

CSC 252: Computer Organization

Spring 2026: Lecture 25

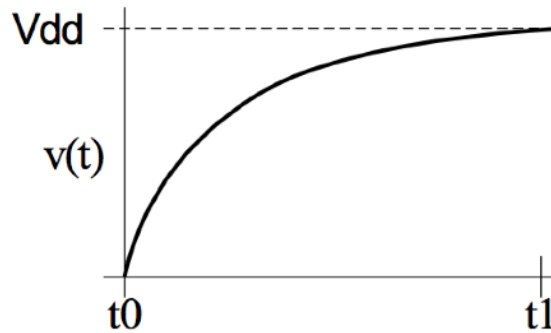
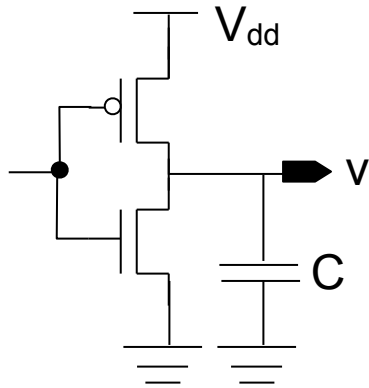
Instructor: Yuhao Zhu

Department of Computer Science
University of Rochester

Today

- Power consumption and dark silicon
- GPU
- Accelerators

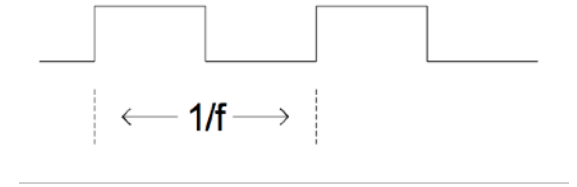
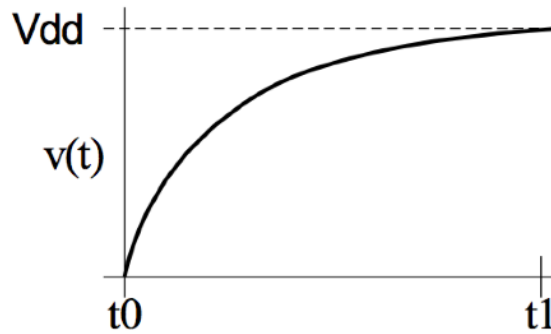
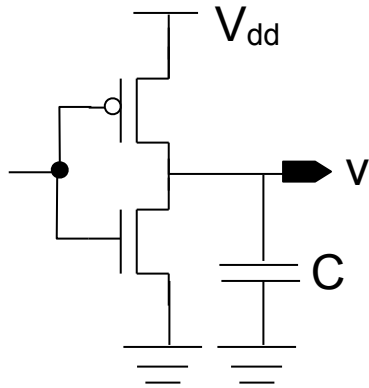
Dynamic Power



Energy dissipated for every transition (0- $\rightarrow$ 1 or 1- $\rightarrow$ 0)

$$\begin{aligned}
 E_{sw} &= \int_{t_0}^{t_1} P(t) dt = \int_{t_0}^{t_1} (V_{dd} - v) \cdot i(t) dt = \int_{t_0}^{t_1} (V_{dd} - v) \cdot c (dv/dt) dt = \\
 &= cV_{dd} \int_{t_0}^{t_1} dv - c \int_{t_0}^{t_1} v \cdot dv = cV_{dd}^2 - 1/2cV_{dd}^2 = \boxed{1/2cV_{dd}^2}
 \end{aligned}$$

Dynamic Power



Average dynamic power of a transistor:

$$P = \alpha \cdot (E / T) = \alpha \cdot E f = \frac{1}{2} \alpha C V_{dd}^2 f$$

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

Dynamic Power

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

- Increasing f requires V to be increased proportionally
 - Intuitively: a higher frequency means a shorter cycle time, which means the critical path of your processor needs to be shorter, which requires faster transistors, which you get by increasing the voltage
 - “Overclocking” just increases the clock speed without increasing voltage => machine might crash (cycle time shorter than the critical path delay)
 - Corollary: reducing voltage requires reducing frequency
 - 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.

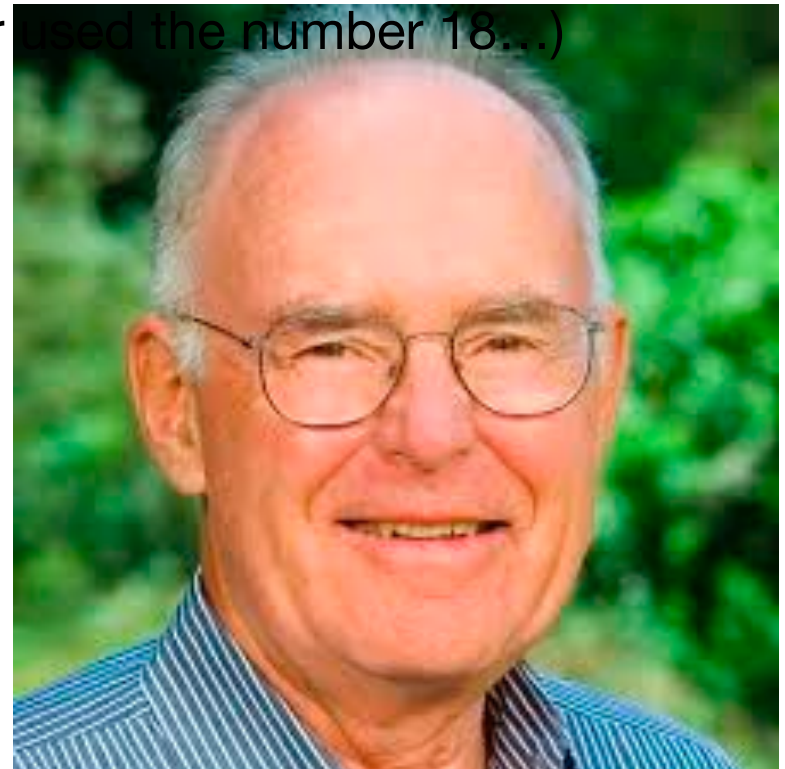
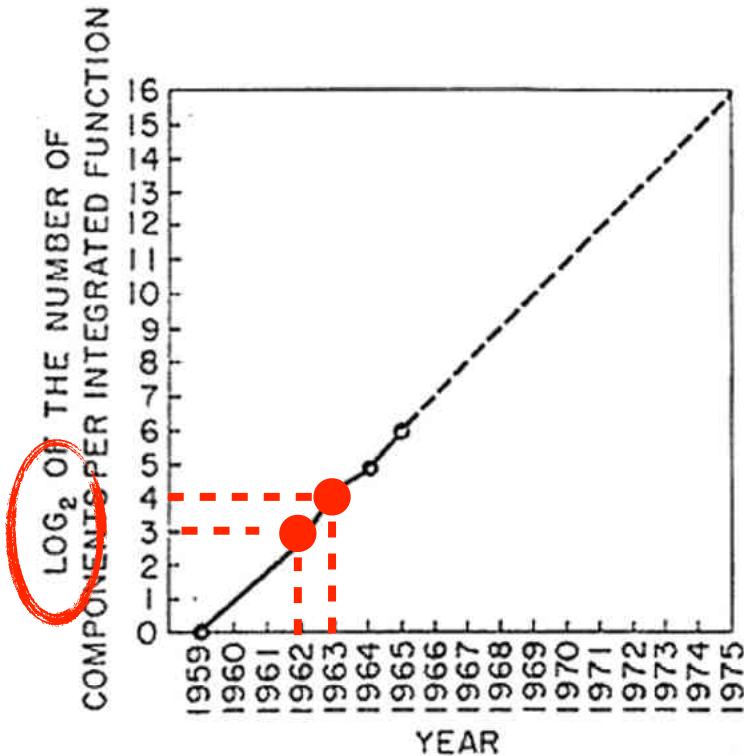
Dynamic Power Favors Parallelisms

$$P = k C f^3$$

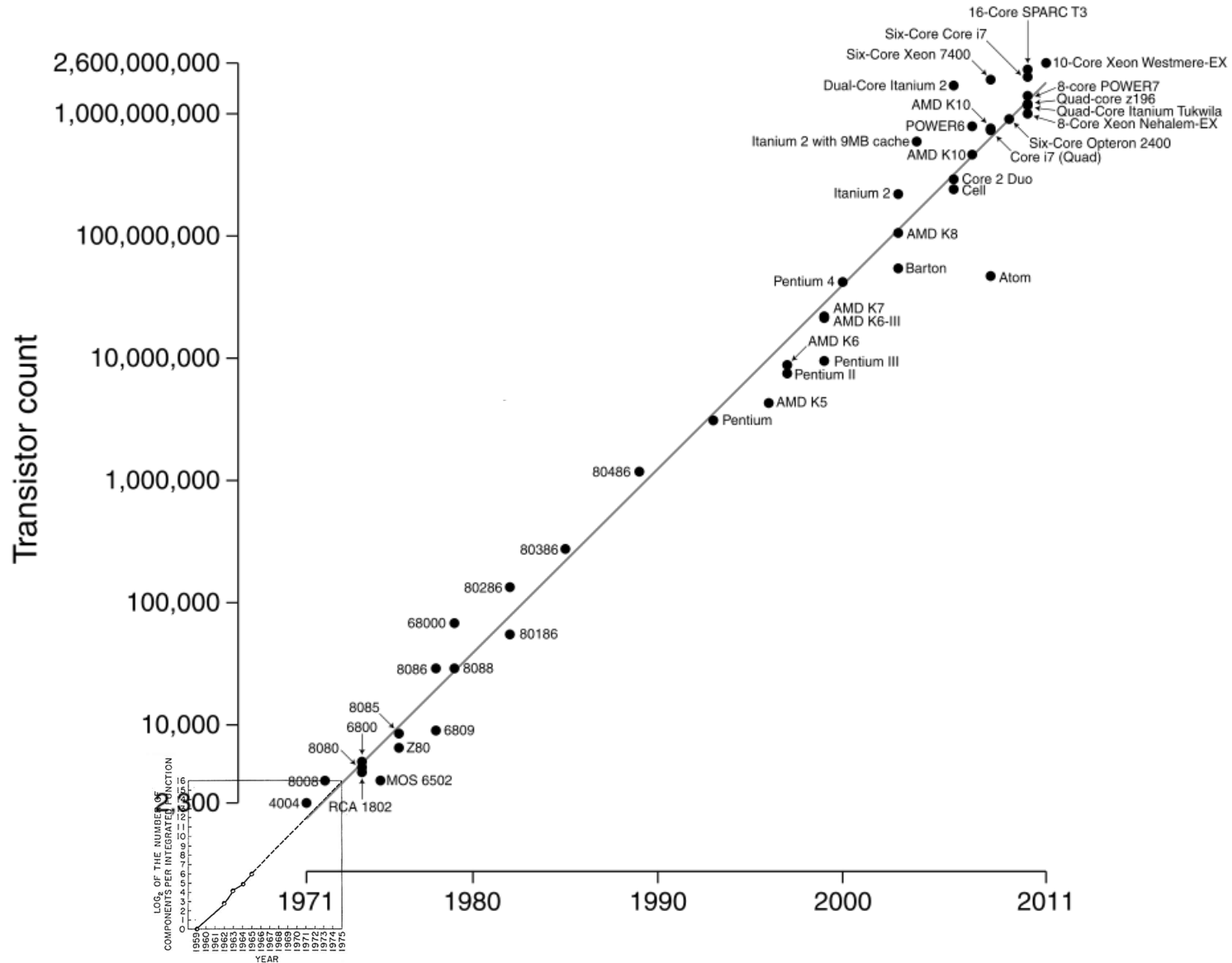
- Dynamic power favors parallel processing over higher clock rate
 - Take a core and replicate it 4 times: 4x speedup & **4x power**
 - Take a core and clock it 4 times faster: 4x speedup but **64x dynamic power!**
- Another way to think about this
 - If a task can be perfectly parallelized by 4 cores,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

Moore's Law

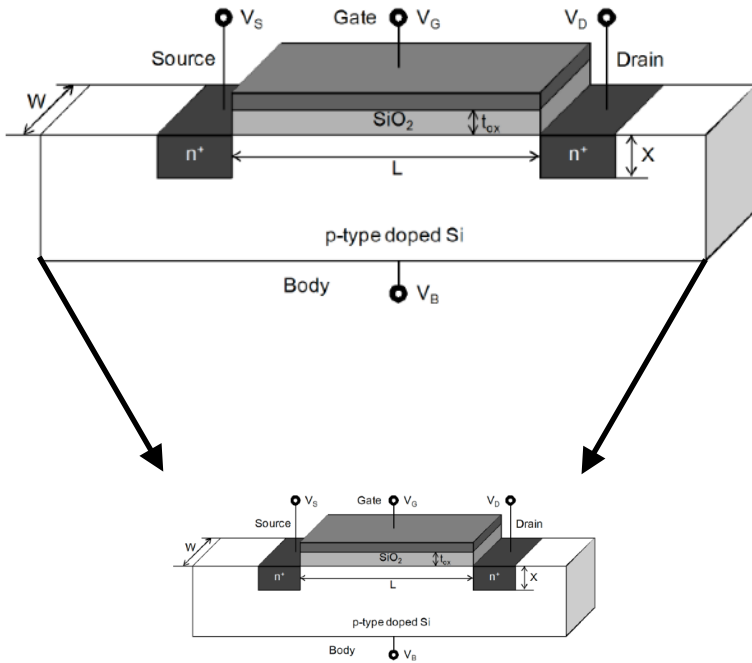
- Gordon Moore in 1965 predicted that the number of transistors doubles every year
- In 1975 he revised the prediction to doubling every 2 years
- Today's widely-known Moore's Law: number of transistors double about every 18 months (Moore never used the number 18...)



Moore's Law



Dennard Scaling



Scale factor $\alpha < 1$
 $\alpha = 0.7 \Rightarrow 2X$ more transistors!

Parameter	Value	Scaled Value
Dopant concentrations	N_a, N_d	$N_a/\alpha, N_d/\alpha$
Dimensions	L, W, T_{ox}	$\alpha L, \alpha W, \alpha T_{ox}$
Field	E	E
Voltage	V	αV
Capacitance	C	αC
Current	I	αI

Transistors/Area	d	d/α^2
------------------	-----	--------------

Propagation time ($\sim CV/I$)	t	αt
Frequency ($1/t$)	f	f/α

Power (CV^2f)	P	$\alpha^2 P$
Power/area (Power density)	P_d	P_d

Implications of Dennard Scaling and Moore's Law

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

2005: End of Dennard Scaling

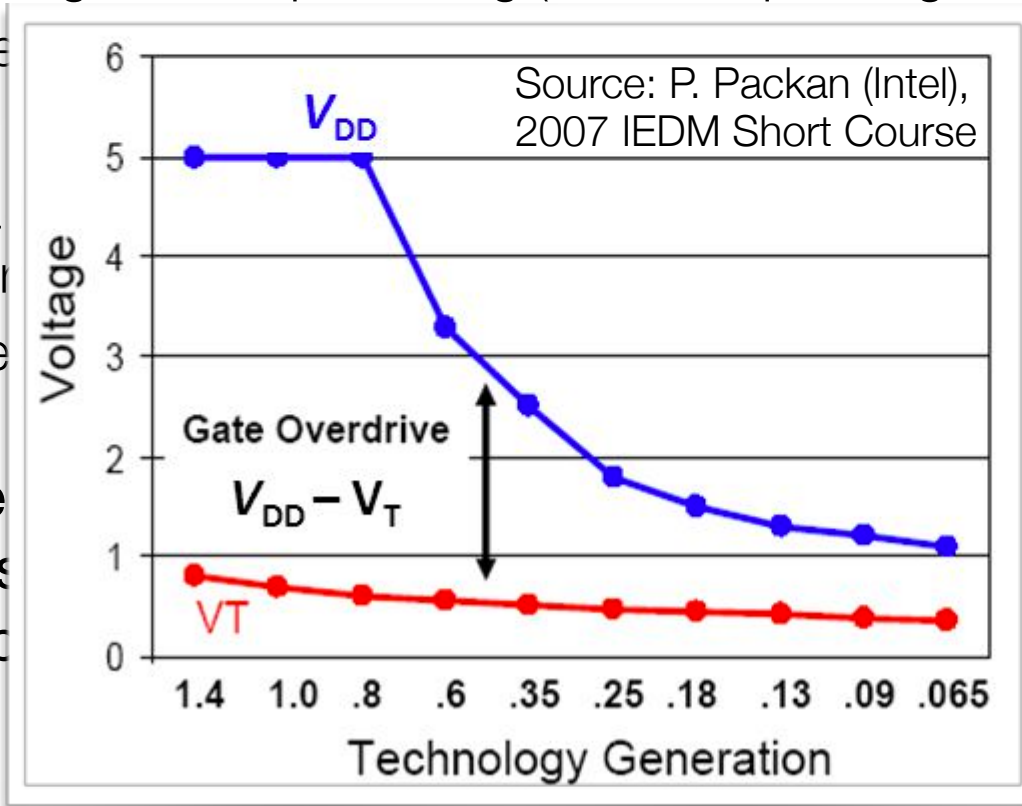
- What Happened?

- Supply voltage V_{DD} stops scaling (Can't drop voltage below ~ 1 V)
- Remember

- Why?

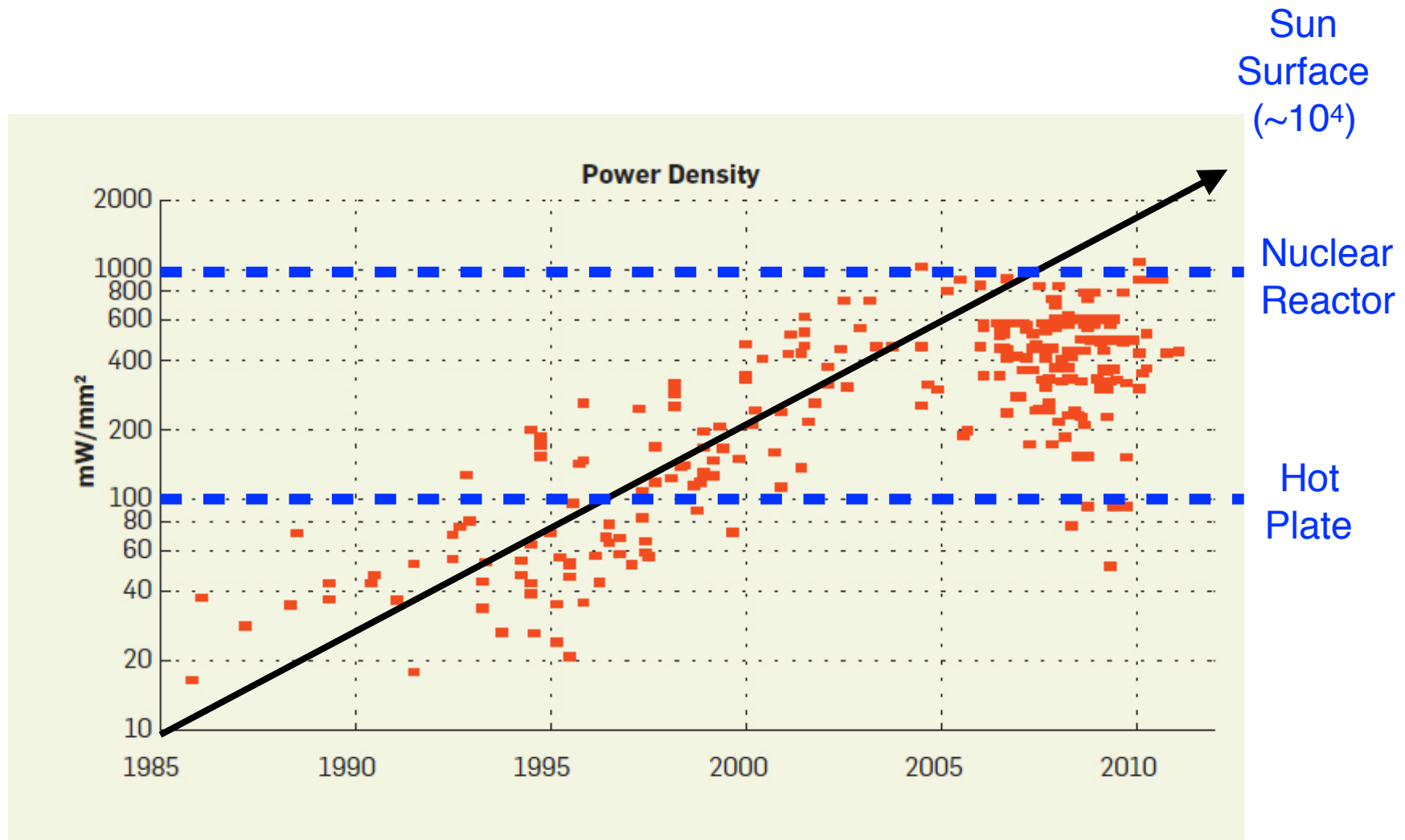
- There is a need to switch a transistor
- V_{th} stopped scaling and becomes a constant

- The demise of Dennard Scaling (power consumption was no longer staying stable)



the need to
variation
g
density
ner than

2005: End of Dennard Scaling



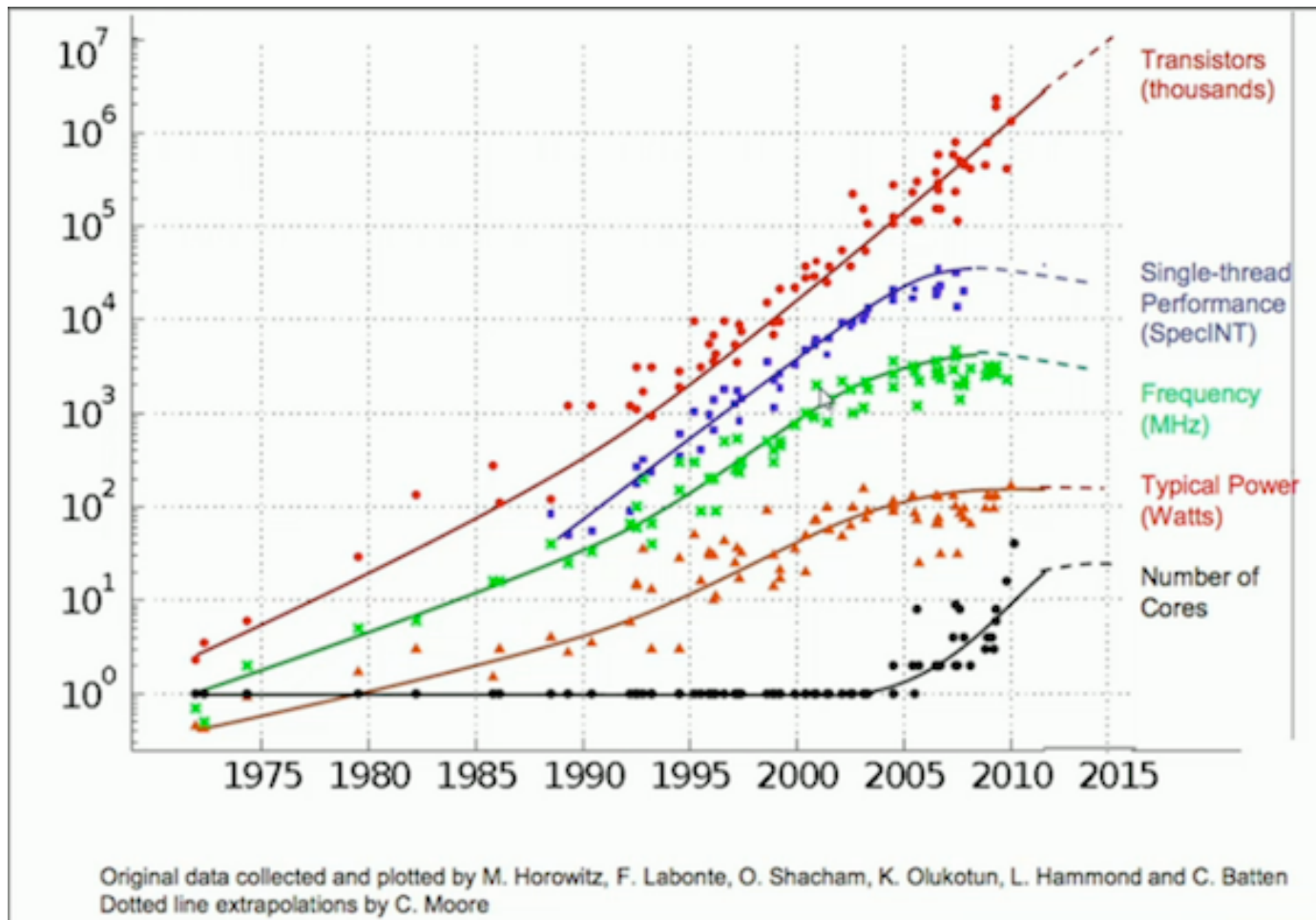
Dark Silicon

n. [därk, sɪl'ɪ-kən, -kɒn']

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

2005: End of Dennard Scaling

- Initial response has been to lower frequency and increase cores / chip
- There is a limit to core scaling. Why?



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 [Computer Systems Organization] General — System architectures

General Terms: Design, Measurement, Performance

Keywords: Dark Silicon, Modeling, Power, Technology Scaling, Multicore

1. INTRODUCTION

Moore's Law [24] (the doubling of *transistors* on chip every 18 months) has been a fundamental driver of computing. For the past three decades, through device, circuit, microarchitecture, architec-

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 and power limits at each technology node. This paper considers all those factors together, projecting upper-bound performance achievable through multicore scaling, and measuring the effects of *non-ideal* device scaling, including the percentage of “dark silicon” (transistor under-utilization) on future multicore chips. Additional projections include best core organization, best chip-level topology, and optimal number of cores.

We consider technology scaling projections, single-core design scaling, multicore design choices, actual application behavior, and microarchitectural features together. Previous studies have also analyzed these features in various combinations, but not all together [8, 9, 10, 15, 16, 21, 23, 28, 29]. This study builds and

2007: A Revolutionary New Computer





No Moore's Law for batteries

Fred Schlachter¹

American Physical Society, Washington, DC 20045

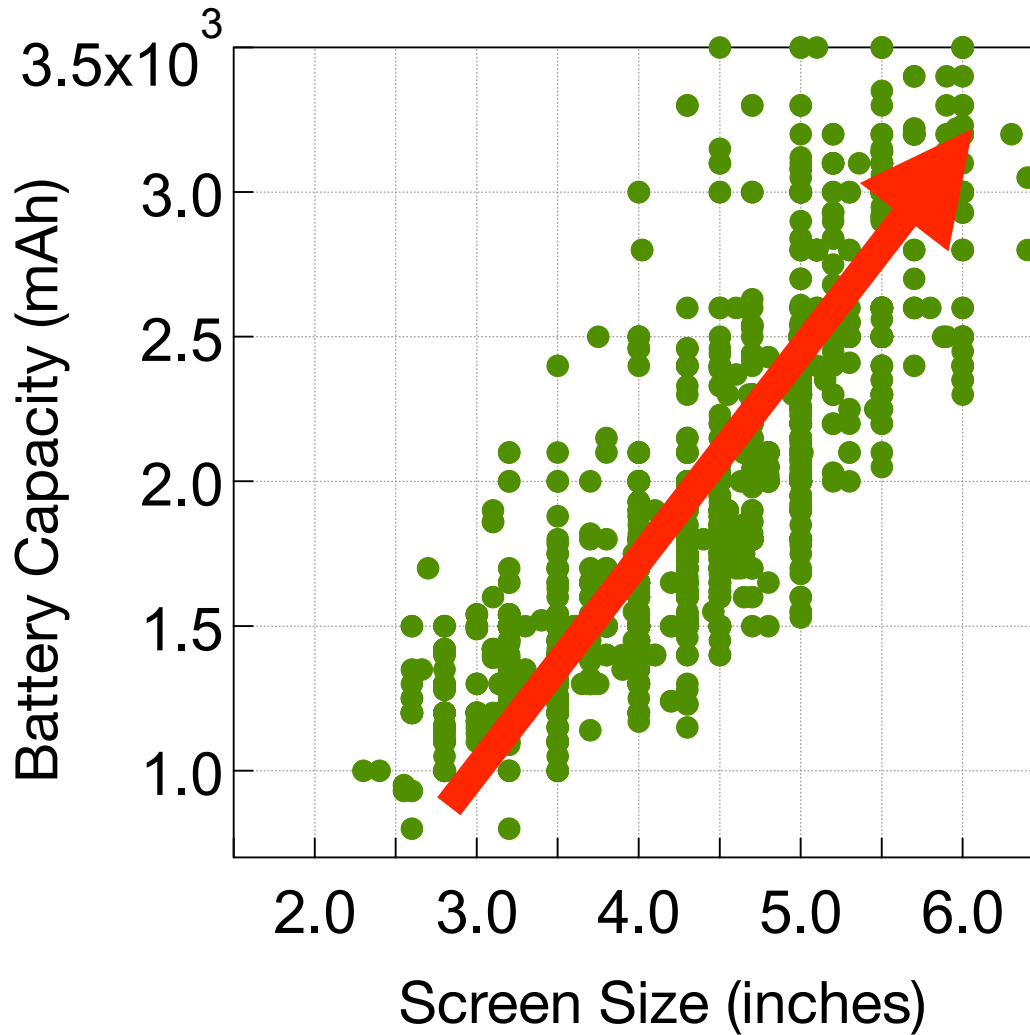
The public has become accustomed to rapid progress in mobile phone technology, computers, and access to information; tablet computers, smart phones, and other powerful new devices are familiar to most people on the planet.

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 co-founder 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



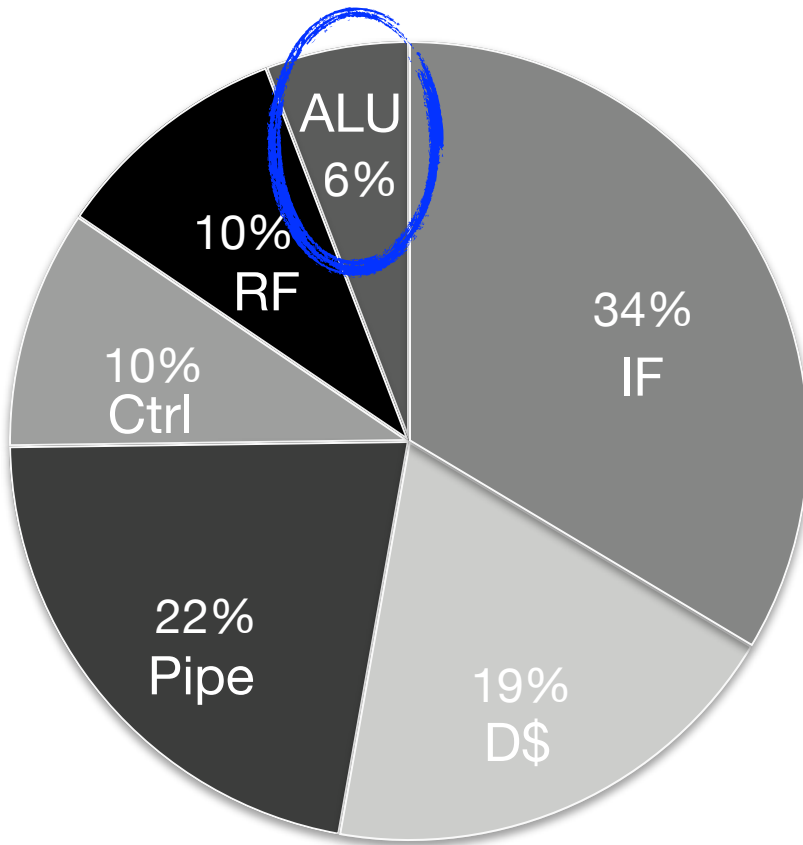
600 smartphone from 2006 to 2014 on <http://www.gsmarena.com/makers.php3>

“Improving” Energy Capacity



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

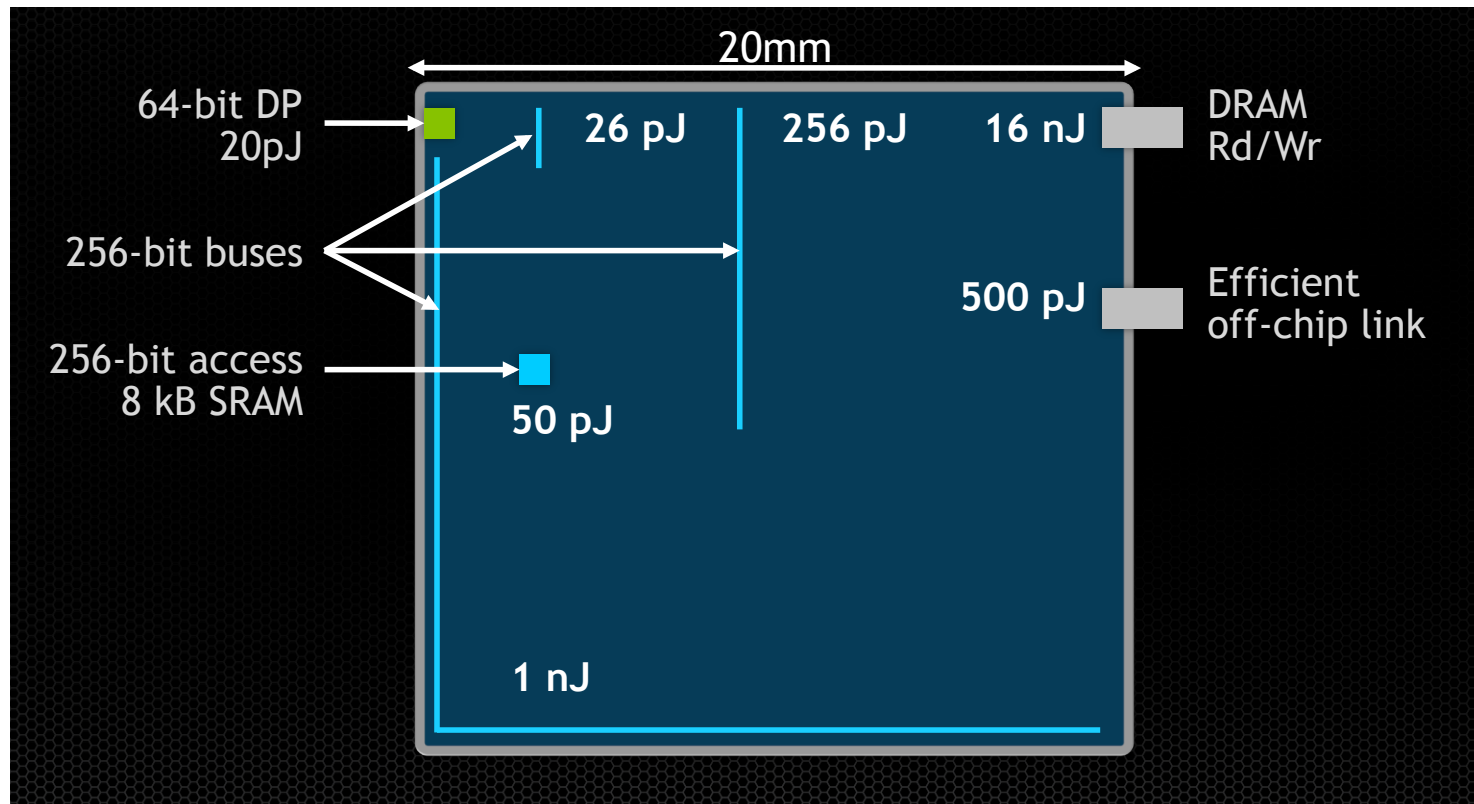
RF: Register file

ALU: Functional units

Doing Actual Work

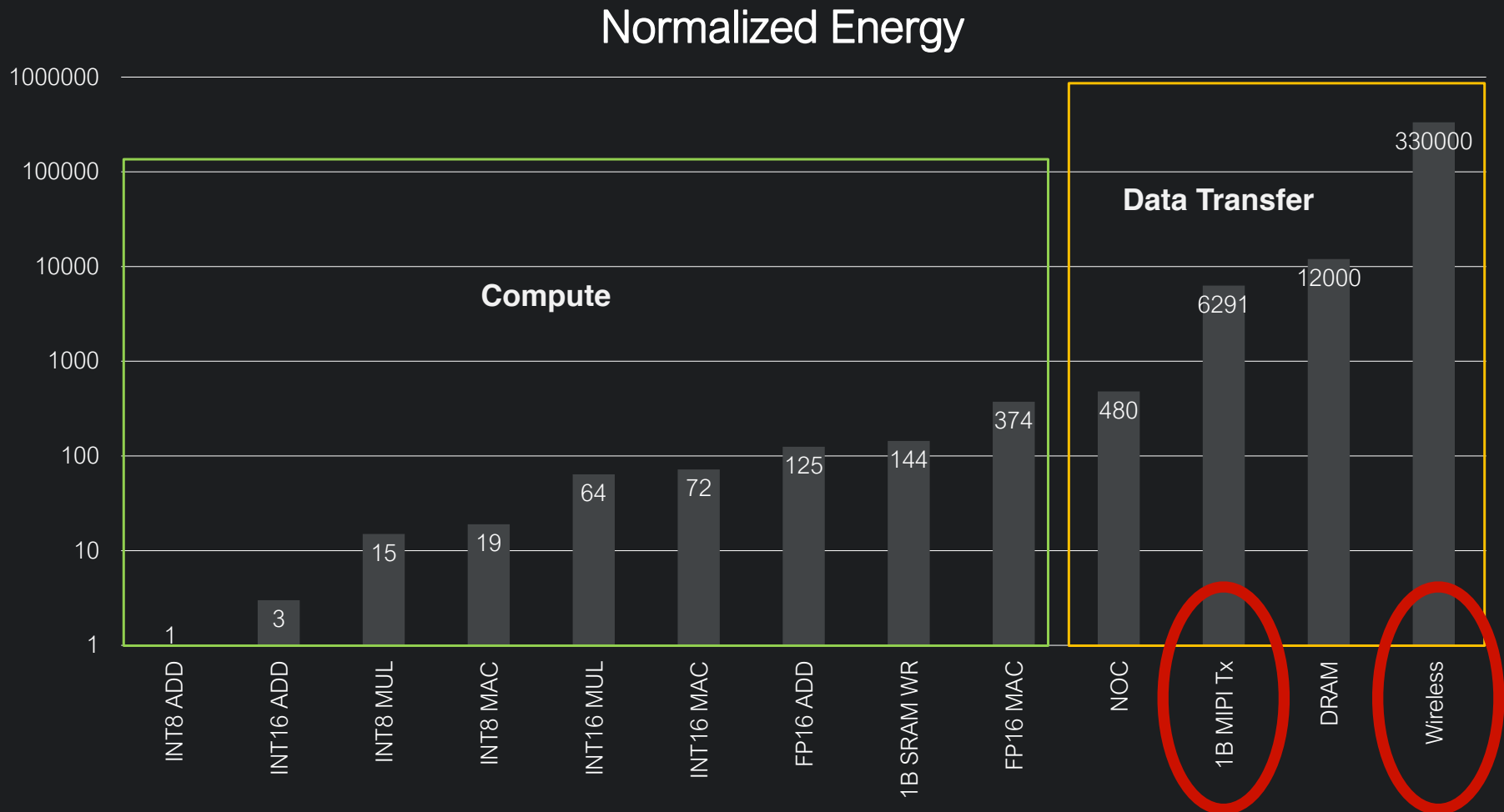
Computation vs. Data Movement

Data movement energy $\gg$ computation energy



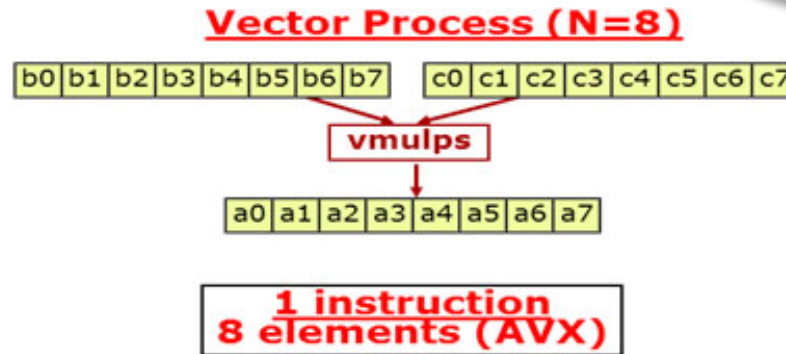
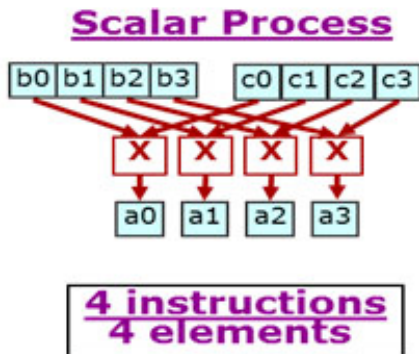
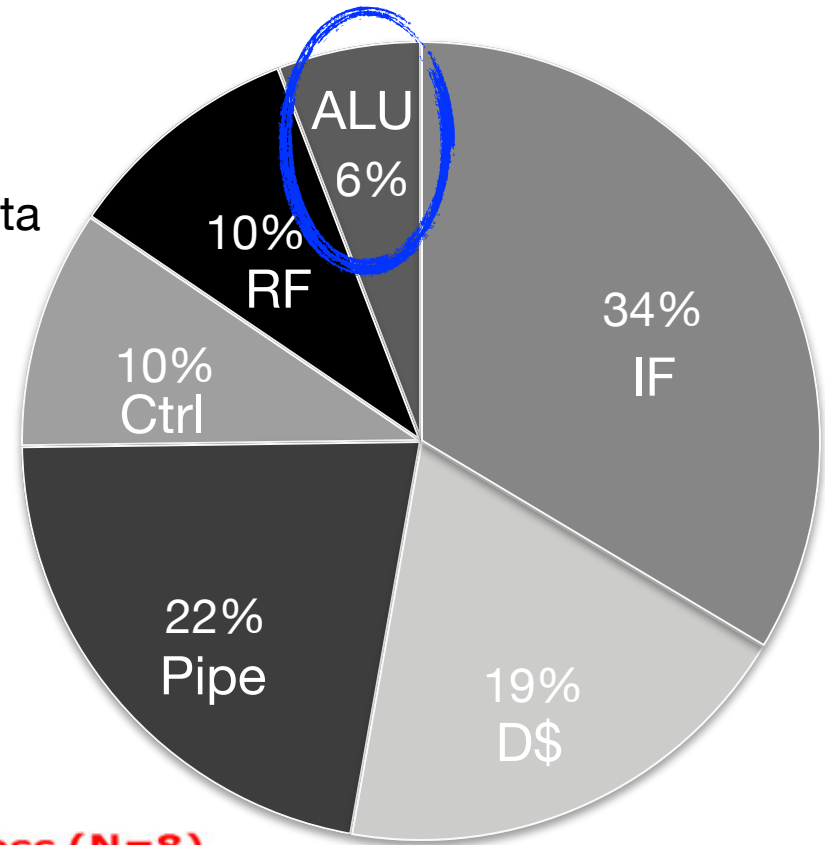
Computation vs. Data Movement

Data movement energy >> computation energy



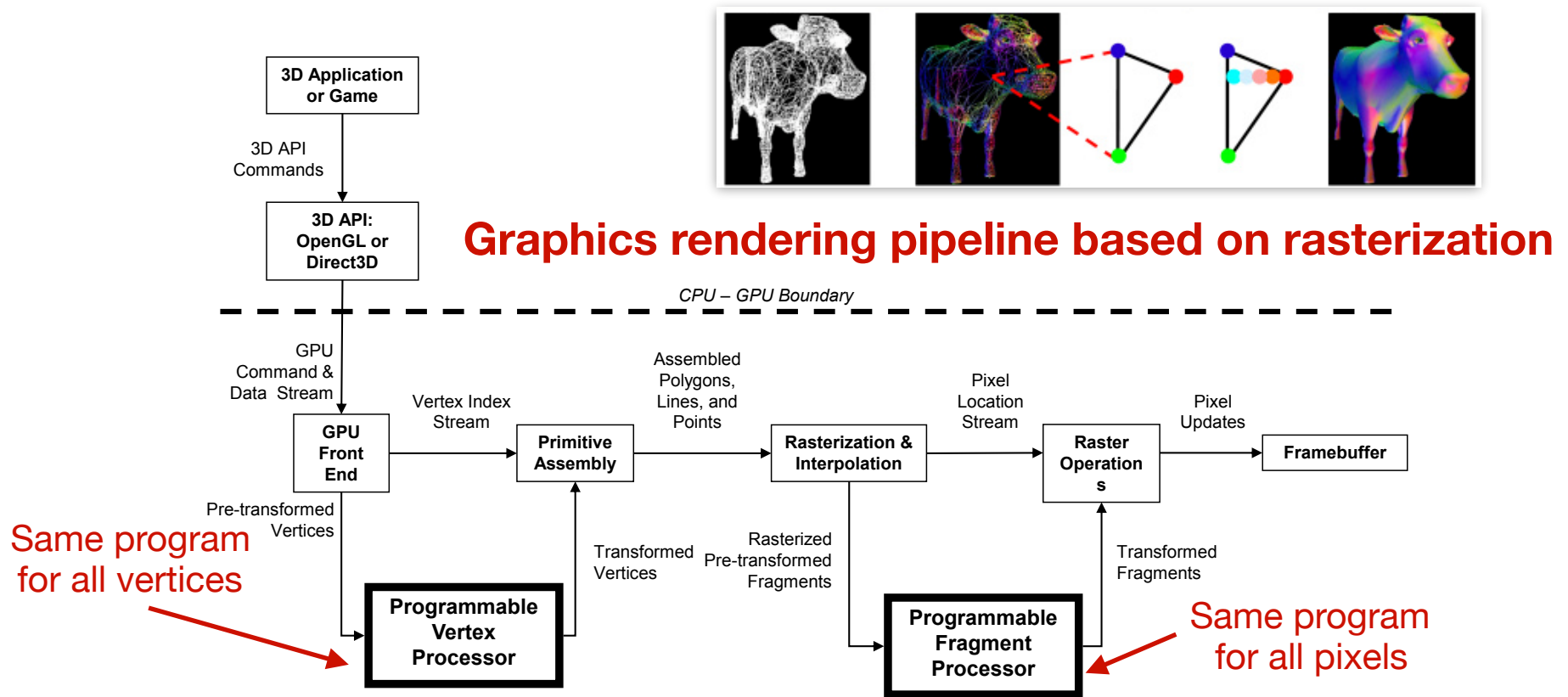
SIMD

- Single Instruction (operating on) Multiple Data
- Amortizing the cost of instruction delivery/control across many execution units (even cores).
- Almost all modern ISAs provide such instructions:
 - x86: MMX/SSE/AVX
 - Arm: Neon



Graphics Processing Units/GPUs (SIMT)

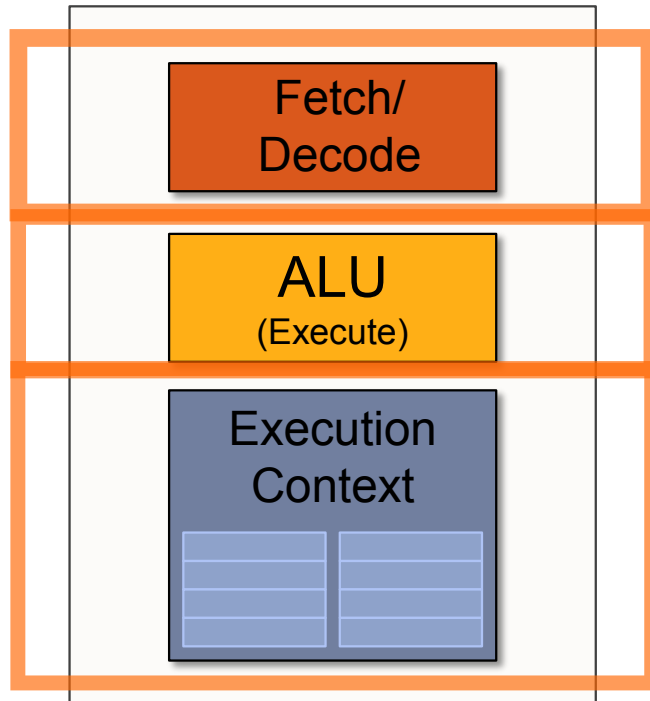
- Designed for graphics rendering, which is massively parallel.



© David Kirk/NVIDIA and
Wen-mei W. Hwu, 2007
ECE 498AL, University of Illinois,
Urbana-Champaign

Computer Architecture

Execute shader

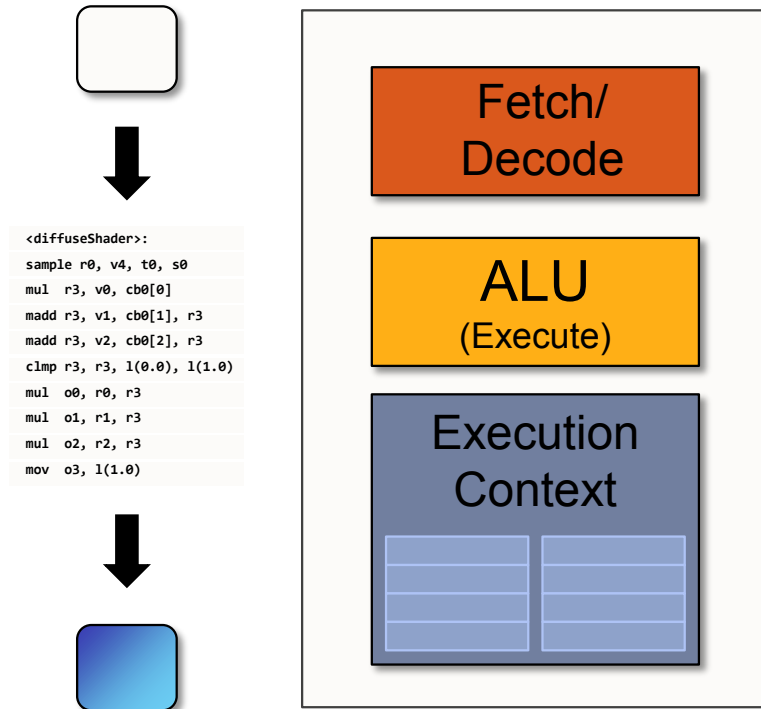


```
<diffuseShader>:
sample r0, v4, t0, s0
mul  r3, v0, cb0[0]
madd r3, v1, cb0[1], r3
madd r3, v2, cb0[2], r3
clmp r3, r3, l(0.0), l(1.0)
mul  o0, r0, r3
mul  o1, r1, r3
mul  o2, r2, r3
mov  o3, l(1.0)
```

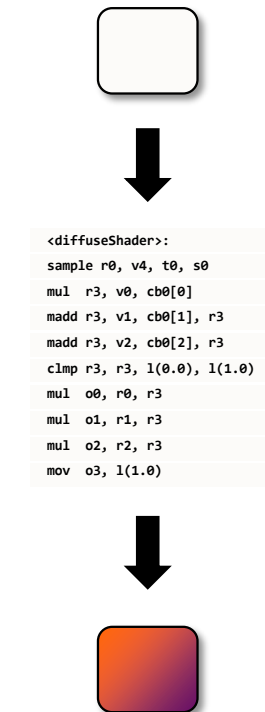


Two cores (two fragments in parallel)

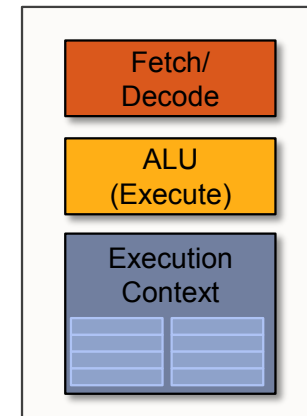
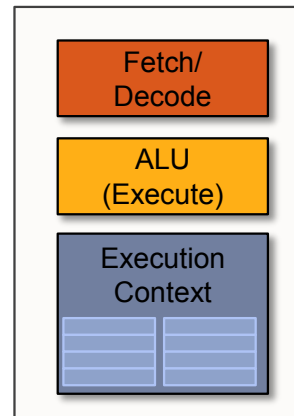
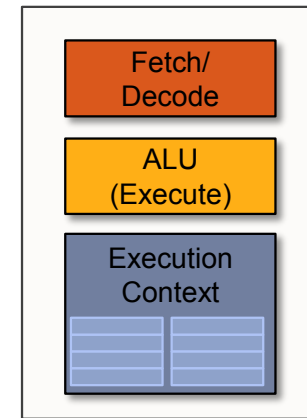
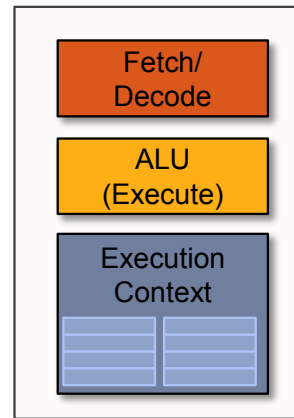
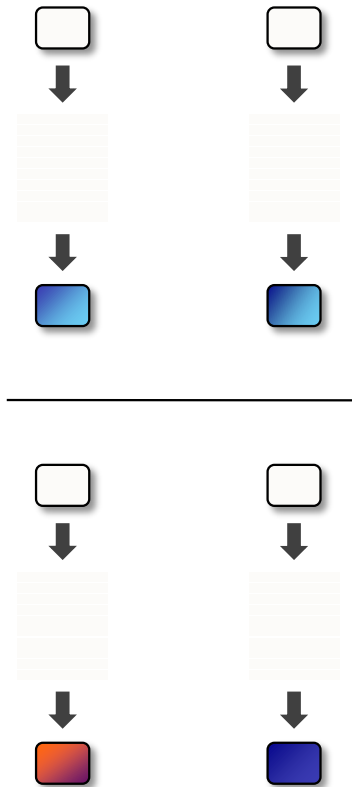
fragment 1



fragment 2



Four cores (four fragments in parallel)



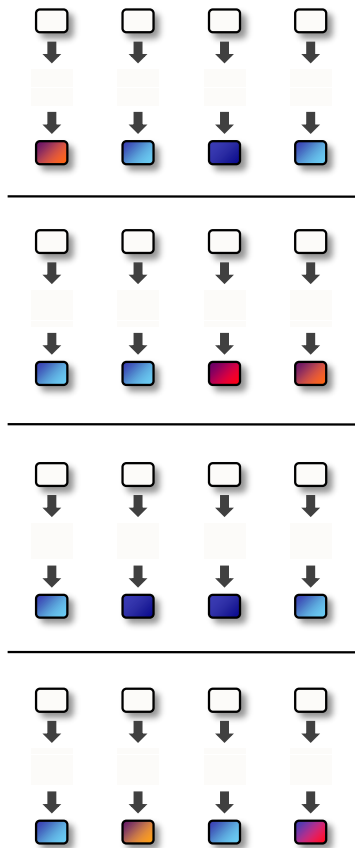
Sixteen cores (sixteen fragments in parallel)



16 cores = 16 simultaneous instruction streams



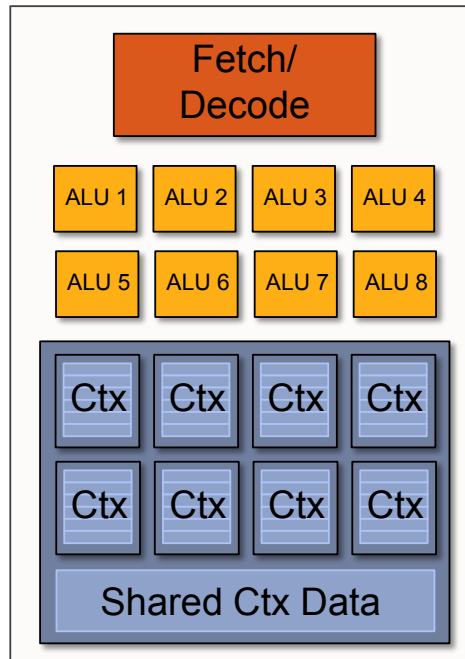
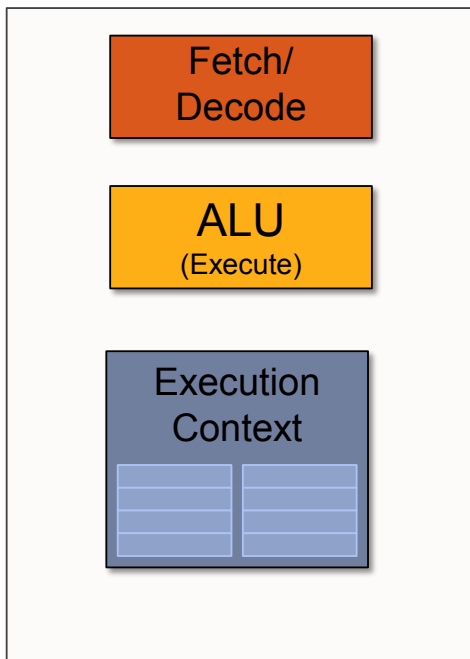
Instruction stream coherence



But... many fragments should be able to share an instruction stream!

```
<diffuseShader>:
sample r0, v4, t0, s0
mul  r3, v0, cb0[0]
madd r3, v1, cb0[1], r3
madd r3, v2, cb0[2], r3
clmp r3, r3, 1(0.0), 1(1.0)
mul  o0, r0, r3
mul  o1, r1, r3
mul  o2, r2, r3
mov  o3, 1(1.0)
```



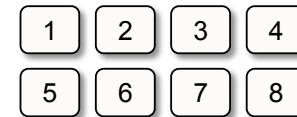
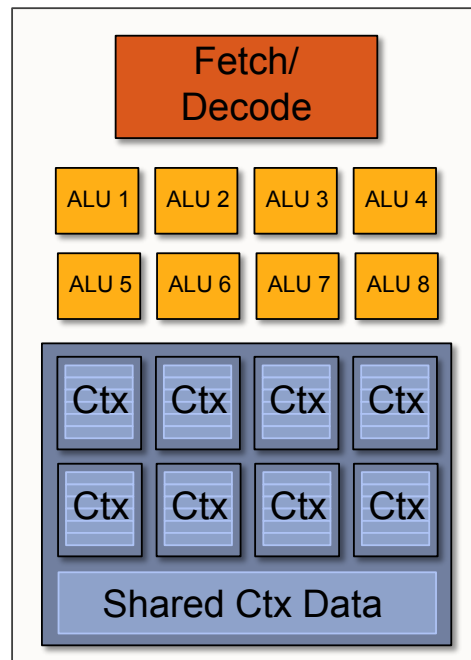


Amortize cost/complexity of managing an instruction stream across many ALUs

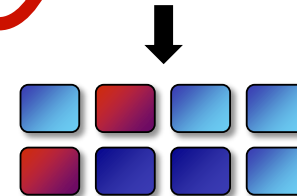
SIMD processing



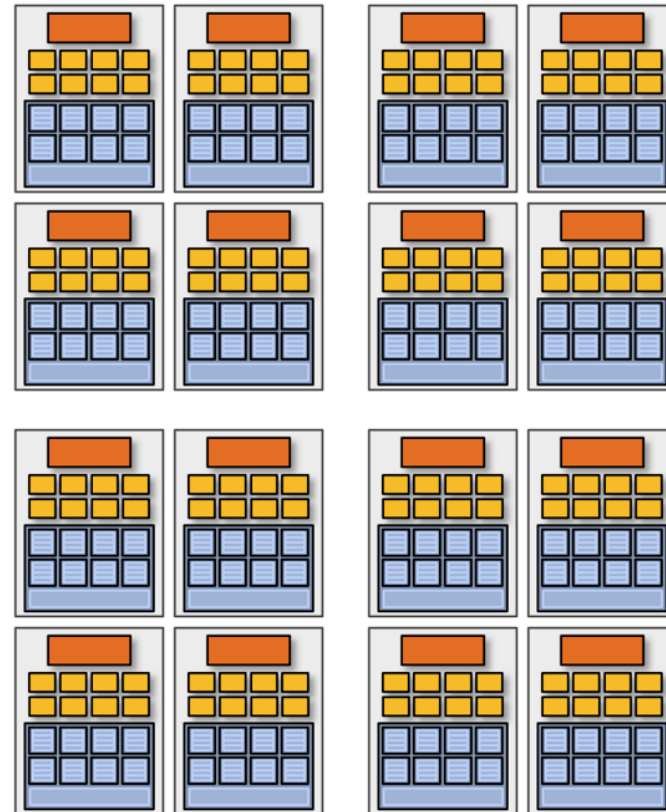
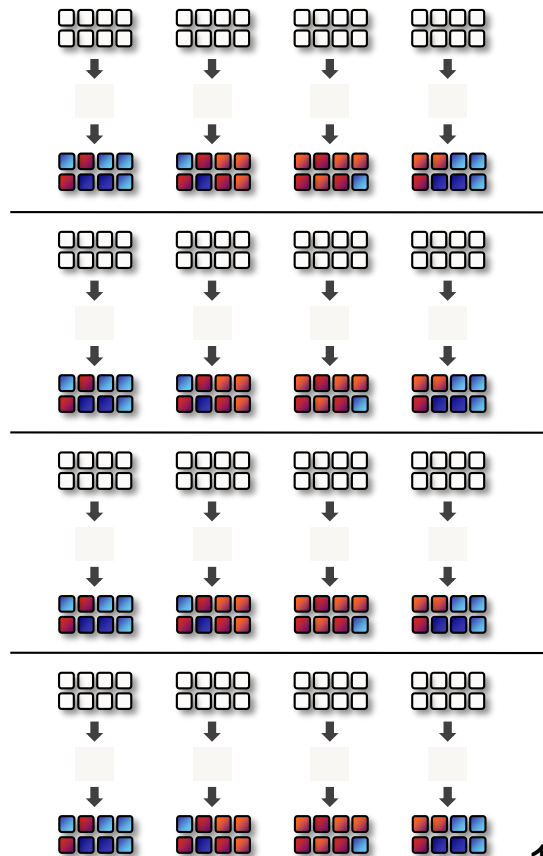
SIMD/vector instructions, each operates on a vector of 8 elements here.



VEC8_diffuseShader>:
 VEC8_sample vec_r0, vec_v4, t0, vec_s0
 VEC8_mul vec_r3, vec_v0, cb0[0]
 VEC8_madd vec_r3, vec_v1, cb0[1], vec_r3
 VEC8_madd vec_r3, vec_v2, cb0[2], vec_r3
 VEC8_clmp vec_r3, vec_r3, 1(0.0), 1(1.0)
 VEC8_mul vec_o0, vec_r0, vec_r3
 VEC8_mul vec_o1, vec_r1, vec_r3
 VEC8_mul vec_o2, vec_r2, vec_r3
 VEC8_mov vec_o3, 1(1.0)



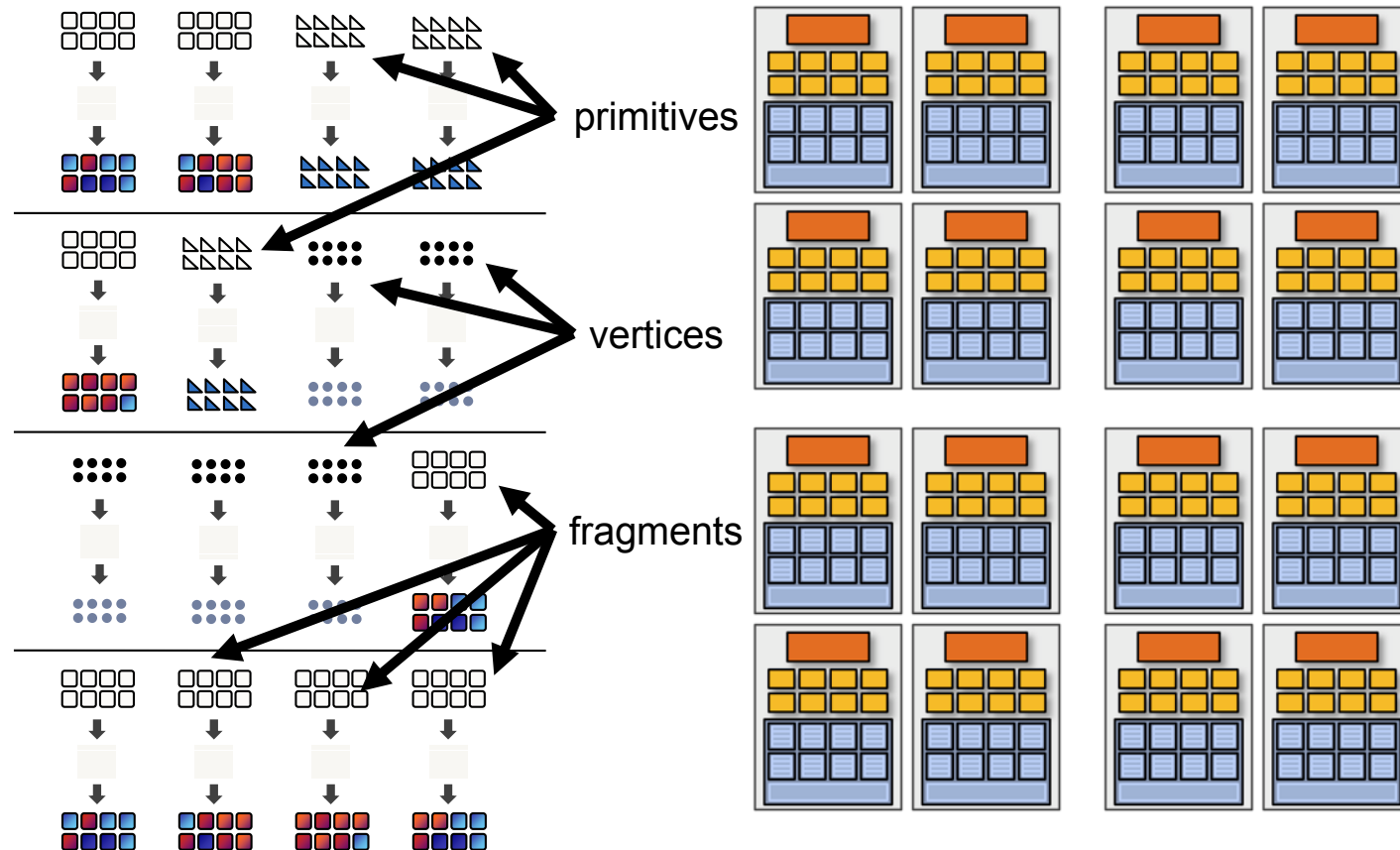
16 cores, each with 8 ALUs. Each core here runs the same program (fragment shader)



16 cores = 128 ALUs

= 16 simultaneous instruction streams

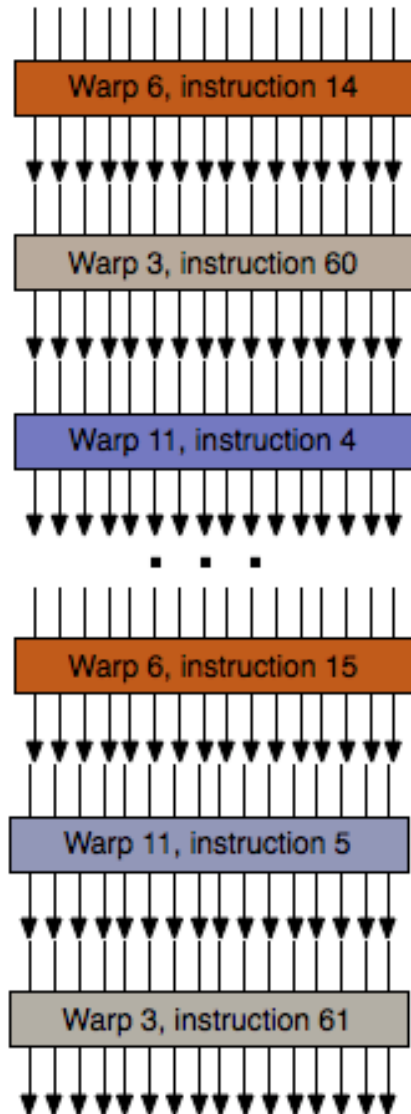
16 cores, each with 8 ALUs. Cores here run different programs
 (some are processing vertices, some are processing fragments)



Each Core Does Fine-Grained Multi-threading

No need for branch prediction and out-of-order execution.
Simple core design. Each thread has its own set of registers in hardware to minimize context switch overhead.

Warp: a group of threads (8 here)



Time

