18-640 Foundations of Computer Architecture

Lecture 19: “Power Reduction and Energy Efficiency”

John Paul Shen
November 11, 2014

- Required Reading Assignments:
  - “Dynamic and Static Power in CMOS” by Vishwani D. Agrawal and Srivaths Ravi, Low Power Design & Test, Hyderabad, July 30-31, 2007
  - “Mitigating Amdahl's Law through EPI Throttling,” by M. Annavaram, E. Grochowski, J. Shen. In 32nd ISCA 2005.

Electrical & Computer ENGINEERING
Carnegie Mellon University

18-640 Foundations of Computer Architecture

Lecture 19: “Power Reduction and Energy Efficiency”

A. CMOS Power Consumption
   a. Dynamic and Static Power
   b. Power Efficiency Metrics
B. Energy Per Instruction (EPI)
   a. EPI Throttling and Asymmetric Cores
C. Extreme Energy Efficiency Challenges
   a. Active Power Saving
   b. Standby Power Saving
   c. Energy Harvesting

Electrical & Computer ENGINEERING
Carnegie Mellon University

Law #5 – Power and Performance

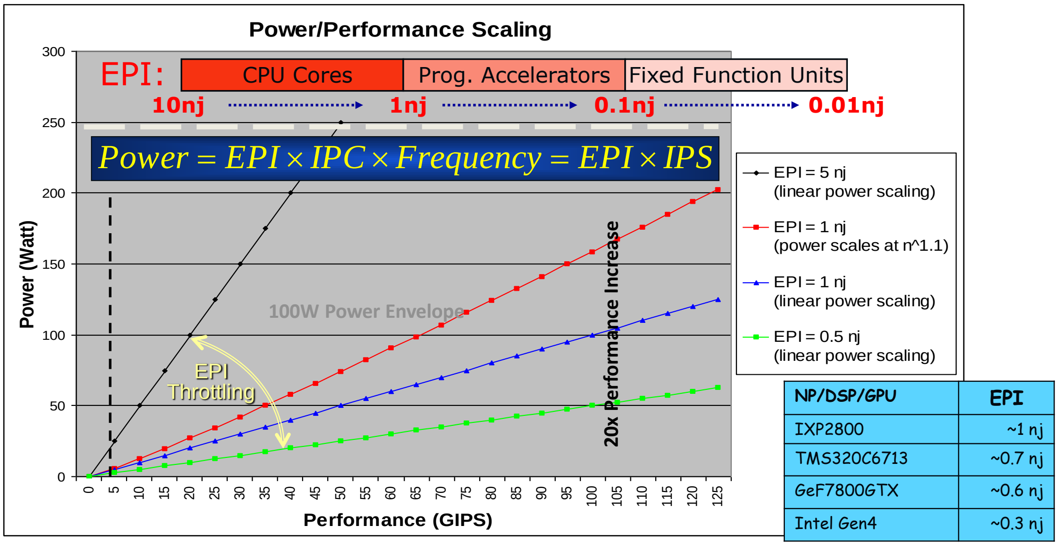
$$Watt = \frac{Joule}{second} = \frac{Joule}{instruction} \times \frac{instruction}{cycle} \times \frac{cycle}{second}$$

$$Power = \boxed{EPI \times IPC \times Frequency}$$

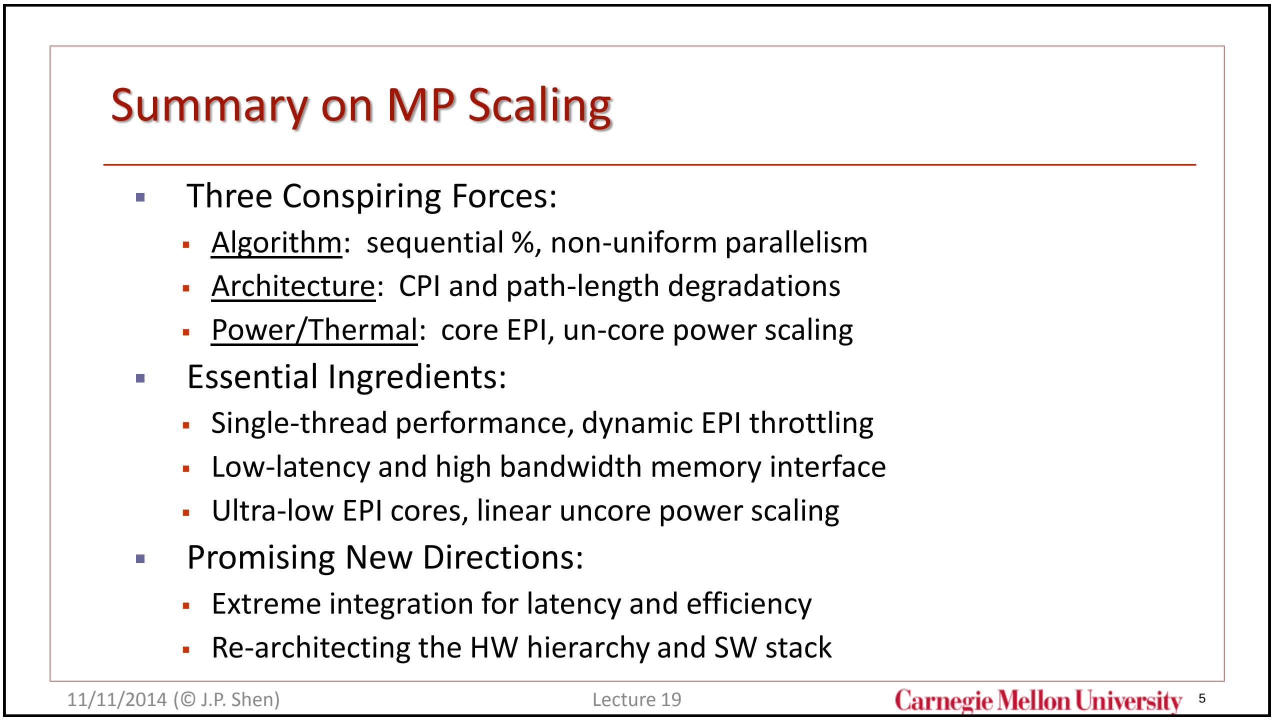
$$Performance = \frac{Frequency}{PathLength \times CPI} = \frac{IPC \times Frequency}{PathLength}$$

$$Power = \boxed{EPI \times Performance \times PathLength}$$

Dynamic EPI Optimization for MP Architectures

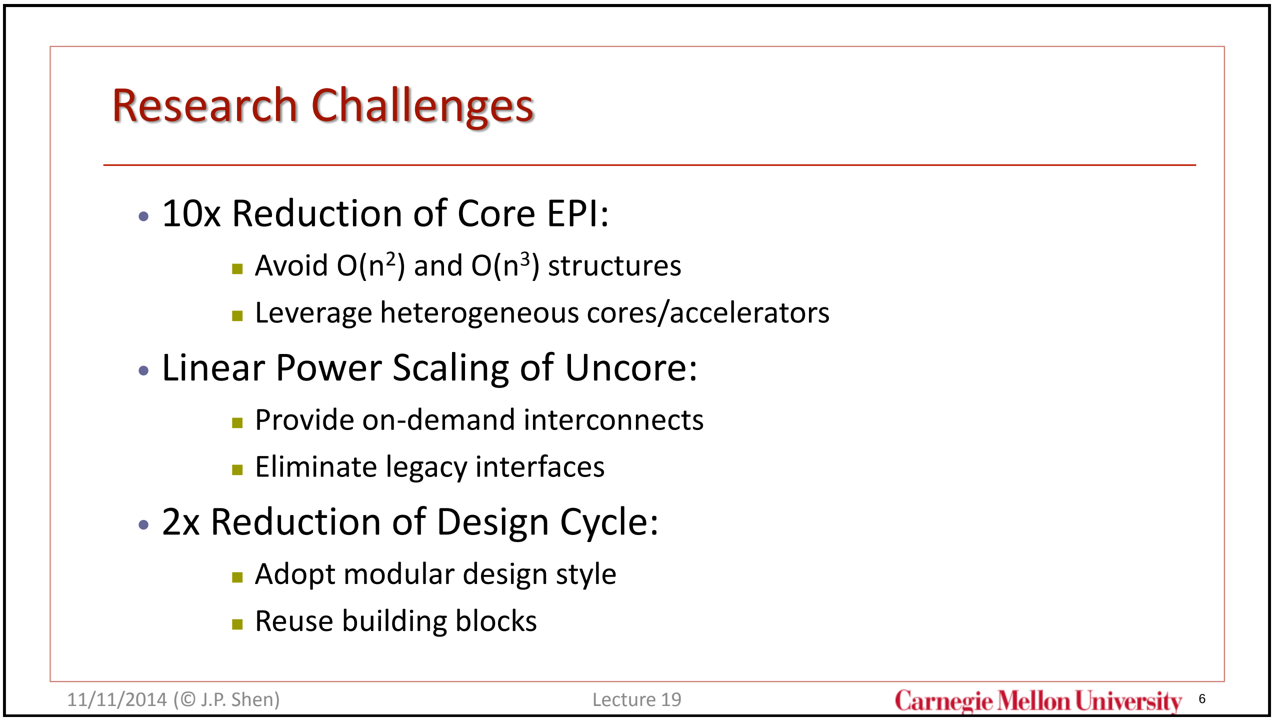


NP/DSP/GPU	EPI
IXP2800	~1 nj
TMS320C6713	~0.7 nj
GeF7800GTX	~0.6 nj
Intel Gen4	~0.3 nj



Summary on MP Scaling

- Three Conspiring Forces:
  - Algorithm: sequential %, non-uniform parallelism
  - Architecture: CPI and path-length degradations
  - Power/Thermal: core EPI, un-core power scaling
- Essential Ingredients:
  - Single-thread performance, dynamic EPI throttling
  - Low-latency and high bandwidth memory interface
  - Ultra-low EPI cores, linear uncore power scaling
- Promising New Directions:
  - Extreme integration for latency and efficiency
  - Re-architecting the HW hierarchy and SW stack



Research Challenges

- 10x Reduction of Core EPI:
  - Avoid $O(n^2)$ and $O(n^3)$ structures
  - Leverage heterogeneous cores/accelerators
- Linear Power Scaling of Uncore:
  - Provide on-demand interconnects
  - Eliminate legacy interfaces
- 2x Reduction of Design Cycle:
  - Adopt modular design style
  - Reuse building blocks

CMOS Scaling

2005 SIA	2005	2008	2011	2014	2017	2020
	0.09μm	0.06μm	0.036μm	0.028μm	0.020μm	0.014μm

Historic CMOS scaling

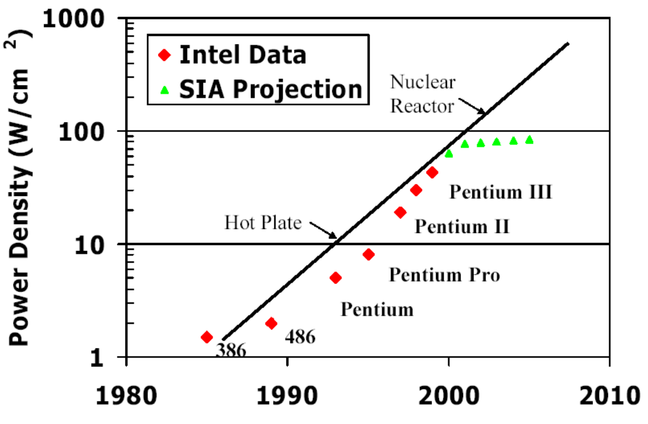
- Doubling every two years (Moore's law)
  - Feature size decrease
  - Device density increase
- Device switching speed improves 30-40%/generation
- Supply & threshold voltages decrease ($V_{dd,}$ V_{th})

Projected CMOS scaling

- Feature size, device density scaling continues
  - ~10 year roadmap out to 10nm generation
- Switching speed improves ~20%/generation
- Voltage scaling tapers off quickly
  - Unreliable device behavior at sub-1.0V V_{dd}

Power Density [Hu et al, MICRO '03 tutorial]

- Power density increasing exponentially
  - Power delivery, packaging, thermal implications
  - Thermal effects on leakage, delay, reliability, etc.



Useful Chart on Power, Voltage, Resistance, Current

Formula 1 – Electrical power equation:

Power $P = I \times V = R \times I^2 = V^2 / R$

where power P is in watts, voltage V is in volts and current I is in amperes (DC).

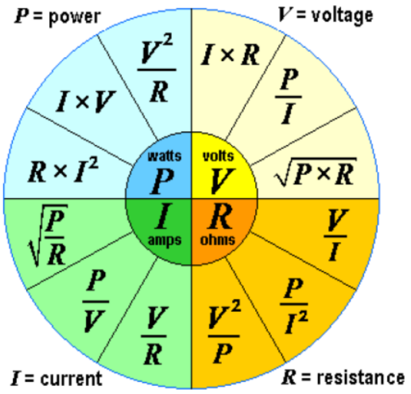
If there is AC, look also at the power factor $PF = \cos \varphi$ and φ = power factor angle (phase angle) between voltage and amperage.

Formula 2 – Mechanical power equation:

Power $P = E / t = W / t$

where power P is in watts, **Energy** E is in joules, and time t is in seconds. 1 W = 1 J/s.

Electric **Energy** is $E = P \times t$ – measured in watt-hours, or also in kWh. 1J = 1N×m = 1W×s



http://www.sengpielaudio.com/calculator-ohm.htm

A. CMOS Power Consumption [Vishwani D. Agrawal, 2007]

- Dynamic
  - Signal transitions
    - Logic activity
    - Glitches
  - Short-circuit
- Static
  - Leakage

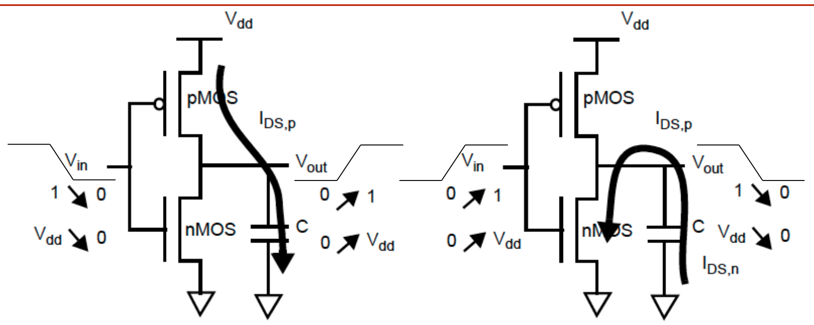
$$P_{total} = P_{dyn} + P_{stat}$$
$$P_{tran} + P_{sc} + P_{stat}$$

http://www.eng.auburn.edu/~vagrawal/hyd.html

A.a. Dynamic and Static Power Consumption

- Power has three components
  - Dynamic Capacitive Power: due to charging and discharging of load capacitance
  - Dynamic Short-Circuit Power: direct current from V_{DD} to G_{nd} when both transistors are on
  - Static Power: when input isn't switching due to leakage

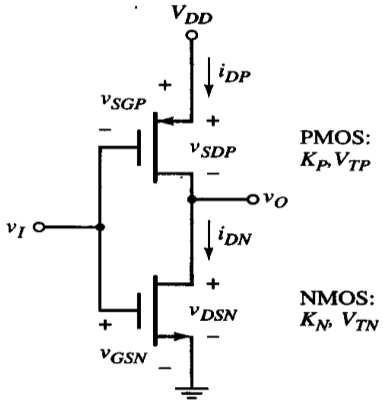
Dynamic Capacitive (Signal Transition) Power

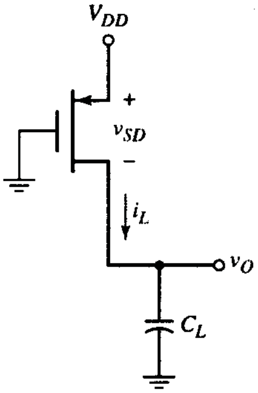


(a) Input switches from 1 to 0 **(b) Input switches from 0 to 1**

- Dynamic power is consumed when device changes ON->OFF and OFF->ON
  - 1->0 current flows to move charge to the capacitance
  - 0->1 current flows from capacitance to ground
- Charging and discharging of capacitance causes power dissipation

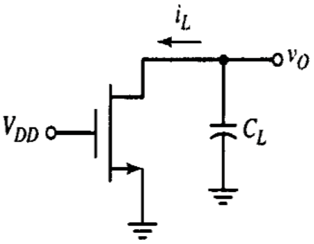
CMOS Inverter Dynamic Capacitive Power





Case I: When the input is at logic 0 and output is 1

Case II: When the input is high and output is low



CMOS Inverter Dynamic Capacitive Power

Case I: When the input is at logic 0: Under this condition the PMOS is conducting and NMOS is in cutoff mode and the load capacitor must be charged through the PMOS device. Power dissipation in the PMOS transistor is given by,

$$P_P=i_L V_{SD}= i_L(V_{DD}-V_O)$$

The current and output voltages are related by,

$$i_L=C_L dv_O/dt$$

Similarly the energy dissipation in the PMOS device can be written as the output switches from low to high,

$$E_P = ½ • CL • V_{dd}^2$$

Above equation show the energy stored in the capacitor C_L when the output is high.

CMOS Inverter Dynamic Capacitive Power

Case II: when the input is high and output is low:

During switching all the energy stored in the load capacitor is dissipated in the NMOS device because NMOS is conducting and PMOS is in cutoff mode. The energy dissipated in the NMOS inverter can be written as,

$$E_N = ½ \bullet CL \bullet V_{dd}{}^2$$

The total energy dissipated during one switching cycle is,

$$E_T = E_P + E_N = \frac{1}{2} C_L V_{DD}{}^2 + \frac{1}{2} C_L V_{DD}{}^2 = C_L V_{DD}{}^2$$

The power dissipated in terms of frequency can be written as

$$E_T = Pt \Rightarrow \quad P = \frac{E_T}{t} \Rightarrow \quad P = fE_T \Rightarrow \quad fC_L V_{DD}{}^2$$

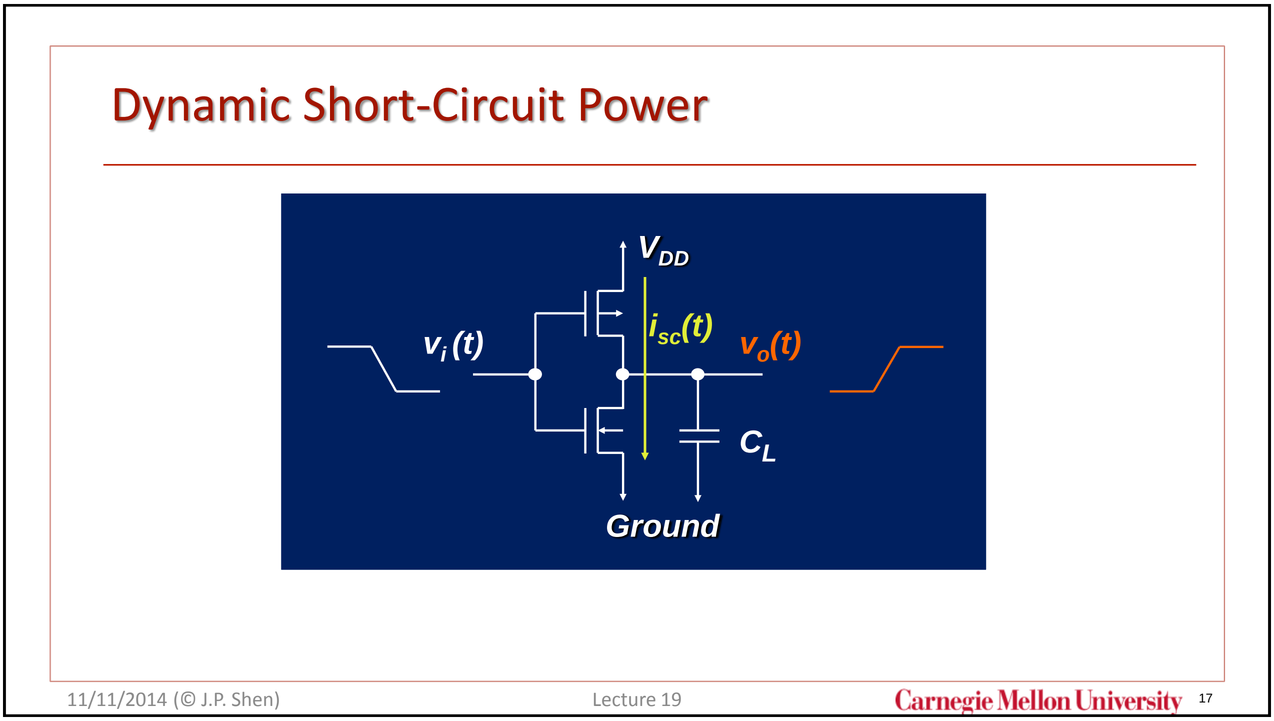
This implied that the power dissipation in the CMOS inverter is directly proportional to switching frequency and $V_{DD}{}^2$

Dynamic Capacitive Power Equation

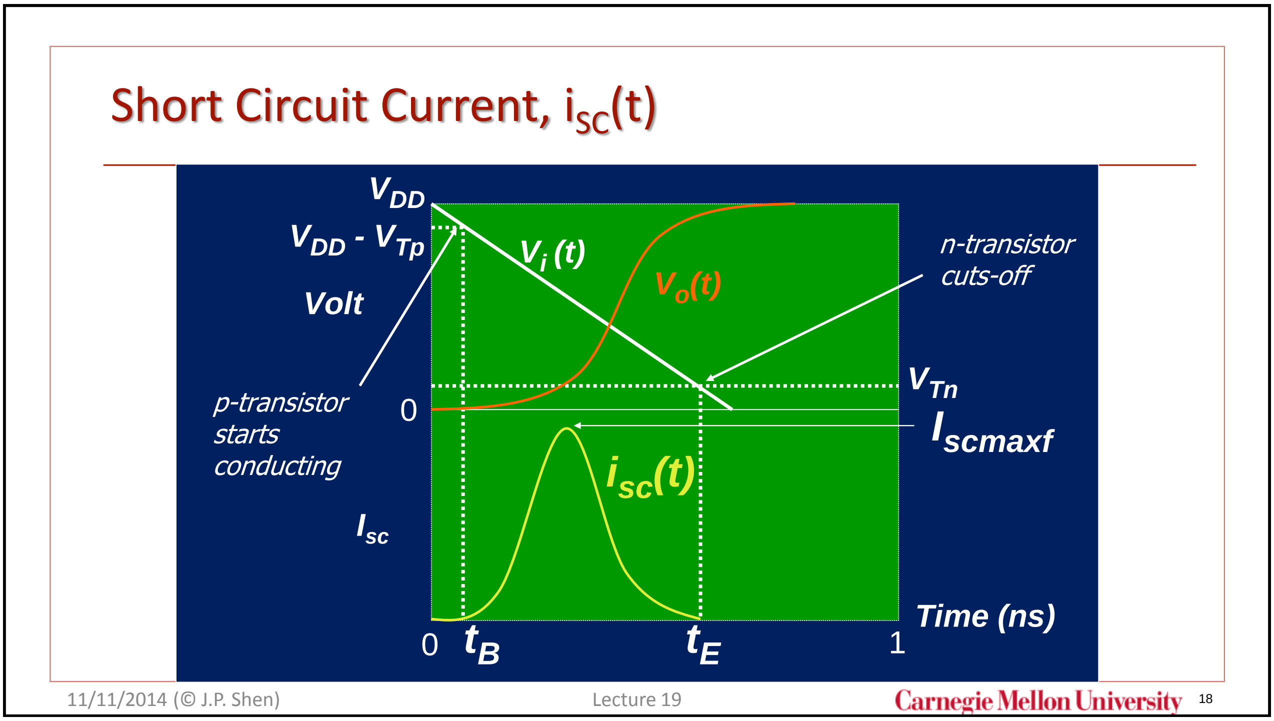
Formula for dynamic power: $P_{dyn} = C_L V_{DD}^2 f$

- Observations
  - Does not (directly) depend on device sizes
  - Does not depend on switching delay
  - Applies to general CMOS gate in which:
    - Switched capacitances are lumped into C_L
    - Output swings from Gnd to V_{DD}
    - Input signal approximated as step function
    - Gate switches with frequency f

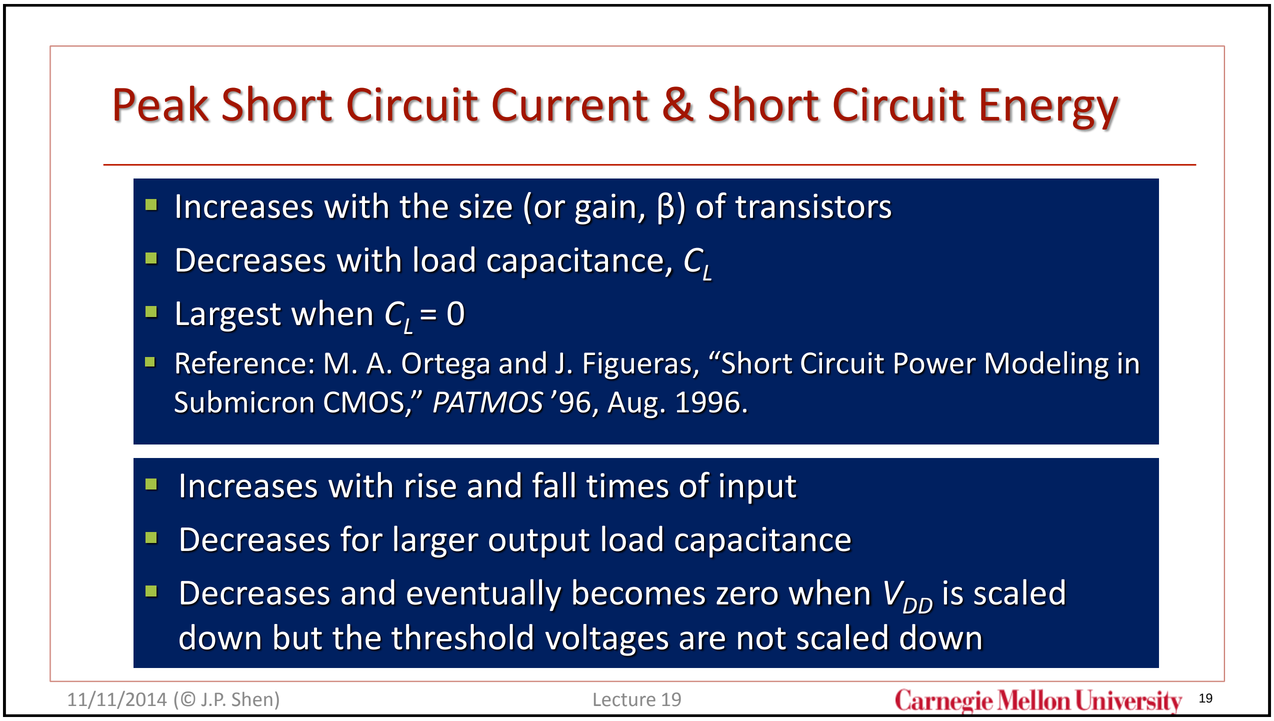
Not a function of transistor sizes!
Data dependent - a function of switching activity!



Dynamic Short-Circuit Power



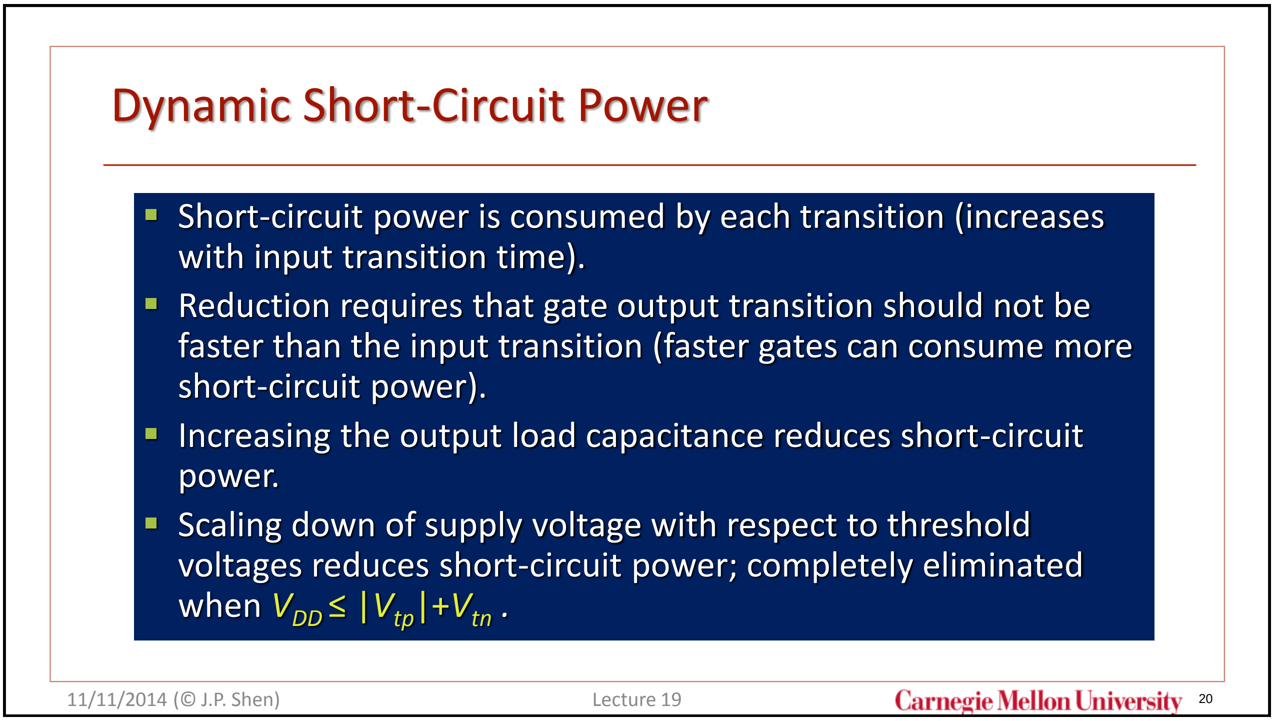
Short Circuit Current, $i_{SC}(t)$



Peak Short Circuit Current & Short Circuit Energy

- Increases with the size (or gain, β) of transistors
- Decreases with load capacitance, C_L
- Largest when $C_L = 0$
- Reference: M. A. Ortega and J. Figueras, “Short Circuit Power Modeling in Submicron CMOS,” *PATMOS* ’96, Aug. 1996.

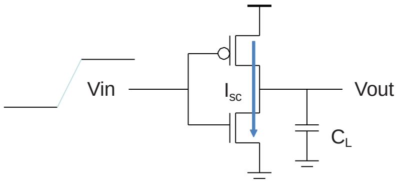
- Increases with rise and fall times of input
- Decreases for larger output load capacitance
- Decreases and eventually becomes zero when V_{DD} is scaled down but the threshold voltages are not scaled down

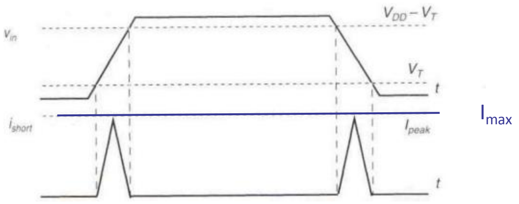


Dynamic Short-Circuit Power

- Short-circuit power is consumed by each transition (increases with input transition time).
- Reduction requires that gate output transition should not be faster than the input transition (faster gates can consume more short-circuit power).
- Increasing the output load capacitance reduces short-circuit power.
- Scaling down of supply voltage with respect to threshold voltages reduces short-circuit power; completely eliminated when $V_{DD} \leq |V_{tp}| + V_{tn}$.

Short Circuit Power Consumption





Finite slope of the input signal causes a direct current path between V_{DD} and GND for a short period of time during switching when both the NMOS and PMOS transistors are conducting.

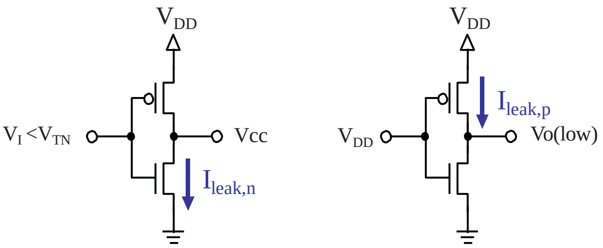
- **Approximate short-circuit current as a triangular wave**
- **Energy per cycle:**

$$E_{sc} = V_{CC}\frac{I_{\max}t_r}{2} + V_{CC}\frac{I_{\max}t_f}{2} = \frac{t_r + t_f}{2}V_{CC}I_{\max}$$

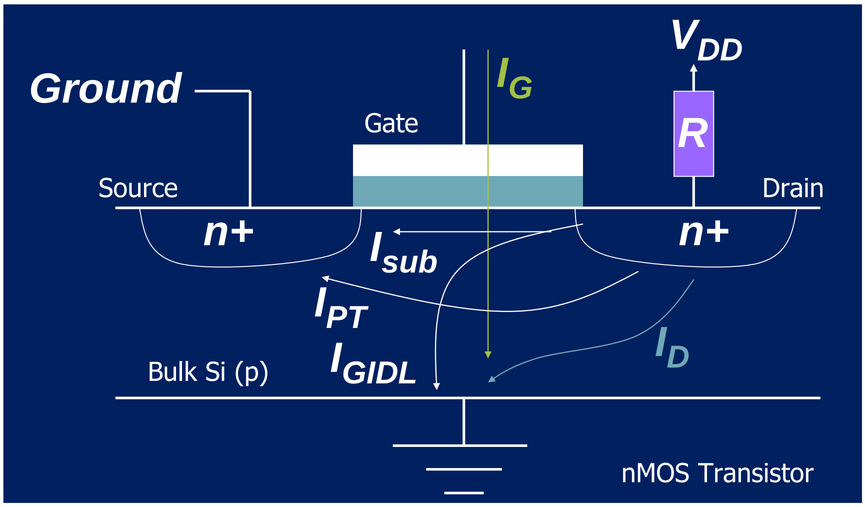
$$P_{sc} = \frac{t_r + t_f}{2}V_{CC}I_{\max}f$$

CMOS Inverter Static Power

- Static Power Consumption:
  - Static current: in CMOS there is no static current as long as $V_{in} < V_{TN}$ or $V_{in} > V_{DD}+V_{TP}$
  - Leakage Current: determined by “off” transistor
  - Influenced by transistor width, supply voltage, transistor threshold voltages



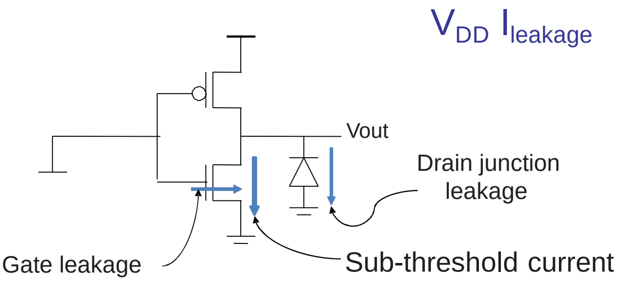
Leakage Power



Leakage Current

- Subthreshold conduction, I_{sub}
- Reverse bias pn junction conduction, I_D
- Gate induced drain leakage, I_{GIDL} due to tunneling at the gate-drain overlap
- Drain source punchthrough, I_{PT} due to short channel and high drain-source voltage
- Gate tunneling, I_G through thin oxide; *may become significant with scaling*

Leakage (Static) Power Consumption



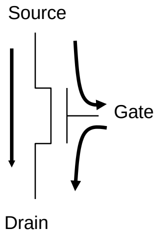
Sub-threshold current is the dominant factor.
All increase exponentially with temperature!

Static Leakage Power

- Leakage power as a fraction of the total power increases as clock frequency drops. *Turning supply off in unused parts can save power (power gating)*.
- For a gate it is a small fraction of the total power; it can be significant for very large circuits.
- Scaling down features requires lowering the threshold voltage, which increases leakage power; roughly doubles with each shrinking.
- Multiple-threshold devices are used to reduce leakage power.

Static Leakage Power

- Transistors aren't perfect on/off switches
- Even in static CMOS, transistors leak
  - Channel (source/drain) leakage
  - Gate leakage through insulator
    - High-K dielectric replacing SiO_2 will help
- Leakage compounded by
  - Low threshold voltage
    - Low V_{th} => fast switching, more leakage
    - High V_{th} => slow switching, less leakage
  - Higher temperature
    - Temperature increases with power
    - Power increases with C, V^2, A, f
- Rough approximation: leakage proportional to area
  - Transistors aren't free
- Huge problem in future technologies
  - Estimates are 40%-50% of total power

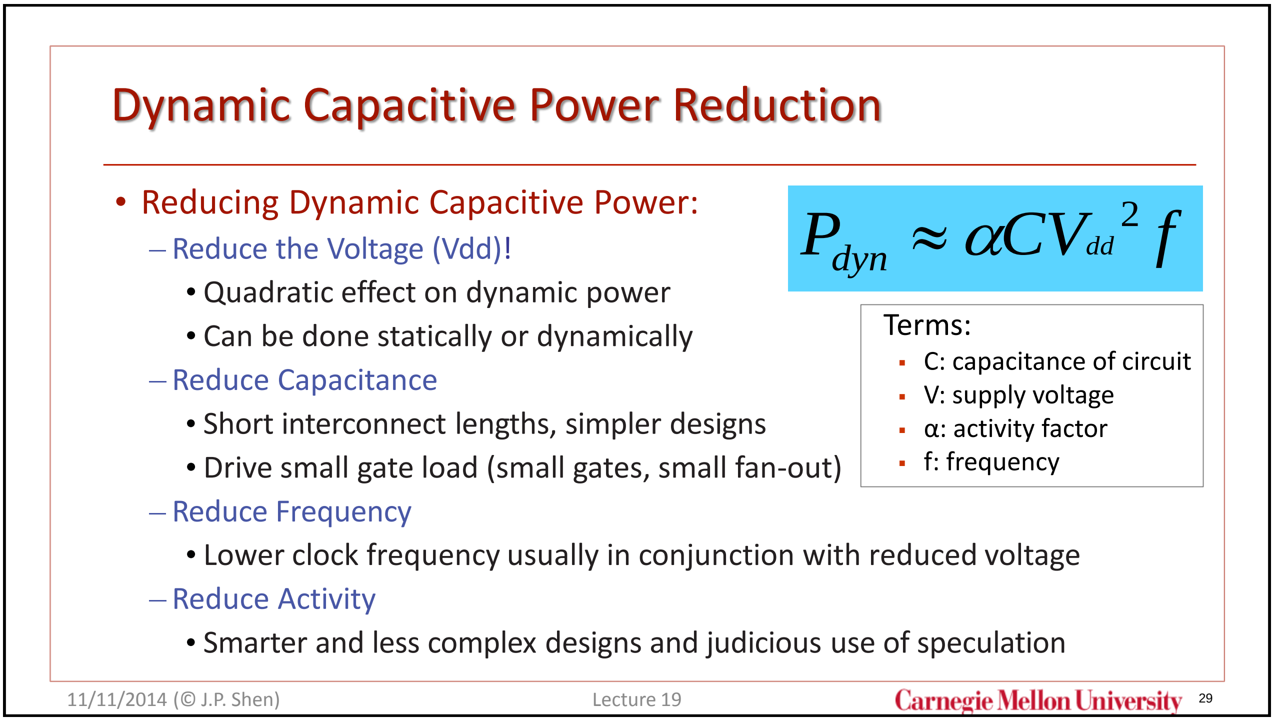


Total Power Consumption (Dynamic & Static)

- Total power consumption

$$P_{tot} = P_{dyn} + P_{sc} + P_{stat}$$

$$P_{tot} = C_L V_{CC}^2 f + V_{CC} I_{\max} \left(\frac{t_r + t_f}{2} \right) f + V_{CC} I_{leak}$$



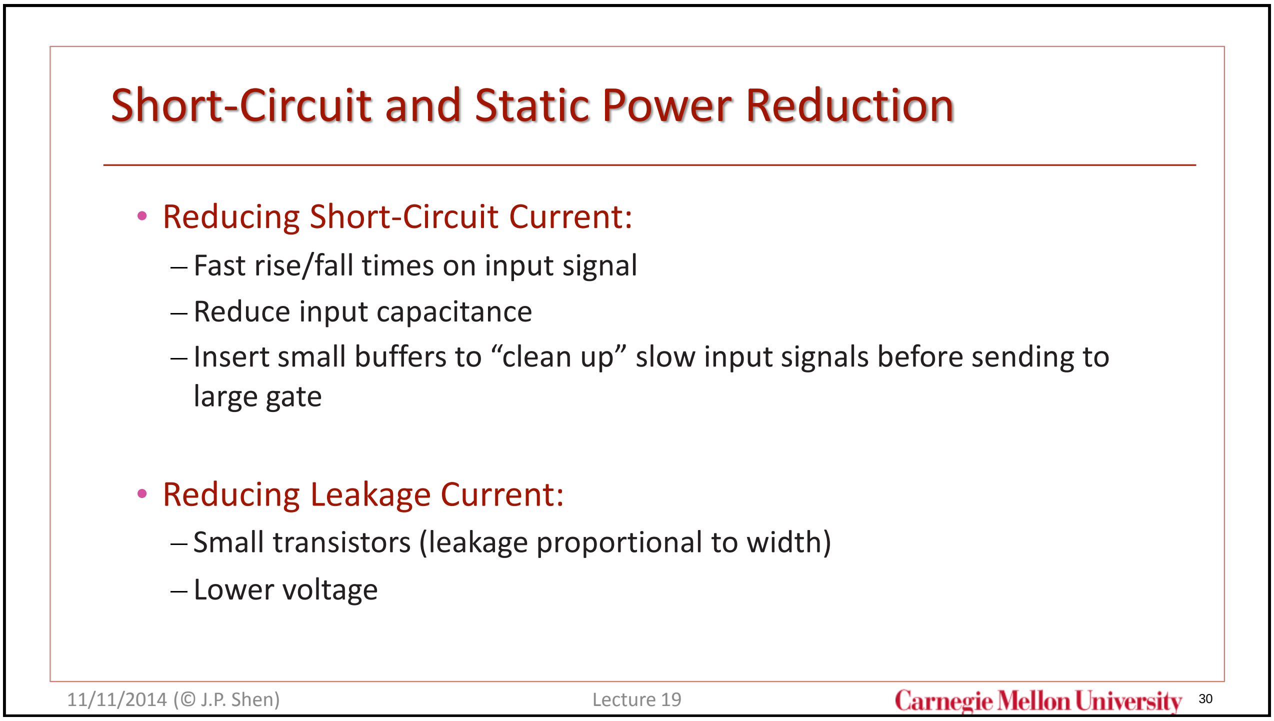
Dynamic Capacitive Power Reduction

- Reducing Dynamic Capacitive Power:
  - Reduce the Voltage (Vdd)!
    - Quadratic effect on dynamic power
    - Can be done statically or dynamically
  - Reduce Capacitance
    - Short interconnect lengths, simpler designs
    - Drive small gate load (small gates, small fan-out)
  - Reduce Frequency
    - Lower clock frequency usually in conjunction with reduced voltage
  - Reduce Activity
    - Smarter and less complex designs and judicious use of speculation

$$P_{dyn} \approx \alpha C V_{dd}^{2} f$$

Terms:

- C: capacitance of circuit
- V: supply voltage
- α: activity factor
- f: frequency



Short-Circuit and Static Power Reduction

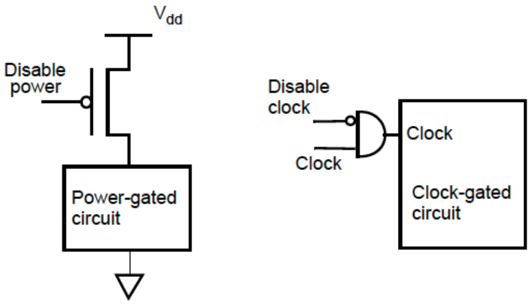
- Reducing Short-Circuit Current:
  - Fast rise/fall times on input signal
  - Reduce input capacitance
  - Insert small buffers to “clean up” slow input signals before sending to large gate
- Reducing Leakage Current:
  - Small transistors (leakage proportional to width)
  - Lower voltage

Circuit-Level Power Reduction Techniques

- Multiple voltages
  - Realize non-critical circuits with slower transistors
  - Voltage islands: V_{dd} and V_{th} are lower
    - Problem: supplying multiple V_{dd}
  - MTCMOS: only V_{th} is lower
- Multiple frequencies
  - Globally Asynchronous Locally Synchronous (GALS)
- Exploiting safety margins
  - Average case vs. worst case design
  - Razor latch [UMichigan]: Sample latch input twice, then compare, recover
- Body biasing
  - Reduce leakage by adapting V_{th}

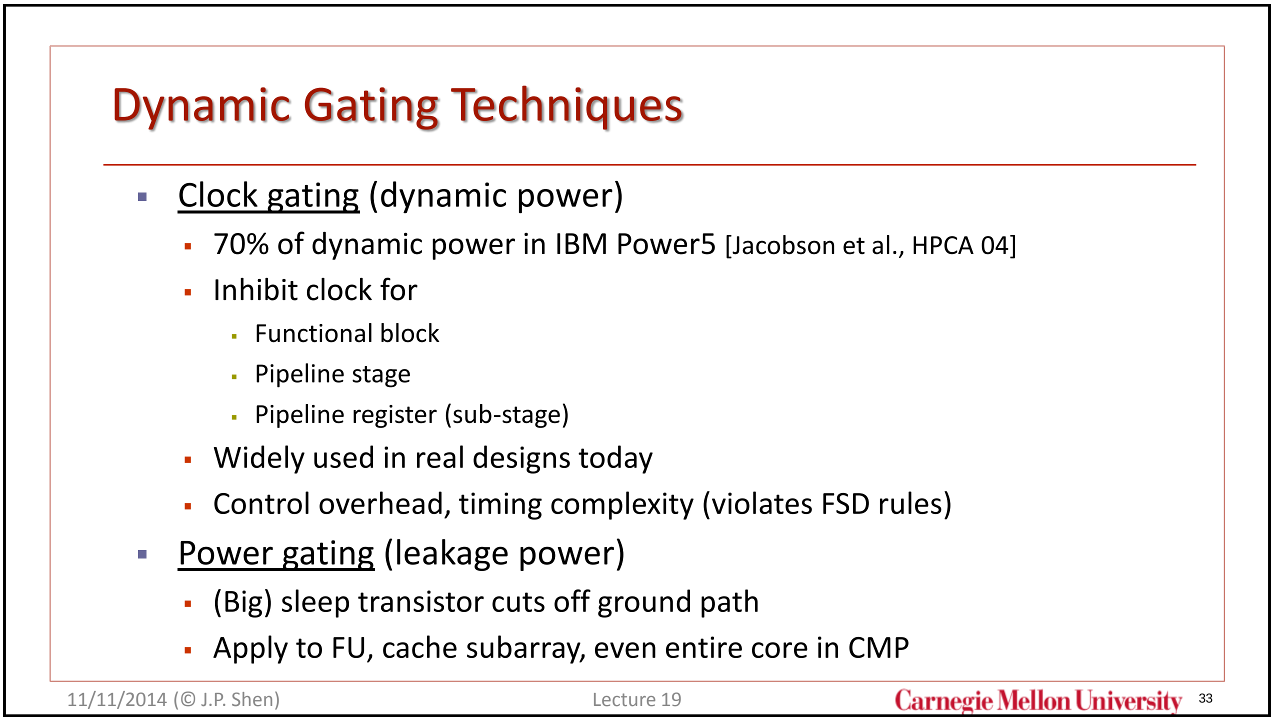
Reducing Power via Dynamic Gating

- Power gating cuts off power from idle units
- Clock gating cuts off power-hungry clock



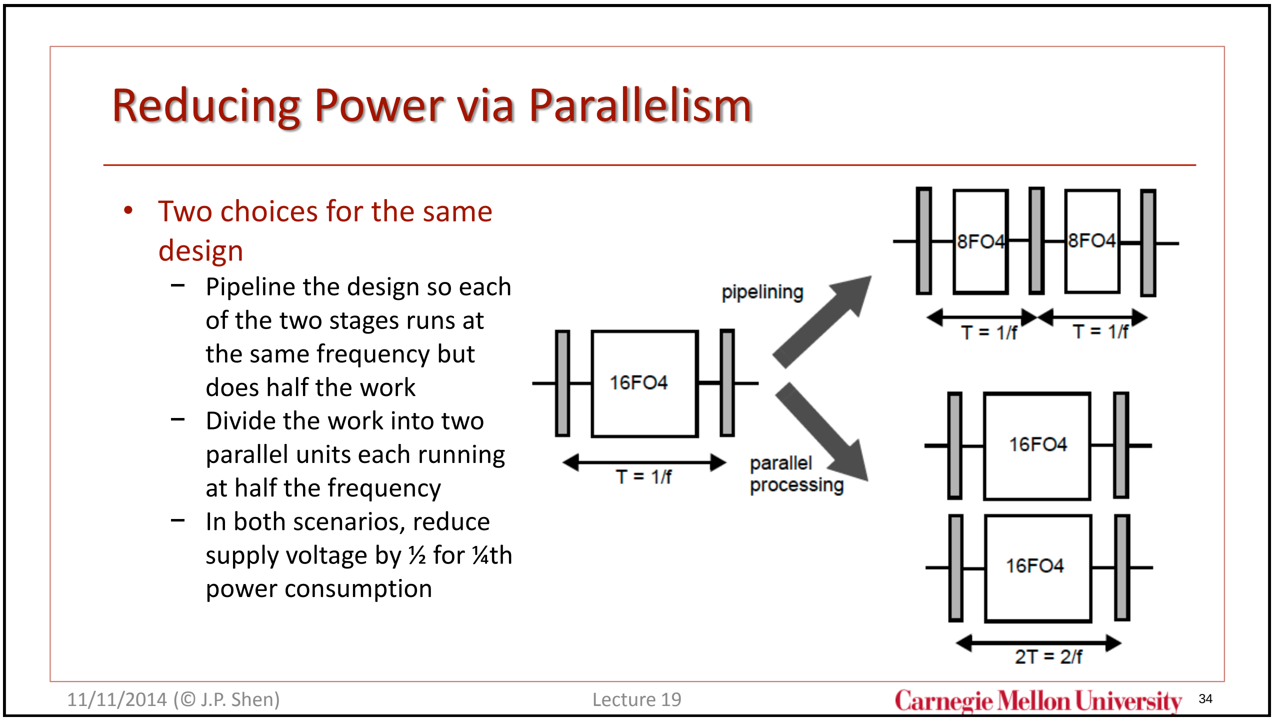
(a) Power gating

(b) Clock gating



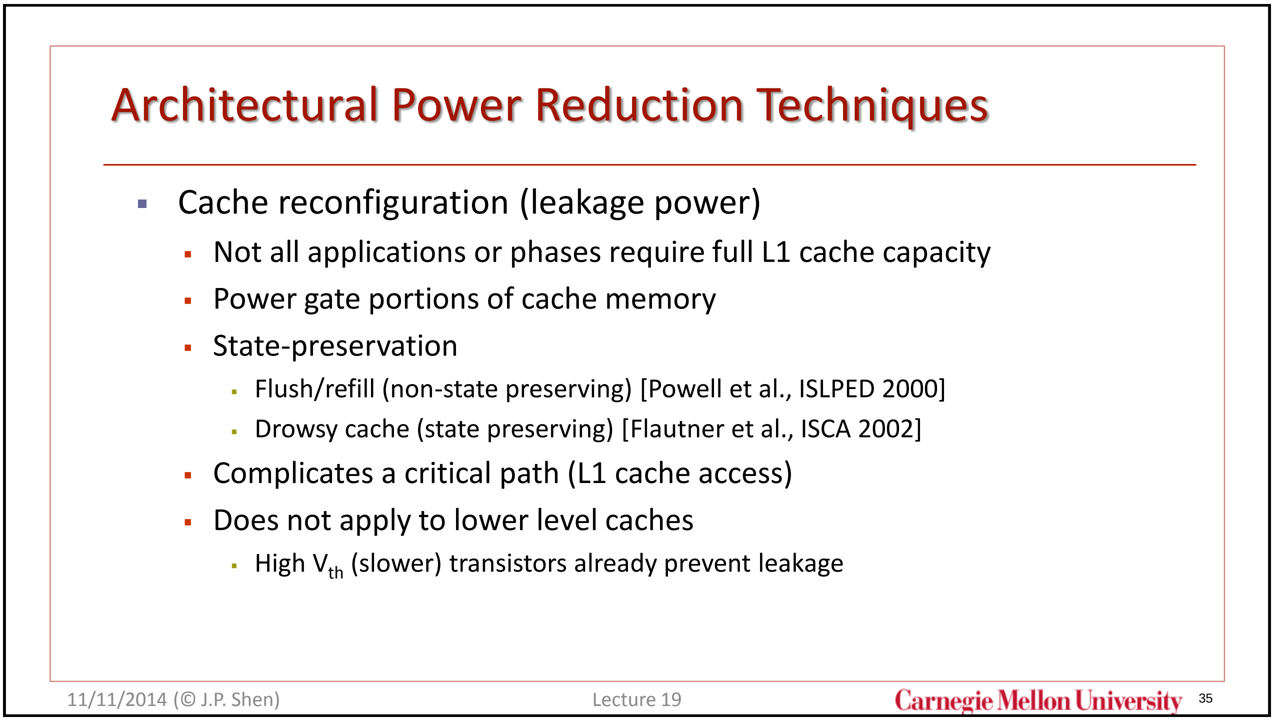
Dynamic Gating Techniques

- Clock gating (dynamic power)
  - 70% of dynamic power in IBM Power5 [Jacobson et al., HPCA 04]
  - Inhibit clock for
    - Functional block
    - Pipeline stage
    - Pipeline register (sub-stage)
  - Widely used in real designs today
  - Control overhead, timing complexity (violates FSD rules)
- Power gating (leakage power)
  - (Big) sleep transistor cuts off ground path
  - Apply to FU, cache subarray, even entire core in CMP



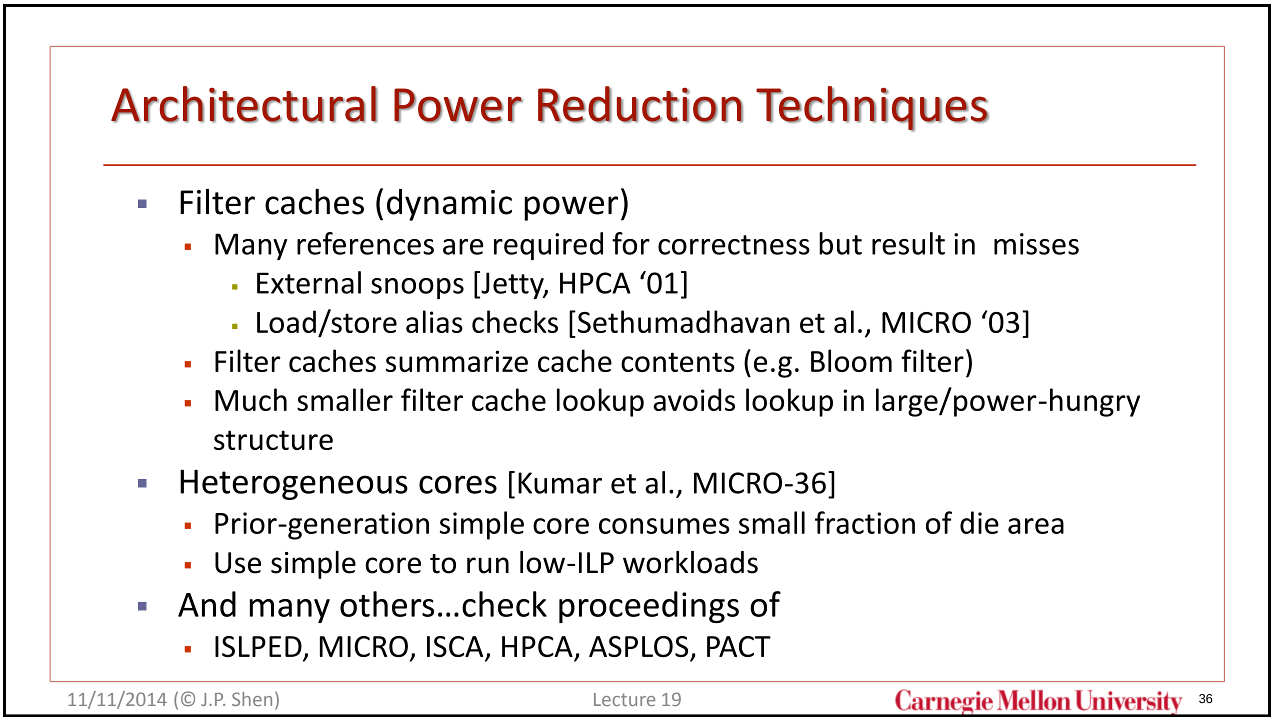
Reducing Power via Parallelism

- Two choices for the same design
  - Pipeline the design so each of the two stages runs at the same frequency but does half the work
  - Divide the work into two parallel units each running at half the frequency
  - In both scenarios, reduce supply voltage by ½ for ¼th power consumption



Architectural Power Reduction Techniques

- Cache reconfiguration (leakage power)
  - Not all applications or phases require full L1 cache capacity
  - Power gate portions of cache memory
  - State-preservation
    - Flush/refill (non-state preserving) [Powell et al., ISLPED 2000]
    - Drowsy cache (state preserving) [Flautner et al., ISCA 2002]
  - Complicates a critical path (L1 cache access)
  - Does not apply to lower level caches
    - High V_{th} (slower) transistors already prevent leakage



Architectural Power Reduction Techniques

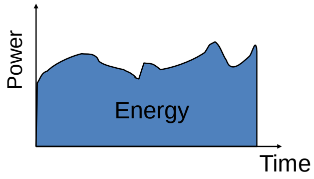
- Filter caches (dynamic power)
  - Many references are required for correctness but result in misses
    - External snoops [Jetty, HPCA '01]
    - Load/store alias checks [Sethumadhavan et al., MICRO '03]
  - Filter caches summarize cache contents (e.g. Bloom filter)
  - Much smaller filter cache lookup avoids lookup in large/power-hungry structure
- Heterogeneous cores [Kumar et al., MICRO-36]
  - Prior-generation simple core consumes small fraction of die area
  - Use simple core to run low-ILP workloads
- And many others…check proceedings of
  - ISLPED, MICRO, ISCA, HPCA, ASPLOS, PACT

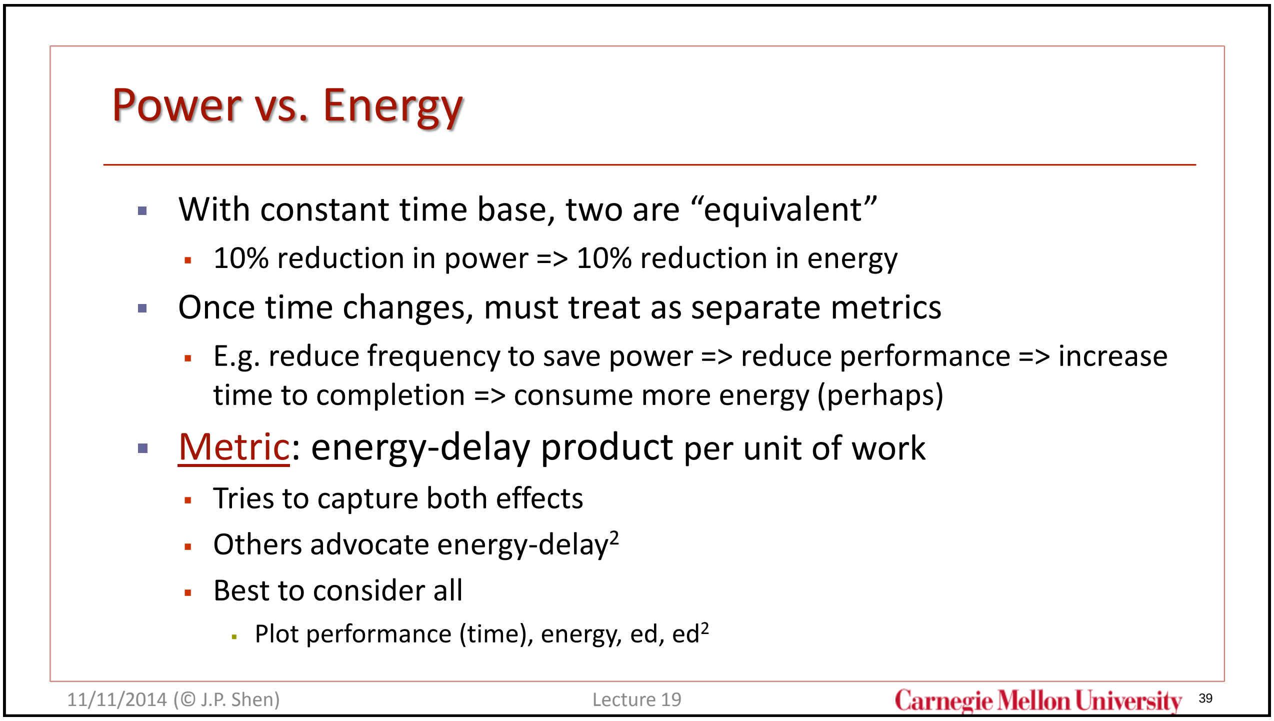
Architecture-Level Power Modeling

- Active power
  - Develop parameterizable high-level energy estimates for critical components
    - Caches, functional units, issue queues, reorder buffers, etc.
    - Difficult to do without access to real designs
    - Difficult to project into future designs and process technologies
  - Augment microarchitectural models to count activity at
    - Unit-level (e.g. Wattch [Brooks et al.])
    - Pipestage level (e.g. IBM Turandot)
    - Bit/gate level [Nam et al., ISLPED 2004]
  - Compute dot-product of activity and energy estimate vectors to determine total energy
  - Normalize by cycle count & frequency to determine power
- Critically important to get this right for future designs!

Power vs. Energy

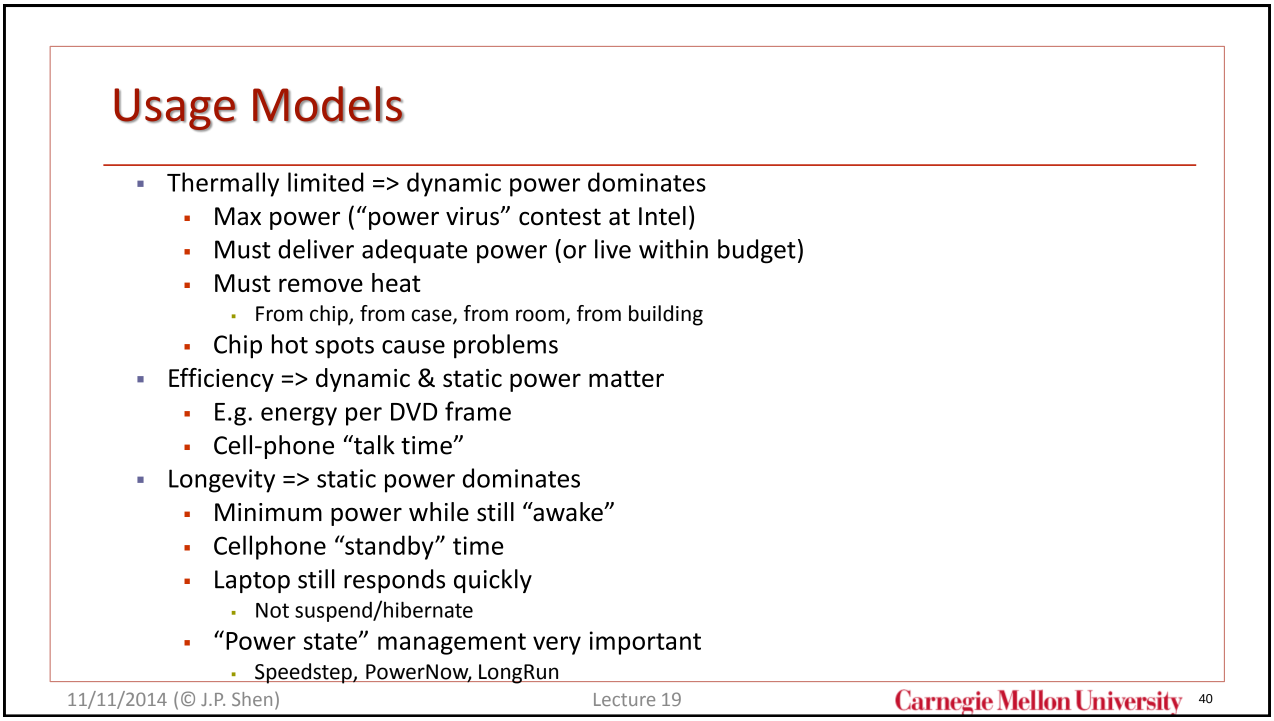
- Energy: integral of power (area under the curve)
  - Energy & power driven by different design constraints
- Power issues:
  - Power delivery (supply current @ right voltage)
  - Thermal (don't fry the chip)
  - Reliability effects (chip lifetime)
- Energy issues:
  - Limited energy capacity (battery)
  - Efficiency (work per unit energy)
- Different usage models drive tradeoffs





Power vs. Energy

- With constant time base, two are "equivalent"
  - 10% reduction in power => 10% reduction in energy
- Once time changes, must treat as separate metrics
  - E.g. reduce frequency to save power => reduce performance => increase time to completion => consume more energy (perhaps)
- Metric: energy-delay product per unit of work
  - Tries to capture both effects
  - Others advocate energy-delay^2
  - Best to consider all
    - Plot performance (time), energy, ed, ed^2

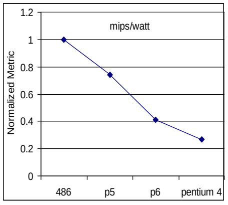
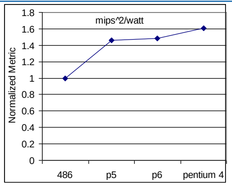
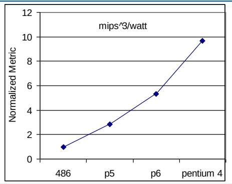


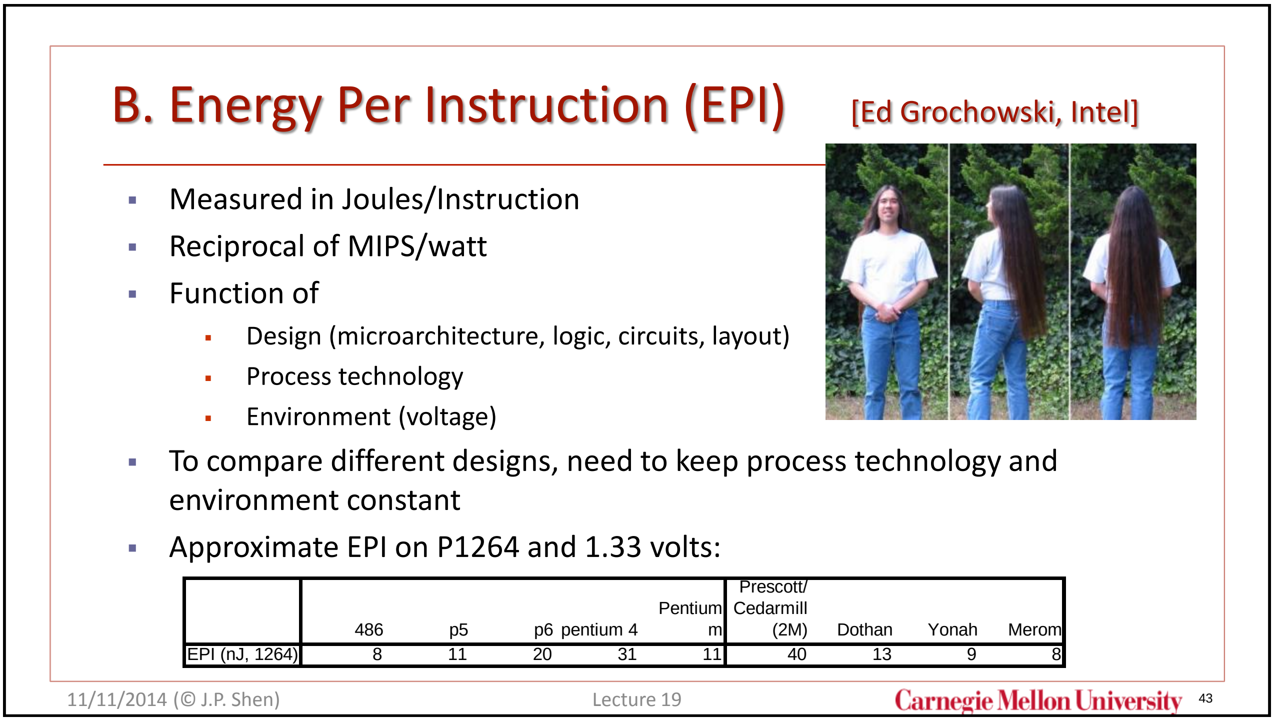
Usage Models

- Thermally limited => dynamic power dominates
  - Max power ("power virus" contest at Intel)
  - Must deliver adequate power (or live within budget)
  - Must remove heat
    - From chip, from case, from room, from building
  - Chip hot spots cause problems
- Efficiency => dynamic & static power matter
  - E.g. energy per DVD frame
  - Cell-phone "talk time"
- Longevity => static power dominates
  - Minimum power while still "awake"
  - Cellphone "standby" time
  - Laptop still responds quickly
    - Not suspend/hibernate
  - "Power state" management very important
    - Speedstep, PowerNow, LongRun

A.b. Power Efficiency Metrics

- Power is good metric for deciding on the thermal envelope of the processor
- Energy is good metric in battery constrained environments
  - Task executed at ½ speed but ¼ power means ½ the energy (2T * ¼ P = ½ E)
  - 2X battery life!
- Energy*Delay metric gives higher weight to performance
  - Same example above, ED ($(2T)^2$ * ¼ P) stays same
- Energy*Delay^2 gives even more weight to performance
  - Same example above shows that ½ speed is 2X worse on ED^2 metric

MIPS/watt	Common measure of power efficiency Equivalent to energy per instruction Independent of time Ideal metric for throughput performance $\frac{Mips}{Watt} = \frac{\frac{Instructions}{Second}}{\frac{Joules}{Second}} = \frac{Instructions}{Joule}$	
MIPS^2/watt	Equivalent to energy • delay Common metric for comparing logic families	
MIPS^3/watt	Equivalent to energy • delay^2 Assign increasing weight to time Appropriate metric for latency performance	

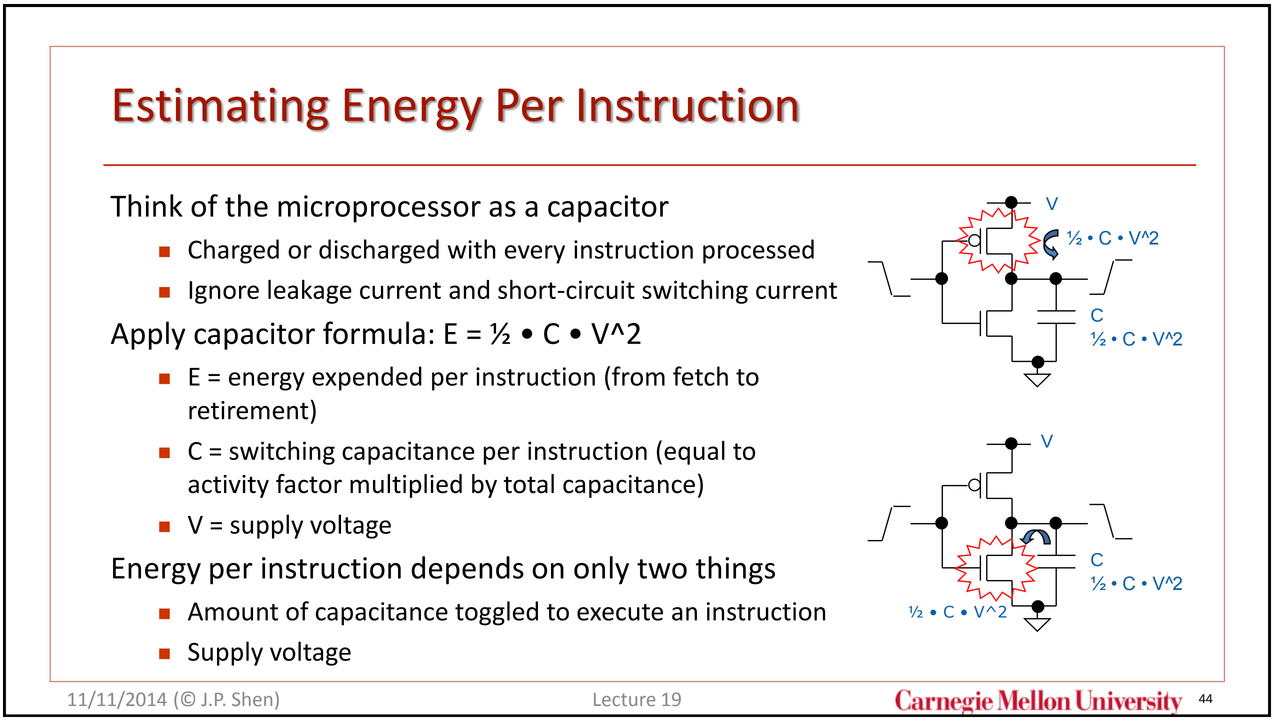


B. Energy Per Instruction (EPI)

[Ed Grochowski, Intel]

- Measured in Joules/Instruction
- Reciprocal of MIPS/watt
- Function of
  - Design (microarchitecture, logic, circuits, layout)
  - Process technology
  - Environment (voltage)
- To compare different designs, need to keep process technology and environment constant
- Approximate EPI on P1264 and 1.33 volts:

	486	p5	p6	pentium 4	Pentium m	Prescott/ Cedarmill (2M)	Dothan	Yonah	Merom
EPI (nJ, 1264)	8	11	20	31	11	40	13	9	8



Estimating Energy Per Instruction

Think of the microprocessor as a capacitor

- Charged or discharged with every instruction processed
- Ignore leakage current and short-circuit switching current

Apply capacitor formula: E = ½ • C • V^2

- E = energy expended per instruction (from fetch to retirement)
- C = switching capacitance per instruction (equal to activity factor multiplied by total capacitance)
- V = supply voltage

Energy per instruction depends on only two things

- Amount of capacitance toggled to execute an instruction
- Supply voltage

Two Classes of Computer Workloads

<u>Scalar workloads</u> • IPC 0.2 – 2 • Examples: software development tools, office productivity suites, OS kernel routines	<u>Scalar uarch</u> • Reduce effective execution latency! • Uarch techniques: speculative execution, branch prediction, caching • Example: Pentium® 4 processor • High energy/instruction
<u>Parallel workloads</u> • IPC of 4 to several thousand • Examples: 3D graphics, media processing, scientific applications • Utilize instruction-level parallelism and thread-level parallelism	<u>Parallel uarch</u> • Provide large execution bandwidth! • Uarch techniques: wide superscalar, multiprocessing, multithreading • Examples: IBM/Sony Cell, Sun Niagara • Low energy/instruction

- Multithreaded programs contain both sequential and parallel phases!
  - Want to minimize run-time for both sequential and parallel phases
  - Amdahl's law: speedup via parallelization will be limited by sequential component

Power Constrained Environment

Assumptions

- Very large chip-level multiprocessors are possible
- Software is multithreaded
- Power is the limiter

Goal: maintain power consumption within a fixed budget regardless of what the processor or software do

- The key to designing a microprocessor that can achieve both high scalar performance and high throughput performance is to *dynamically vary the amount of energy expended to process each instruction according to the amount of parallelism available in the software*.

P = EPI • IPS

- P = fixed power budget
- EPI = energy per instruction
- IPS = aggregate number of instructions retired per second

How to Vary Energy/Instruction

Four techniques

- Voltage/frequency scaling -> control V
- Asymmetric cores -> control C
- Variable-size cores -> control C
- Speculation control -> control C

Quantify min-max range of possible energy/instruction

- In contrast to incremental energy/instruction

Assume inactive cores consume negligible power

- Clock gating, sleep transistors, and/or body bias are required

B.a. EPI Throttling and Asymmetric Cores

Chip-level multiprocessor built with two types of cores

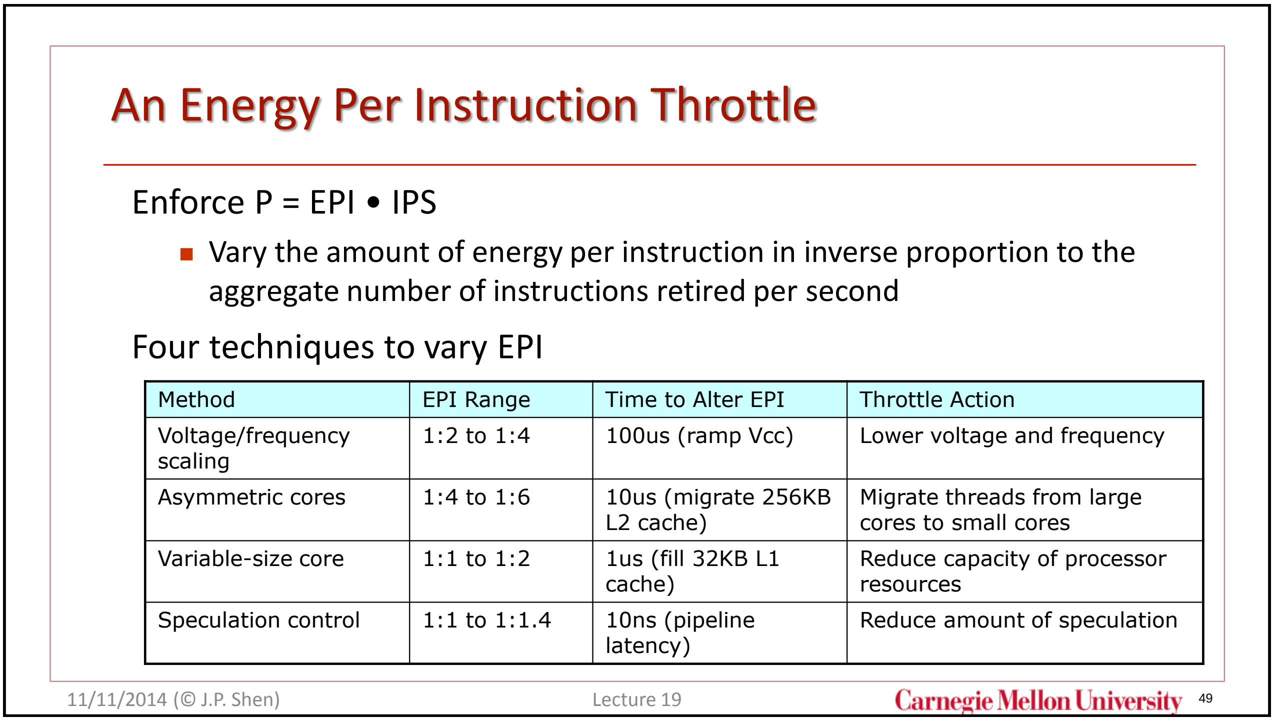
- For example, 1 large core and 25 small cores
- Migrate threads between large/small cores

Energy/instruction

- 1:4 with i486™ and Pentium® 4 processors

	Large core	Small core
Microarchitecture	Out-of-order, 128 entry ROB	In-order
Width	3-4	1
Pipeline depth	20-30	5
Normalized performance	5-8x	1x
Normalized power	20-50x	1x
Normalized EPI	4-6x	1x

Operation of a chip-level multiprocessor	Low thread parallelism	Run large core
	High thread parallelism	Run many small cores



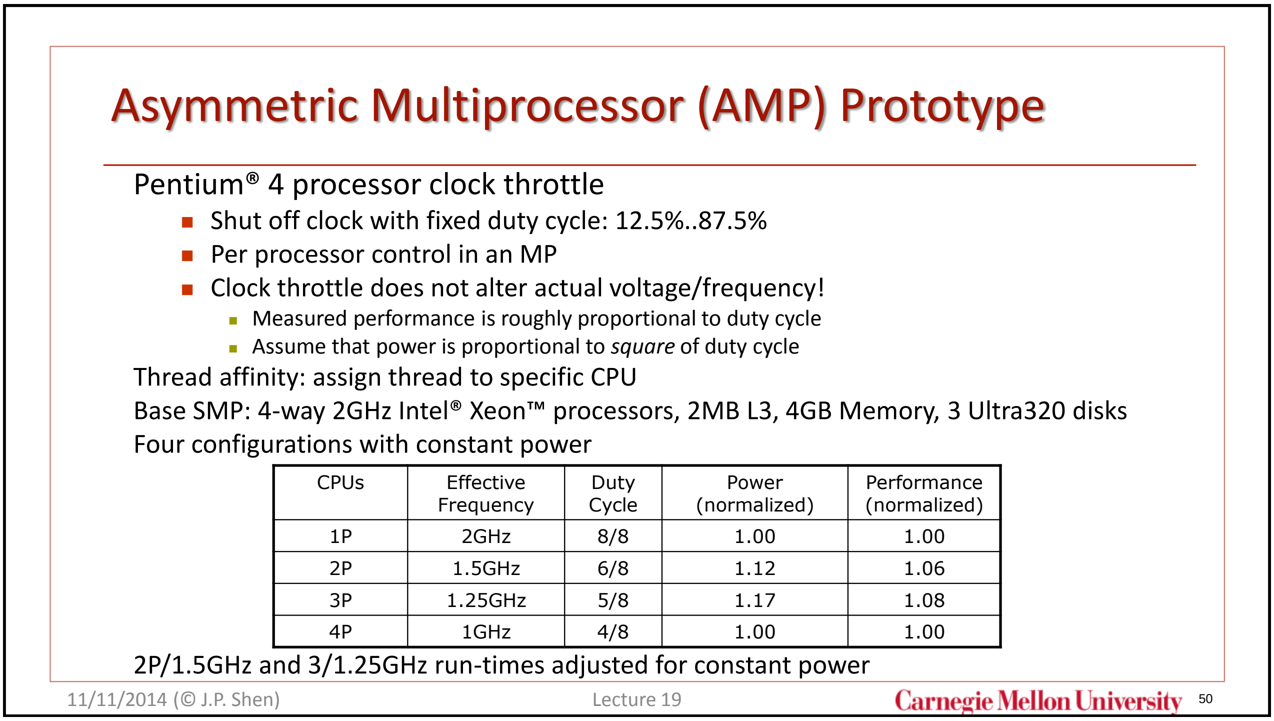
An Energy Per Instruction Throttle

Enforce P = EPI • IPS

- Vary the amount of energy per instruction in inverse proportion to the aggregate number of instructions retired per second

Four techniques to vary EPI

Method	EPI Range	Time to Alter EPI	Throttle Action
Voltage/frequency scaling	1:2 to 1:4	100us (ramp Vcc)	Lower voltage and frequency
Asymmetric cores	1:4 to 1:6	10us (migrate 256KB L2 cache)	Migrate threads from large cores to small cores
Variable-size core	1:1 to 1:2	1us (fill 32KB L1 cache)	Reduce capacity of processor resources
Speculation control	1:1 to 1:1.4	10ns (pipeline latency)	Reduce amount of speculation



Asymmetric Multiprocessor (AMP) Prototype

Pentium® 4 processor clock throttle

- Shut off clock with fixed duty cycle: 12.5%..87.5%
- Per processor control in an MP
- Clock throttle does not alter actual voltage/frequency!
  - Measured performance is roughly proportional to duty cycle
  - Assume that power is proportional to *square* of duty cycle

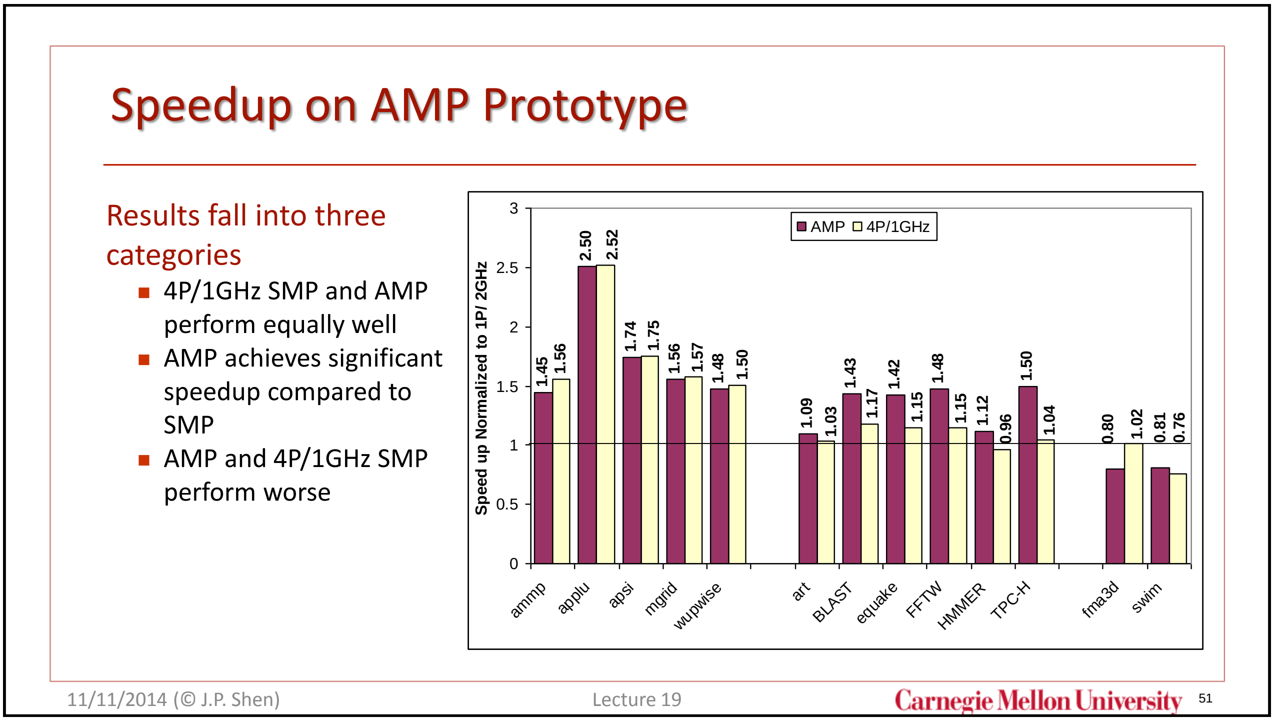
Thread affinity: assign thread to specific CPU

Base SMP: 4-way 2GHz Intel® Xeon™ processors, 2MB L3, 4GB Memory, 3 Ultra320 disks

Four configurations with constant power

CPUs	Effective Frequency	Duty Cycle	Power (normalized)	Performance (normalized)
1P	2GHz	8/8	1.00	1.00
2P	1.5GHz	6/8	1.12	1.06
3P	1.25GHz	5/8	1.17	1.08
4P	1GHz	4/8	1.00	1.00

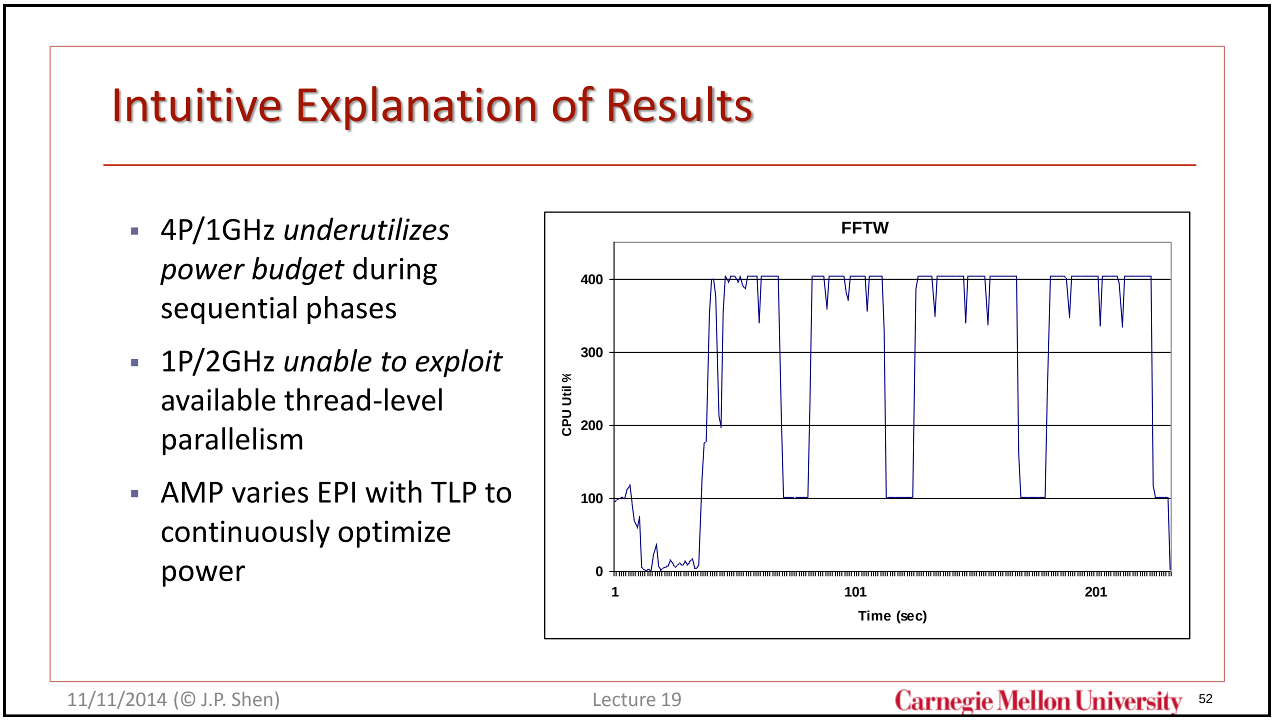
2P/1.5GHz and 3/1.25GHz run-times adjusted for constant power



Speedup on AMP Prototype

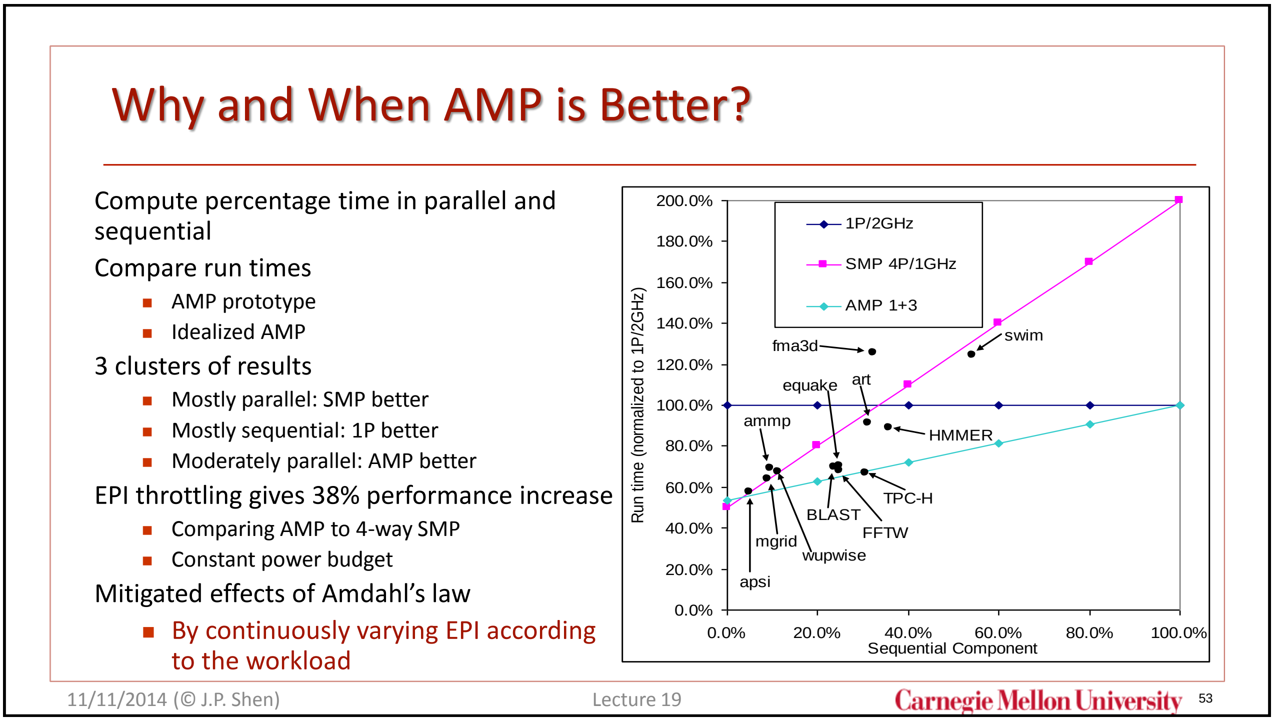
Results fall into three categories

- 4P/1GHz SMP and AMP perform equally well
- AMP achieves significant speedup compared to SMP
- AMP and 4P/1GHz SMP perform worse



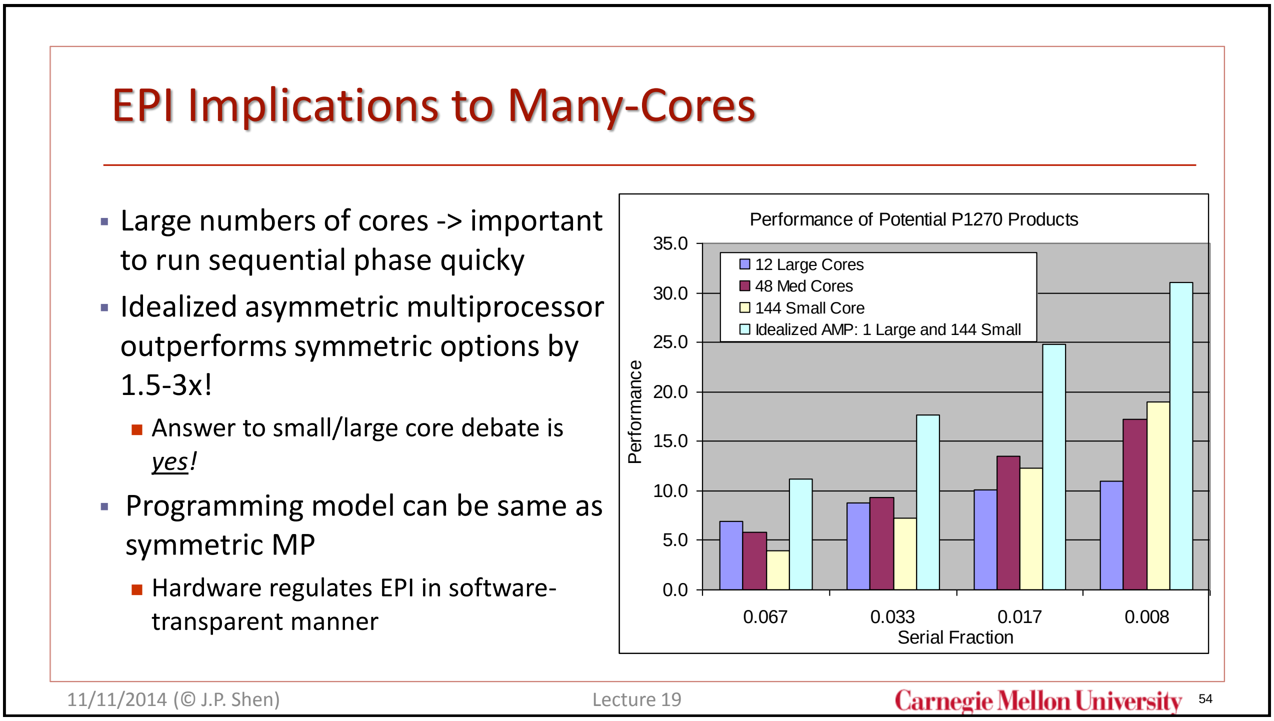
Intuitive Explanation of Results

- 4P/1GHz *underutilizes power budget* during sequential phases
- 1P/2GHz *unable to exploit* available thread-level parallelism
- AMP varies EPI with TLP to continuously optimize power



Why and When AMP is Better?

- Compute percentage time in parallel and sequential
- Compare run times
  - AMP prototype
  - Idealized AMP
- 3 clusters of results
  - Mostly parallel: SMP better
  - Mostly sequential: 1P better
  - Moderately parallel: AMP better
- EPI throttling gives 38% performance increase
  - Comparing AMP to 4-way SMP
  - Constant power budget
- Mitigated effects of Amdahl’s law
  - By continuously varying EPI according to the workload



EPI Implications to Many-Cores

- Large numbers of cores -> important to run sequential phase quicky
- Idealized asymmetric multiprocessor outperforms symmetric options by 1.5-3x!
  - Answer to small/large core debate is *yes*!
- Programming model can be same as symmetric MP
  - Hardware regulates EPI in software-transparent manner

C. Extreme Energy Efficiency Challenges

a. Active Power Saving
b. Standby Power Saving
c. Energy Harvesting

Electrical & Computer ENGINEERING

Energy Storage Challenge

How long between recharges?

iPad2

- 25Wh = 90kJ
- 10h use = avg 2.5W

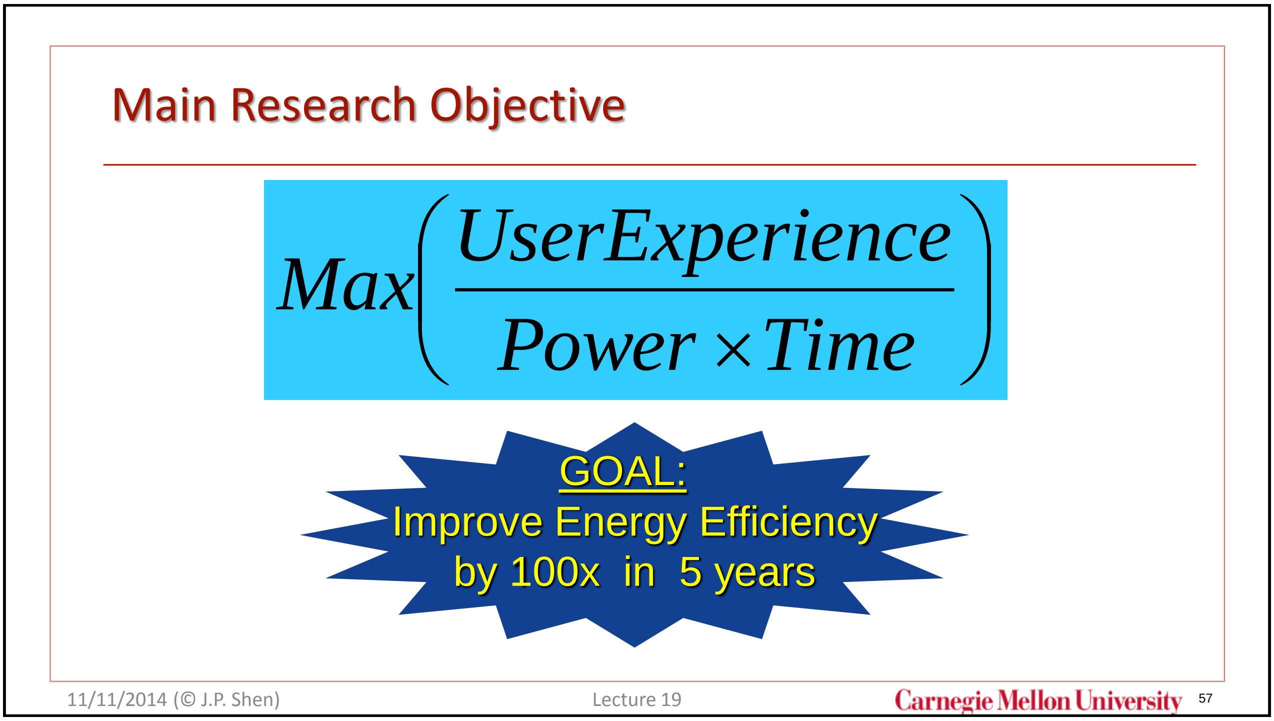
Kindle3

- 6.5Wh = 23kJ
- “30 day use” = 30h = avg 220mW

Typical smartphone

- 32g, 13cc, 5.5Wh = 18kJ
- +5h charging @ max 1W
- 20mW static power = 10 days standby
- 150mW notifications = 1.3 day standby
- “typical usage” 5kJ active + 13kJ standby = 1 battery charge

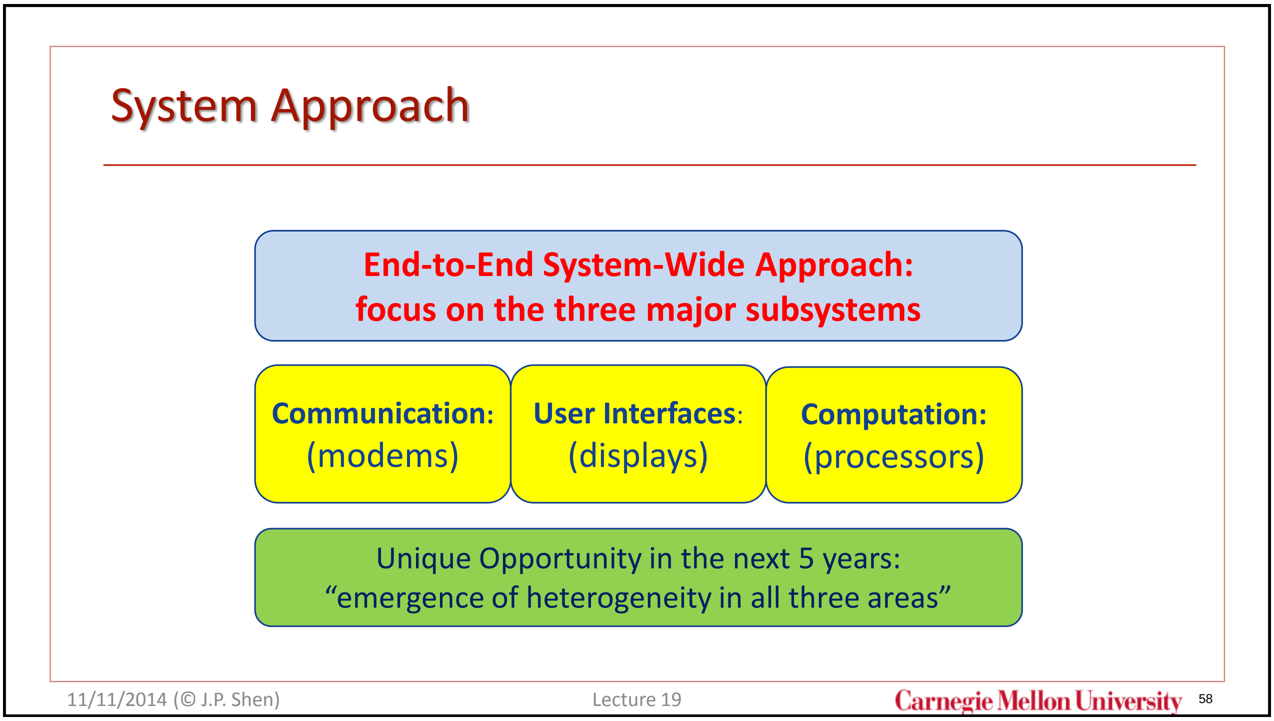
iPad3



Main Research Objective

$$Max\left(\frac{UserExperience}{Power \times Time}\right)$$

GOAL:
Improve Energy Efficiency
by 100x in 5 years

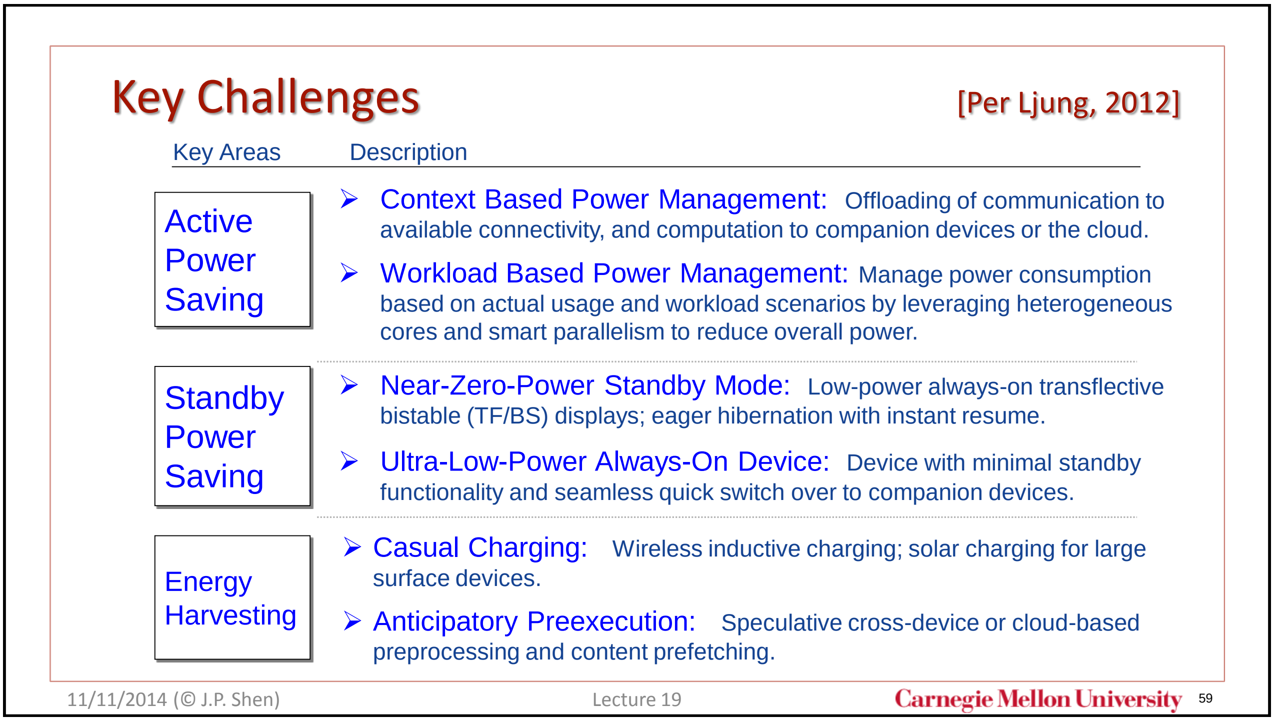


System Approach

**End-to-End System-Wide Approach:
focus on the three major subsystems**

Communication: (modems)	**User Interfaces:** (displays)	**Computation:** (processors)

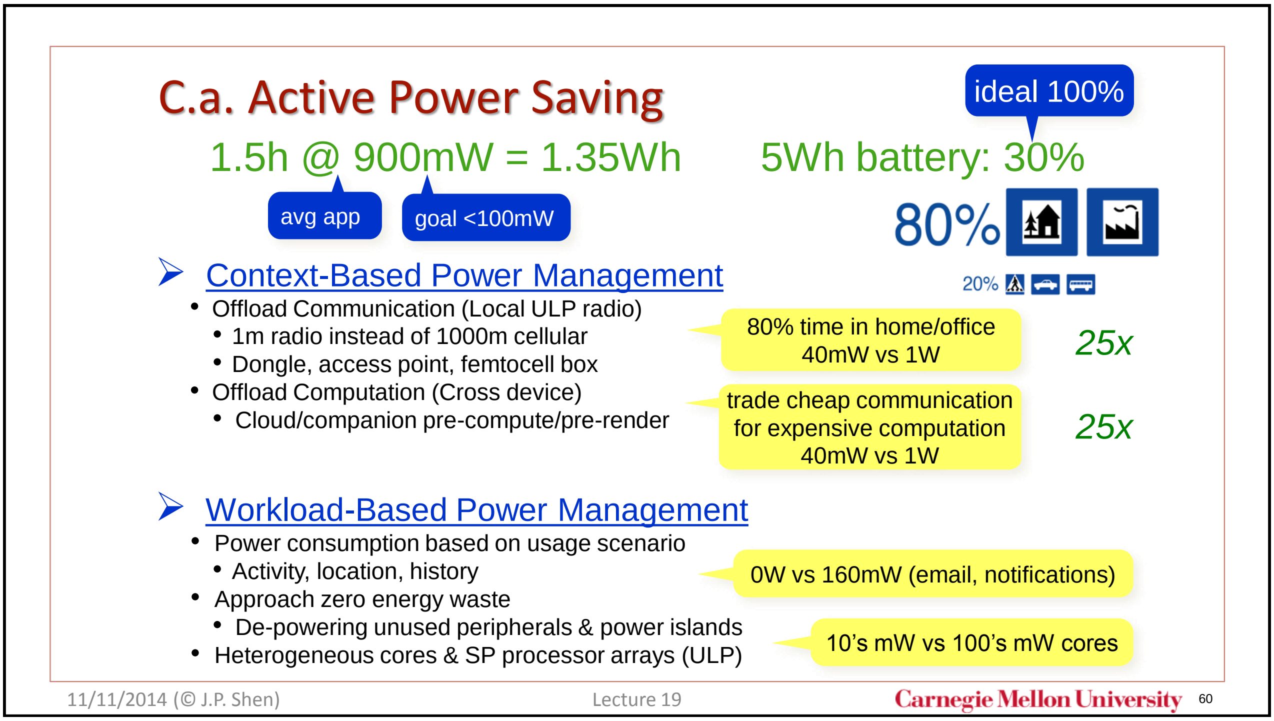
Unique Opportunity in the next 5 years:
"emergence of heterogeneity in all three areas"



Key Challenges

[Per Ljung, 2012]

Key Areas	Description
Active Power Saving	➢ Context Based Power Management: Offloading of communication to available connectivity, and computation to companion devices or the cloud. ➢ Workload Based Power Management: Manage power consumption based on actual usage and workload scenarios by leveraging heterogeneous cores and smart parallelism to reduce overall power.
Standby Power Saving	➢ Near-Zero-Power Standby Mode: Low-power always-on transflective bistable (TF/BS) displays; eager hibernation with instant resume. ➢ Ultra-Low-Power Always-On Device: Device with minimal standby functionality and seamless quick switch over to companion devices.
Energy Harvesting	➢ Casual Charging: Wireless inductive charging; solar charging for large surface devices. ➢ Anticipatory Preexecution: Speculative cross-device or cloud-based preprocessing and content prefetching.



C.a. Active Power Saving

1.5h @ 900mW = 1.35Wh

avg app

goal <100mW

5Wh battery: 30%

ideal 100%

80%

20%

- ➢ Context-Based Power Management
  - Offload Communication (Local ULP radio)
    - 1m radio instead of 1000m cellular
    - Dongle, access point, femtocell box
  - Offload Computation (Cross device)
    - Cloud/companion pre-compute/pre-render

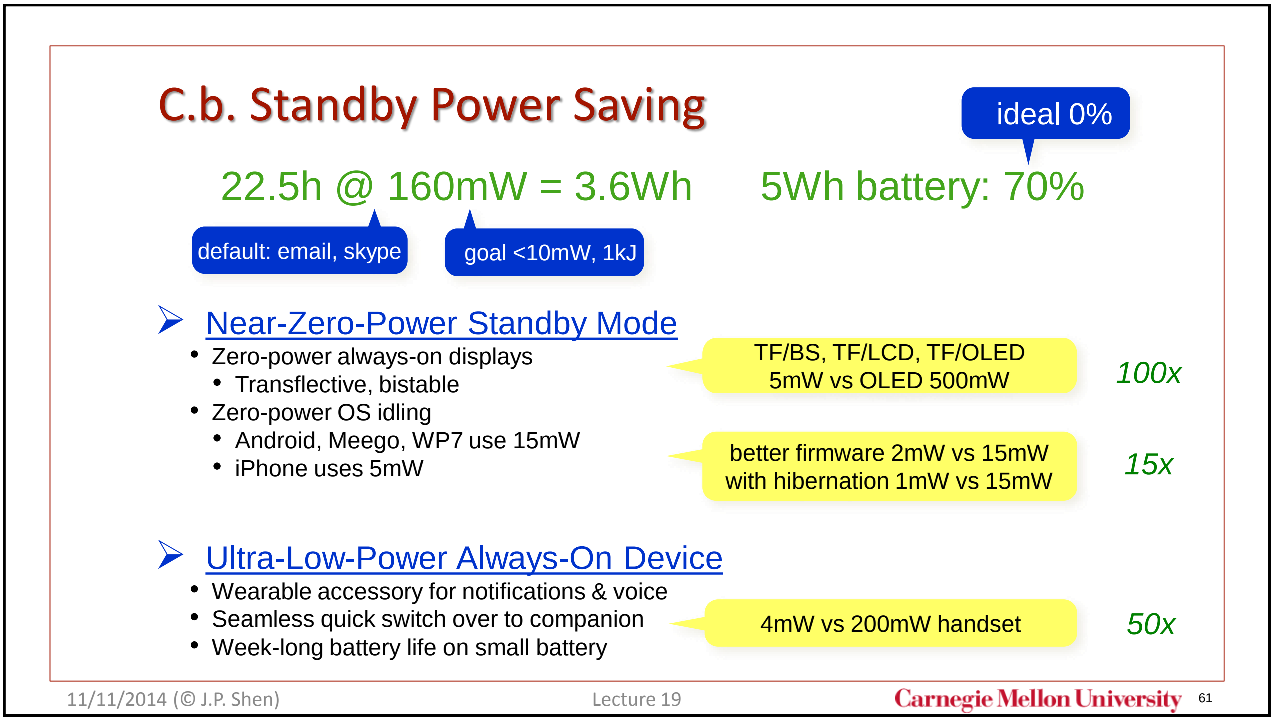
80% time in home/office 40mW vs 1W — 25x

trade cheap communication for expensive computation 40mW vs 1W — 25x

- ➢ Workload-Based Power Management
  - Power consumption based on usage scenario
    - Activity, location, history
  - Approach zero energy waste
    - De-powering unused peripherals & power islands
  - Heterogeneous cores & SP processor arrays (ULP)

0W vs 160mW (email, notifications)

10's mW vs 100's mW cores



C.b. Standby Power Saving

ideal 0%

22.5h @ 160mW = 3.6Wh 5Wh battery: 70%

default: email, skype

goal <10mW, 1kJ

Near-Zero-Power Standby Mode

- Zero-power always-on displays
  - Transflective, bistable
- Zero-power OS idling
  - Android, Meego, WP7 use 15mW
  - iPhone uses 5mW

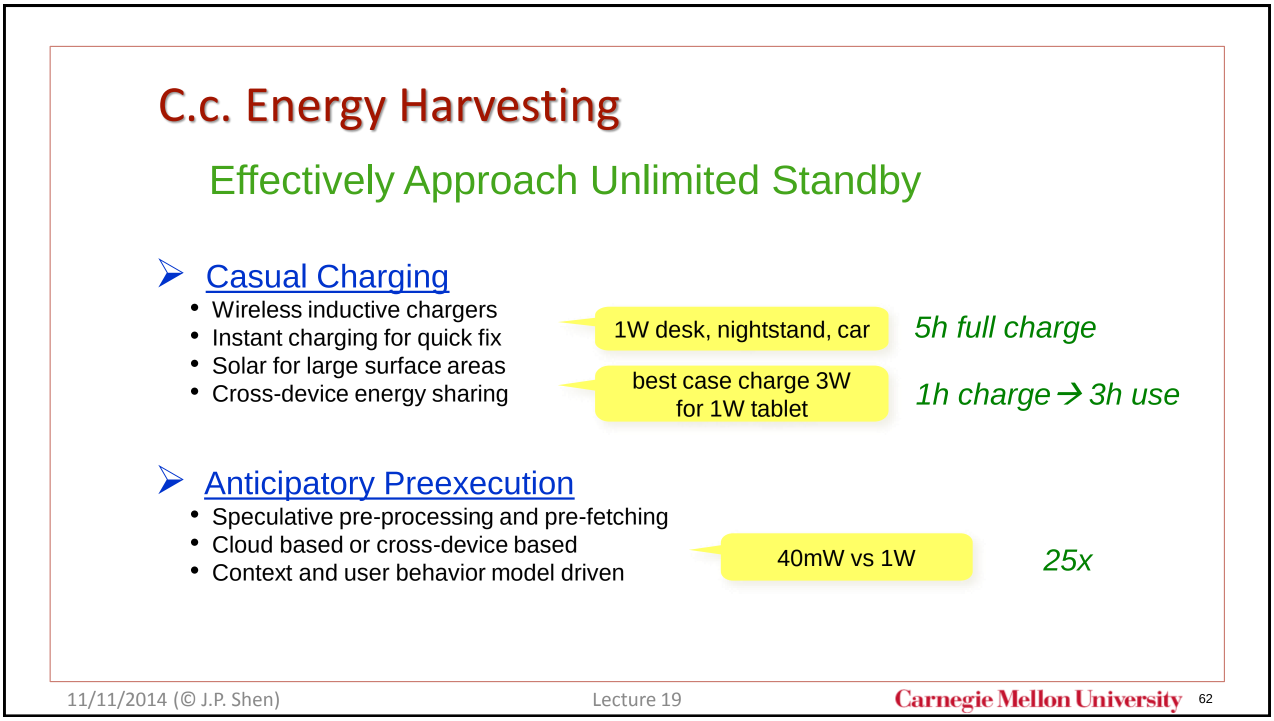
TF/BS, TF/LCD, TF/OLED 5mW vs OLED 500mW *100x*

better firmware 2mW vs 15mW with hibernation 1mW vs 15mW *15x*

Ultra-Low-Power Always-On Device

- Wearable accessory for notifications & voice
- Seamless quick switch over to companion
- Week-long battery life on small battery

4mW vs 200mW handset *50x*



C.c. Energy Harvesting

Effectively Approach Unlimited Standby

Casual Charging

- Wireless inductive chargers
- Instant charging for quick fix
- Solar for large surface areas
- Cross-device energy sharing

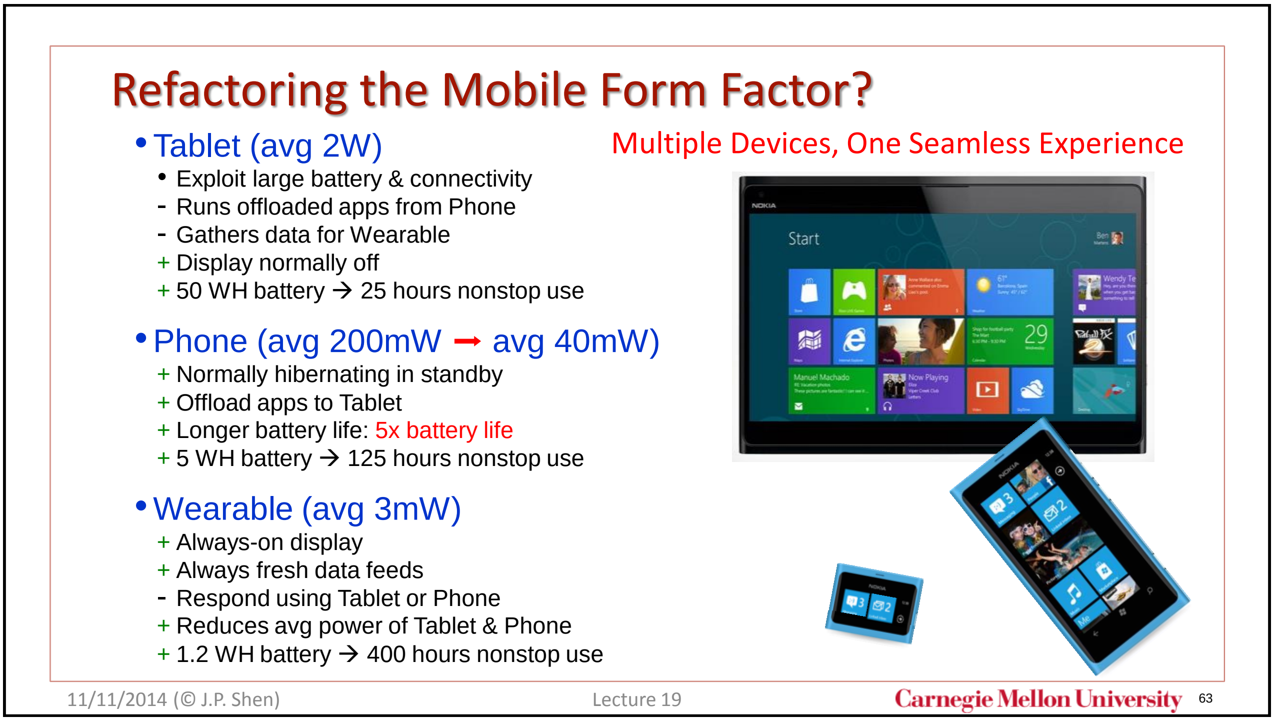
1W desk, nightstand, car *5h full charge*

best case charge 3W for 1W tablet *1h charge → 3h use*

Anticipatory Preexecution

- Speculative pre-processing and pre-fetching
- Cloud based or cross-device based
- Context and user behavior model driven

40mW vs 1W *25x*

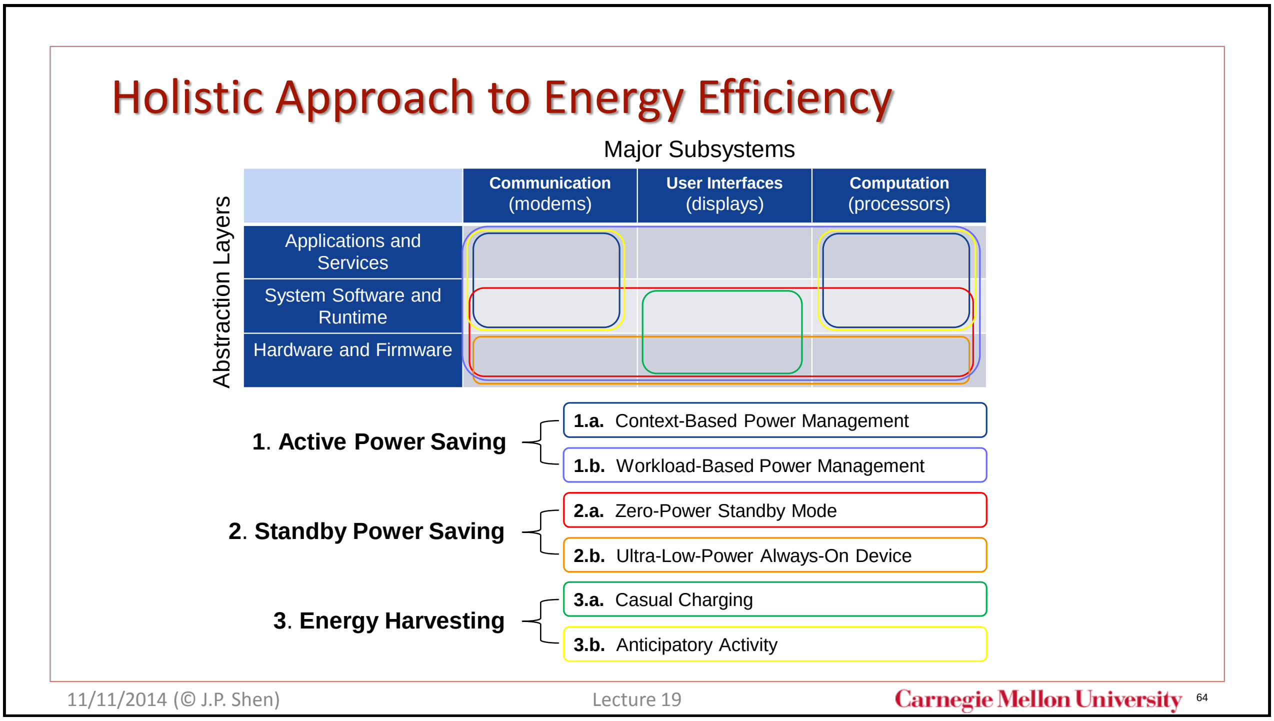


Refactoring the Mobile Form Factor?

Multiple Devices, One Seamless Experience

- Tablet (avg 2W)
  - • Exploit large battery & connectivity
  - - Runs offloaded apps from Phone
  - - Gathers data for Wearable
  - + Display normally off
  - + 50 WH battery → 25 hours nonstop use
- Phone (avg 200mW → avg 40mW)
  - + Normally hibernating in standby
  - + Offload apps to Tablet
  - + Longer battery life: 5x battery life
  - + 5 WH battery → 125 hours nonstop use
- Wearable (avg 3mW)
  - + Always-on display
  - + Always fresh data feeds
  - - Respond using Tablet or Phone
  - + Reduces avg power of Tablet & Phone
  - + 1.2 WH battery → 400 hours nonstop use

11/11/2014 (© J.P. Shen) Lecture 19 Carnegie Mellon University 63



Holistic Approach to Energy Efficiency

Major Subsystems

Abstraction Layers

	Communication (modems)	User Interfaces (displays)	Computation (processors)
Applications and Services			
System Software and Runtime			
Hardware and Firmware			

1. **Active Power Saving**
   - **1.a.** Context-Based Power Management
   - **1.b.** Workload-Based Power Management
2. **Standby Power Saving**
   - **2.a.** Zero-Power Standby Mode
   - **2.b.** Ultra-Low-Power Always-On Device
3. **Energy Harvesting**
   - **3.a.** Casual Charging
   - **3.b.** Anticipatory Activity

11/11/2014 (© J.P. Shen) Lecture 19 Carnegie Mellon University 64