CSC 572: Mobile Systems Architecture
Fall 2018: Intro

Instructor: Yuhao Zhu

Department of Computer Science
University of Rochester

Where to Find Stuff

- http://cs.rochester.edu/courses/572/fall2018/
  - General info
  - Various course materials
- Slack for ***all*** communication
  - Only place for all announcements
  - Q&A

Basic Building Block: Transistors

Basic Building Block: Transistors

MOS = Metal Oxide Semiconductor

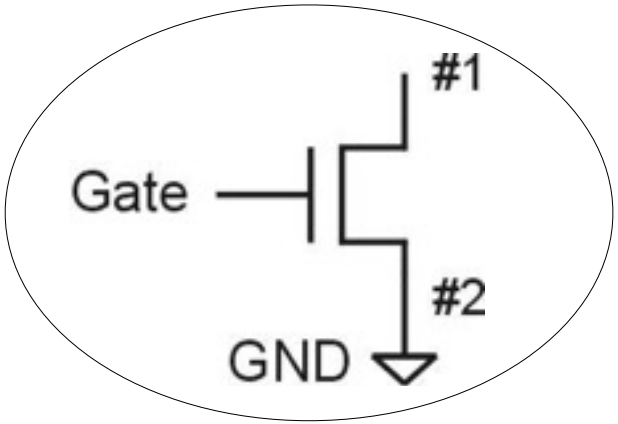
- two types: n-type and p-type

Basic Building Block: Transistors

MOS = Metal Oxide Semiconductor

- two types: n-type and p-type

n-type (NMOS)



Terminal #2 must be connected to GND (0V).

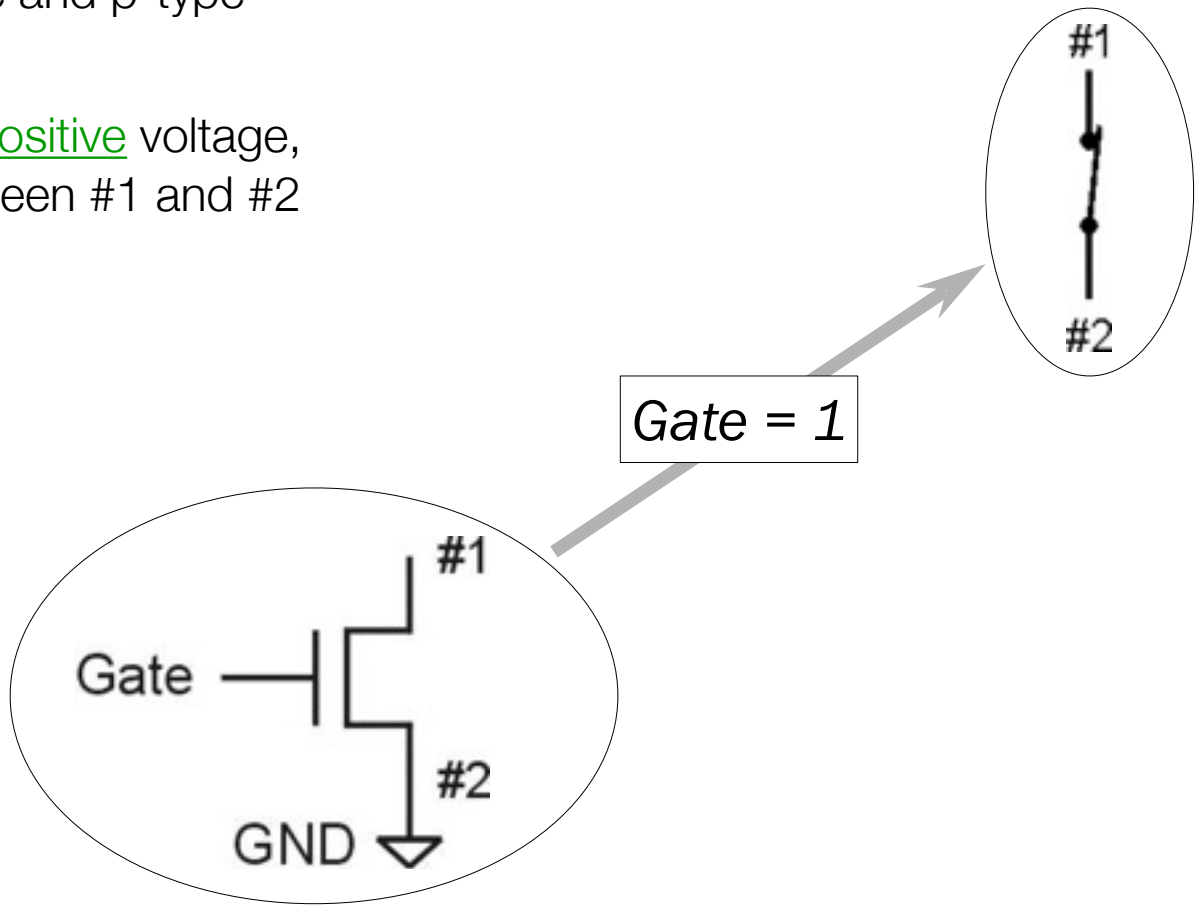
Basic Building Block: Transistors

MOS = Metal Oxide Semiconductor

- two types: n-type and p-type

n-type (NMOS)

- when Gate has positive voltage, short circuit between #1 and #2 (switch closed)



Terminal #2 must be connected to GND (0V).

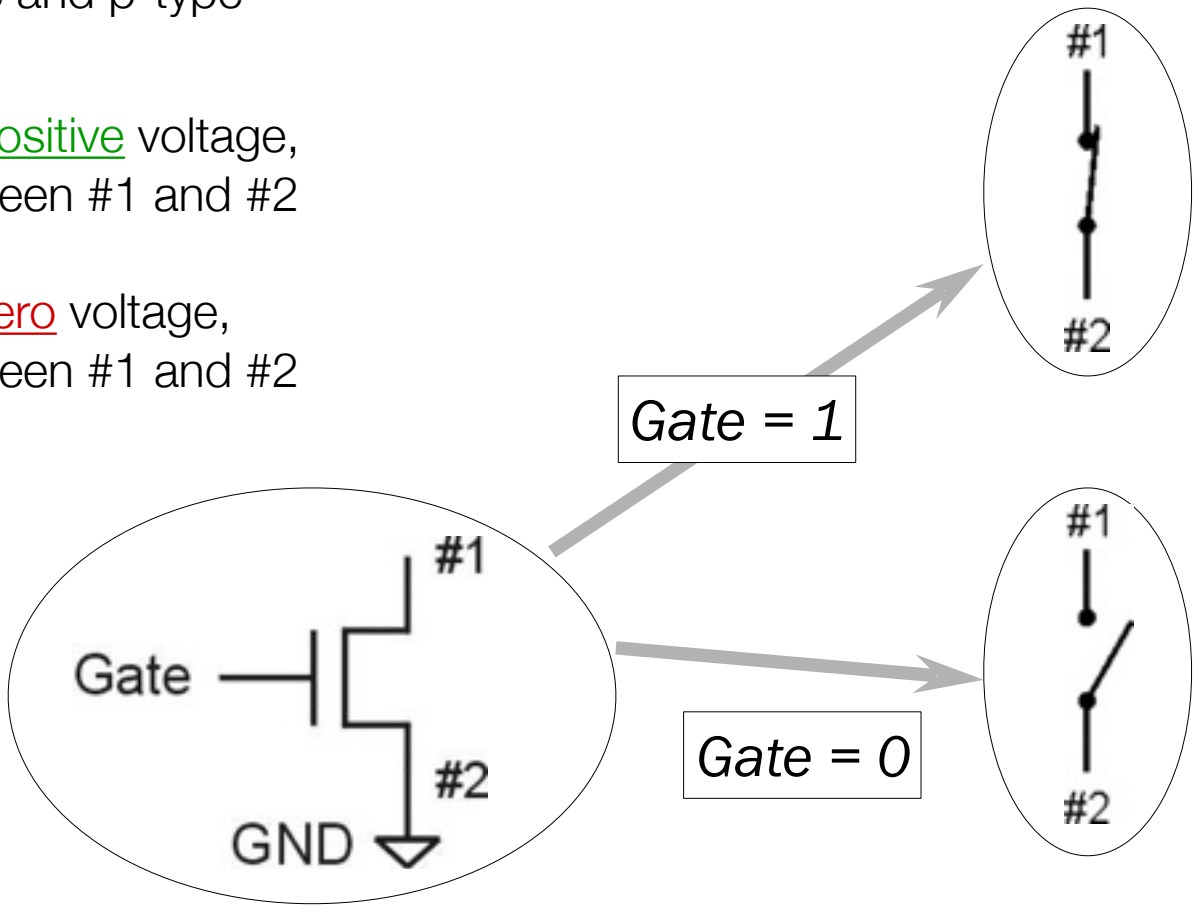
Basic Building Block: Transistors

MOS = Metal Oxide Semiconductor

- two types: n-type and p-type

n-type (NMOS)

- when Gate has positive voltage, short circuit between #1 and #2 (switch closed)
- when Gate has zero voltage, open circuit between #1 and #2 (switch open)

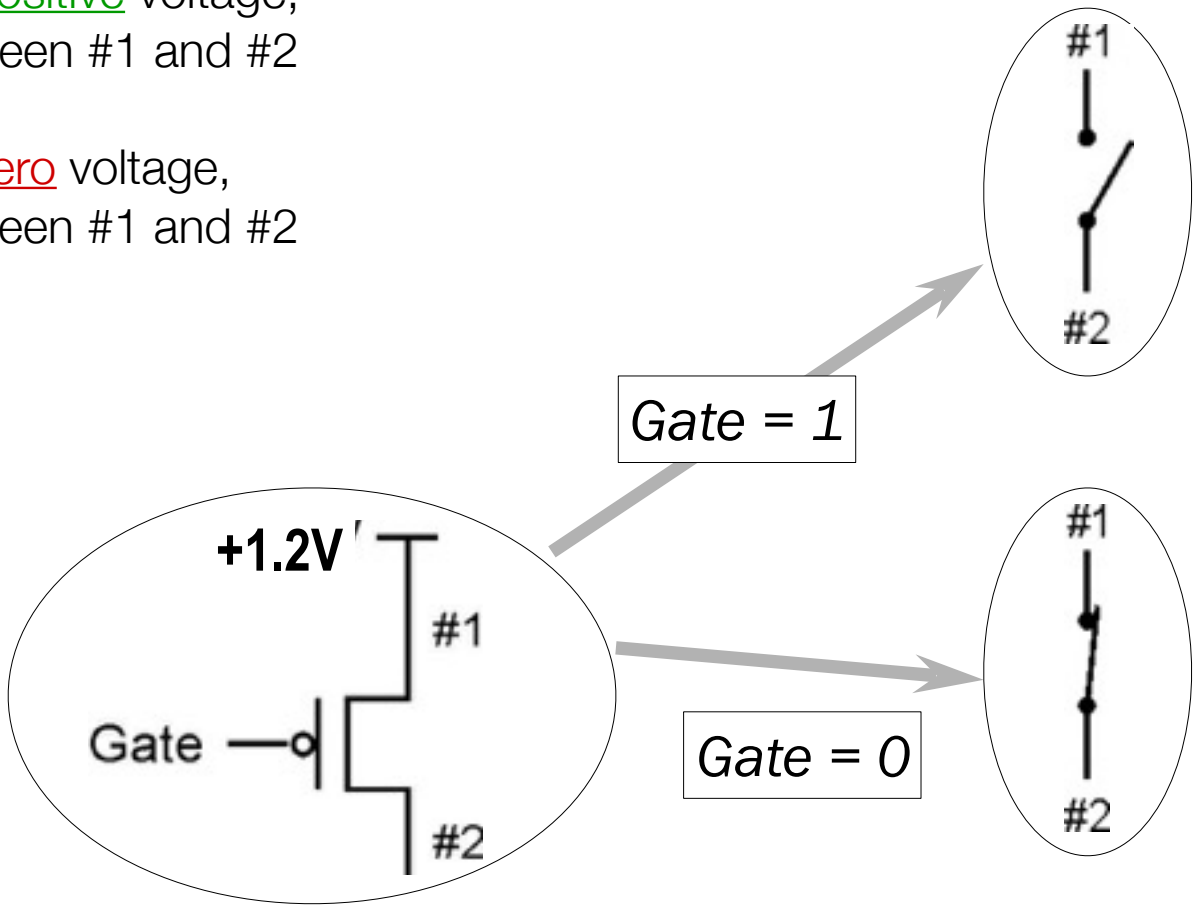


Terminal #2 must be connected to GND (0V).

Basic Building Block: Transistors

p-type is *complementary* to n-type (PMOS)

- when Gate has positive voltage, open circuit between #1 and #2 (switch open)
- when Gate has zero voltage, short circuit between #1 and #2 (switch closed)

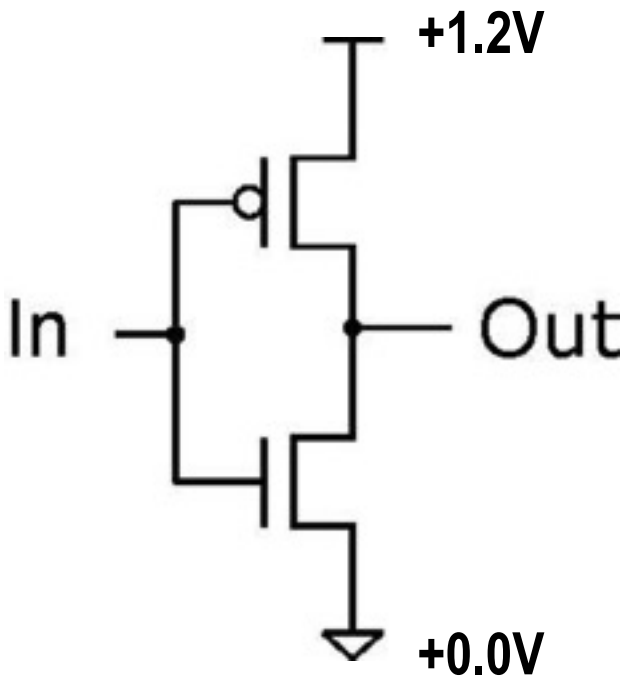


Terminal #1 must be connected to +1.2V

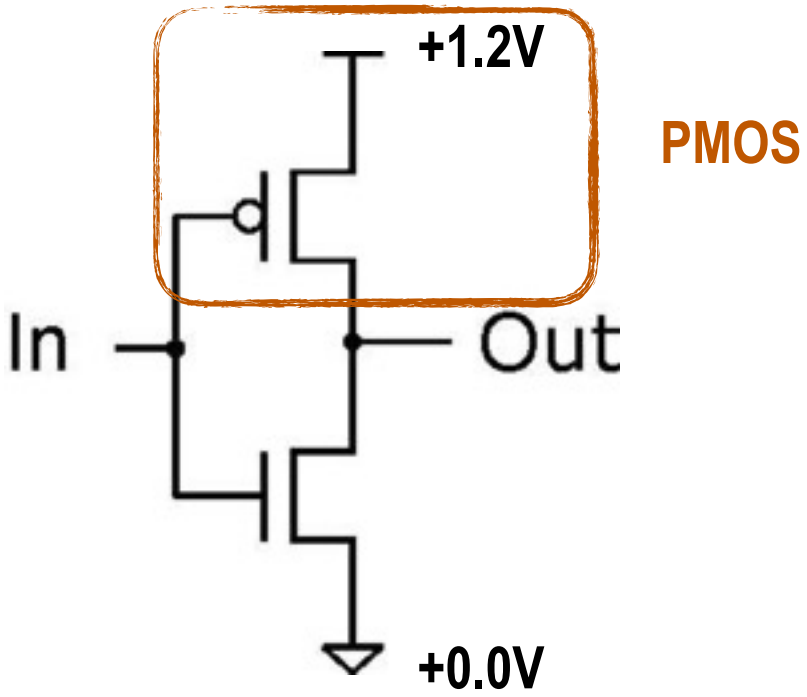
CMOS Circuit

- Complementary MOS
- Uses both n-type and p-type MOS transistors
  - p-type
    - Attached to + voltage
    - Pulls output voltage UP when input is zero
  - n-type
    - Attached to GND
    - Pulls output voltage DOWN when input is one

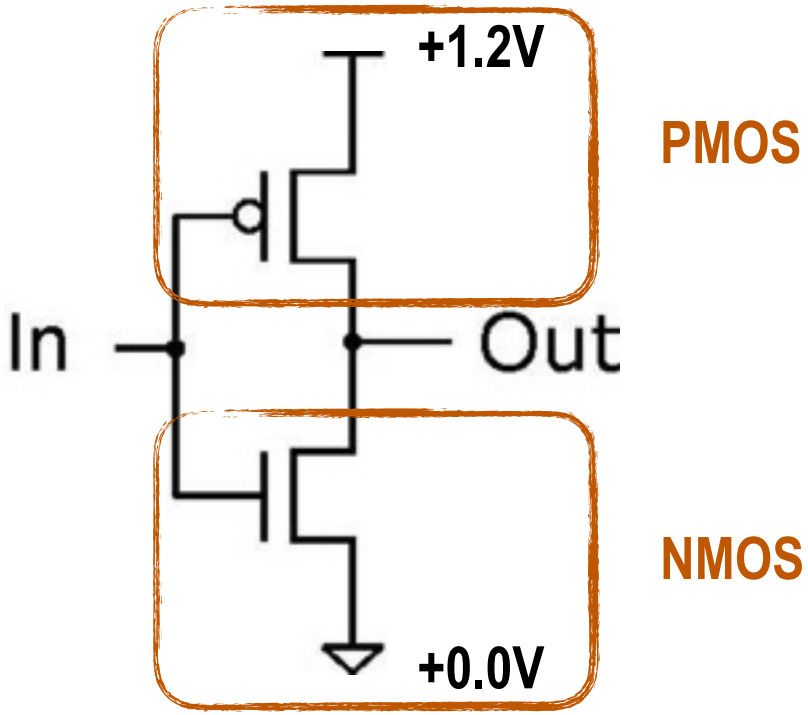
Inverter (NOT Gate)



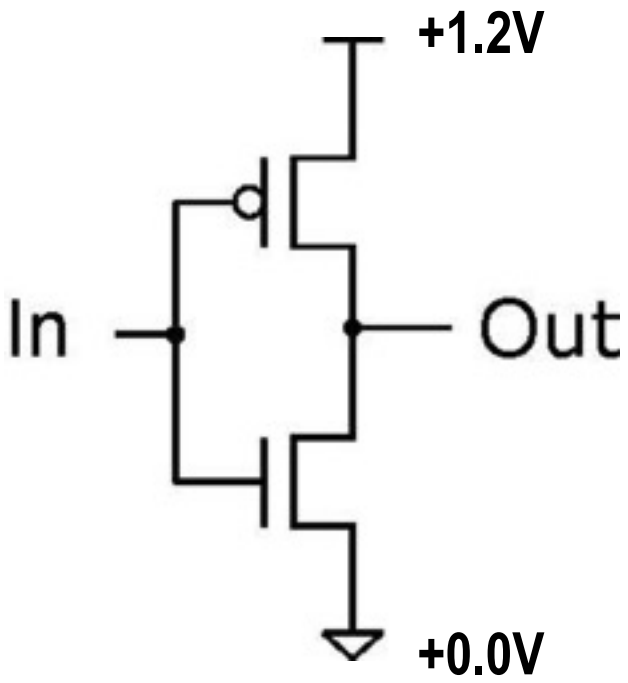
Inverter (NOT Gate)

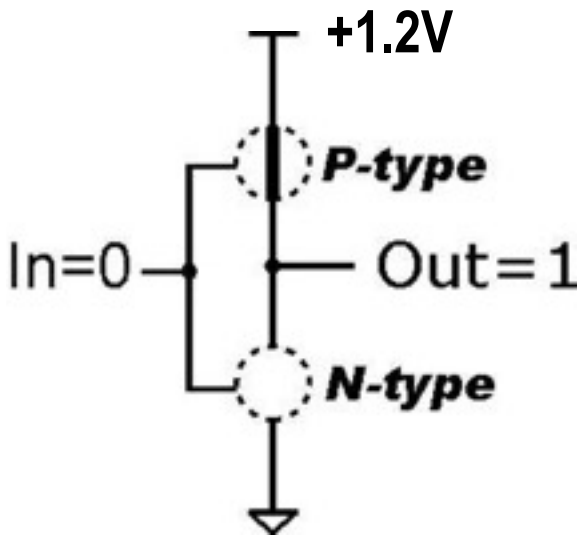


Inverter (NOT Gate)

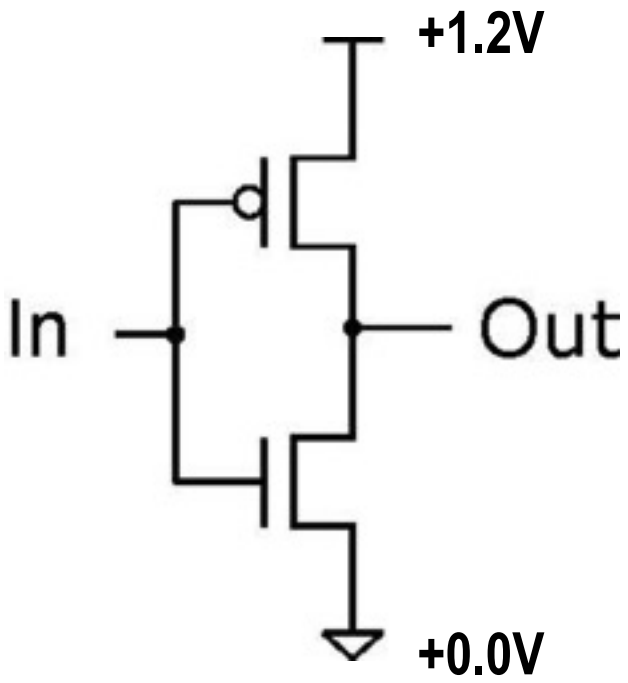


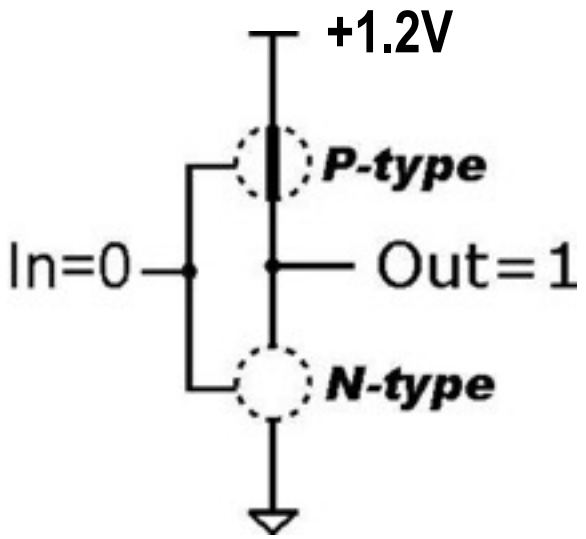
Inverter (NOT Gate)

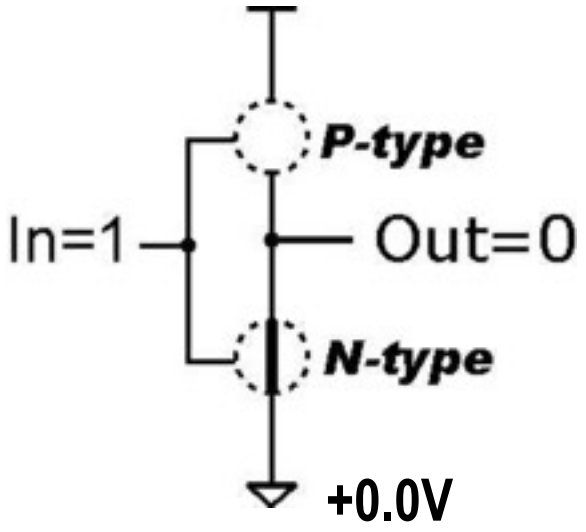




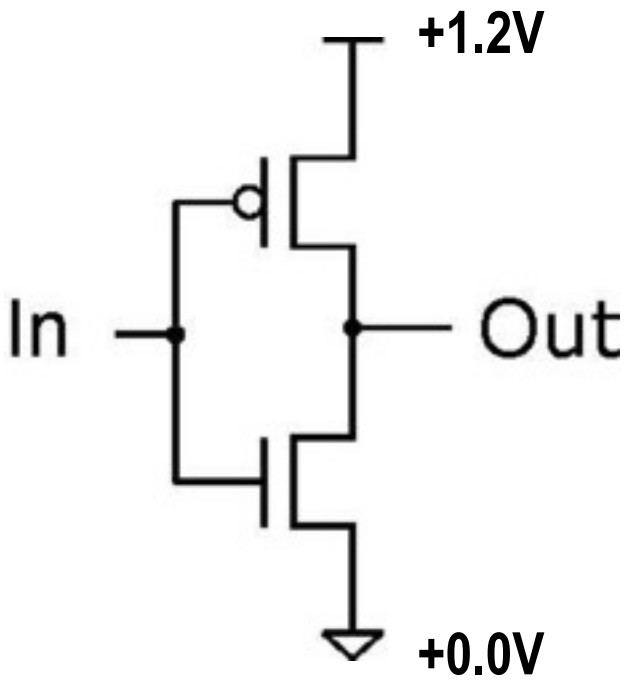
Inverter (NOT Gate)



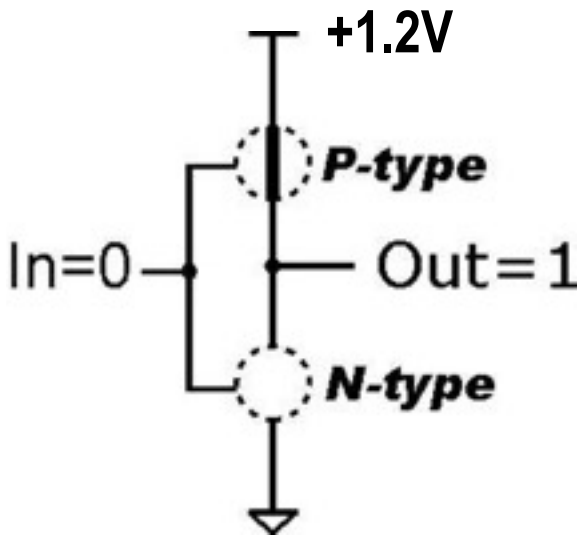


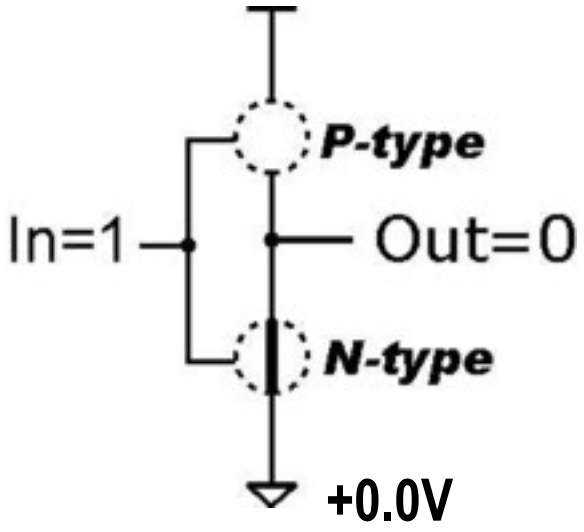


Inverter (NOT Gate)

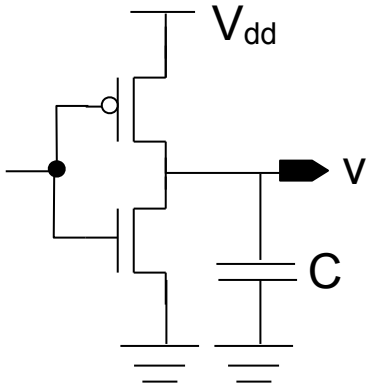


In	Out
0	1
1	0

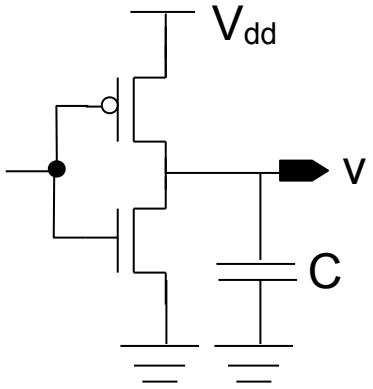


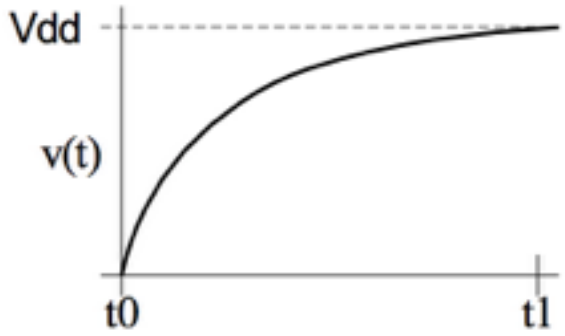


Dynamic Power

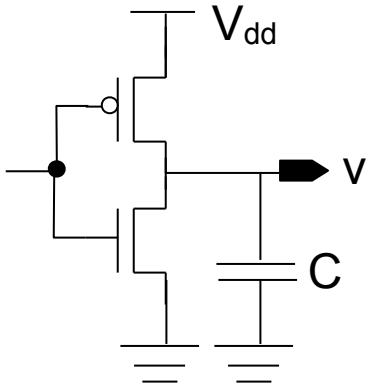


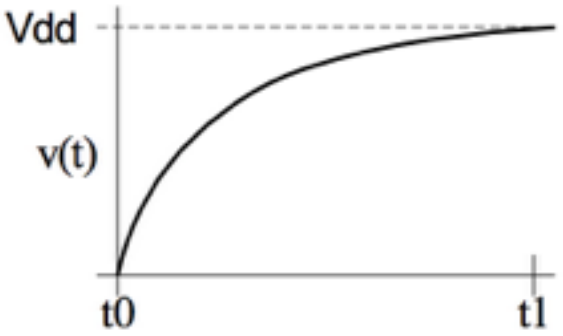
Dynamic Power





Dynamic Power

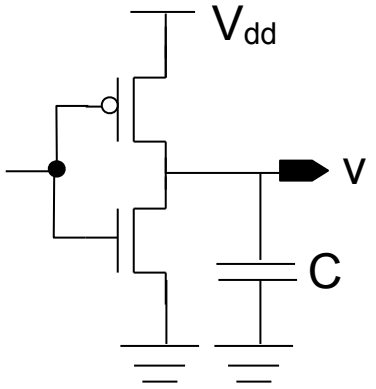


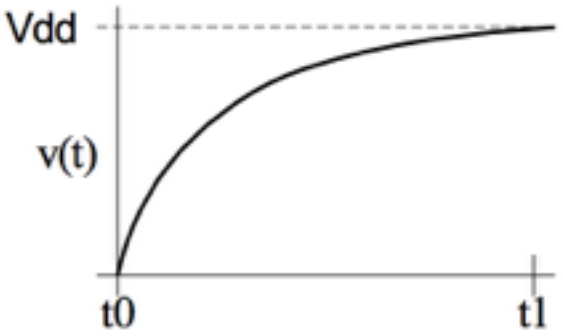


$$E_{sw} = \int_{t_0}^{t_1} P(t)dt = \int_{t_0}^{t_1} (V_{dd} - v) \cdot i(t)dt = \int_{t_0}^{t_1} (V_{dd} - v) \cdot c(dv/dt)\,dt =$$

$$= cV_{dd} \int_{t_0}^{t_1} dv - c\int_{t_0}^{t_1} v \cdot dv = cV_{dd}^{\,2} - 1/2\,cV_{dd}^{\,2} = \boxed{1/2\,cV_{dd}^{\,2}}$$

Dynamic Power



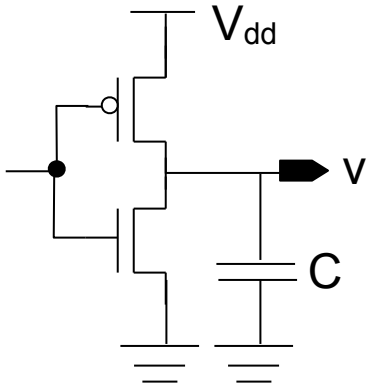


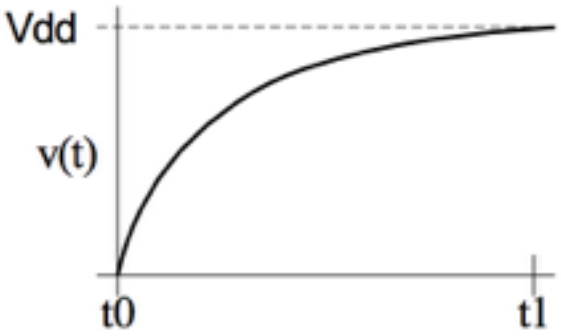
$$E_{sw} = \int_{t_0}^{t_1} P(t)dt = \int_{t_0}^{t_1} (V_{dd} - v) \cdot i(t)dt = \int_{t_0}^{t_1} (V_{dd} - v) \cdot c(dv/dt)\,dt =$$

$$= cV_{dd} \int_{t_0}^{t_1} dv - c\int_{t_0}^{t_1} v \cdot dv = cV_{dd}^{\,2} - 1/2\,cV_{dd}^{\,2} = \boxed{1/2\,cV_{dd}^{\,2}}$$

Energy dissipated for every transition (0->1 or 1->0): ½ C_L V_{dd}^2

Dynamic Power





$$E_{sw} = \int_{t_0}^{t_1} P(t)dt = \int_{t_0}^{t_1} (V_{dd} - v) \cdot i(t)dt = \int_{t_0}^{t_1} (V_{dd} - v) \cdot c(dv/dt)\,dt =$$

$$= cV_{dd} \int_{t_0}^{t_1} dv - c\int_{t_0}^{t_1} v \cdot dv = cV_{dd}^{\,2} - 1/2\,cV_{dd}^{\,2} = \boxed{1/2\,cV_{dd}^{\,2}}$$

Energy dissipated for every transition (0->1 or 1->0): ½ C_L V_{dd}^2

$P = \alpha \cdot (E / T) = \alpha \cdot E\, f = ½\, \alpha\, C\, V_{dd}^2\, f$

α: switch activity factor. No switching, no dynamic power consumption

Dynamic Power

$$P = k\,C\,V^2\,f$$

Dynamic Power

$$P = k\,C\,V^2\,f$$

- Frequency f is proportional to Voltage V

Dynamic Power

$$P = k\,C\,V^2\,f$$

- Frequency f is proportional to Voltage V
  - Intuitively: higher voltage moves electrons faster, so the clock speed can go up also ("overclocking" just increases clock speed without increasing voltage => machine might crash)

Dynamic Power

$$P = k\,C\,V^2\,f$$

- Frequency f is proportional to Voltage V
  - Intuitively: higher voltage moves electrons faster, so the clock speed can go up also ("overclocking" just increases clock speed without increasing voltage => machine might crash)
  - 15% reduction in voltage requires about 15% slow down in frequency

Dynamic Power

$$P = k\,C\,V^2\,f$$

- Frequency f is proportional to Voltage V
  - Intuitively: higher voltage moves electrons faster, so the clock speed can go up also ("overclocking" just increases clock speed without increasing voltage => machine might crash)
  - 15% reduction in voltage requires about 15% slow down in frequency
  - What's the impact on dynamic power? $0.85^3 \approx 60\%$ -> 40% dynamic power reduction.

Multi-core Reduces Dynamic Power

$$P = k\ C\ f^3$$

Multi-core Reduces Dynamic Power

$$P = k\ C\ f^3$$

- Dynamic power favors parallel processing over higher clock rate
  - Take a core and replicate it 4 times: 4x speedup & **4x power**

Multi-core Reduces Dynamic Power

$$P = k\,C\,f^3$$

- Dynamic power favors parallel processing over higher clock rate
  - Take a core and replicate it 4 times: 4x speedup & **4x power**
  - Take a core and clock it 4 times faster: 4x speedup but **64x dynamic power**!

Multi-core Reduces Dynamic Power

$$P = k\,C\,f^3$$

- Dynamic power favors parallel processing over higher clock rate
  - Take a core and replicate it 4 times: 4x speedup & **4x power**
  - Take a core and clock it 4 times faster: 4x speedup but **64x dynamic power**!
- Another way to think about this

Multi-core Reduces Dynamic Power

$$P = k\,C\,f^3$$

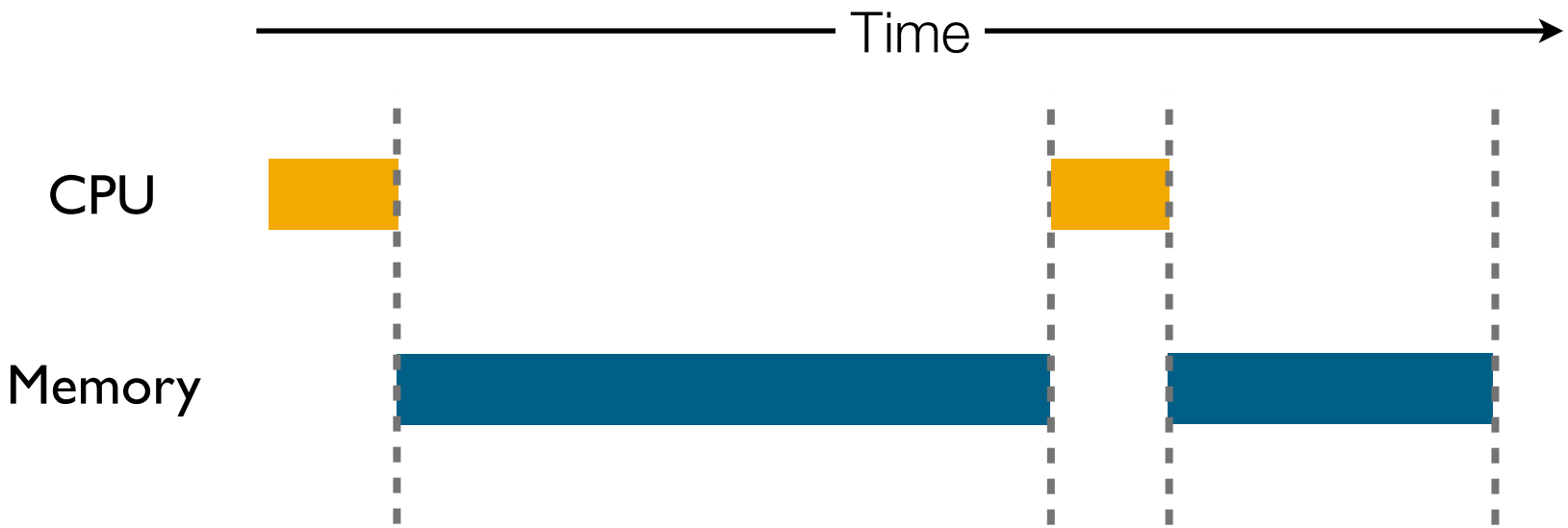
- Dynamic power favors parallel processing over higher clock rate
  - Take a core and replicate it 4 times: 4x speedup & **4x power**
  - Take a core and clock it 4 times faster: 4x speedup but **64x dynamic power**!
- Another way to think about this
  - If a task can be perfectly parallelized by 4 cores, we can reduce the clock frequency of each core to 1/4 while retaining the same performance

Multi-core Reduces Dynamic Power

$$P = k\,C\,f^3$$

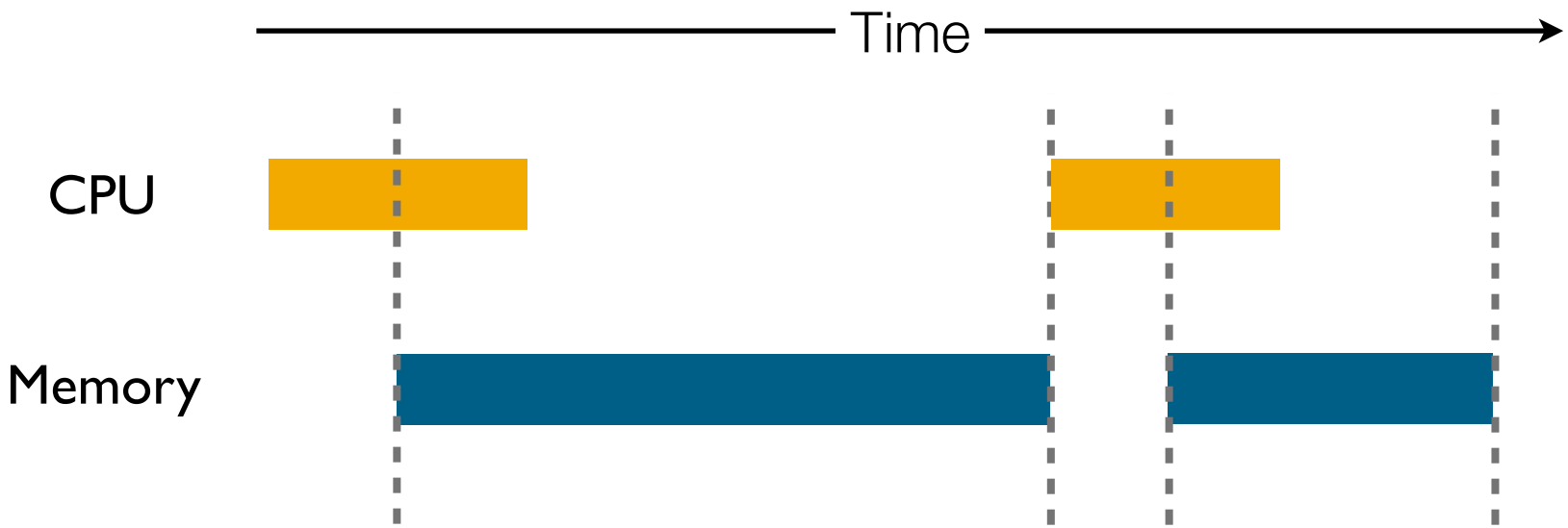
- Dynamic power favors parallel processing over higher clock rate
  - Take a core and replicate it 4 times: 4x speedup & **4x power**
  - Take a core and clock it 4 times faster: 4x speedup but **64x dynamic power**!
- Another way to think about this
  - If a task can be perfectly parallelized by 4 cores, we can reduce the clock frequency of each core to 1/4 while retaining the same performance
  - Dynamic power becomes 4 x $(1/4)^3$ = 1/16

Dynamic Voltage and Frequency Scaling



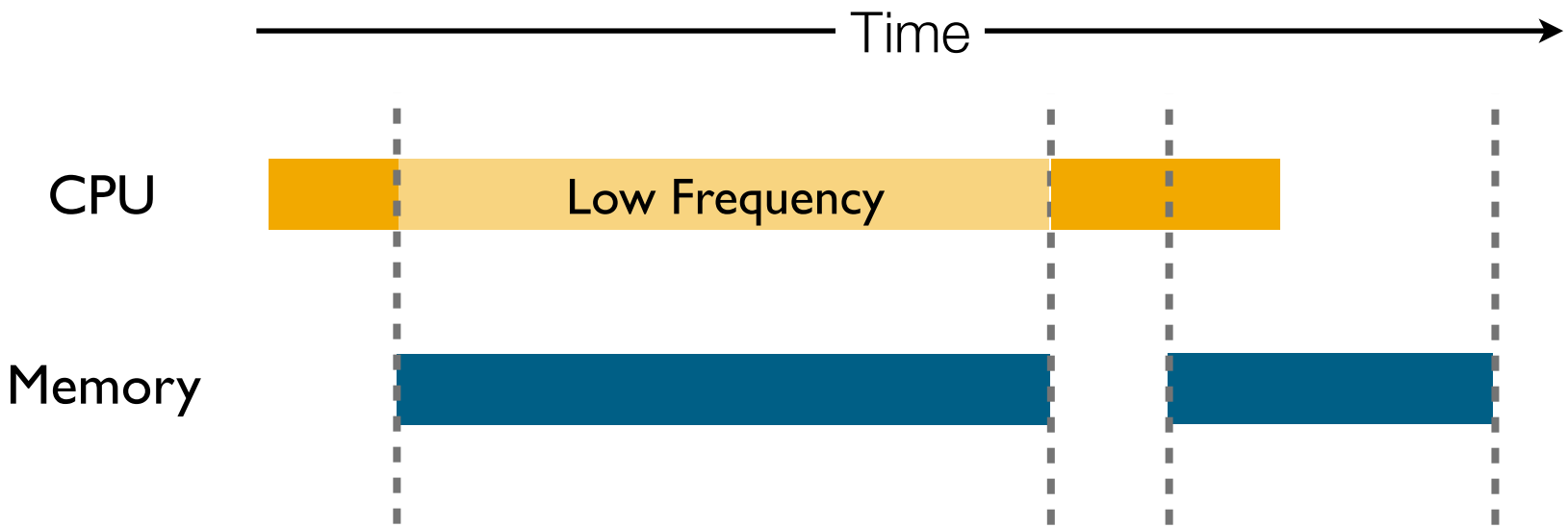
https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

Dynamic Voltage and Frequency Scaling



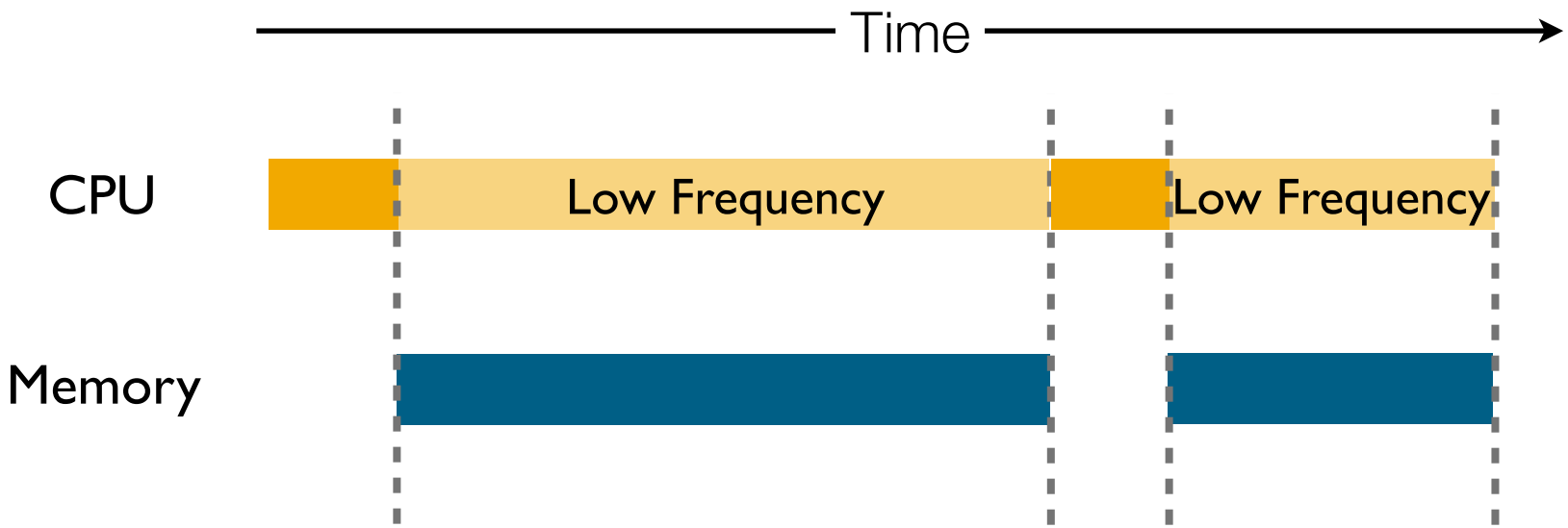
https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

Dynamic Voltage and Frequency Scaling



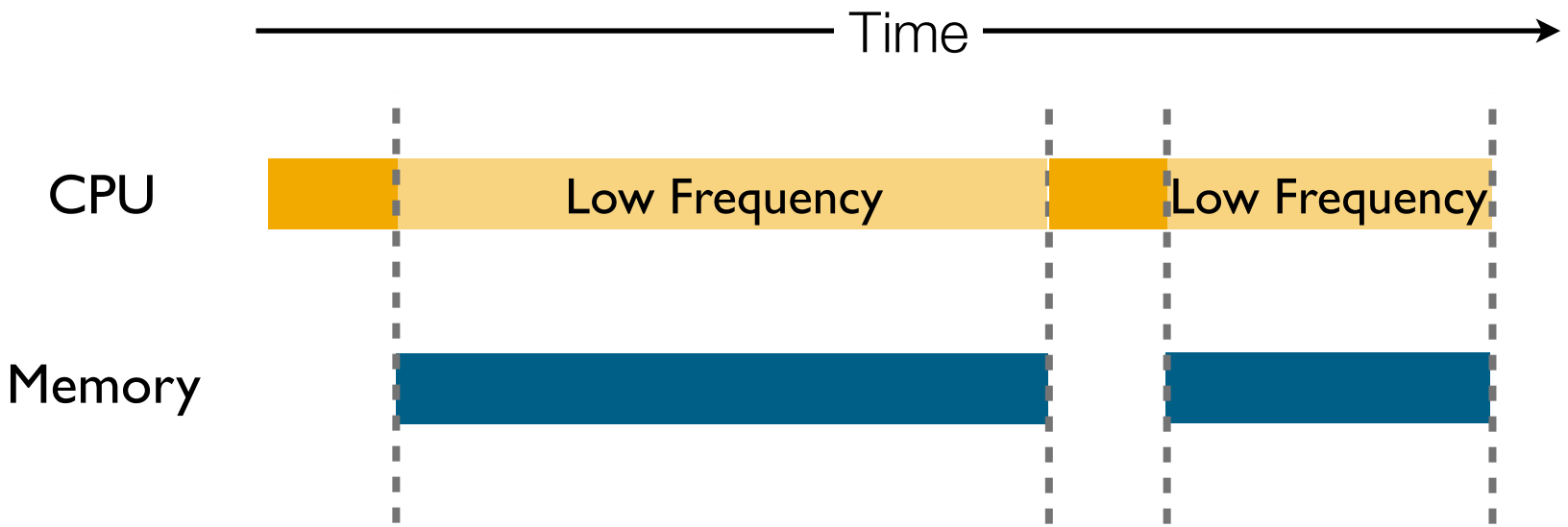
https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

Dynamic Voltage and Frequency Scaling



https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

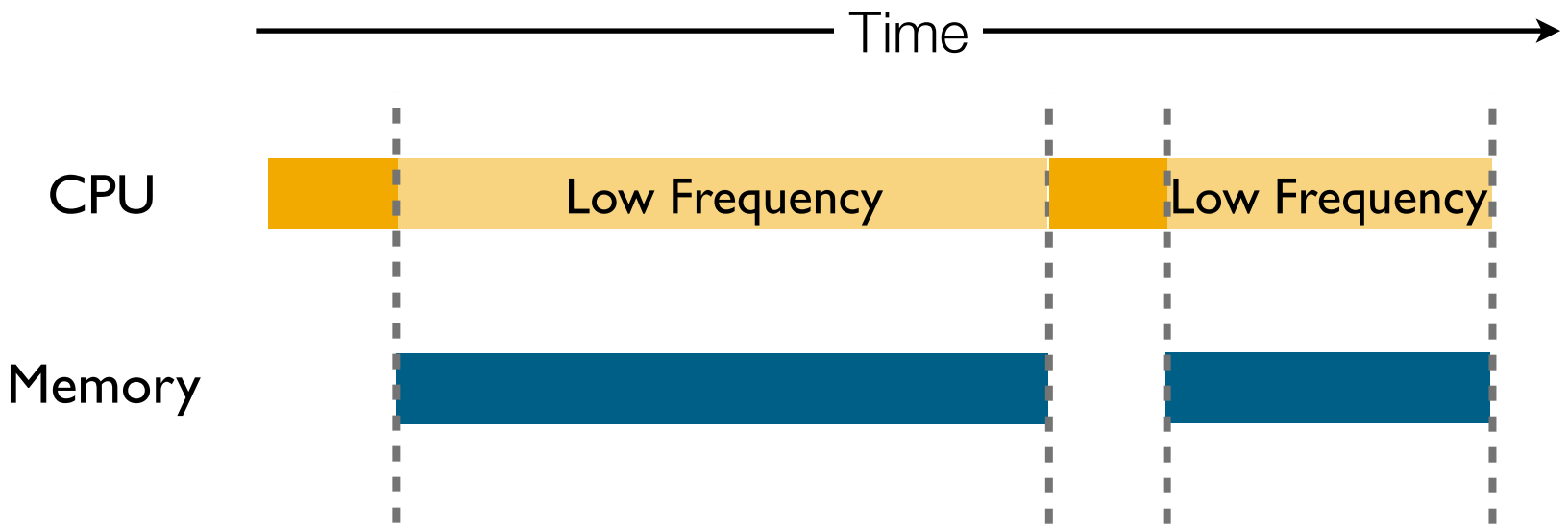
Dynamic Voltage and Frequency Scaling



- DVFS: dynamic scales voltage and frequency depending on how much slack there is to reduce power with little performance impact

https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

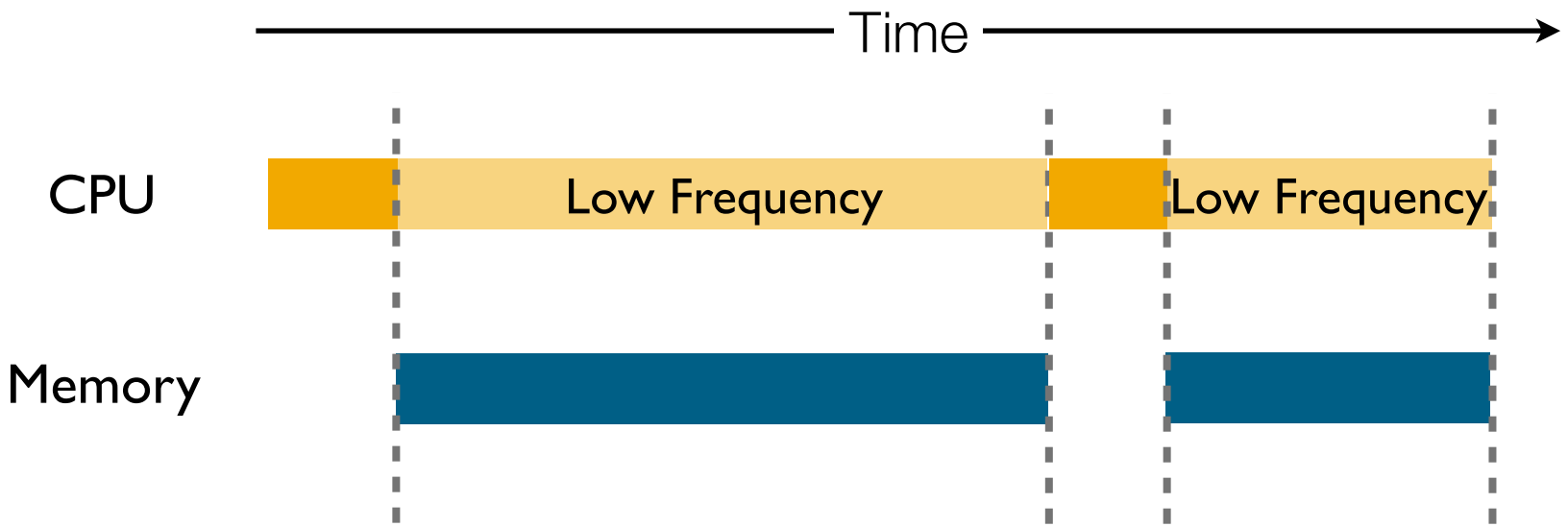
Dynamic Voltage and Frequency Scaling



- DVFS: dynamic scales voltage and frequency depending on how much slack there is to reduce power with little performance impact
- Mostly done in OS today as the “CPU frequency governor”: based on CPU utilization, coarse-grained (e.g., ~10 ms)

https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

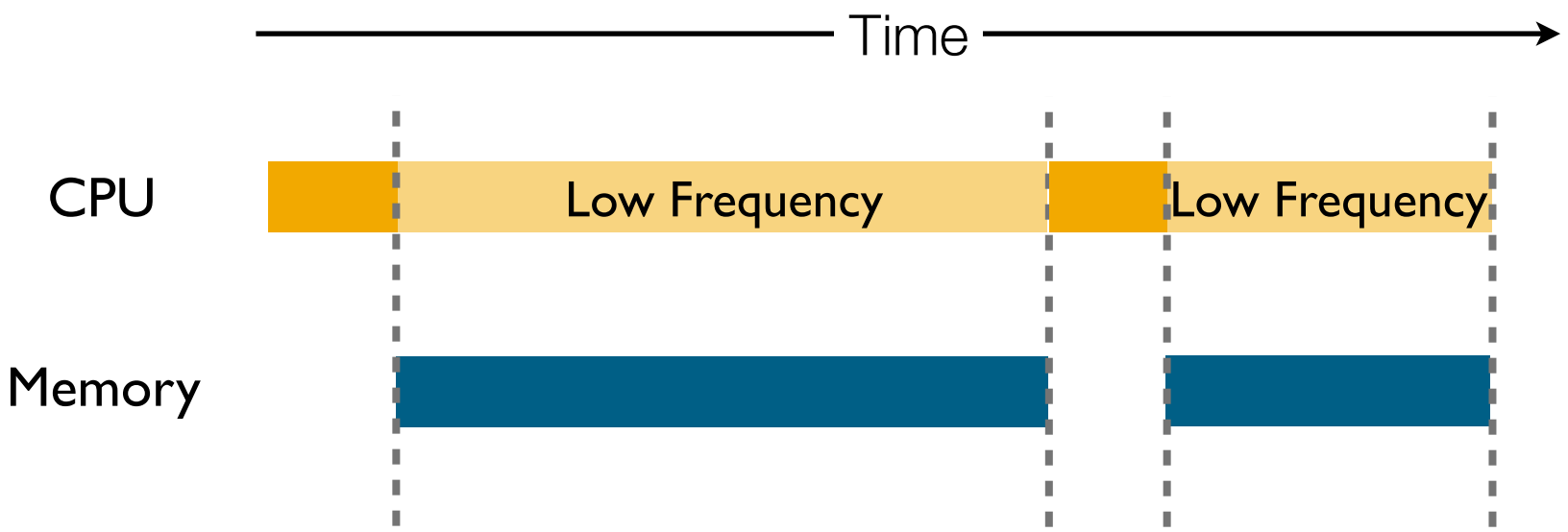
Dynamic Voltage and Frequency Scaling



- DVFS: dynamic scales voltage and frequency depending on how much slack there is to reduce power with little performance impact
- Mostly done in OS today as the “CPU frequency governor”: based on CPU utilization, coarse-grained (e.g., ~10 ms)
- You can control it. Try it yourself.

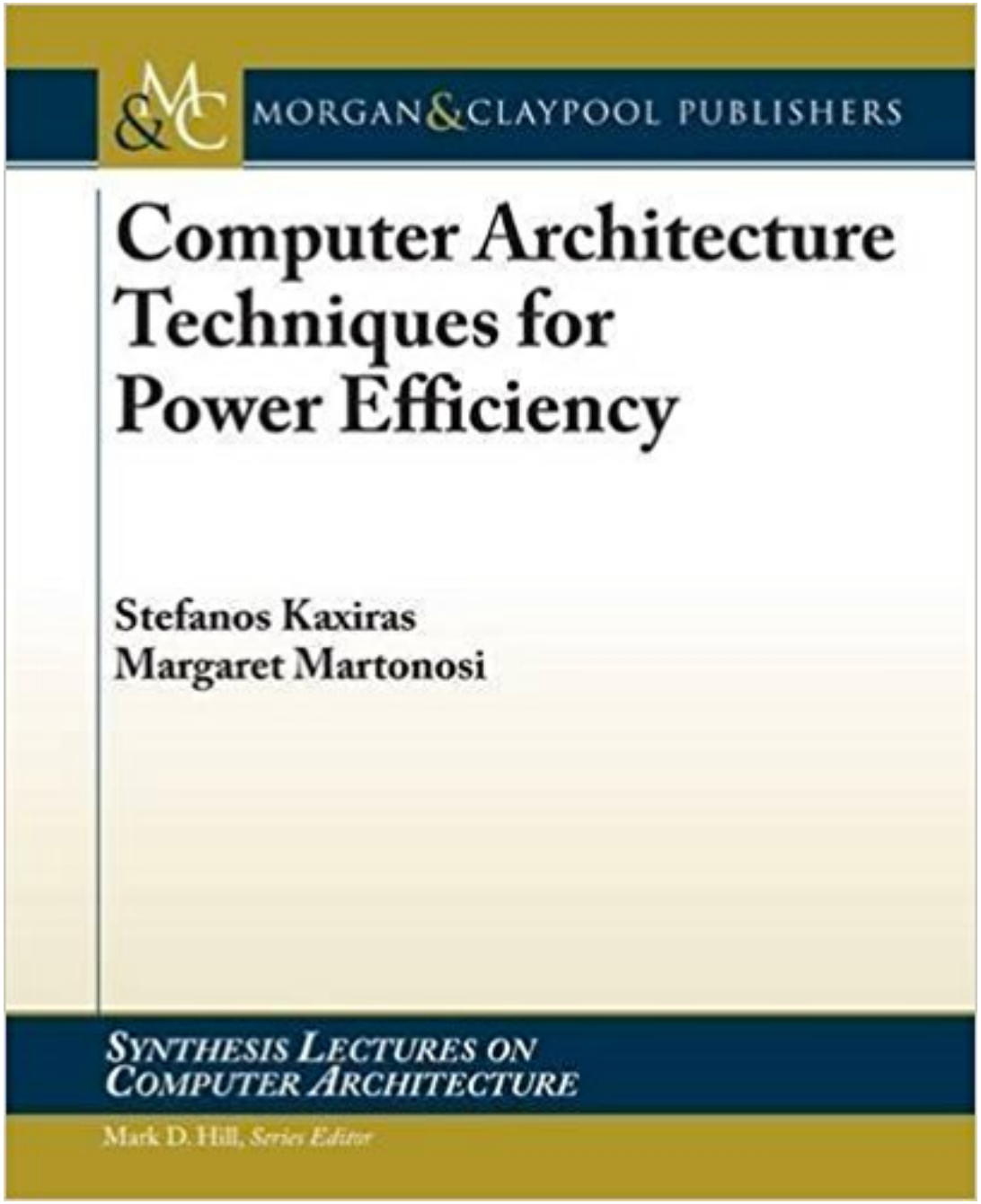
https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

Dynamic Voltage and Frequency Scaling



- DVFS: dynamic scales voltage and frequency depending on how much slack there is to reduce power with little performance impact
- Mostly done in OS today as the “CPU frequency governor”: based on CPU utilization, coarse-grained (e.g., ~10 ms)
- You can control it. Try it yourself.
- Hardware can do it in finer granularity, but more complex circuits

https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

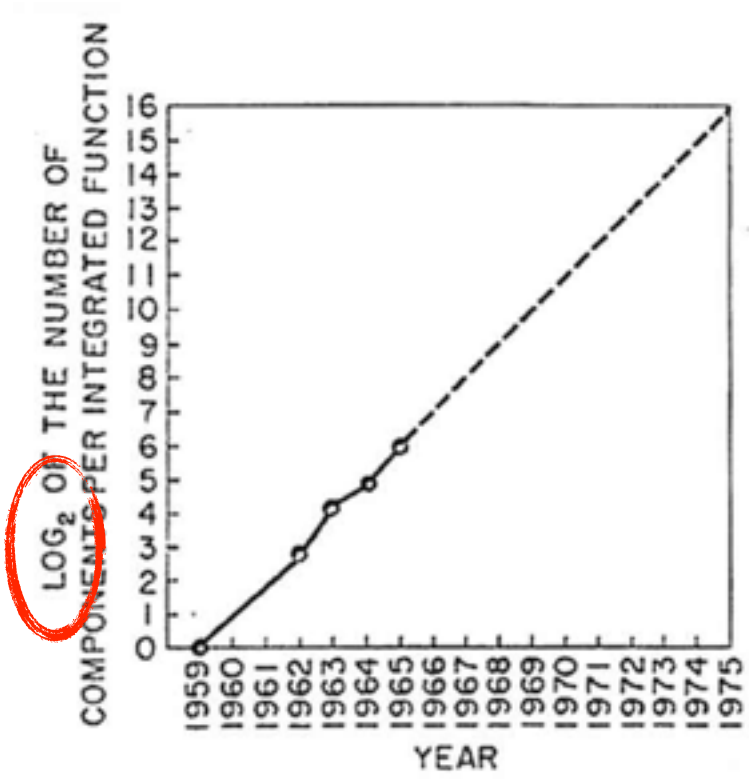

MORGAN & CLAYPOOL PUBLISHERS
Computer Architecture Techniques for Power Efficiency
Stefanos Kaxiras
Margaret Martonosi
SYNTHESIS LECTURES ON COMPUTER ARCHITECTURE
Mark D. Hill, Series Editor

Moore’s Law

- Gordon Moore in 1965 predicted that the number of transistors doubles every year

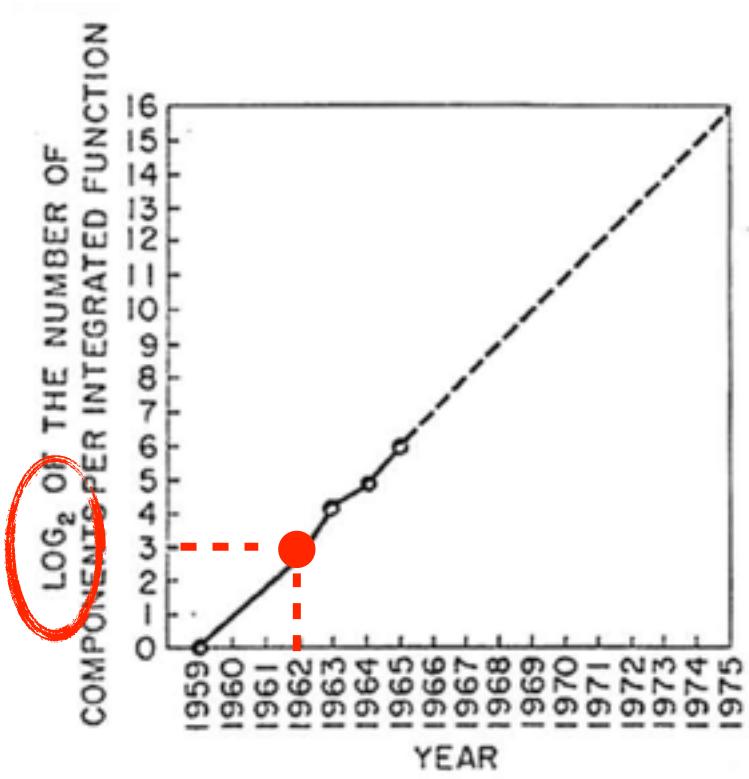
Moore's Law

- Gordon Moore in 1965 predicted that the number of transistors doubles every year



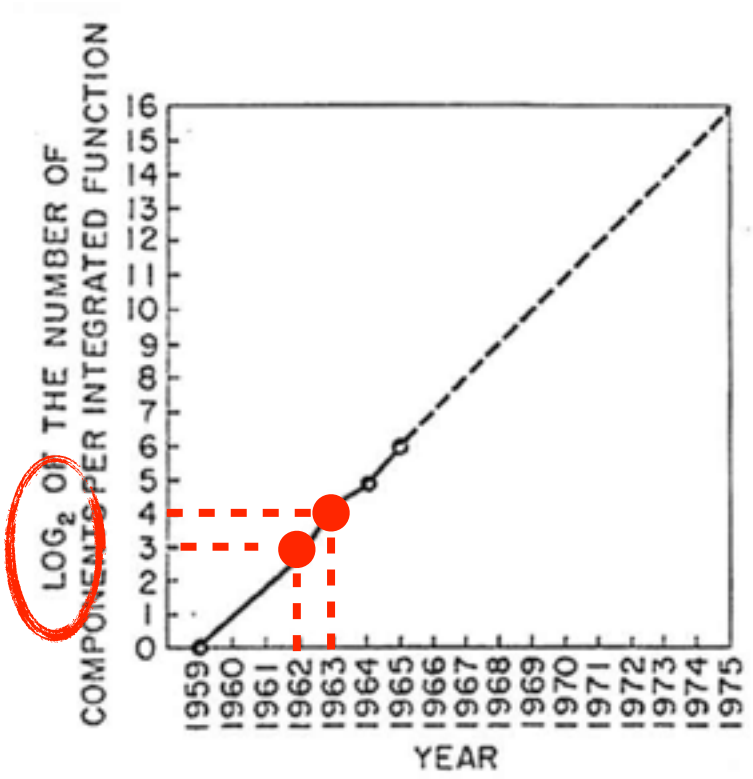
Moore's Law

- Gordon Moore in 1965 predicted that the number of transistors doubles every year



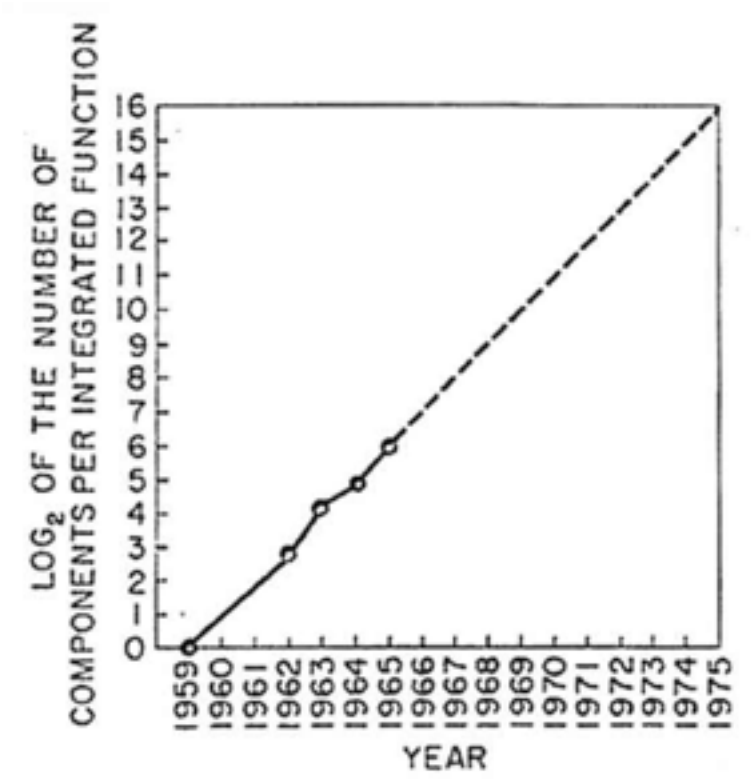
Moore's Law

- Gordon Moore in 1965 predicted that the number of transistors doubles every year



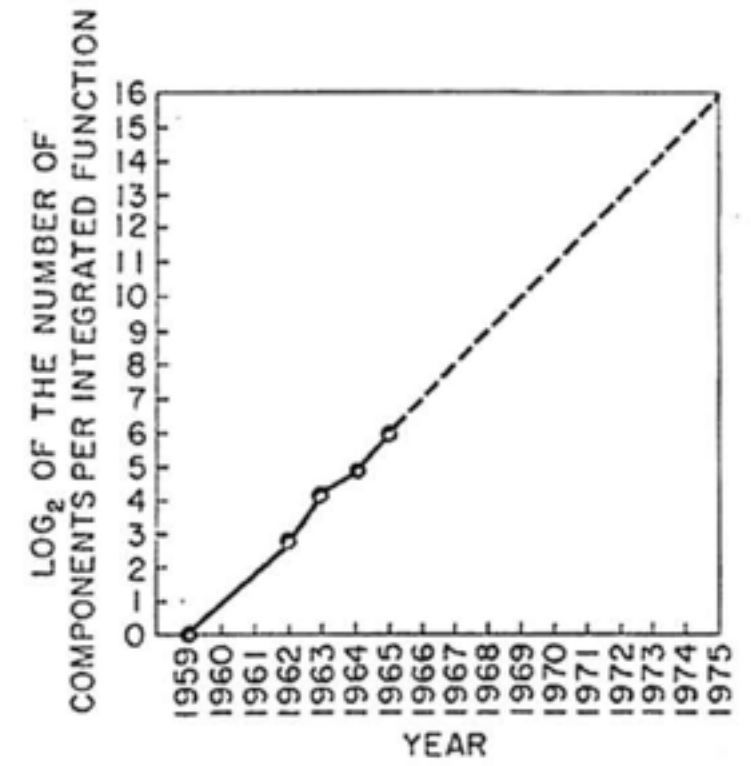
Moore's Law

- Gordon Moore in 1965 predicted that the number of transistors doubles every year
- In 1975 he revised the prediction to doubling every 2 years

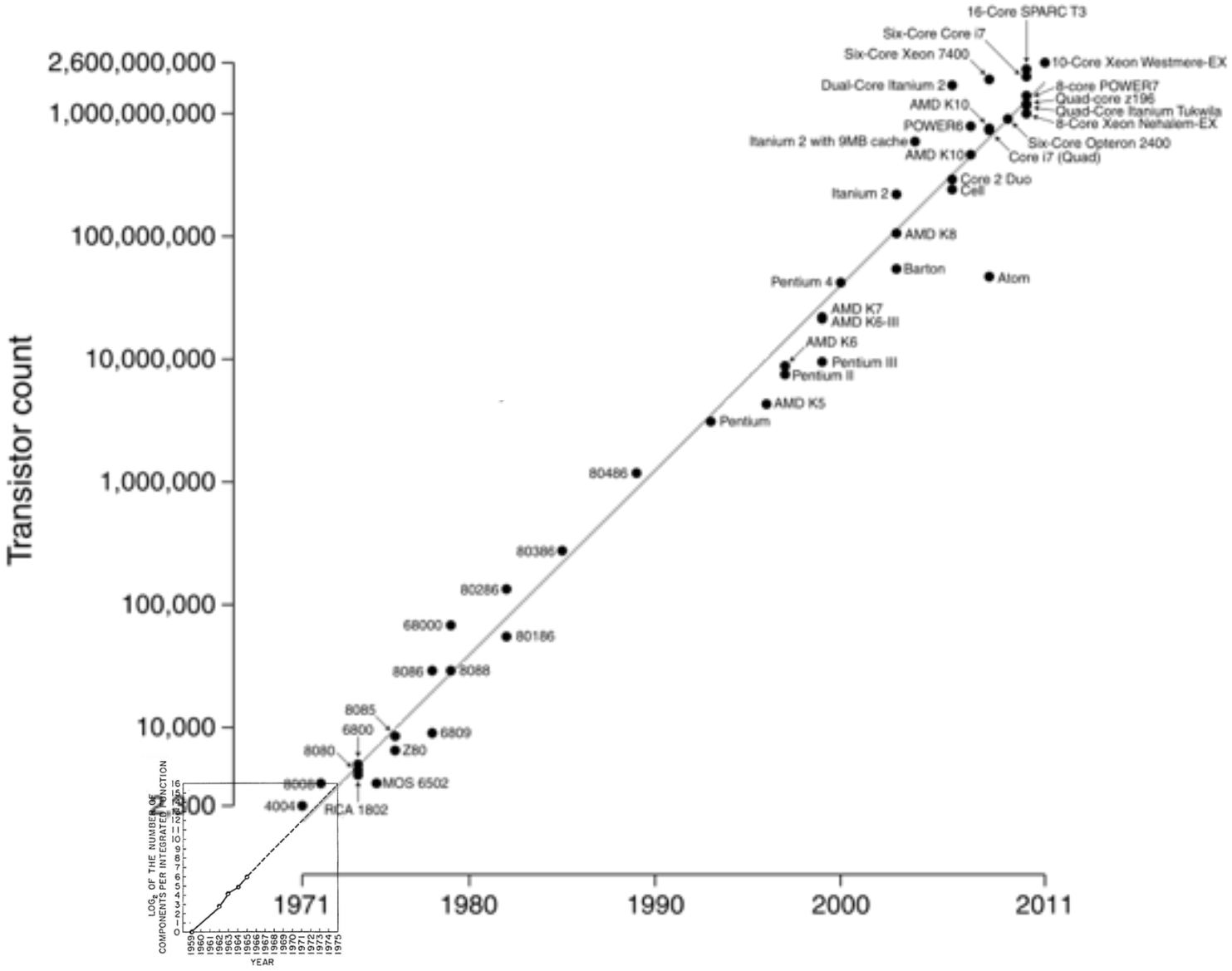


Moore's Law

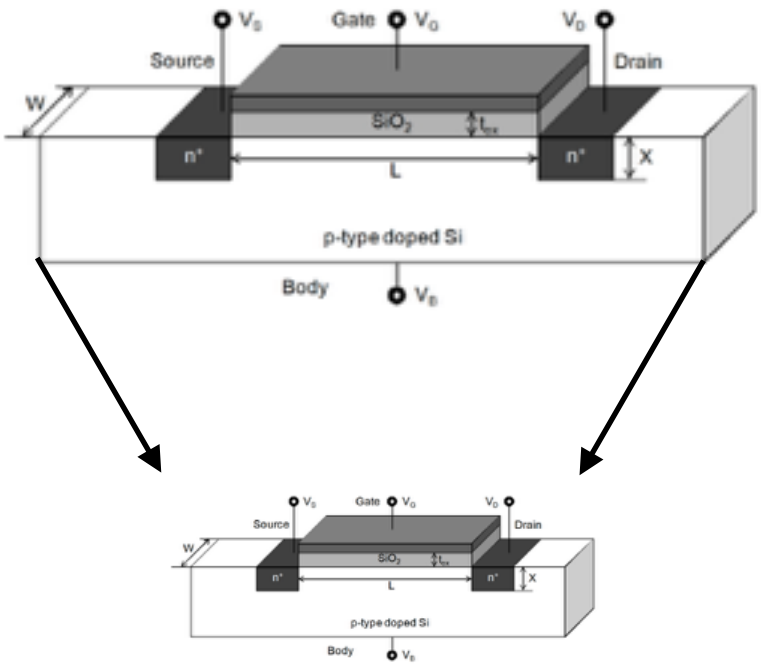
- Gordon Moore in 1965 predicted that the number of transistors doubles every year
- In 1975 he revised the prediction to doubling every 2 years
- Today's widely-known Moore's Law: number of transistors double about every 18 months (Moore never used the number 18…)



Moore's Law


Transistor count
2,600,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
1971
1980
1990
2000
2011
16-Core SPARC T3
Six-Core Core i7
Six-Core Xeon 7400
10-Core Xeon Westmere-EX
Dual-Core Itanium 2
8-core POWER7
Quad-core z196
AMD K10
Quad-Core Itanium Tukwila
POWER6
8-Core Xeon Nehalem-EX
Itanium 2 with 9MB cache
Six-Core Opteron 2400
AMD K10
Core i7 (Quad)
Core 2 Duo
Itanium 2
Cell
AMD K8
Barton
Atom
Pentium 4
AMD K7
AMD K6-III
AMD K6
Pentium III
Pentium II
AMD K5
Pentium
80486
80386
80286
68000
80186
8086
8088
8085
6800
6809
8080
Z80
8008
MOS 6502
4004
RCA 1802
LOG2 OF THE NUMBER OF COMPONENTS PER INTEGRATED FUNCTION
YEAR

Dennard Scaling

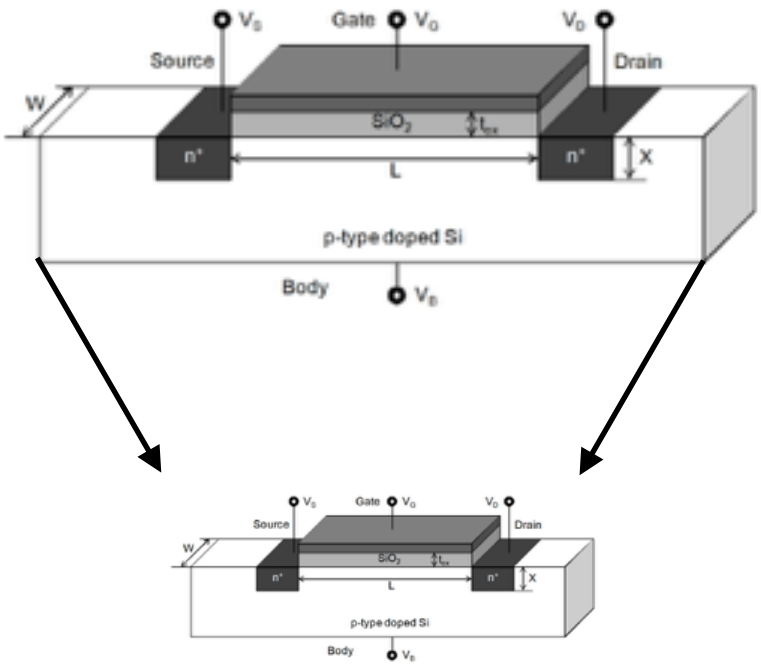


Scale factor α<1

$\alpha = 0.7$ => 2X more transistors!

Design of Ion-Implanted MOSFET's with Very Small Physical Dimensions. Proc. of IEEE. Dennart et al.

Dennard Scaling



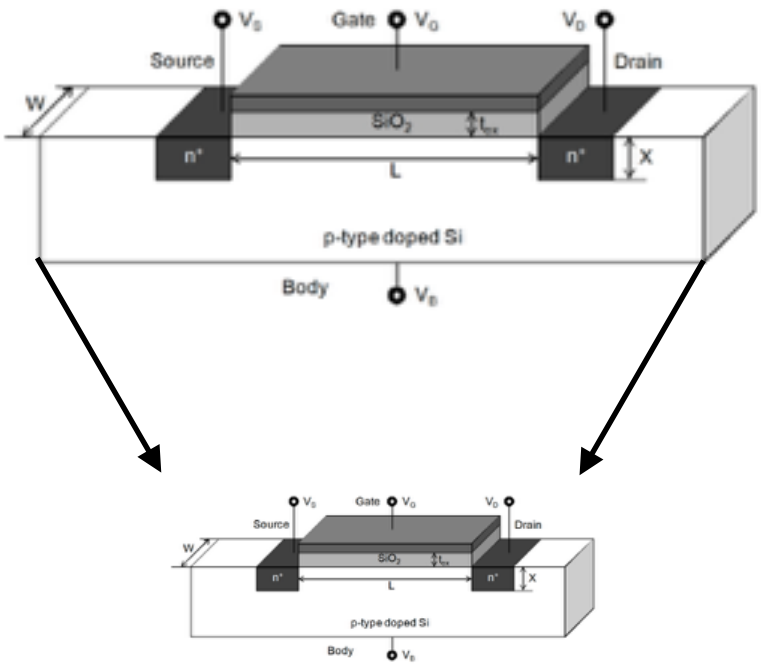
Parameter	Value	Scaled Value
Dopant concentrations	Na, Nd	Na/α, Nd/α
Dimensions	L, W, Tox	αL, αW, αTox
Field	E	E
Voltage	V	αV
Capacitance	C	αC
Current	I	αI

Scale factor α<1

$\alpha = 0.7$ => 2X more transistors!

Design of Ion-Implanted MOSFET's with Very Small Physical Dimensions. Proc. of IEEE. Dennart et al.

Dennard Scaling



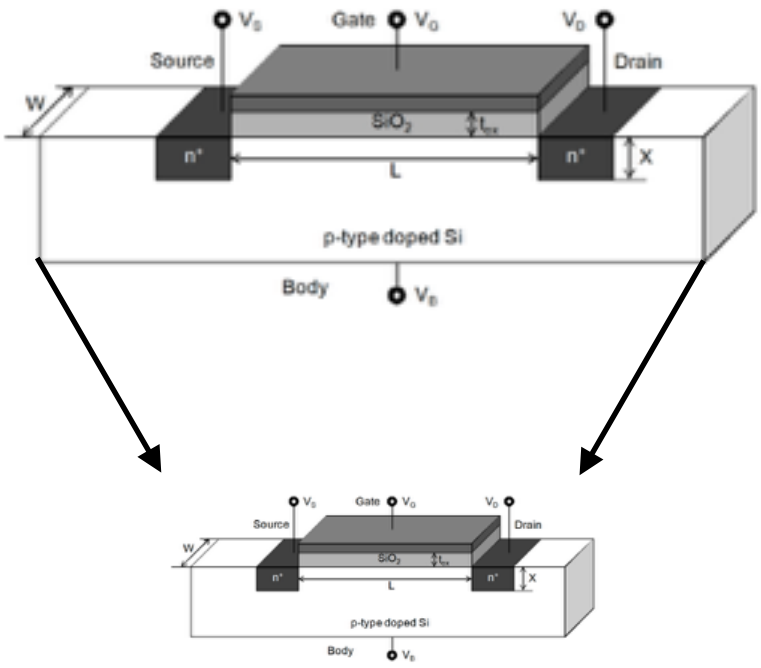
Parameter	Value	Scaled Value
Dopant concentrations	Na, Nd	Na/α, Nd/α
Dimensions	L, W, Tox	αL, αW, αTox
Field	E	E
Voltage	V	αV
Capacitance	C	αC
Current	I	αI
Transistors/Area	d	d/α^2

Scale factor α<1

$\alpha = 0.7$ => 2X more transistors!

Design of Ion-Implanted MOSFET's with Very Small Physical Dimensions. Proc. of IEEE. Dennart et al.

Dennard Scaling



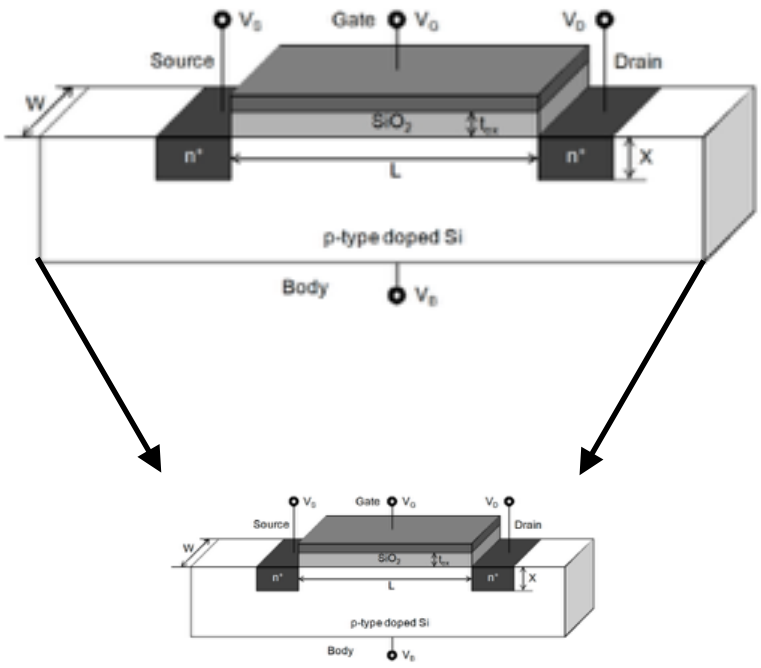
Parameter	Value	Scaled Value
Dopant concentrations	Na, Nd	Na/α, Nd/α
Dimensions	L, W, Tox	αL, αW, αTox
Field	E	E
Voltage	V	αV
Capacitance	C	αC
Current	I	αI
Transistors/Area	d	d/α^2
Propagation time (~CV/I)	t	αt
Frequency (1/t)	f	f/α

Scale factor α<1

$\alpha = 0.7$ => 2X more transistors!

Design of Ion-Implanted MOSFET's with Very Small Physical Dimensions. Proc. of IEEE. Dennart et al.

Dennard Scaling



Parameter	Value	Scaled Value
Dopant concentrations	Na, Nd	Na/α, Nd/α
Dimensions	L, W, Tox	αL, αW, αTox
Field	E	E
Voltage	V	αV
Capacitance	C	αC
Current	I	αI
Transistors/Area	d	d/α^2
Propagation time (~CV/I)	t	αt
Frequency (1/t)	f	f/α
Power (CV^2f)	P	$\alpha^2 P$
Power/area (Power density)	**Pd**	**pd**

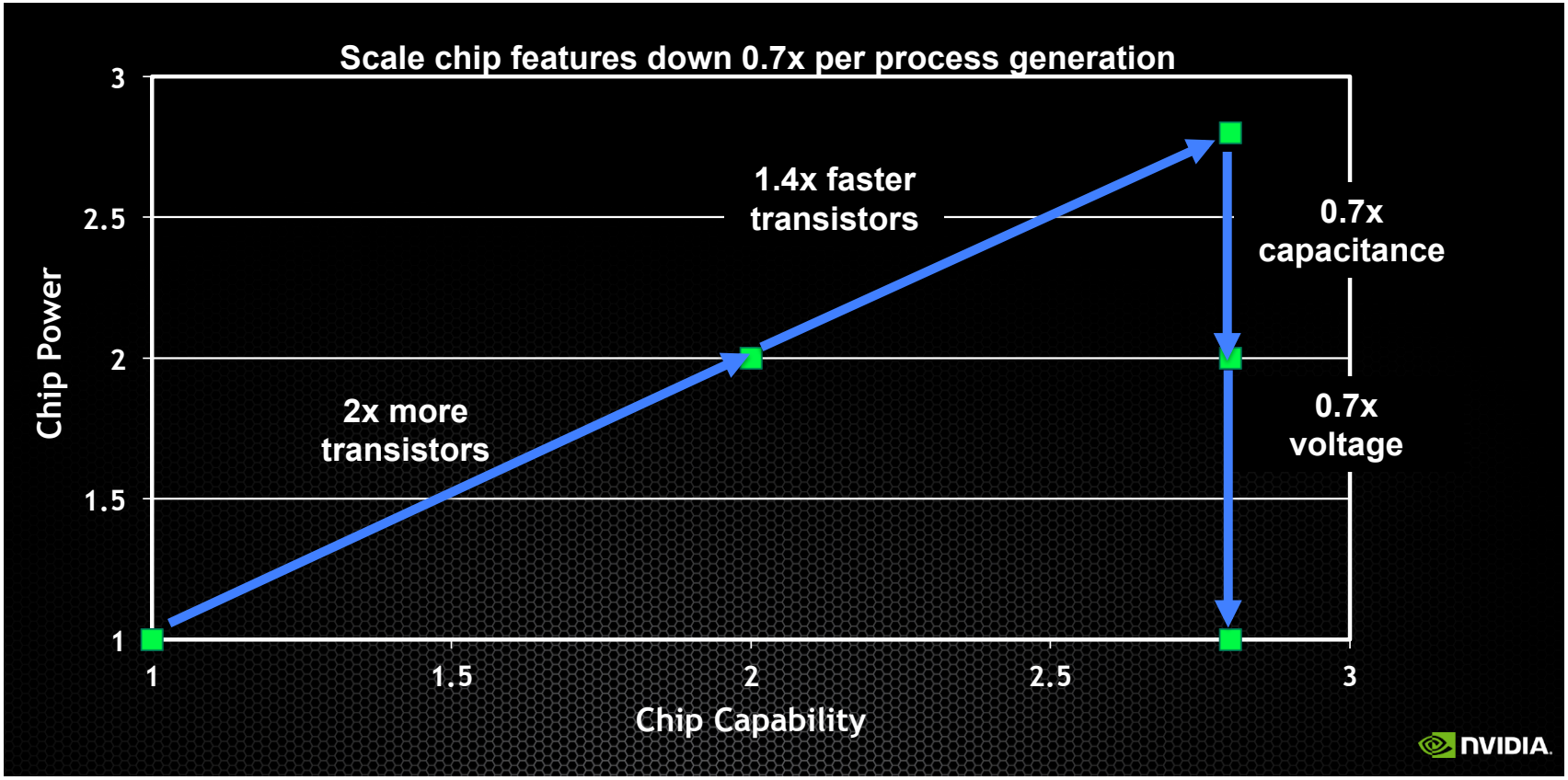
Scale factor α<1

$\alpha = 0.7$ => 2X more transistors!

Design of Ion-Implanted MOSFET's with Very Small Physical Dimensions. Proc. of IEEE. Dennart et al.

Dennard Scaling

- Every generation: 2.8X chip capability in same power



Challenges for future computing systems, Bill Dally, 2015

Implications of Dennard Scaling

Implications of Dennard Scaling

- Each new processor generation can have more transistors (Moore's Law), and will run at higher frequency but won't consume more power given the same area budget

Implications of Dennard Scaling

- Each new processor generation can have more transistors (Moore's Law), and will run at higher frequency but won't consume more power given the same area budget
- More transistors means better microarchitecture, which leads to better performance even under the same frequency

Implications of Dennard Scaling

- Each new processor generation can have more transistors (Moore's Law), and will run at higher frequency but won't consume more power given the same area budget
- More transistors means better microarchitecture, which leads to better performance even under the same frequency
- Higher frequency means better performance even under the same microarchitecture

Implications of Dennard Scaling

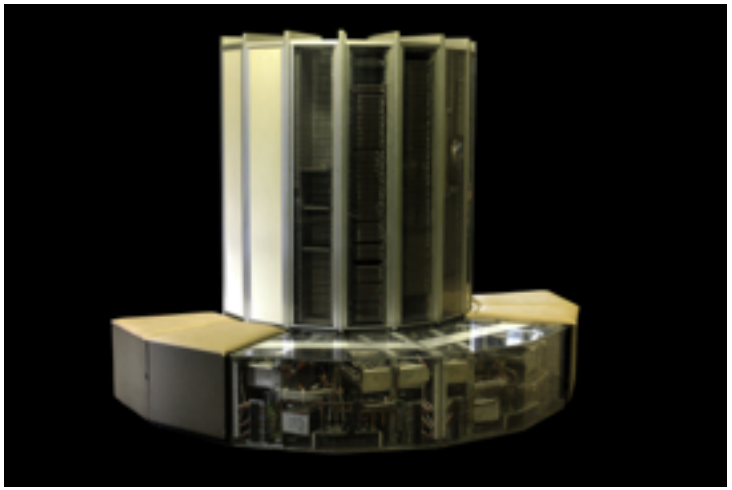
- Each new processor generation can have more transistors (Moore's Law), and will run at higher frequency but won't consume more power given the same area budget
- More transistors means better microarchitecture, which leads to better performance even under the same frequency
- Higher frequency means better performance even under the same microarchitecture
- Overall, software gets a free ride: wait for the next generation of hardware and performance will naturally increase without consuming more power

Implications of Dennard Scaling

- Each new processor generation can have more transistors (Moore's Law), and will run at higher frequency but won't consume more power given the same area budget
- More transistors means better microarchitecture, which leads to better performance even under the same frequency
- Higher frequency means better performance even under the same microarchitecture
- Overall, software gets a free ride: wait for the next generation of hardware and performance will naturally increase without consuming more power

> Moore's law gave us more transistors;
>
> Dennard scaling made them useful.
>
> Bob Colwell, DAC 2013, June 4, 2013

Moore's Law + Dennard Scaling Have Meant



- 1976 Cray 1
  - Frequency: 80 MHz
  - Speed: 250 M Ops/second
  - Scale: 2.5 Million transistors
  - 5,000 kg, 115 KW
  - Price: $9M
  - 80 manufactured

- 2014 iPhone 6
  - Frequency: 1.4 GHz
  - Speed: > 4 B Ops/second
  - Scale: > 3 Billion transistors
  - 120 g, < 5 W
  - Price: $649
  - 10 million sold in first 3 days

The CRAY-1 computer system, Russell, CACM 1978

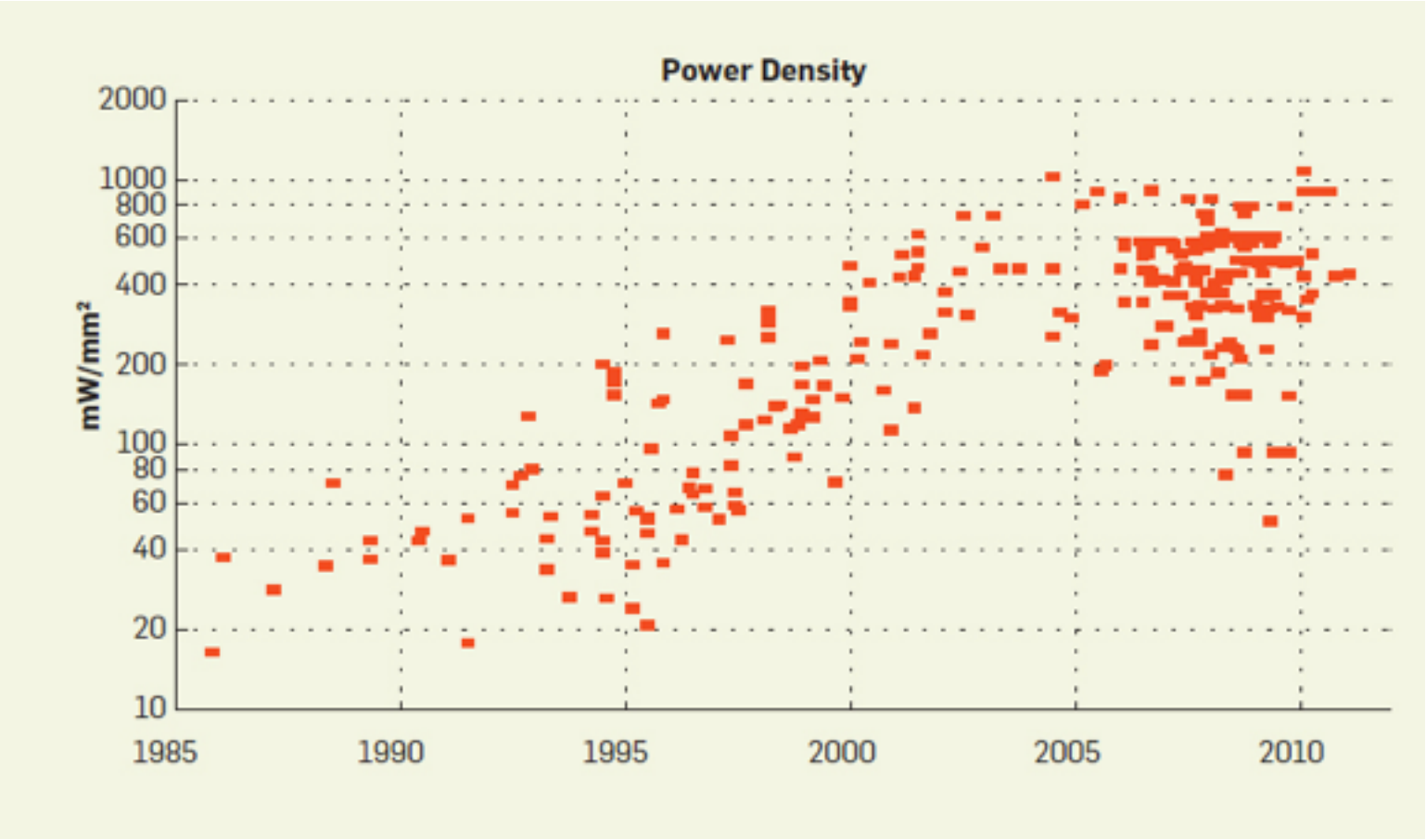
Real Technology Scaling

- Slightly worse than ideal Dennard Scaling
  - Power density steadily increasing, but still manageable

Parameter	Value	Scaled Value	
Dimensions	L, W, Tox	0.7 L, 0.7 W, 0.7 Tox	
Dopant concentrations	Na, Nd	1.4 Na, 1.4 Nd	
Voltage	V	~~0.7 V~~	**0.9 V**
Frequency	F	1.4 F	
Power/device	P	~~0.5 P~~	**0.8 P**
Power/chip	P	~~1 P~~	**1.2 P**
Power density	P/A	~~P/A~~	**1.2 P/A**

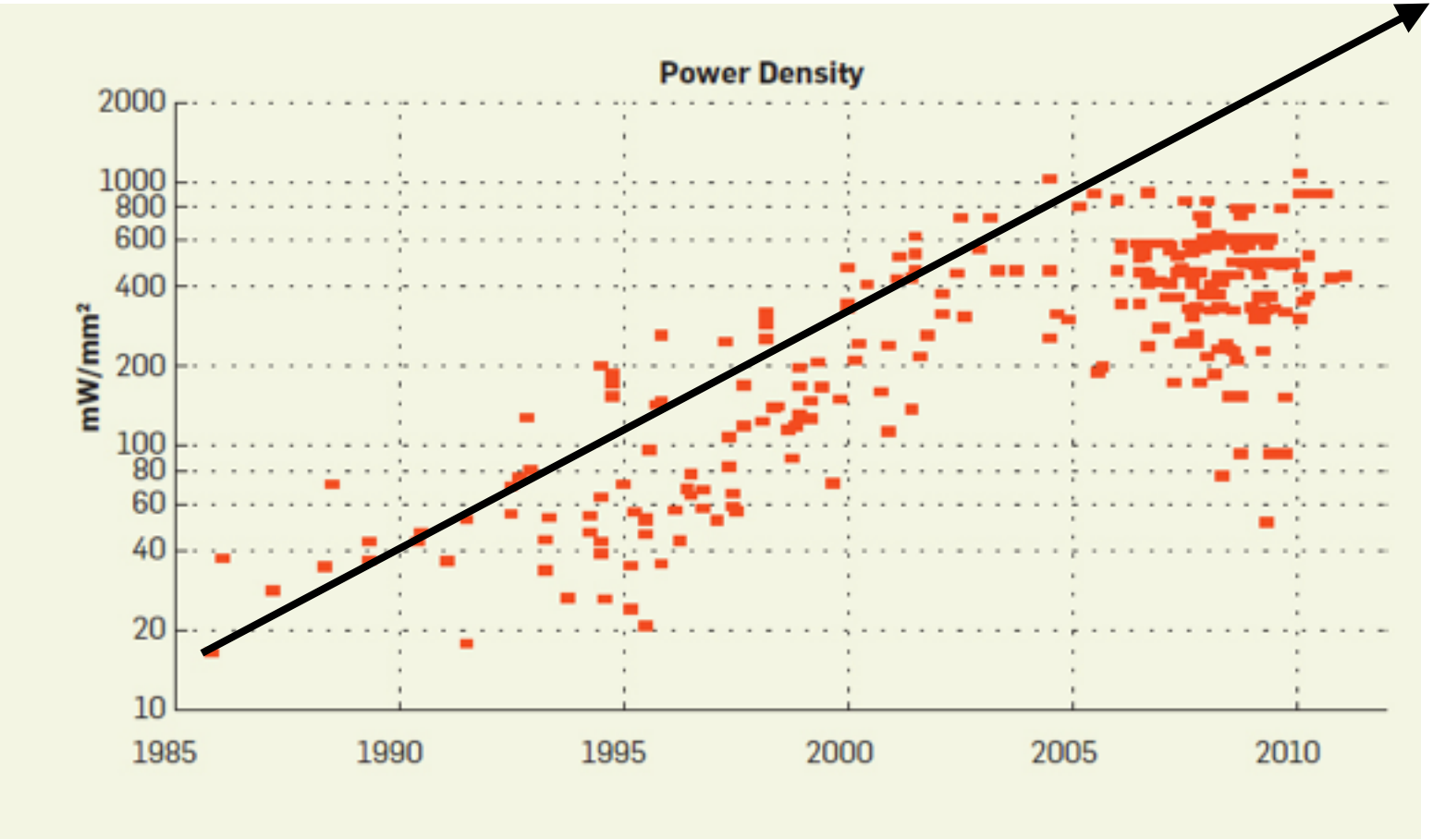
Real Technology Scaling

- Slightly worse than ideal Dennard Scaling
  - Power density steadily increasing, but still manageable



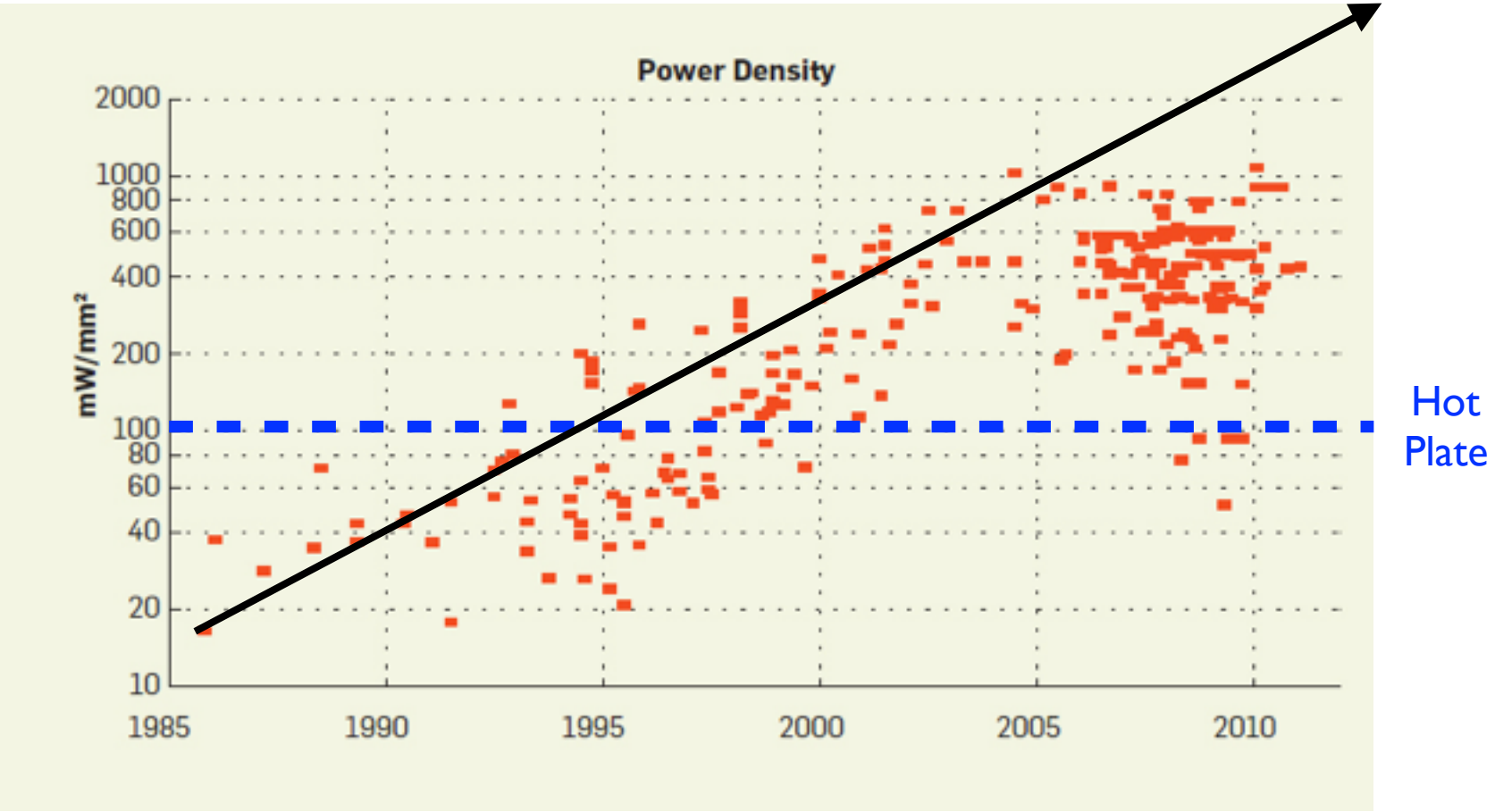
Real Technology Scaling

- Slightly worse than ideal Dennard Scaling
  - Power density steadily increasing, but still manageable



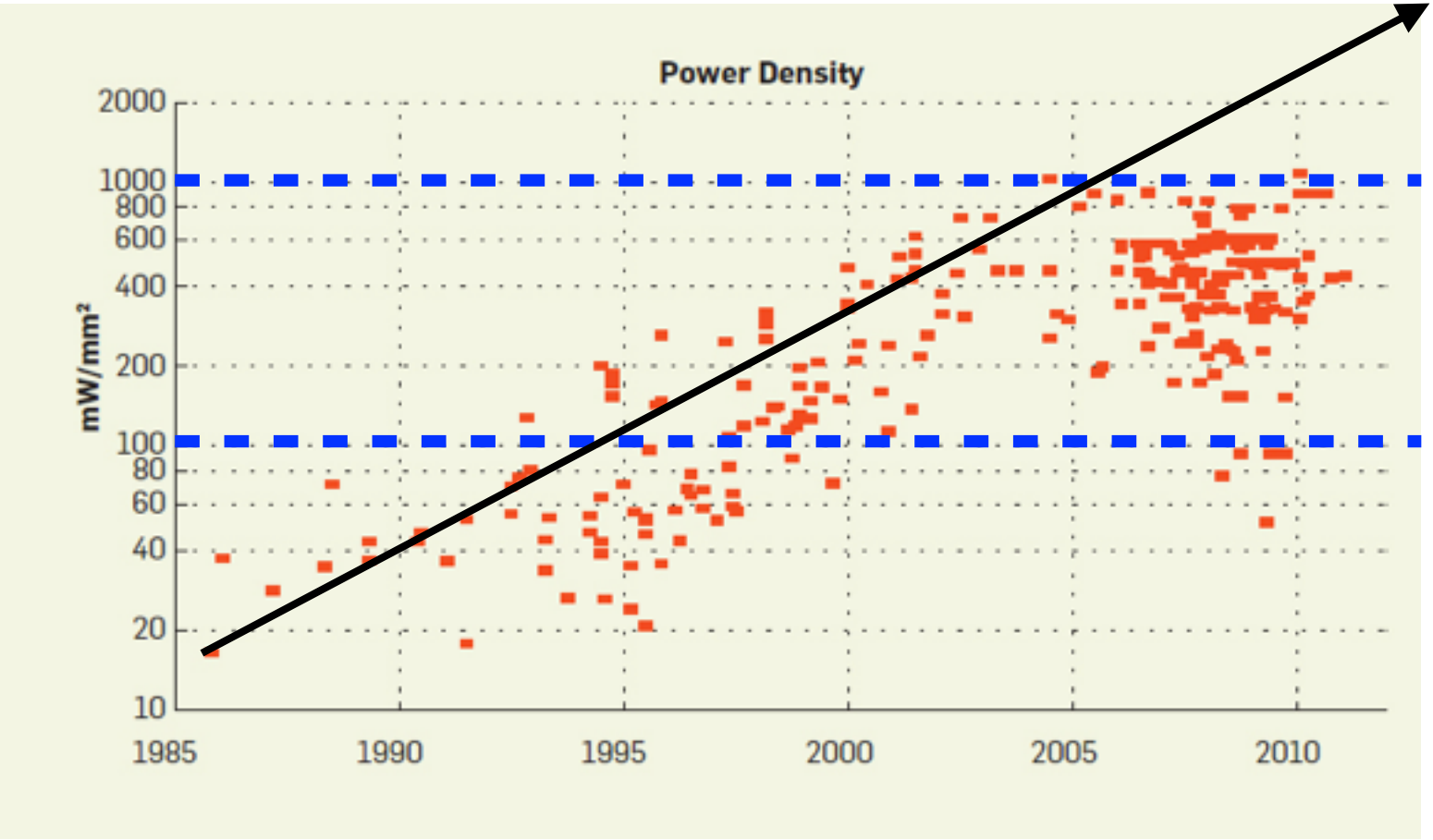
Real Technology Scaling

- Slightly worse than ideal Dennard Scaling
  - Power density steadily increasing, but still manageable



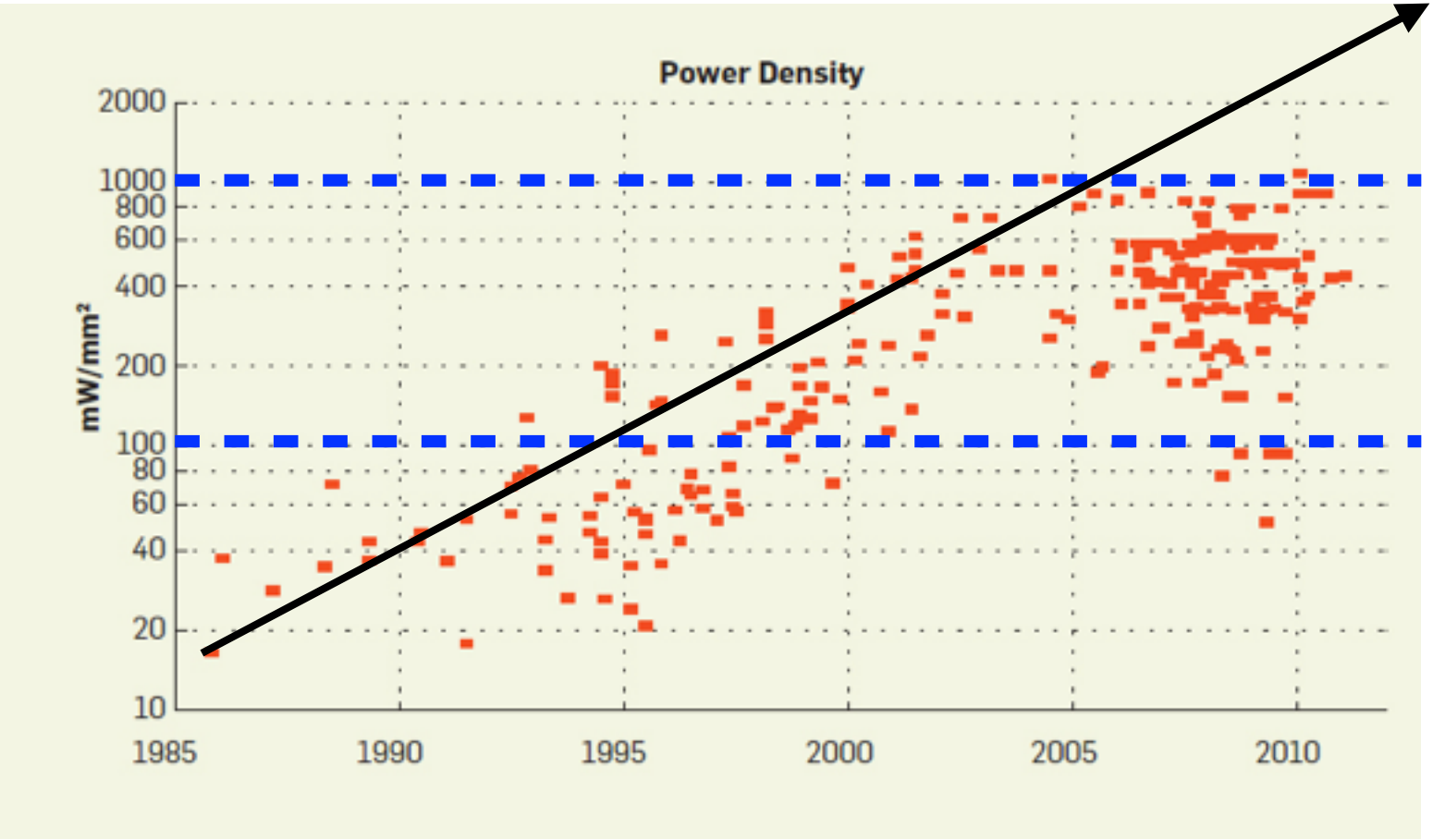
Real Technology Scaling

- Slightly worse than ideal Dennard Scaling
  - Power density steadily increasing, but still manageable



Real Technology Scaling

- Slightly worse than ideal Dennard Scaling
  - Power density steadily increasing, but still manageable



2005: End of Dennard Scaling

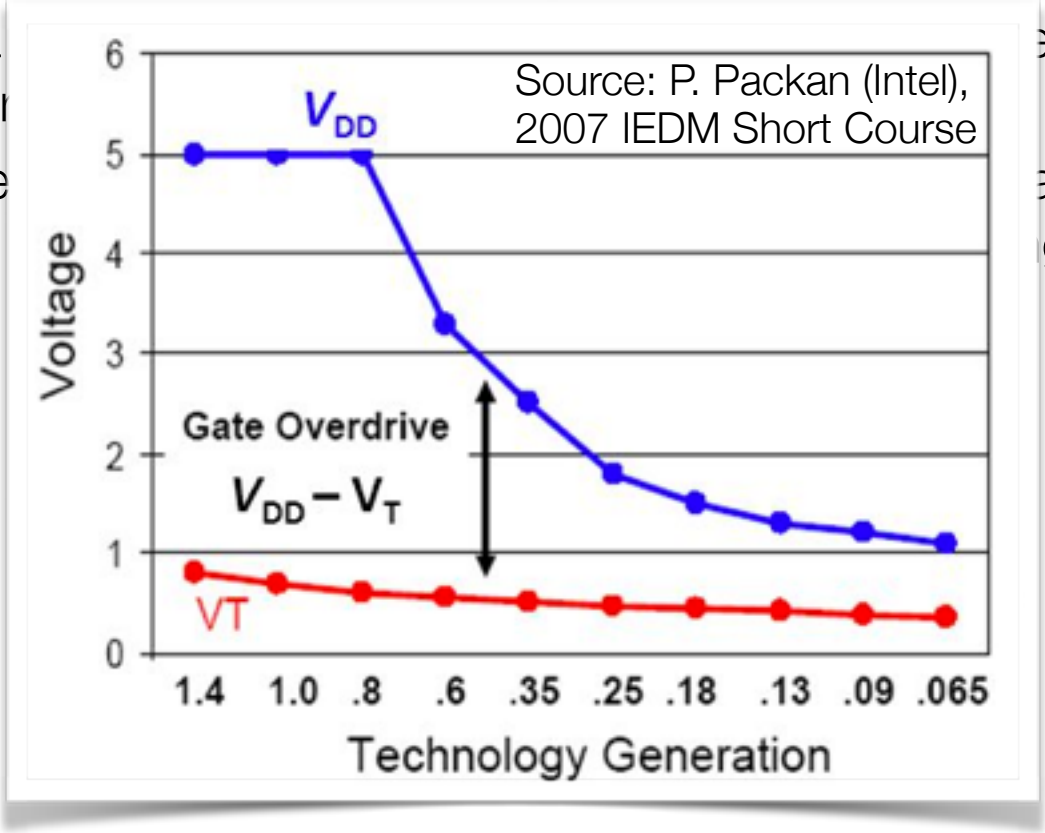
- What Happened?
  - Supply voltage V_{dd} stops scaling (Can't drop voltage below ~1 V)

2005: End of Dennard Scaling

- What Happened?
  - Supply voltage V_{dd} stops scaling (Can't drop voltage below ~1 V)
- Why?
  - There is a fundamental limit as to how much voltage we need to switch a transistor, called threshold voltage (V_{th}).
  - V_{th} stopped scaling because leakage power/reliability/variation becomes huge issues, and accordingly V_{dd} stops scaling

2005: End of Dennard Scaling

- What Happened?
  - Supply voltage V_{dd} stops scaling (Can't drop voltage below ~1 V)
- Why?
  - There is a need to switch a tr
  - V_{th} stoppe ariation becomes g



2005: End of Dennard Scaling

- What Happened?
  - Supply voltage V_{dd} stops scaling (Can't drop voltage below ~1 V)
- Why?
  - There is a fundamental limit as to how much voltage we need to switch a transistor, called threshold voltage (V_{th}).
  - V_{th} stopped scaling because leakage power/reliability/variation becomes huge issues, and accordingly V_{dd} stops scaling
- The demise of Dennard Scaling coupled with power density reaching the limit is a huge crisis for computing industry
  - Power density reached limit even with Dennard scaling, now voltage stops scaling, and things started becoming worse

Dark Silicon

n. [därk, sĭl′ĭ-kən, -kŏn′]

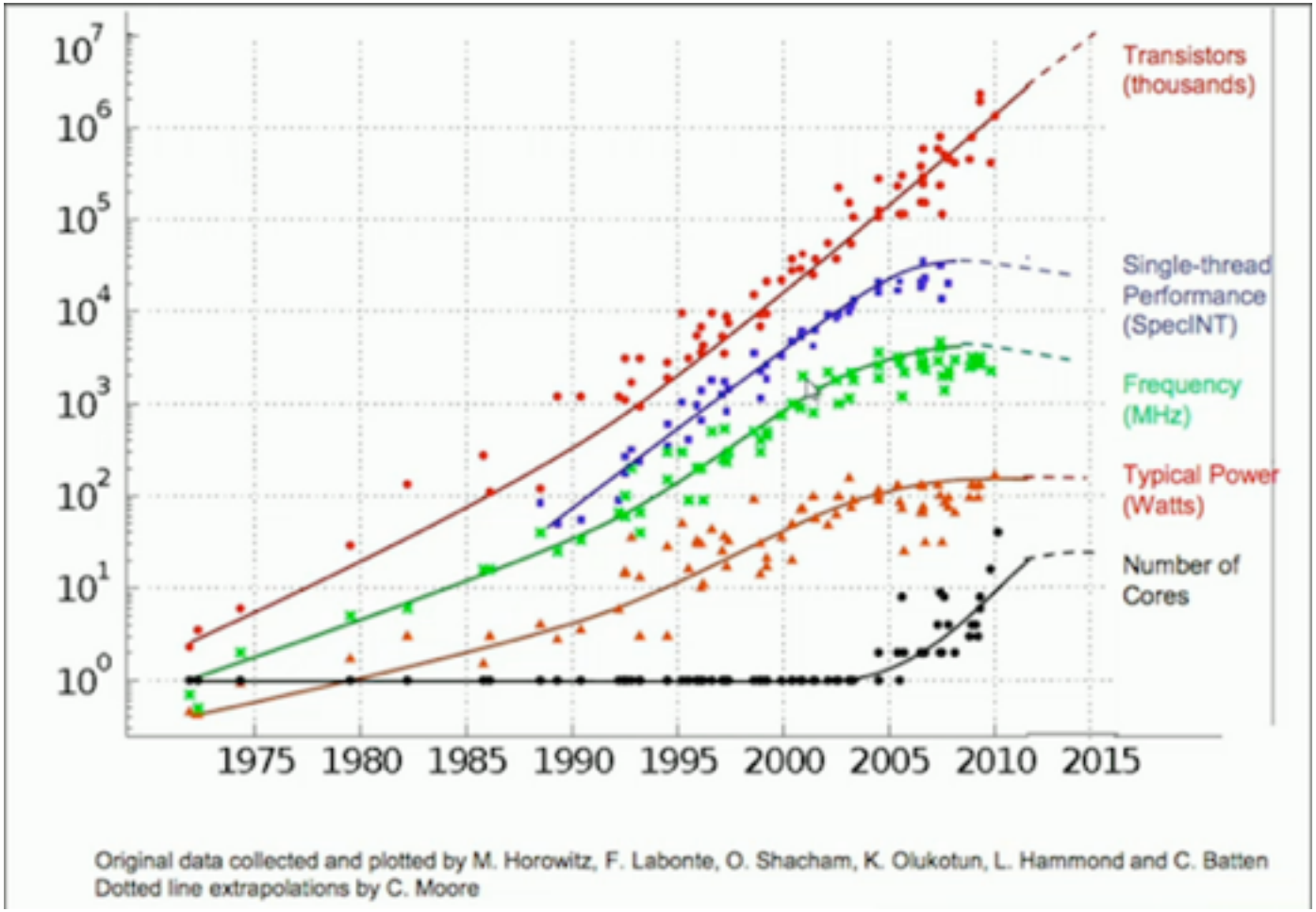
More transistors on chip (Moore's Law), but a growing fraction cannot actually be used due to power limits.

2005: End of Dennard Scaling

- Initial response has been to lower frequency and increase cores / chip

2005: End of Dennard Scaling

- Initial response has been to lower frequency and increase cores / chip



Dark Silicon and the End of Multicore Scaling

Hadi Esmaeilzadeh† Emily Blem‡ Renée St. Amant§ Karthikeyan Sankaralingam‡ Doug Burger◇
†University of Washington ‡University of Wisconsin-Madison
§The University of Texas at Austin ◇Microsoft Research
hadianeh@cs.washington.edu blem@cs.wisc.edu stamant@cs.utexas.edu karu@cs.wisc.edu dburger@microsoft.com

ABSTRACT

Since 2005, processor designers have increased core counts to exploit Moore's Law scaling, rather than focusing on single-core performance. The failure of Dennard scaling, to which the shift to multicore parts is partially a response, may soon limit multicore scaling just as single-core scaling has been curtailed. This paper models multicore scaling limits by combining device scaling, single-core scaling, and multicore scaling to measure the speedup potential for a set of parallel workloads for the next five technology generations. For device scaling, we use both the ITRS projections and a set of more conservative device scaling parameters. To model single-core scaling, we combine measurements from over 150 processors to derive Pareto-optimal frontiers for area/performance and power/performance. Finally, to model multicore scaling, we build a detailed performance model of upper-bound performance and lower-bound core power. The multicore designs we study include single-threaded CPU-like and massively threaded GPU-like multicore chip organizations with symmetric, asymmetric, dynamic, and composed topologies. The study shows that regardless of chip organization and topology, multicore scaling is power limited to a degree not widely appreciated by the computing community. Even at 22 nm (just one year from now), 21% of a fixed-size chip must be powered off, and at 8 nm, this number grows to more than 50%. Through 2024, only 7.9× average speedup is possible across commonly used parallel workloads, leaving a nearly 24-fold gap from a target of doubled performance per generation.

Categories and Subject Descriptors: C.0 [Computer Systems Organization] General — Modeling of computer architecture; C.0

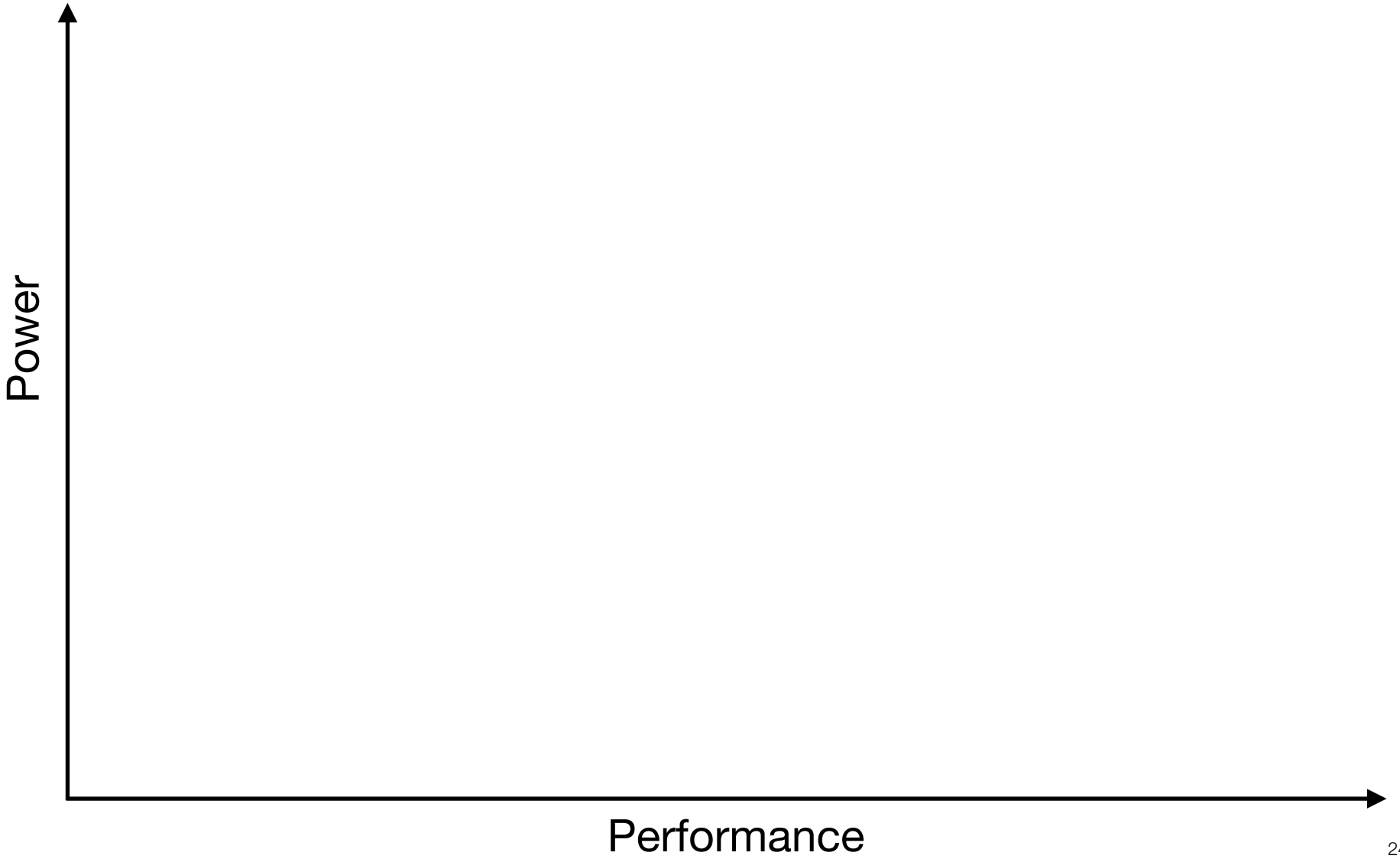
ture, and compiler advances, Moore's Law, coupled with Dennard scaling [11], has resulted in commensurate exponential performance increases. The recent shift to multicore designs has aimed to increase the number of cores along with transistor count increases, and continue the proportional scaling of performance. As a result, architecture researchers have started focusing on 100-core and 1000-core chips and related research topics and called for changes to the undergraduate curriculum to solve the parallel programming challenge for multicore designs at these scales.

With the failure of Dennard scaling–and thus slowed supply voltage scaling–core count scaling may be in jeopardy, which would leave the community with no clear scaling path to exploit continued transistor count increases. Since future designs will be power limited, higher core counts must provide performance gains despite the worsening energy and speed scaling of transistors, and given the available parallelism in applications. By studying these characteristics together, it is possible to predict for how many additional technology generations multicore scaling will provide a clear benefit. Since the energy efficiency of devices is not scaling along with integration capacity, and since few applications (even from emerging domains such as recognition, mining, and synthesis [5]) have parallelism levels that can efficiently use a 100-core or 1000-core chip, it is critical to understand how good multicore performance will be in the long term. In 2024, will processors have 32 times the performance of processors from 2008, exploiting five generations of core doubling?

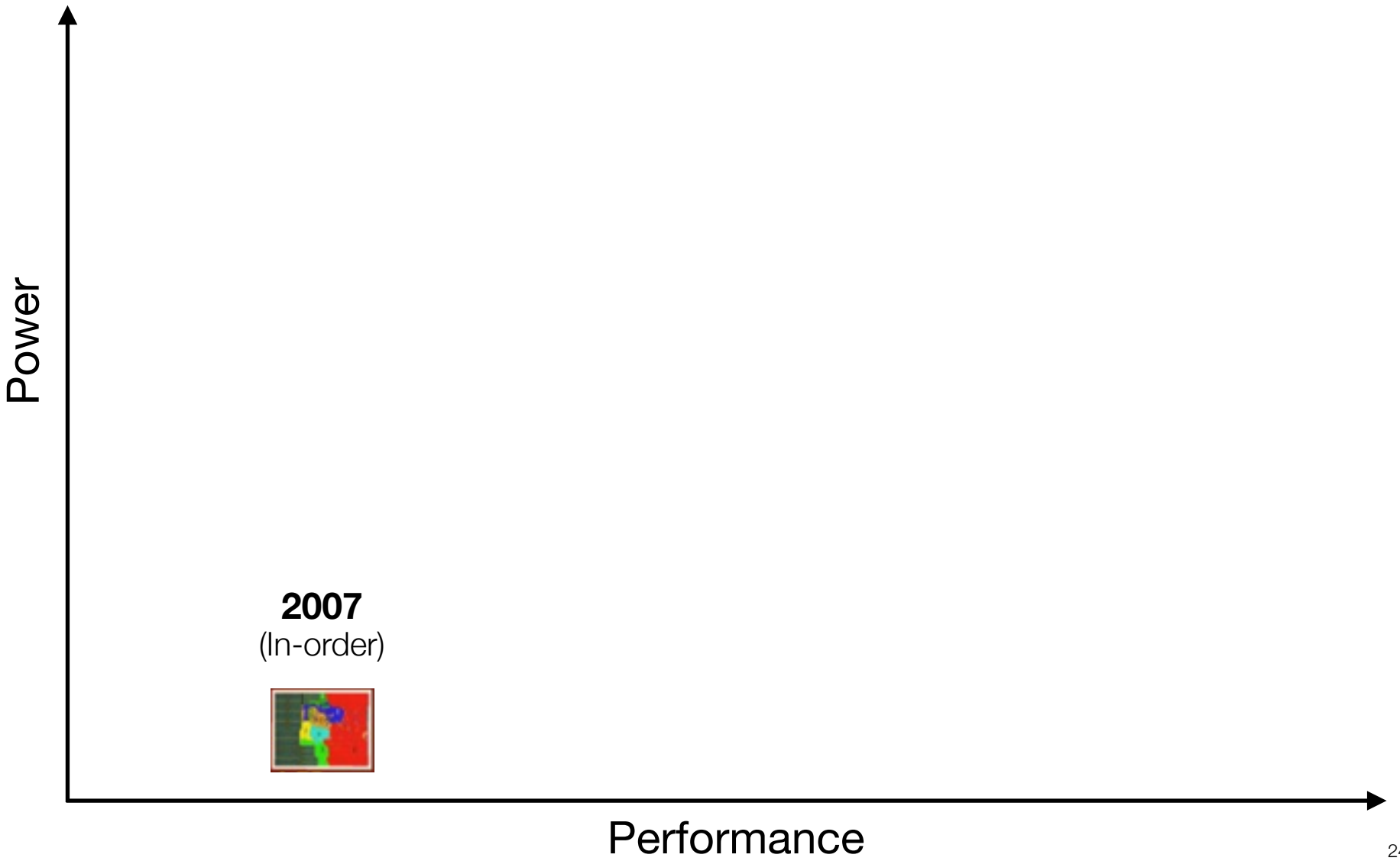
Such a study must consider devices, core microarchitectures, chip organizations, and benchmark characteristics, applying area

2007: A Revolutionary New Computer

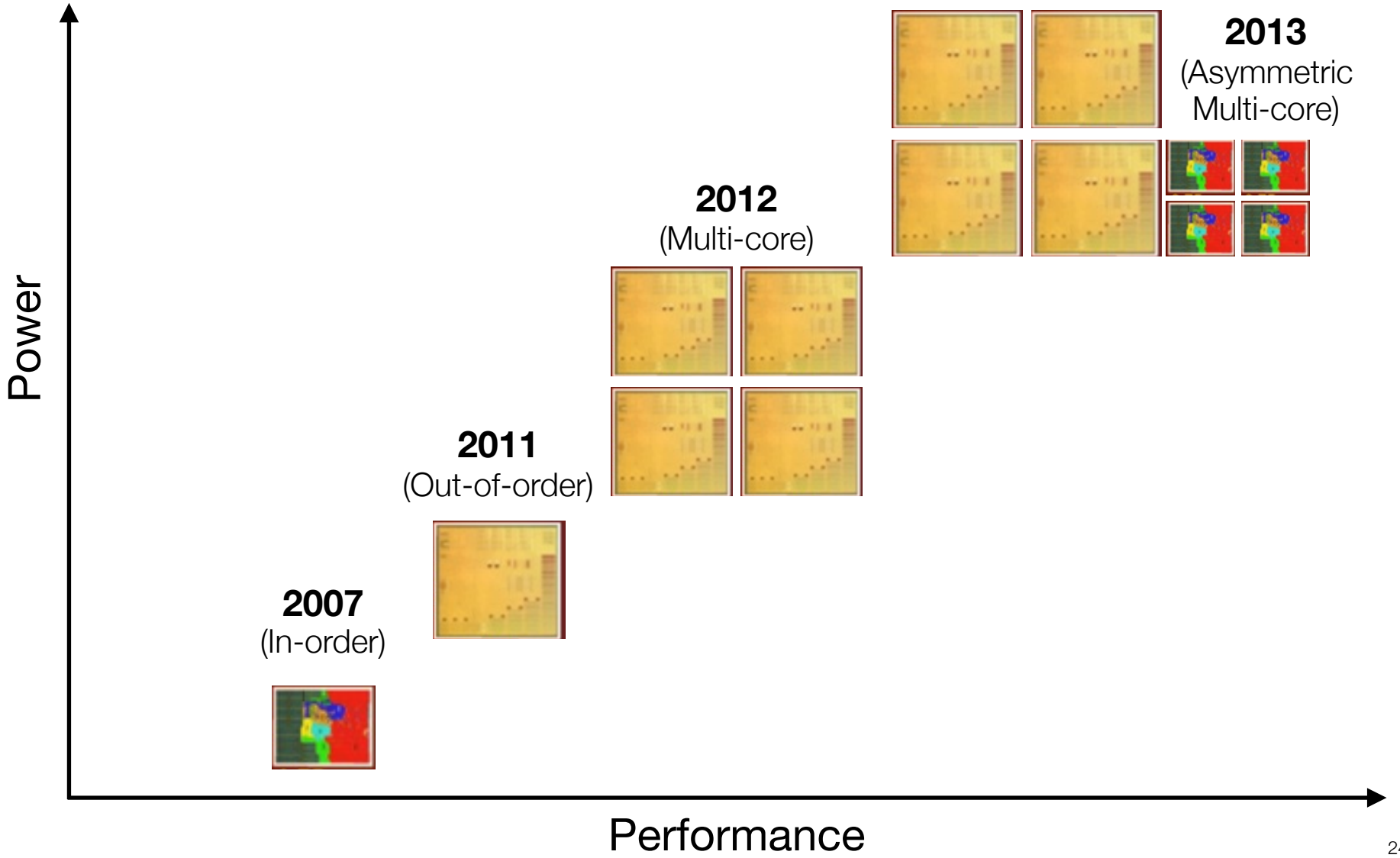
Mobile Processor Design “Strategy”



Mobile Processor Design “Strategy”

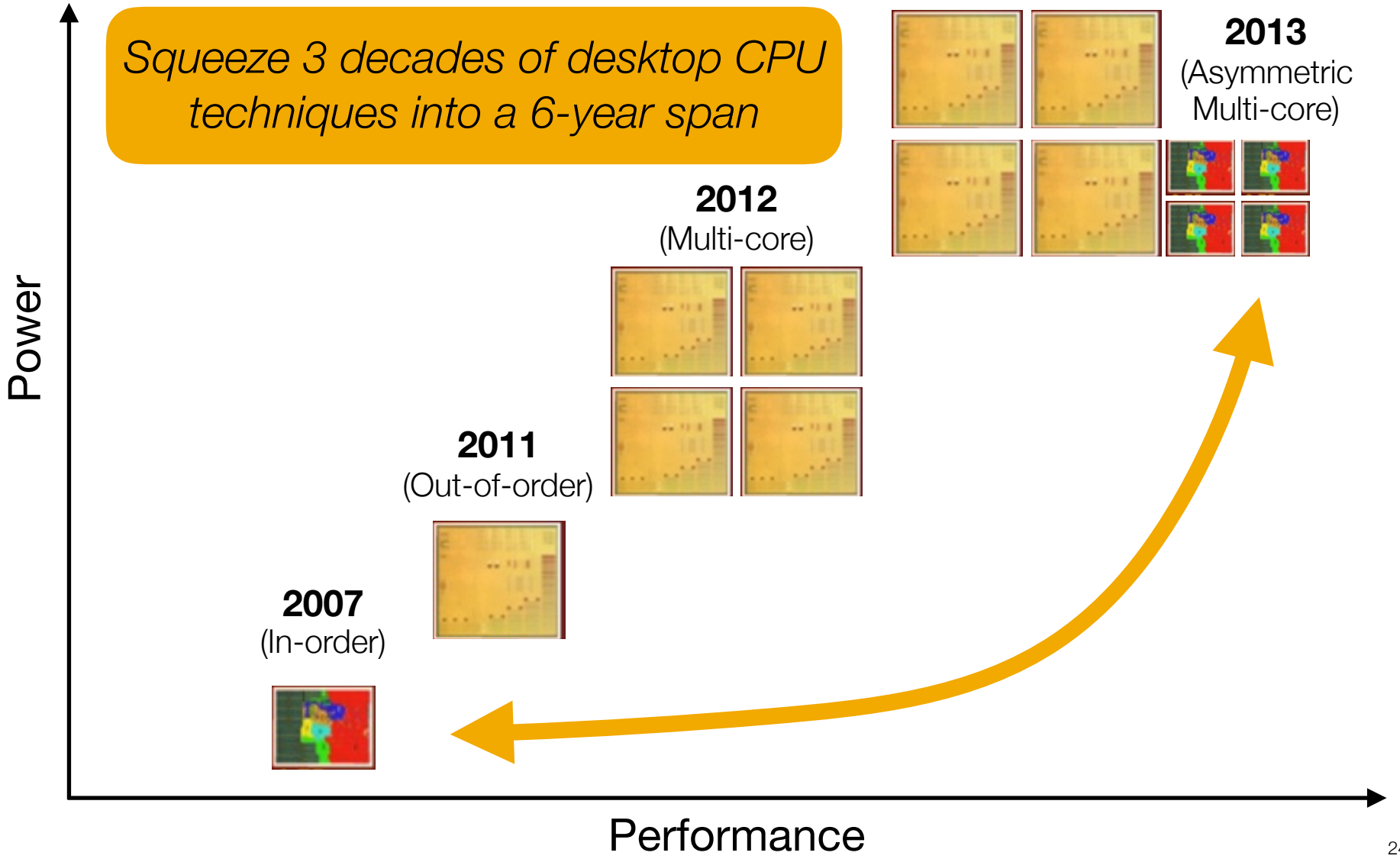


Mobile Processor Design “Strategy”



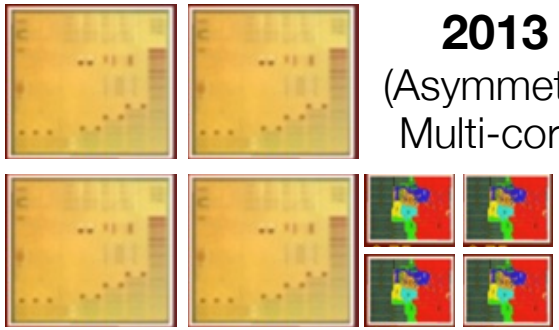
Mobile Processor Design "Strategy"

Squeeze 3 decades of desktop CPU techniques into a 6-year span



Mobile Processor Design "Strategy"

Squeeze 3 decades of desktop CPU techniques into a 6-year span



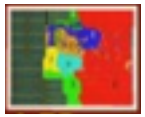
2013
(Asymmetric Multi-core)

2012
(Multi-core)

Ran Into Dark Silicon Issue in 6 Years

(Out-of-order)

2007
(In-order)



Performance

OPINION

No Moore's Law for batteries

Fred Schlachter[1]
American Physical Society, Washington, DC 20045

The public has become accustomed to rapid progress in mobile phone technology, computers, and access to information; tablet computers, smart phones, and other powerful new devices are familiar to most people on the planet.

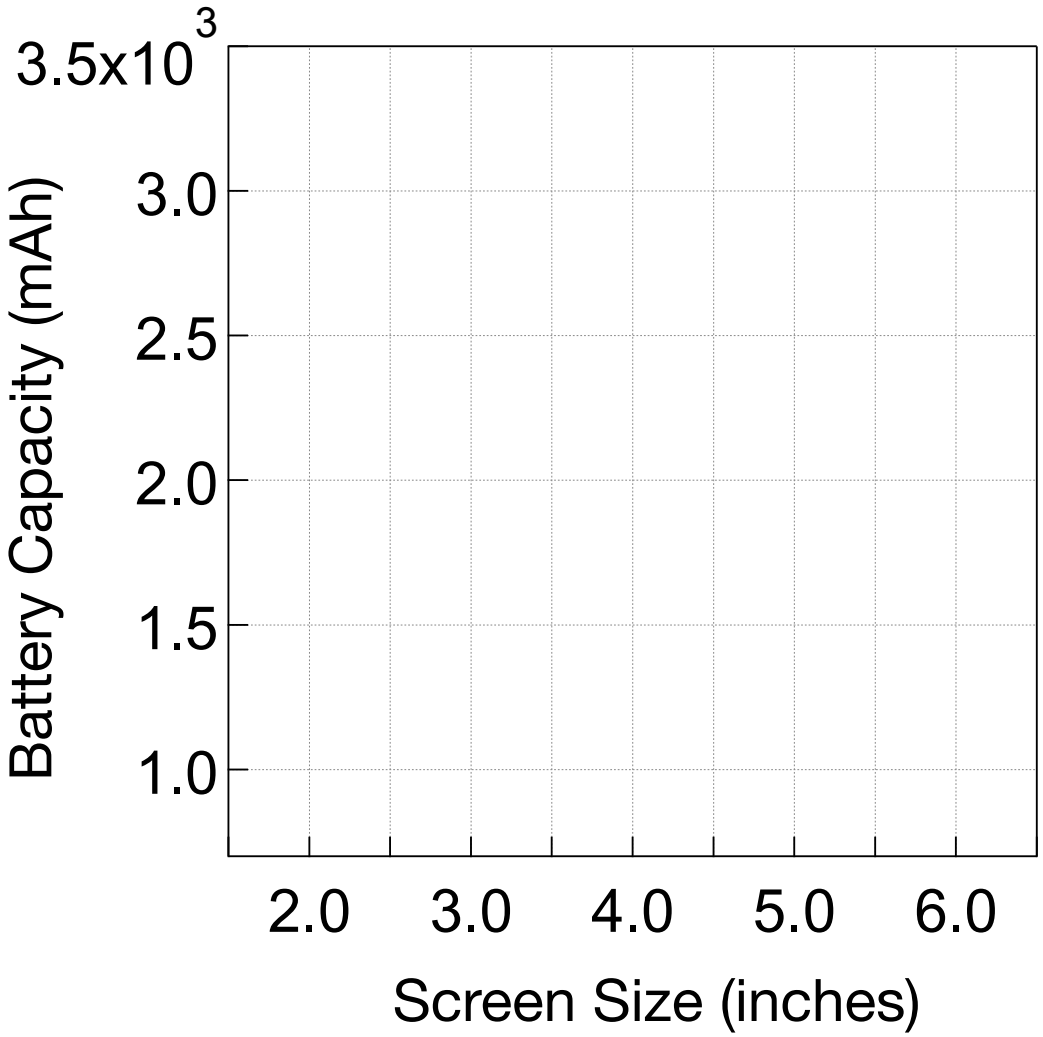
These developments are due in part to the ongoing exponential increase in computer processing power, doubling approximately every 2 years for the past several decades. This pattern is usually called Moore's Law and is named for Gordon Moore, a cofounder of Intel. The law is not a law like that for gravity; it is an empirical observation, which has become a self-fulfilling prophecy. Unfortunately, much of the public has come to expect that all technology does, will, or should follow such a law, which is not consistent with our everyday observations: For example, the maximum there is a Moore's Law for computer processors is that electrons are small and they do not take up space on a chip. Chip performance is limited by the lithography technology used to fabricate the chips; as lithography improves ever smaller features can be made on processors. Batteries are not like this. Ions, which transfer charge in batteries, are large, and they take up space, as do anodes, cathodes, and electrolytes. A D-cell battery stores more energy than an AA-cell. Potentials in a battery are dictated by the relevant chemical reactions, thus limiting eventual battery performance. Significant improvement in battery capacity can only be made by changing to a different chemistry.

Scientists and battery experts, who have been optimistic in the recent past about improving lithium-ion batteries and about de-

“Improving” Energy Capacity

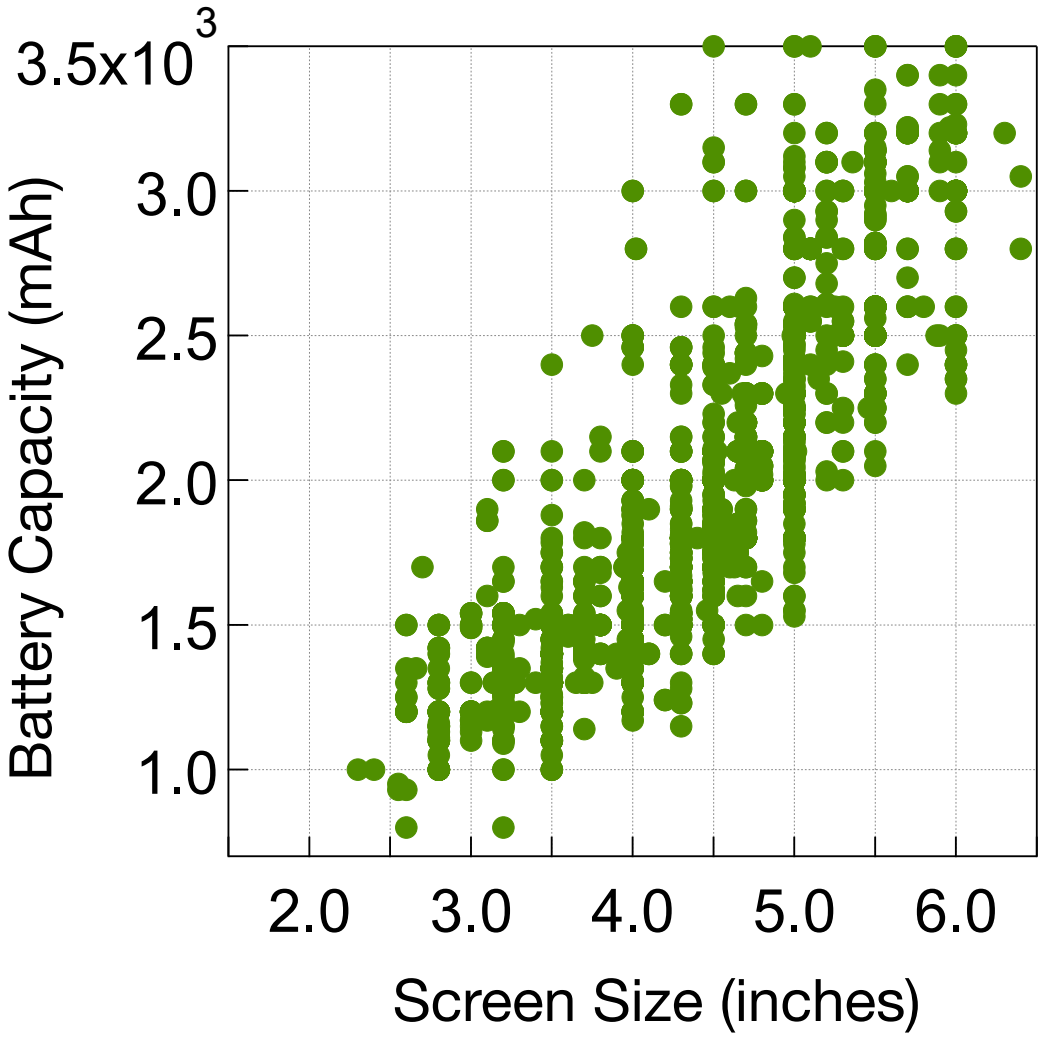
600 smartphone from 2006 to 2014 on http://www.gsmarena.com/makers.php3

“Improving” Energy Capacity



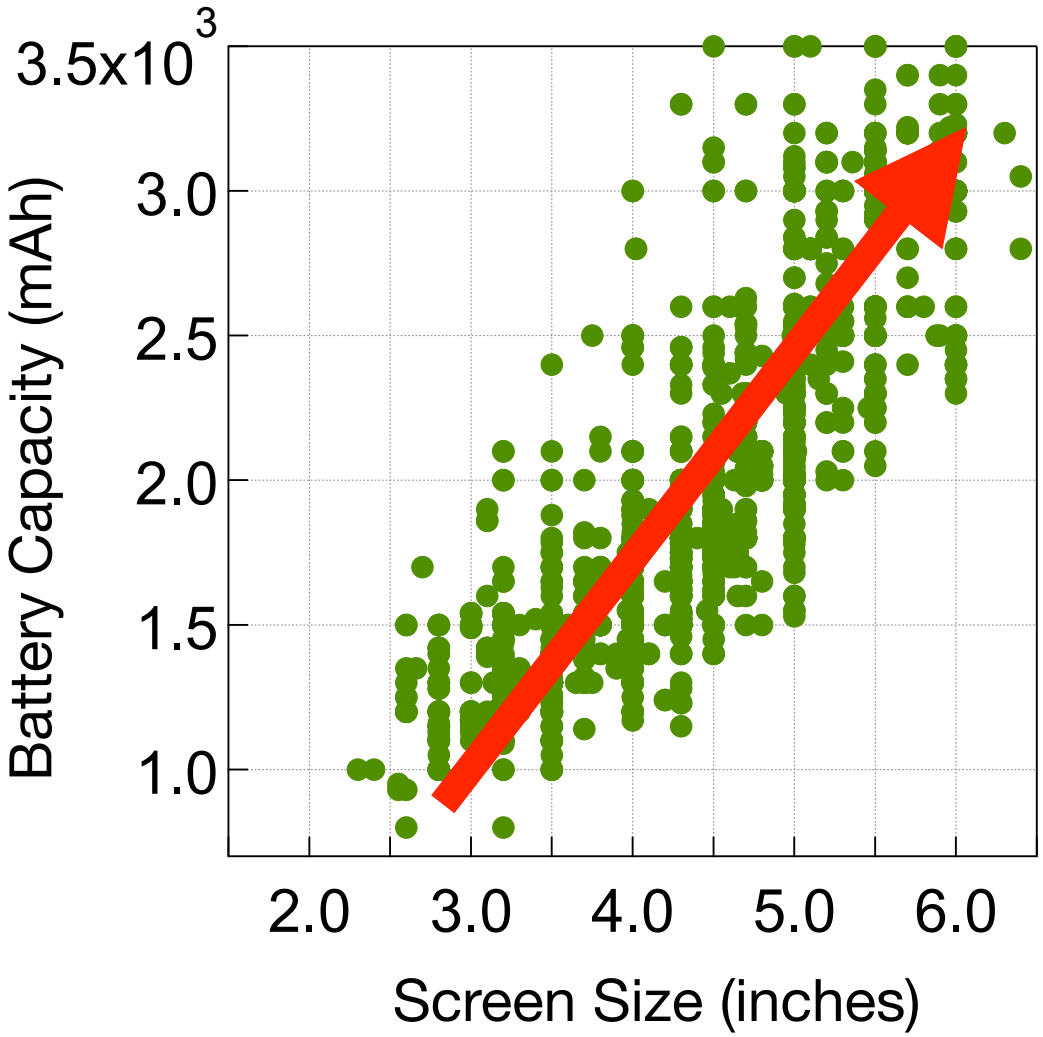
600 smartphone from 2006 to 2014 on http://www.gsmarena.com/makers.php3

“Improving” Energy Capacity



600 smartphone from 2006 to 2014 on http://www.gsmarena.com/makers.php3

“Improving” Energy Capacity



600 smartphone from 2006 to 2014 on http://www.gsmarena.com/makers.php3

“Improving” Energy Capacity

“Improving” Energy Capacity

“Improving” Energy Capacity

“Improving” Energy Capacity

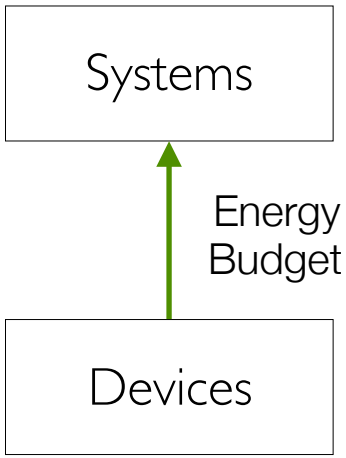
“Improving” Energy Capacity

The (Mobile) Computing Virtuous Cycle

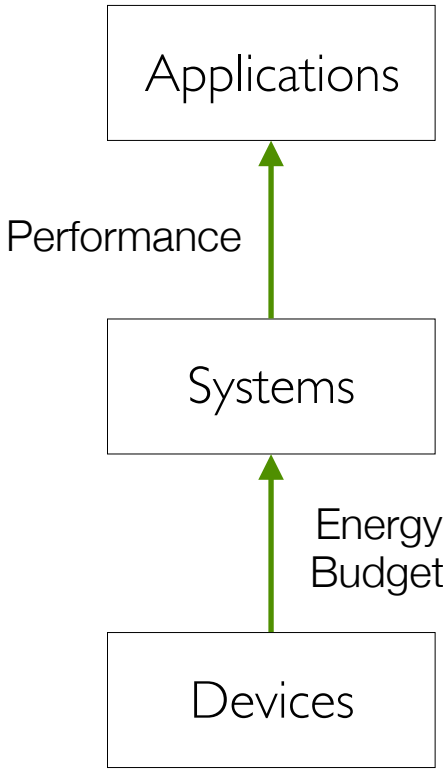
The (Mobile) Computing Virtuous Cycle

Devices

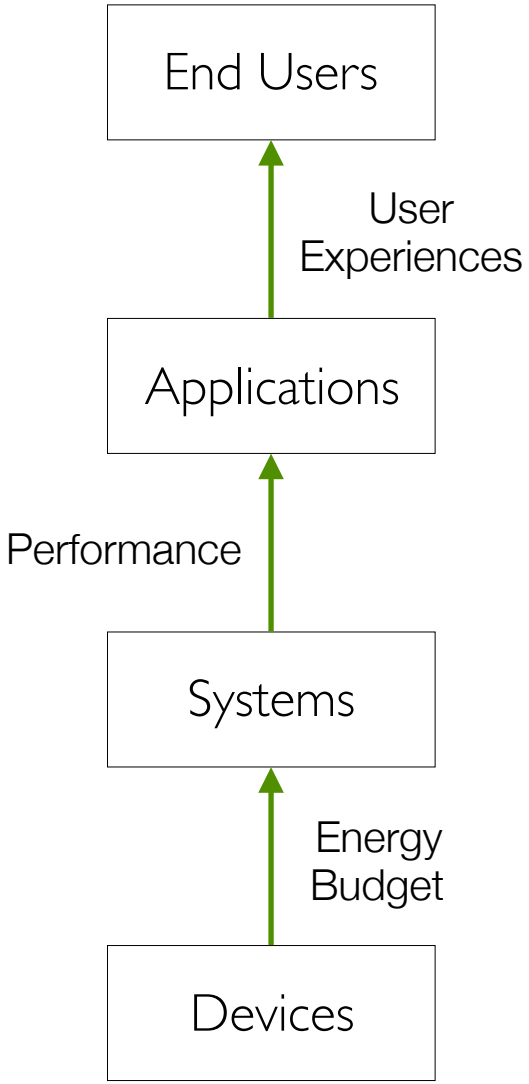
The (Mobile) Computing Virtuous Cycle



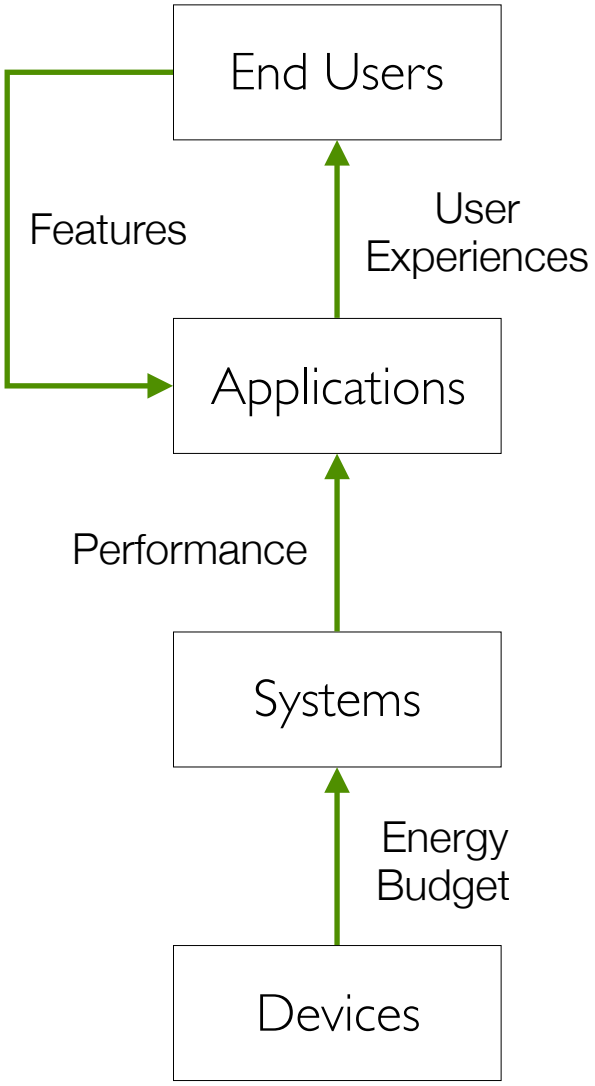
The (Mobile) Computing Virtuous Cycle



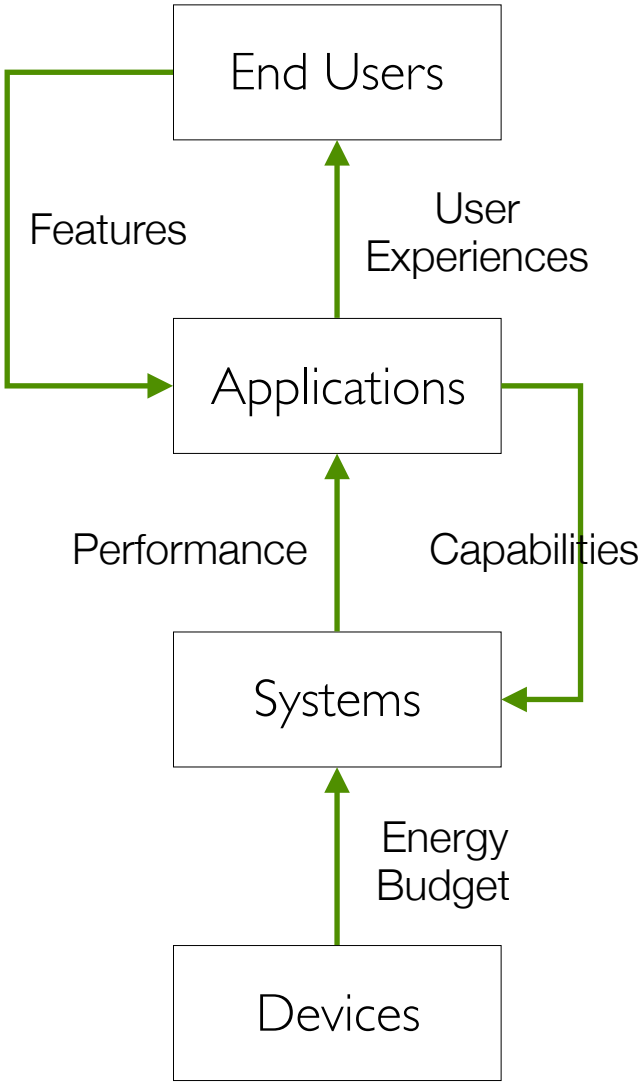
The (Mobile) Computing Virtuous Cycle



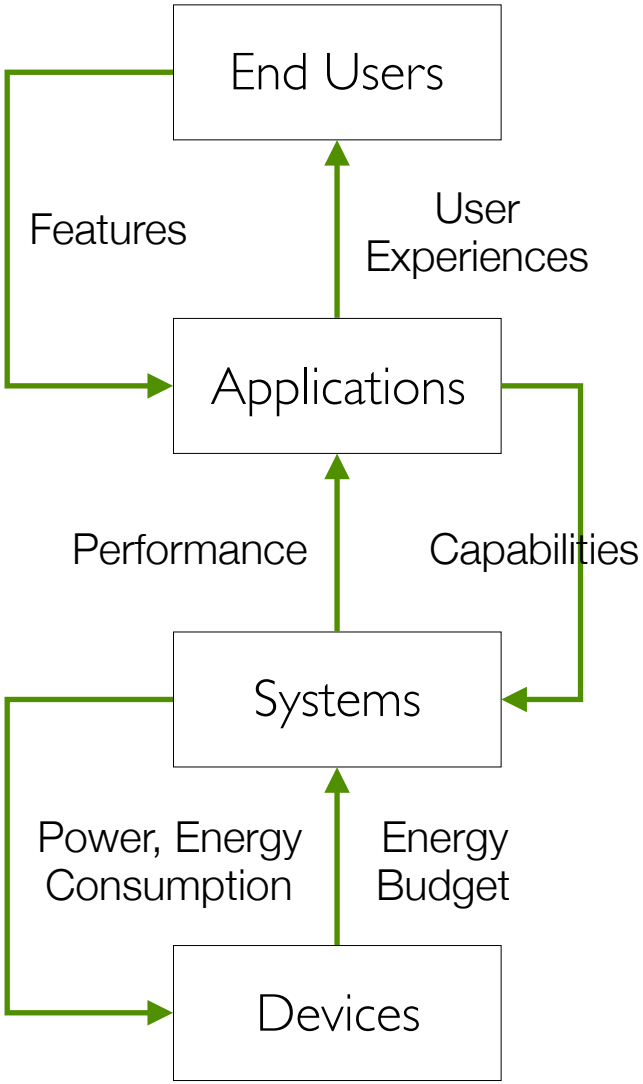
The (Mobile) Computing Virtuous Cycle



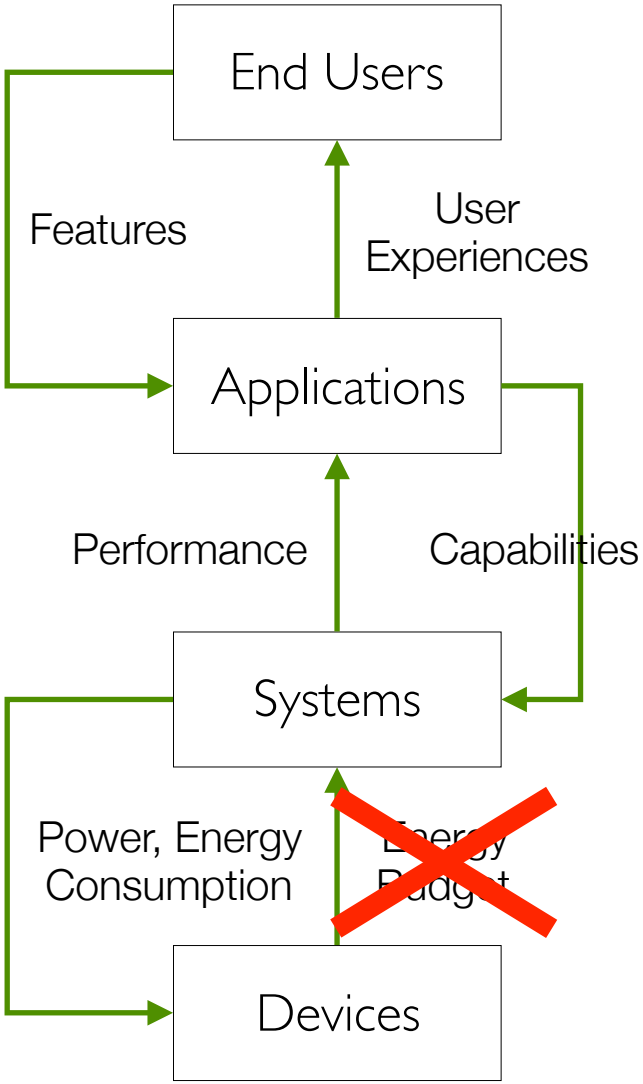
The (Mobile) Computing Virtuous Cycle



The (Mobile) Computing Virtuous Cycle



The (Mobile) Computing Virtuous Cycle



Mobile CPU's Rise to Power: Quantifying the Impact of Generational Mobile CPU Design Trends on Performance, Energy, and User Satisfaction

Matthew Halpern Yuhao Zhu Vijay Janapa Reddi
The University of Texas at Austin, Department of Electrical and Computer Engineering
{matthalp,yzhu}@utexas.edu, vj@ece.utexas.edu

Abstract

In this paper, we assess the past, present, and future of mobile CPU design. We study how mobile CPU designs trends have impacted the end-user, hardware design, and the holistic mobile device. We analyze the evolution of ten cutting-edge mobile CPU designs released over the past seven years. Specifically, we report measured performance, power, energy and user satisfaction trends across mobile CPU generations.

A key contribution of our work is that we contextualize the mobile CPU's evolution in terms of user satisfaction, which has largely been absent from prior mobile hardware studies. To bridge the gap between mobile CPU design and user satisfaction, we construct and conduct a novel crowdsourcing study that spans over 25,000 survey participants using the Amazon Mechanical Turk service. Our methodology allows us to identify what mobile CPU design techniques provide the most benefit to the end-user's quality of user experience.

Our results quantitatively demonstrate that CPUs play a crucial role in modern mobile system-on-chips (SoCs). Over the last seven years, both single- and multicore performance improvements have contributed to end-user satisfaction by reducing user-critical application response latencies. Mobile CPUs aggressively adopted many power-hungry desktop-oriented design techniques to reach these performance levels. Unlike other smartphone components (e.g. display and radio) whose peak power consumption has decreased over time, the mobile CPU's peak power consumption has steadily increased.

As the limits of technology scaling restrict the ability of desktop-like scaling to continue for mobile CPUs, specialized accelerators appear to be a promising alternative that can help sustain the power, performance, and energy improvements that

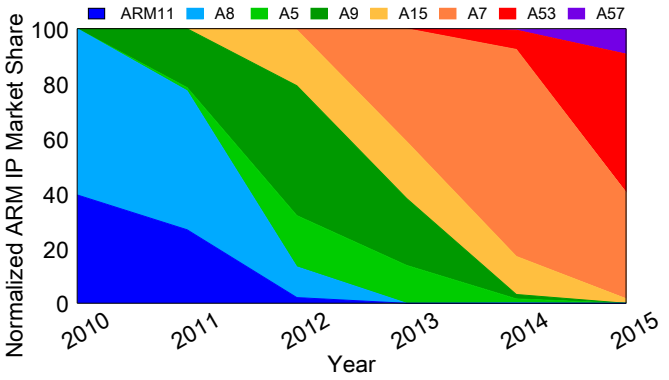


Fig. 1: Breakdown of yearly ARM Cortex-A CPU design market share. Mobile CPU core designs have rapid design iteration and innovation. At least one new core design is released each year and newer designs overshadow the older ones.

rate to keep pace with end-user demands. Fig. 1, based on data mined from over 1700 Android smartphone specifications, conveys the fast pace at which mobile CPU designs have evolved. Considering the ARM-based Cortex-A series alone, the most dominant mobile CPU design in smartphones and tablets to date [1], at least one new CPU core design has been released each year for the last six years – each significantly more advanced than the last. In comparison, x86-based desktop CPU designs did not exhibit as dramatic changes. Intel-based desktop processors only exhibited four significant core design changes throughout the same time span.

The rapid design innovation, pervasiveness in society, and power-constrained nature of mobile hardware necessitate the need to understand the implications of their current design trends on future designs. Mobile CPUs have evolved from em-

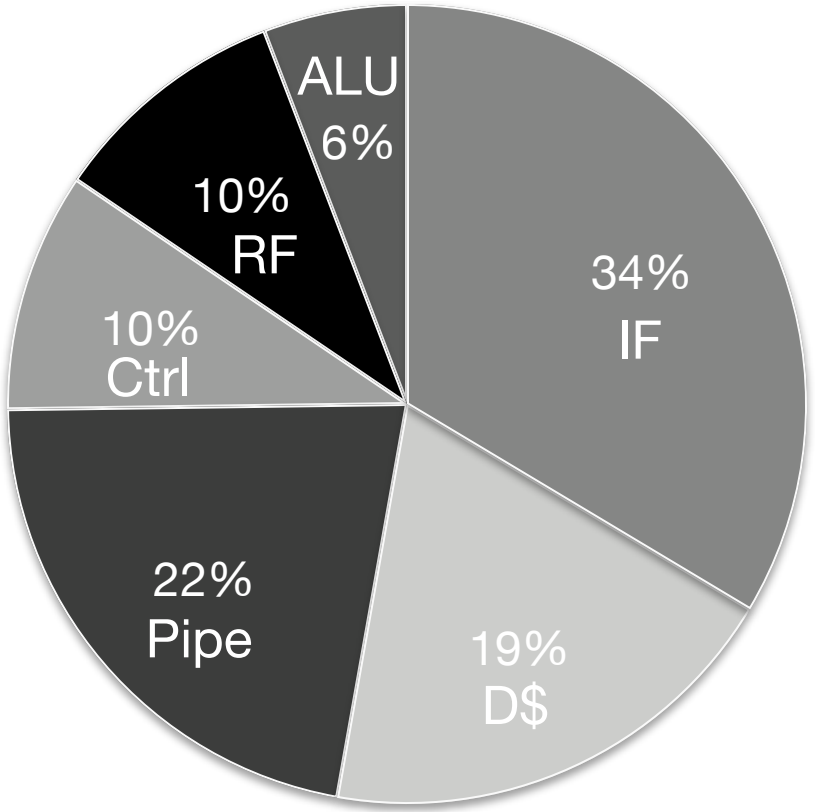
Sources of Energy-Inefficiencies

General-Purpose CPU = Instruction Delivery + Data Feeding + Execution + Control, where instruction delivery, data feeding & control are pure overhead

Understanding sources of inefficiency in general-purpose chips, Hameed et al., ISCA 2010

Sources of Energy-Inefficiencies

General-Purpose CPU = Instruction Delivery + Data Feeding + Execution + Control, where instruction delivery, data feeding & control are pure overhead



IF: Instruction fetch

Ctrl: Other control logics

Pipe: Pipeline reg, bus, clock

D$: Data cache

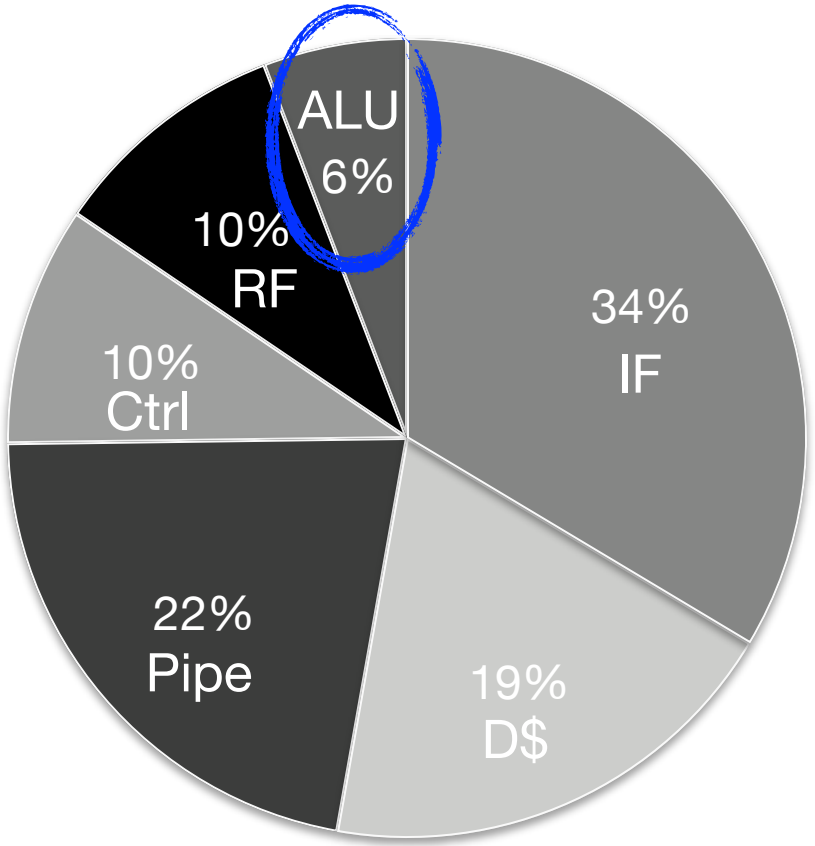
RF: Register file

ALU: Functional units

Understanding sources of inefficiency in general-purpose chips, Hameed et al., ISCA 2010

Sources of Energy-Inefficiencies

General-Purpose CPU = Instruction Delivery + Data Feeding + Execution + Control, where instruction delivery, data feeding & control are pure overhead



IF: Instruction fetch

Ctrl: Other control logics

Pipe: Pipeline reg, bus, clock

D$: Data cache

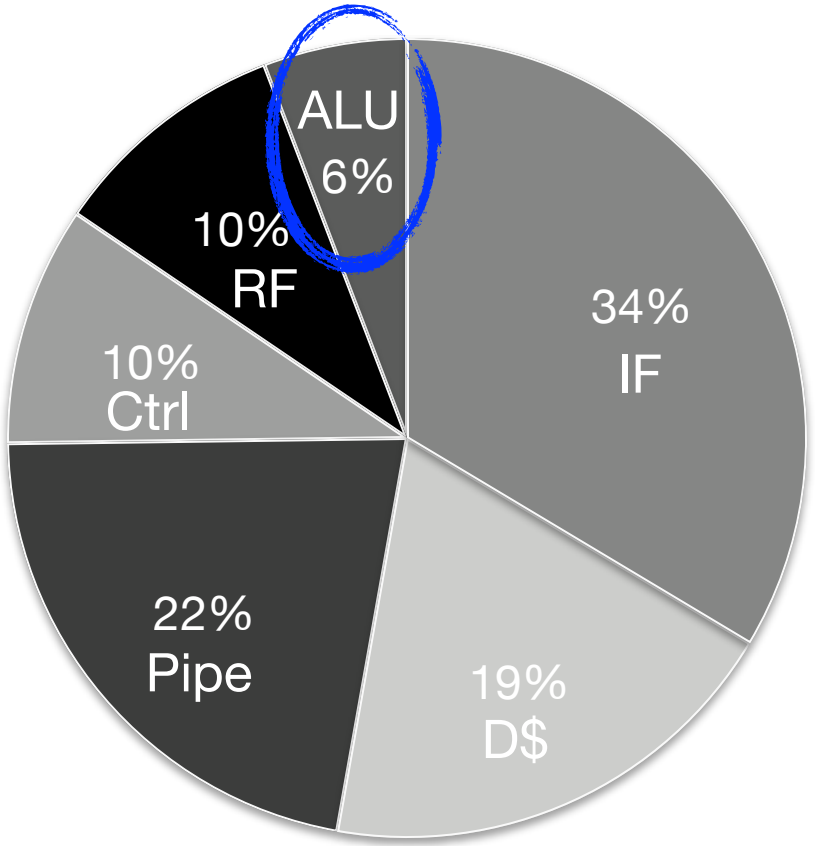
RF: Register file

ALU: Functional units

Understanding sources of inefficiency in general-purpose chips, Hameed et al., ISCA 2010

Sources of Energy-Inefficiencies

General-Purpose CPU = Instruction Delivery + Data Feeding + Execution + Control, where instruction delivery, data feeding & control are pure overhead



Pure Overhead

IF: Instruction fetch

Ctrl: Other control logics

Pipe: Pipeline reg, bus, clock

D$: Data cache

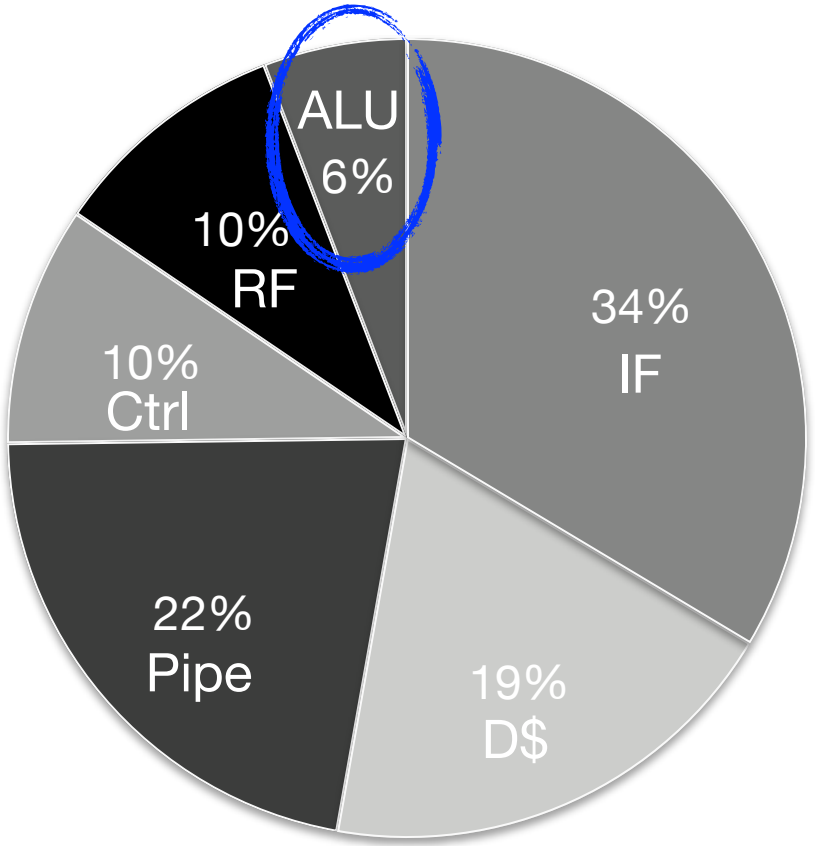
RF: Register file

ALU: Functional units

Understanding sources of inefficiency in general-purpose chips, Hameed et al., ISCA 2010

Sources of Energy-Inefficiencies

General-Purpose CPU = Instruction Delivery + Data Feeding + Execution + Control, where instruction delivery, data feeding & control are pure overhead



Pure Overhead

IF: Instruction fetch

Ctrl: Other control logics

Pipe: Pipeline reg, bus, clock

D$: Data cache

RF: Register file

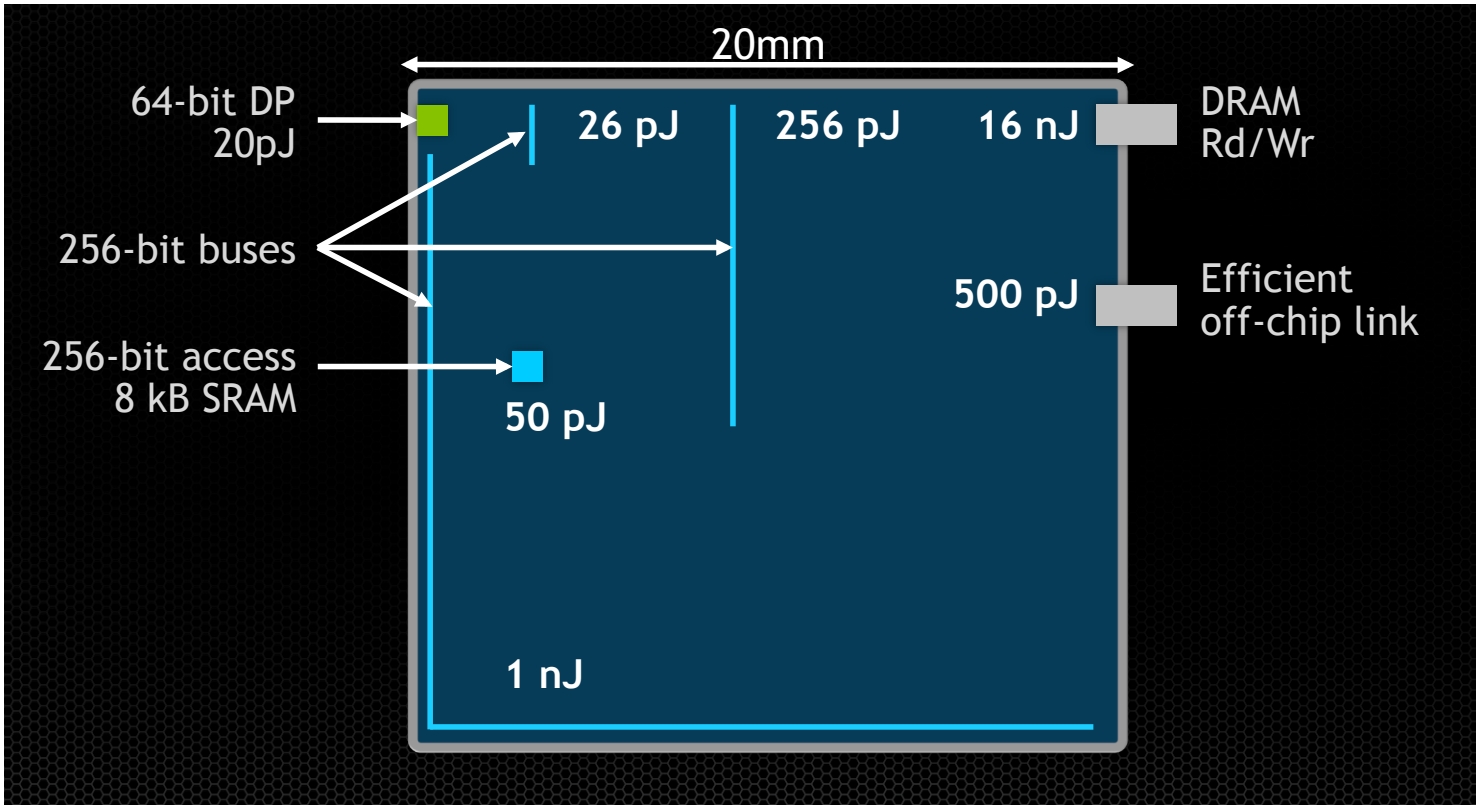
ALU: Functional units

Doing Actual Work

Understanding sources of inefficiency in general-purpose chips, Hameed et al., ISCA 2010

Sources of Energy-Inefficiencies

General-Purpose CPU = Instruction Delivery + Data Feeding + Execution + Control, where instruction delivery, data feeding & control are pure overhead



Challenges for future computing systems, Bill Dally, 2015

Now What?

Entering the Era of Specialization

Entering the Era of Specialization

- Instead of building general-purpose processors that can do everything, but inefficiently, let’s build specialized processors that can only do limited things, but extremely efficiently.

Entering the Era of Specialization

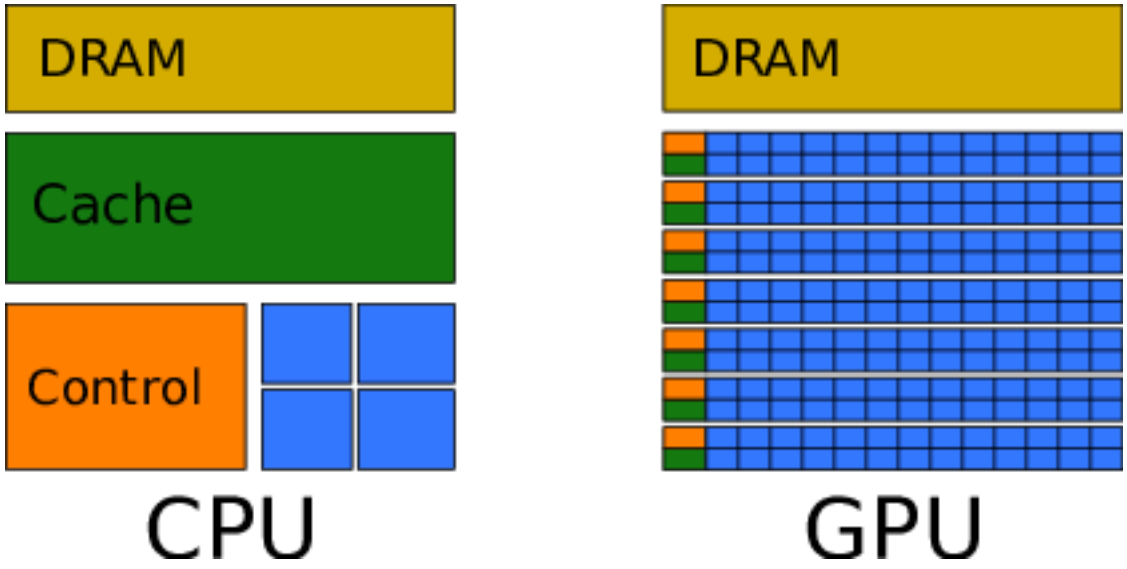
- Instead of building general-purpose processors that can do everything, but inefficiently, let's build specialized processors that can only do limited things, but extremely efficiently.
- Extreme example: Application-Specific Integrated Circuit (ASIC)

Entering the Era of Specialization

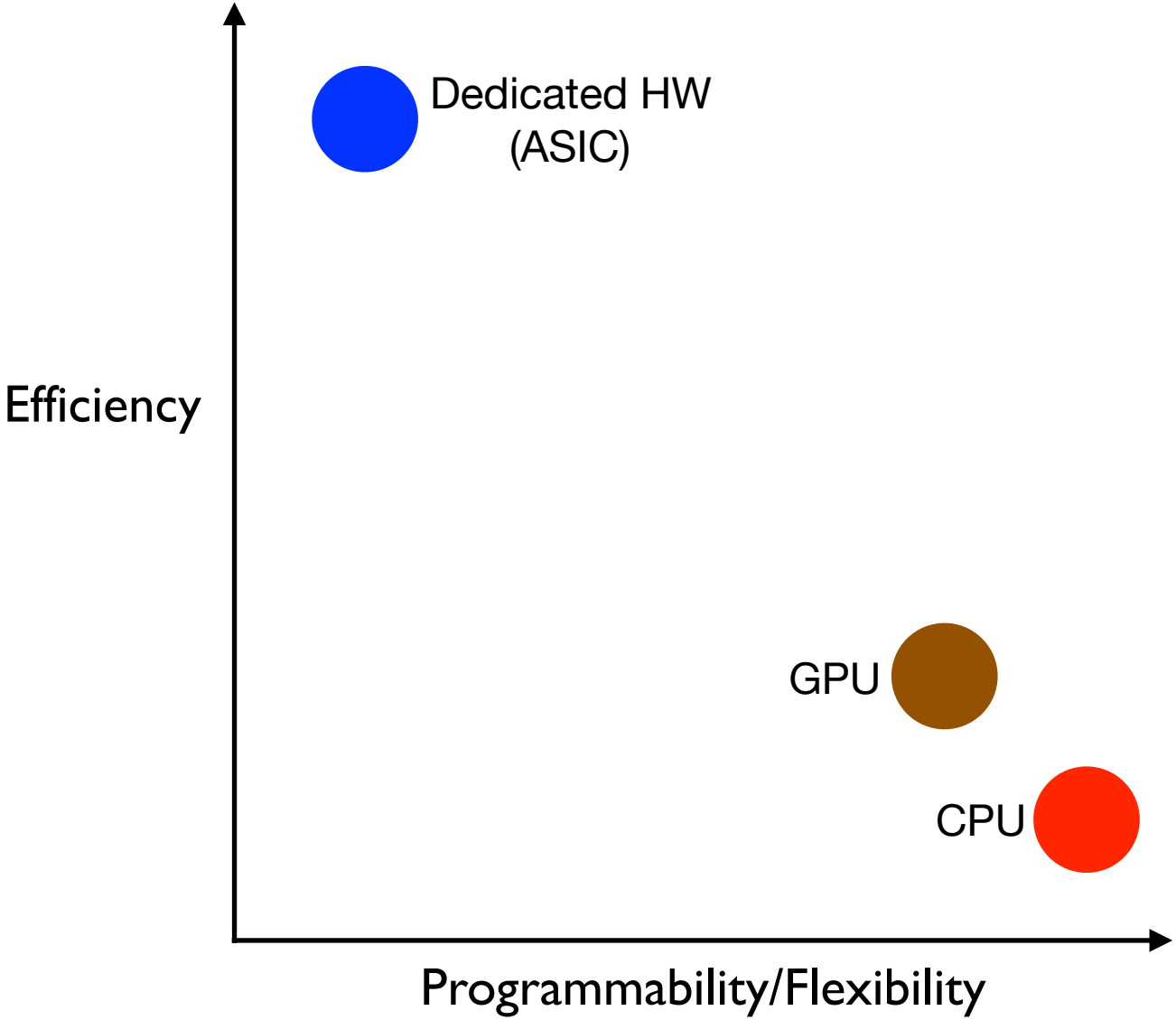
- Instead of building general-purpose processors that can do everything, but inefficiently, let's build specialized processors that can only do limited things, but extremely efficiently.
- Extreme example: Application-Specific Integrated Circuit (ASIC)
- Imagine the entire processor is an FP adder with 2 registers
  - Instruction delivery: none. Instructions are implicit (add).
  - Data feeding: simple.
  - Control: little: clock the adder, simple pipeline, etc.
  - Execution: will be the major power consumer.
  - Flexibility/programmability: very limited: addition only, nothing else!

Entering the Era of Specialization

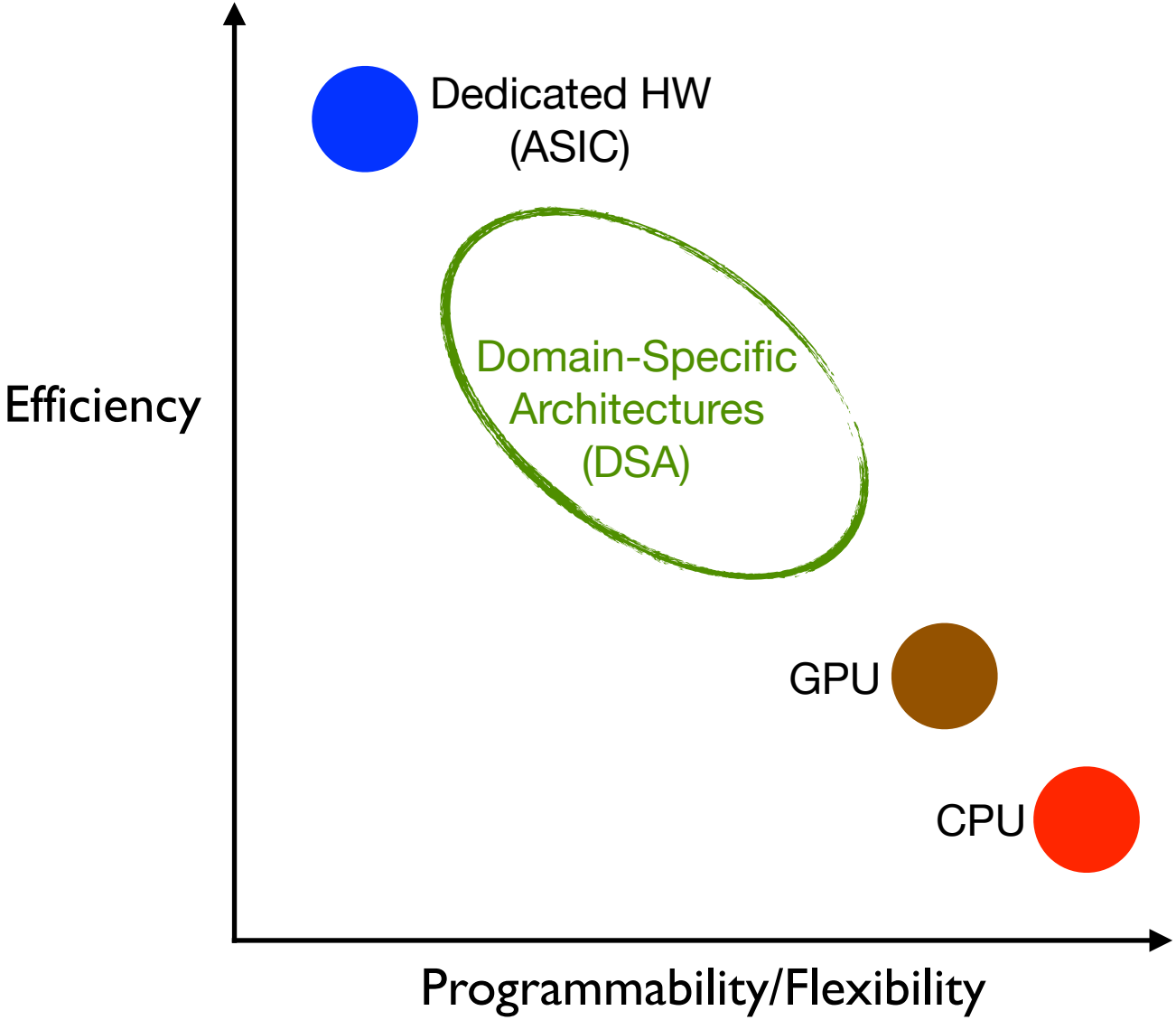
- Another example: Graphics Processing Unit (GPU)
  - SIMT: Single instruction multiple thread
  - SIMT amortizes control/instruction delivery overhead
  - Data feeding is still very complex
  - More efficient execution for applications that are massively parallel



Entering the Era of Specialization



Entering the Era of Specialization



Example: Video (De)-Compression

Numbers credit: Kayvon Fatahalian

Example: Video (De)-Compression

30 second video: 1920 x 1080, @ 30fps
8-bits per color → 24 bits/pixel → 6.2MB/frame (6.2 MB*30 sec*30 fps = 5.2 GB)
Actual H.264 video file size: 65.4 MB (80-to-1 compression ratio)
Compression/encoding performed in real time on iPhone 5s

Numbers credit: Kayvon Fatahalian

Example: Video (De)-Compression

- Main Idea: Exploiting Redundancies

Slides credit: Kayvon Fatahalian

Example: Video (De)-Compression

- Main Idea: Exploiting Redundancies
- Spatial redundancy: value of pixels in neighboring regions of a frame are good predictor of values for other pixels in the frame,

Slides credit: Kayvon Fatahalian

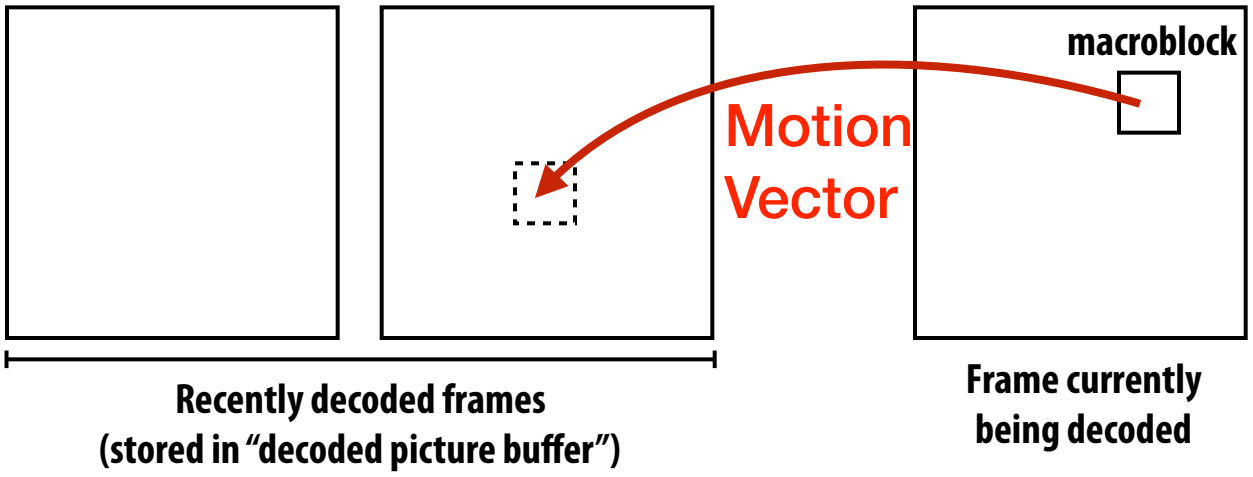
Example: Video (De)-Compression

- Main Idea: Exploiting Redundancies
- Spatial redundancy: value of pixels in neighboring regions of a frame are good predictor of values for other pixels in the frame,
- Temporal redundancy: pixels from nearby frames in time are a good predictor for the current frame's pixels (e.g., objects move slightly on screen between frames)

Slides credit: Kayvon Fatahalian

Example: Video (De)-Compression

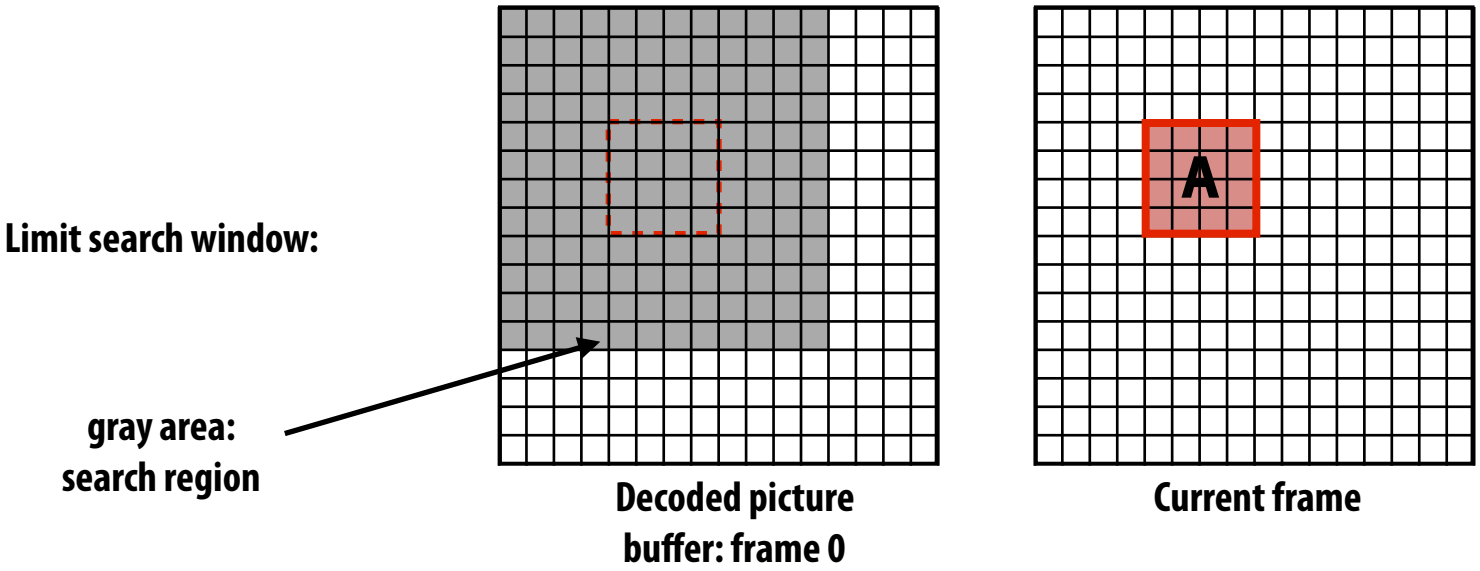
- Main Idea: Exploiting Redundancies
- **Spatial redundancy:** value of pixels in neighboring regions of a frame are good predictor of values for other pixels in the frame,
- **Temporal redundancy:** pixels from nearby frames in time are a good predictor for the current frame's pixels (e.g., objects move slightly on screen between frames)



Slides credit: Kayvon Fatahalian

Example: Video (De)-Compression

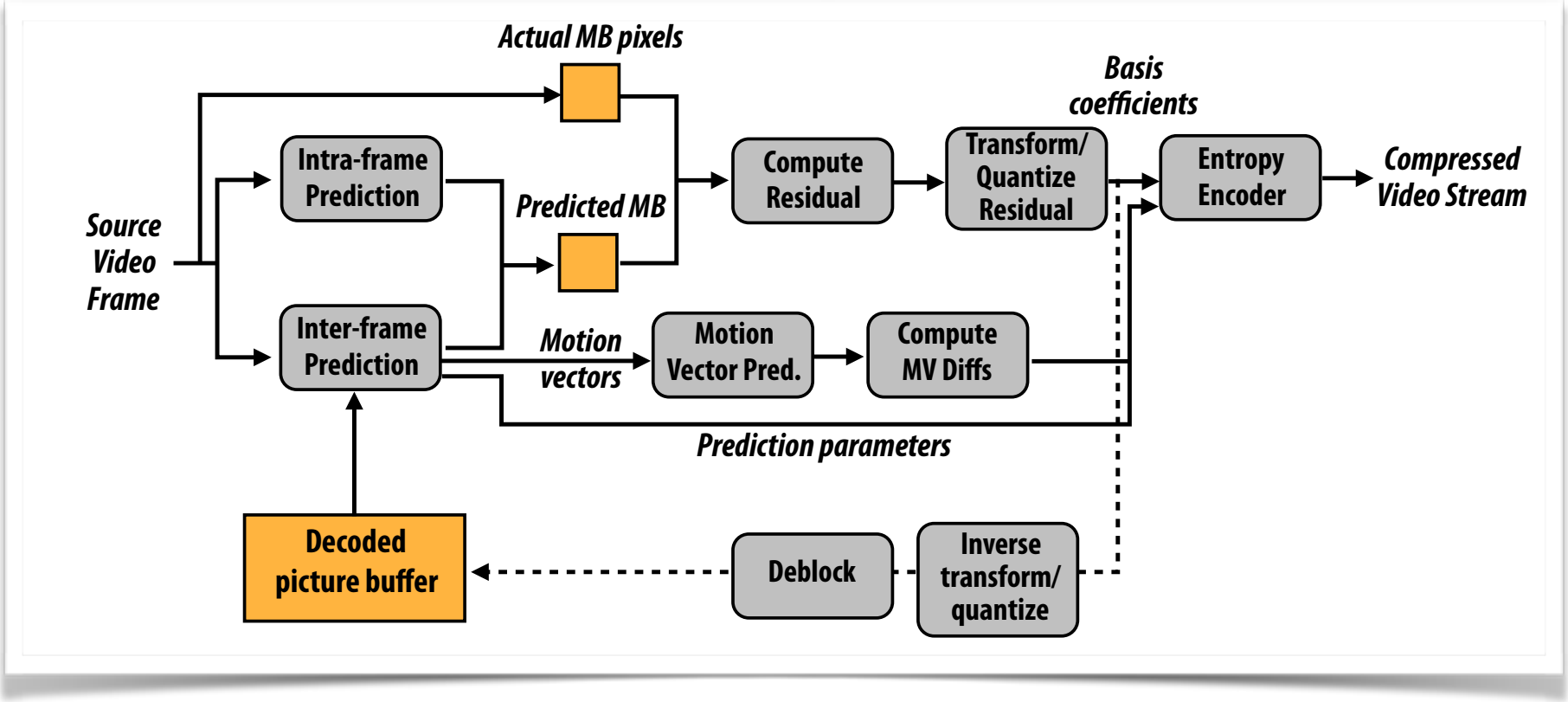
- Key Operation: Motion Estimation
- Remember, must execute motion estimation in real-time for HD video (1920x1080), on a low-power smartphone.



Slides credit: Kayvon Fatahalian

Example: Video (De)-Compression

- Specialized hardware for H.264 video codec is virtually in every (mobile) platforms. Very efficient for video (de)-compression, but can't do much else.



Slides credit: Kayvon Fatahalian

Example: Computational Photography

- High-quality cameras are key product differentiators for today's consumer smartphones.

Example: Computational Photography

- High-quality cameras are key product differentiators for today's consumer smartphones.
- But smartphone cameras aren't as good as high-end digital single-lens reflex cameras.
  - Aperture, focal length, pixels, dynamic range, sensors, resolution, etc.

Example: Computational Photography

- High-quality cameras are key product differentiators for today's consumer smartphones.
- But smartphone cameras aren't as good as high-end digital single-lens reflex cameras.
  - Aperture, focal length, pixels, dynamic range, sensors, resolution, etc.
- How to achieve DSLR-like quality in a smartphone form factor?

Example: Computational Photography

- High-quality cameras are key product differentiators for today's consumer smartphones.
- But smartphone cameras aren't as good as high-end digital single-lens reflex cameras.
  - Aperture, focal length, pixels, dynamic range, sensors, resolution, etc.
- How to achieve DSLR-like quality in a smartphone form factor?
- Computational photography: digitally post-processing images
  - E.g., High-dynamic range (HDR)
  - Used to be just in desktop photo editing software (e.g., PhotoShop), but now in high-end smartphones

Short Exposure

Long Exposure

Short Exposure

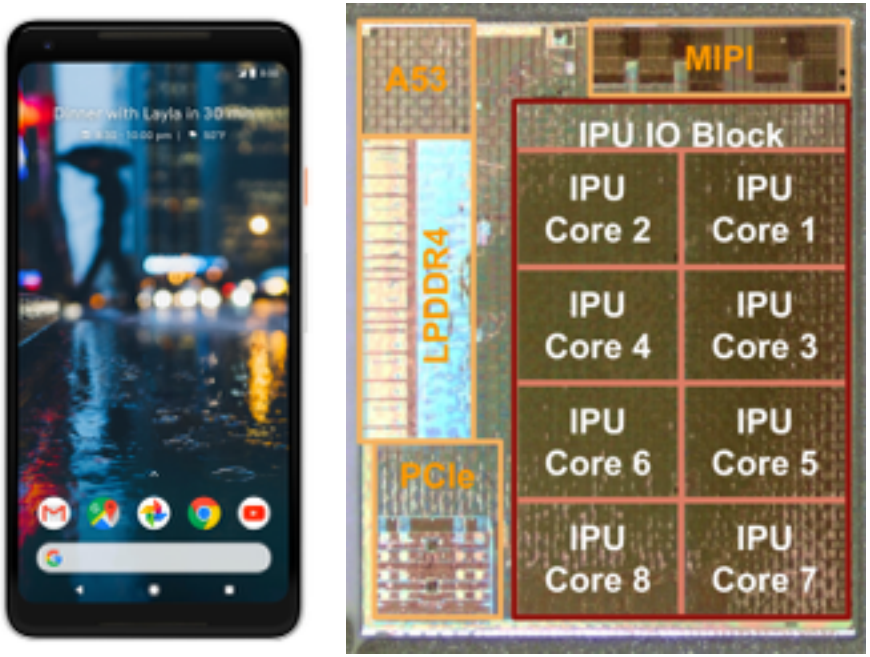
Long Exposure


HYATT

Example: Computational Photography

- ISP (Image Signal Processor): Specialized processors for imaging and computational photography algorithms.
- Fast, energy-efficient, but only for imaging

Google Pixel 2

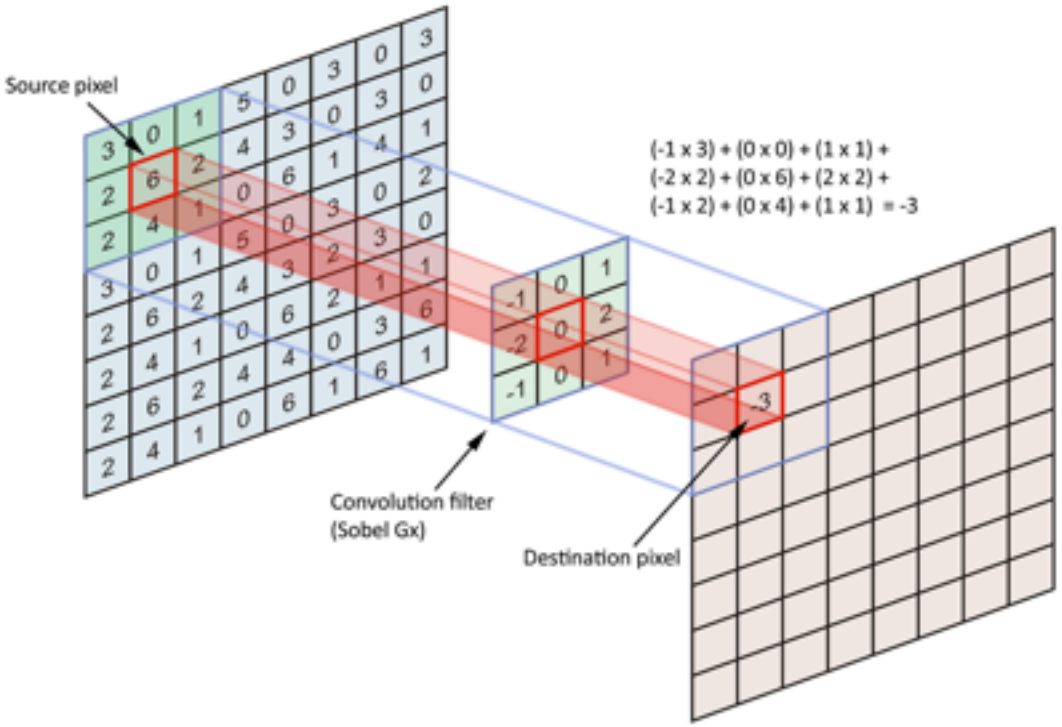


Example: Machine Learning

- Machine learning (e.g., Neural Networks) takes over the world
- Computer vision is the poster-child example
- Natural language processing, Precision medicine, robotics planning, house rent/load prediction, games, etc.

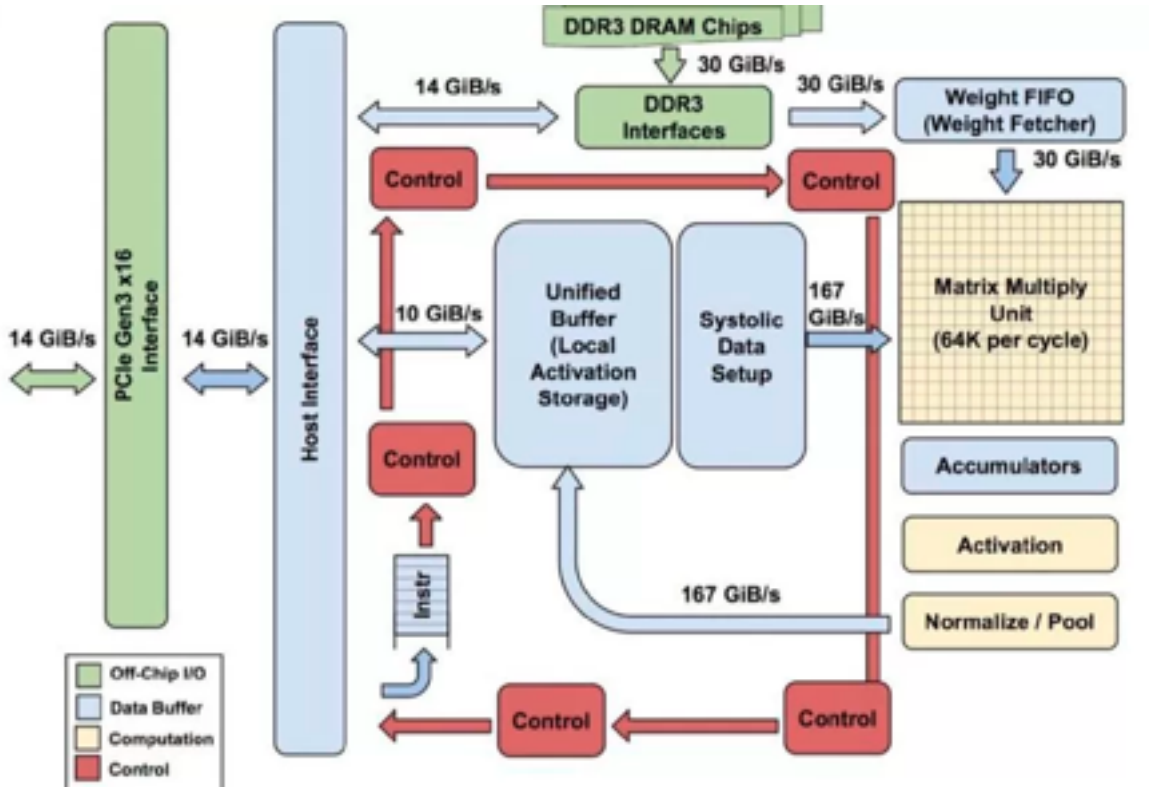
Example: Machine Learning

- Neural networks make heavy use of the convolution operation
  - Convolution can be transformed to matrix-multiplication
  - Convolutional neural networks (CNN) are by far the most popular NN, especially effectively for computer vision tasks

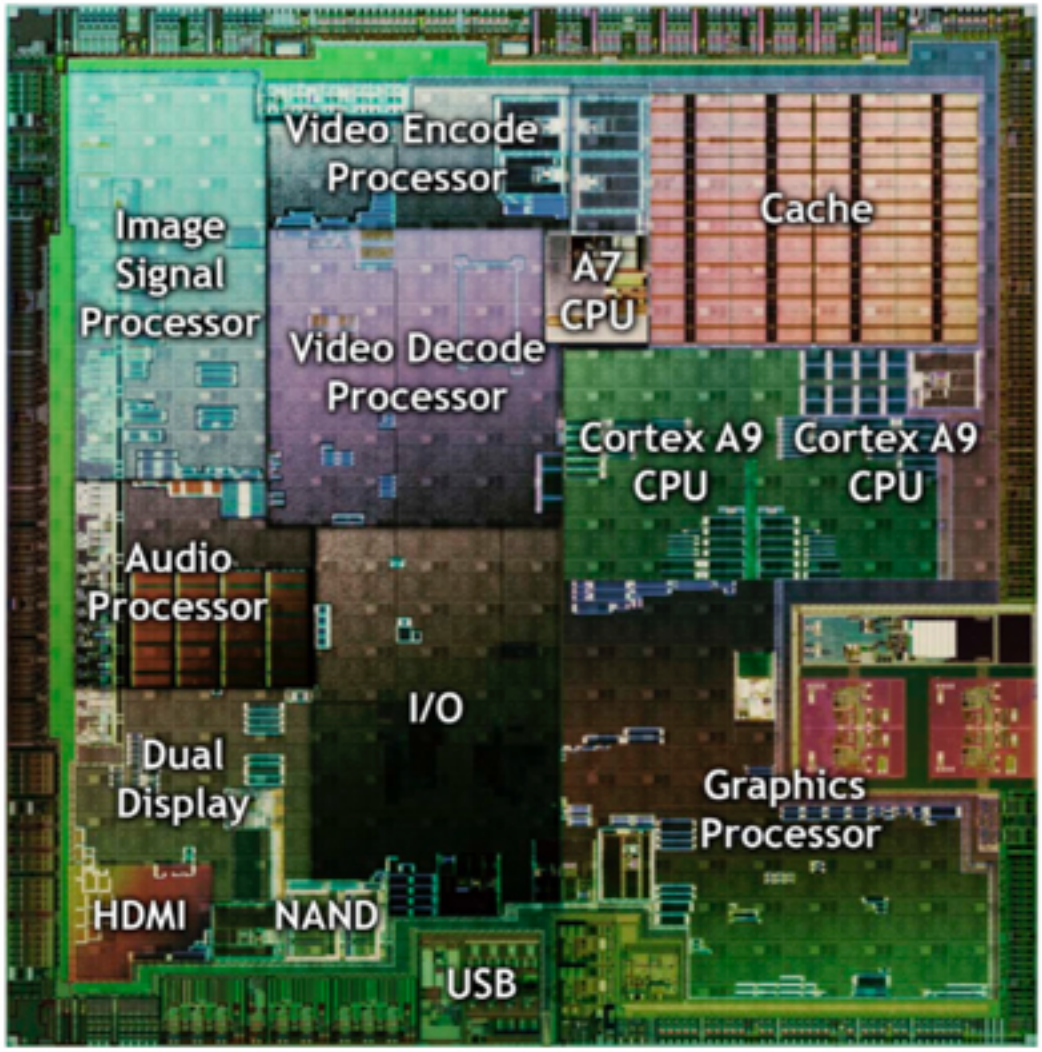


Example: Machine Learning

- Google Tensor Processing Unit (TPU)
  - Specialized processor (i.e., systolic array architecture) for tensor processing (matrix multiply)
  - (Arguably) 30x~80x more power-efficient than GPU



Today's Mobile Processor Chips



The Role of a Computer System Designer

Problem

Algorithm

Program

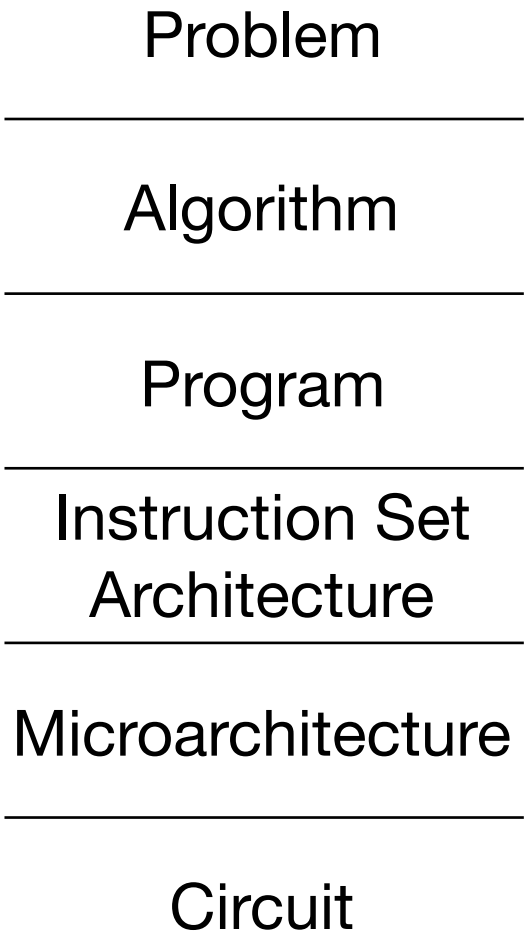
Instruction Set Architecture

Microarchitecture

Circuit

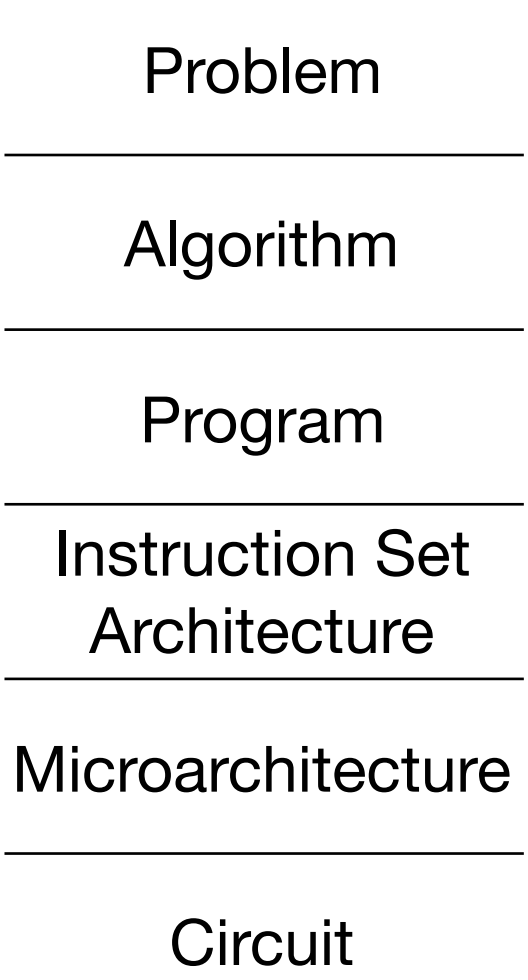
The Role of a Computer System Designer

- Look Up
  - Nature of the problems



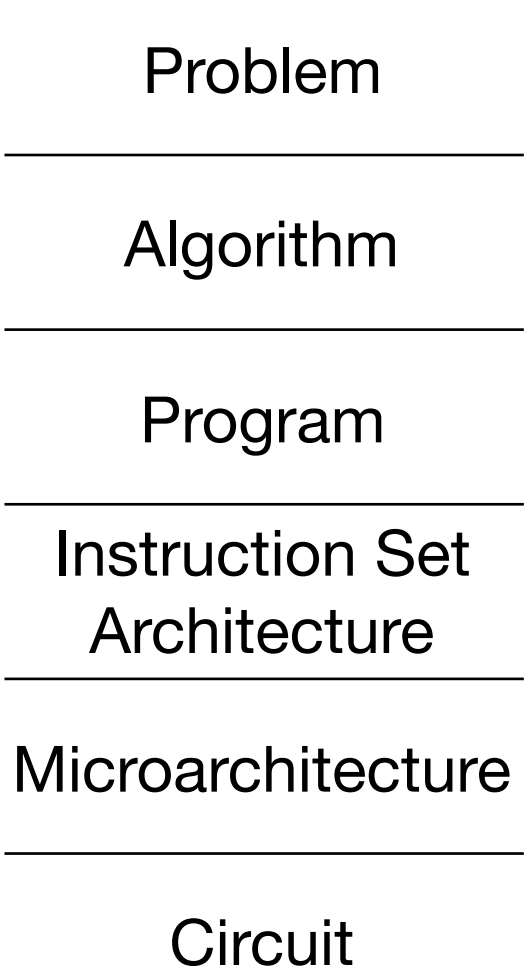
The Role of a Computer System Designer

- Look Up
  - Nature of the problems
- Look Down
  - Nature of the circuit technology and physics



The Role of a Computer System Designer

- Look Up
  - Nature of the problems
- Look Down
  - Nature of the circuit technology and physics
- Look Backward
  - Evaluating old ideas in light of new technologies



The Role of a Computer System Designer

- Look Up
  - Nature of the problems
- Look Down
  - Nature of the circuit technology and physics
- Look Backward
  - Evaluating old ideas in light of new technologies
- Look Forward
  - Listen to dreamers and predict the future