scaling Trends





$$T_{log\,ic}(V_{DD}) = 24 \times FO4(V_{DD})$$

$$T_{flop}(V_{DD}) = 2 \sim 3 \times FO4(V_{DD})$$

e.g., Pentium4: T_{logic} ~20FO4

$$T(V) = a_6 V^6 + a_5 V^5 + \ldots + a_0$$

$$T_{log\,ic}(V_{DD}) + T_{flop}(V_{DD}) = \frac{1}{M} \cdot \left(\frac{T_{log\,ic}(V)}{N} + T_{flop}(V) \right)$$

Switching Power Trends

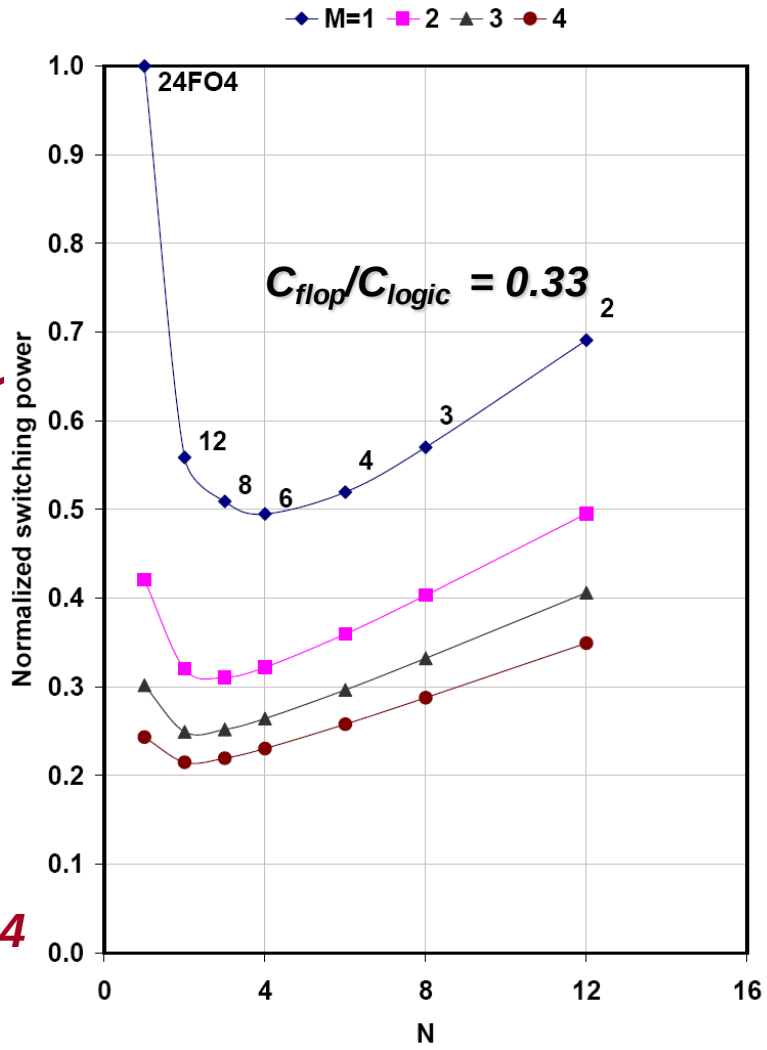
Flop growth factor: $\rho = 1.1 \sim 1.3$

$$P_s^{M-N} = M \cdot \frac{f}{M} \cdot \left(C_{log\,ic} + C_{flop} \cdot N^{\rho}\right) \cdot V^2$$

$$= f \cdot C_{log\,ic} \cdot \left(1 + \frac{C_{flop}}{C_{log\,ic}} \cdot N^{\rho}\right) \cdot V^2$$

$$\frac{P_s^{M-N}}{P_s^{1-1}} = \frac{f \cdot C_{log\,ic} \cdot \left(1 + B \cdot N^{\rho}\right) \cdot V^2}{f \cdot C_{log\,ic} \cdot (1+B) \cdot V_{DD}}$$

$$= \frac{\left(1 + B \cdot N^{\rho}\right) \cdot V^2}{(1+B) \cdot V_{DD}}$$



In high-end µproc., the percentage of flop switching power including CK tree is 0.2~0.4

Wider parallel processing reduces more switching power, but …

Leakage Power Trends

$$I_{gate} = E \cdot \left(\frac{V}{t_{ox}}\right)^2 e^{\left(-F\frac{V}{t_{ox}}\cdot\left(1-\left(1-\frac{V}{\phi_{ox}}\right)^{3/2}\right)\right)}$$

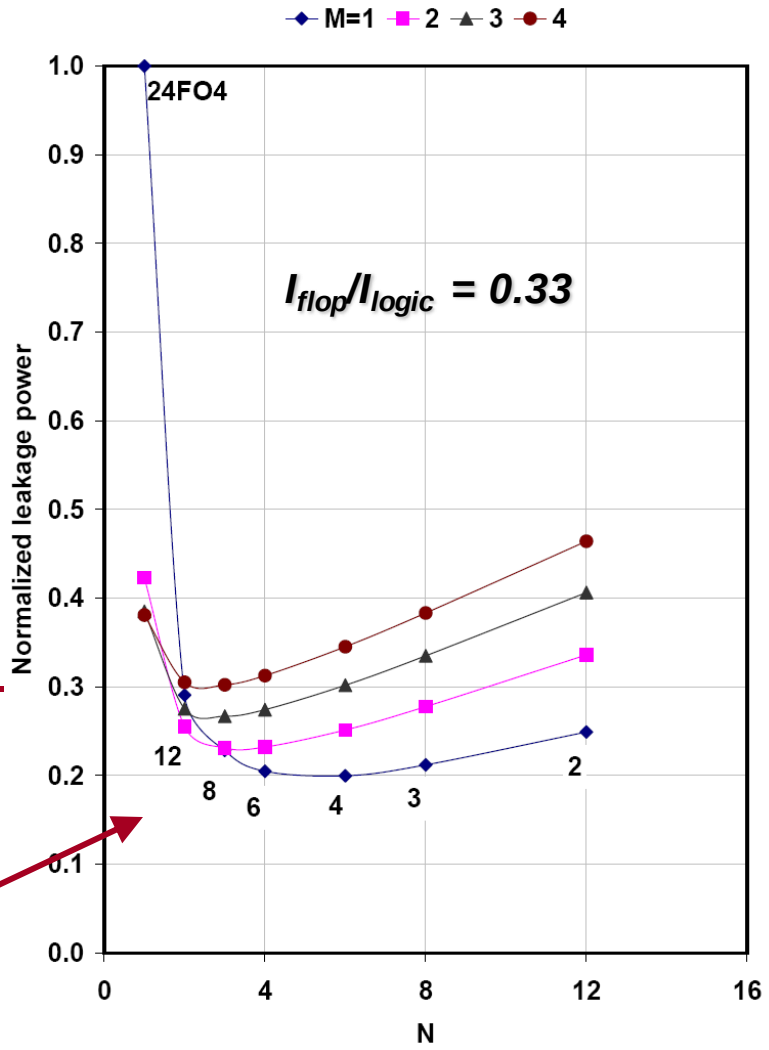
$$+$$

$$I_{sub} = D \cdot e^{\frac{V_{th}-\eta V}{n\cdot(kT/q)}} \cdot \left(1 - e^{\frac{V}{kT/q}}\right)$$

$$I(V) = b_4 V^4 + b_3 V^3 + \ldots + b_0$$

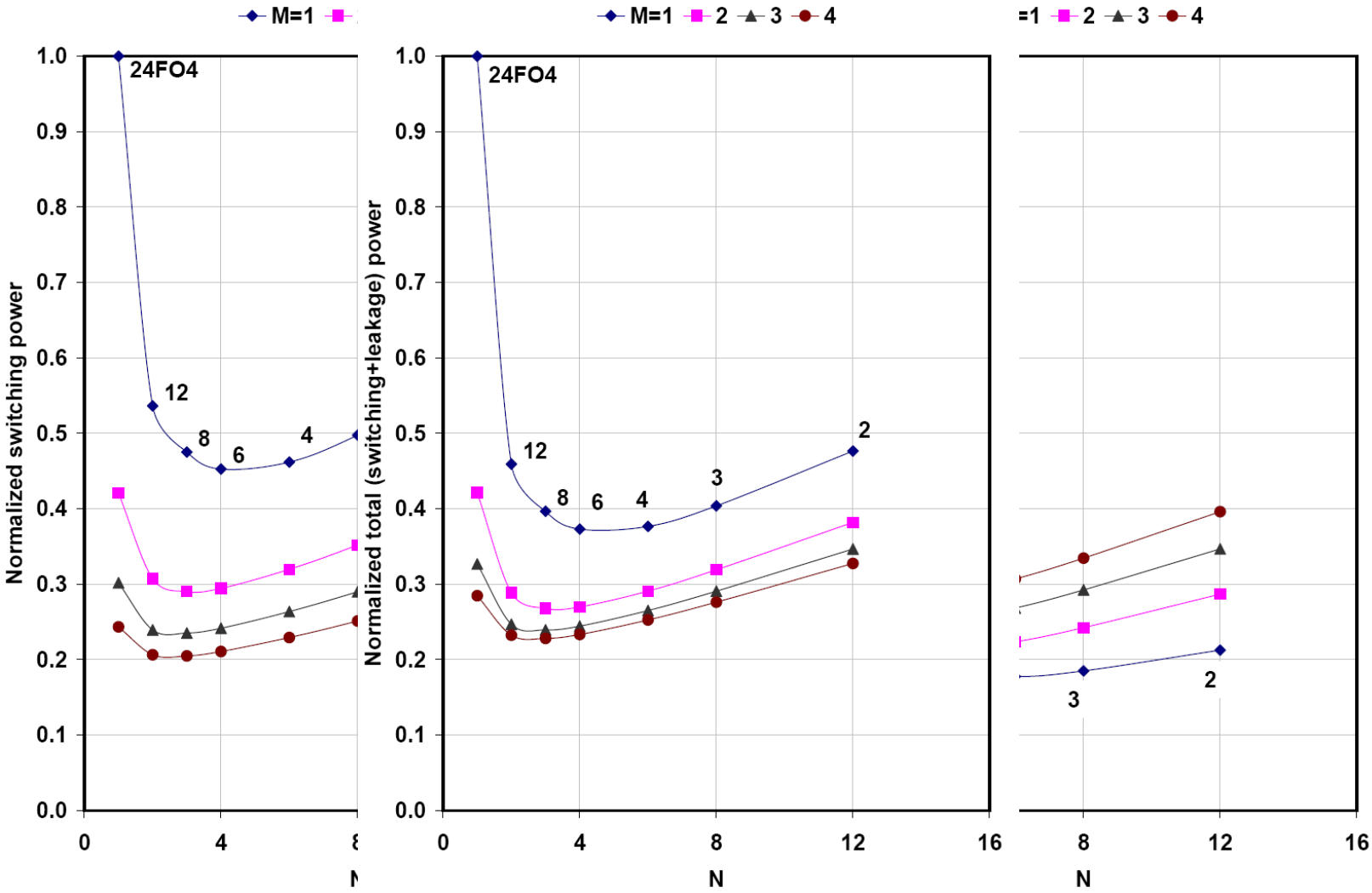
$$P_l^{M-N} = M \cdot \left(I_{logic} + I_{flop} \cdot N^{\rho}\right) \cdot V$$

$$= M \cdot I_{logic} \cdot \left(1 + \frac{I_{flop}}{I_{logic}} \cdot N^{\rho}\right) \cdot V$$



Leakage power makes parallel processing less attractive

Total Power Trends



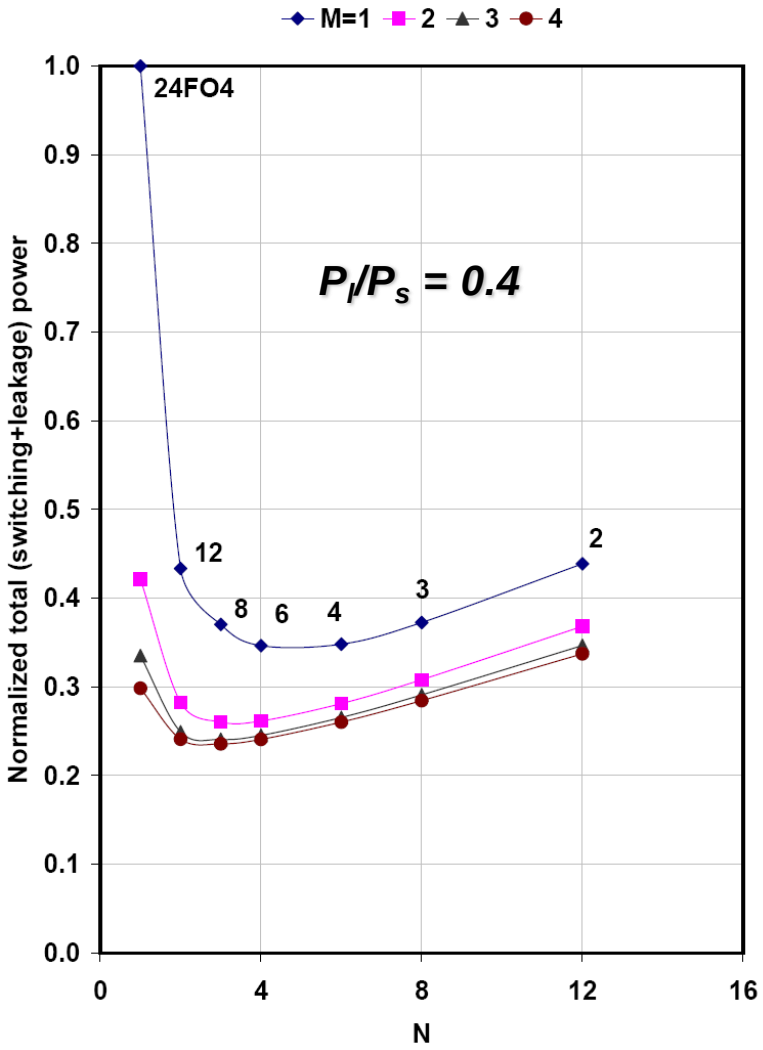
Total Power Trends

$$P_t^{M-N} = P_s^{M-N} + P_l^{M-N}$$
$$= P_s^{M-N} \cdot \left(1 + \frac{P_l^{M-N}}{P_s^{M-N}}\right)$$

In high-end µproc., the percentage of leakage power (P_l/P_s) is 0.3~0.5

Knobs:

P_l/P_s	= 0.4
T_{flop}/T_{logic}	= 2/24
C_{flop}/C_{logic}	= 0.25
I_{flop}/I_{logic}	= 0.25
ρ	= 1.2



12FO4 logic depth w/ 3 parallel processing width is total power optimal!