EECS151 : Introduction to Digital Design and Ics

Lecture 1 – Introduction

Bora Nikolic and Sophia Shao

Mondays and Wednesdays 2:30-4pm

Cory 540AB

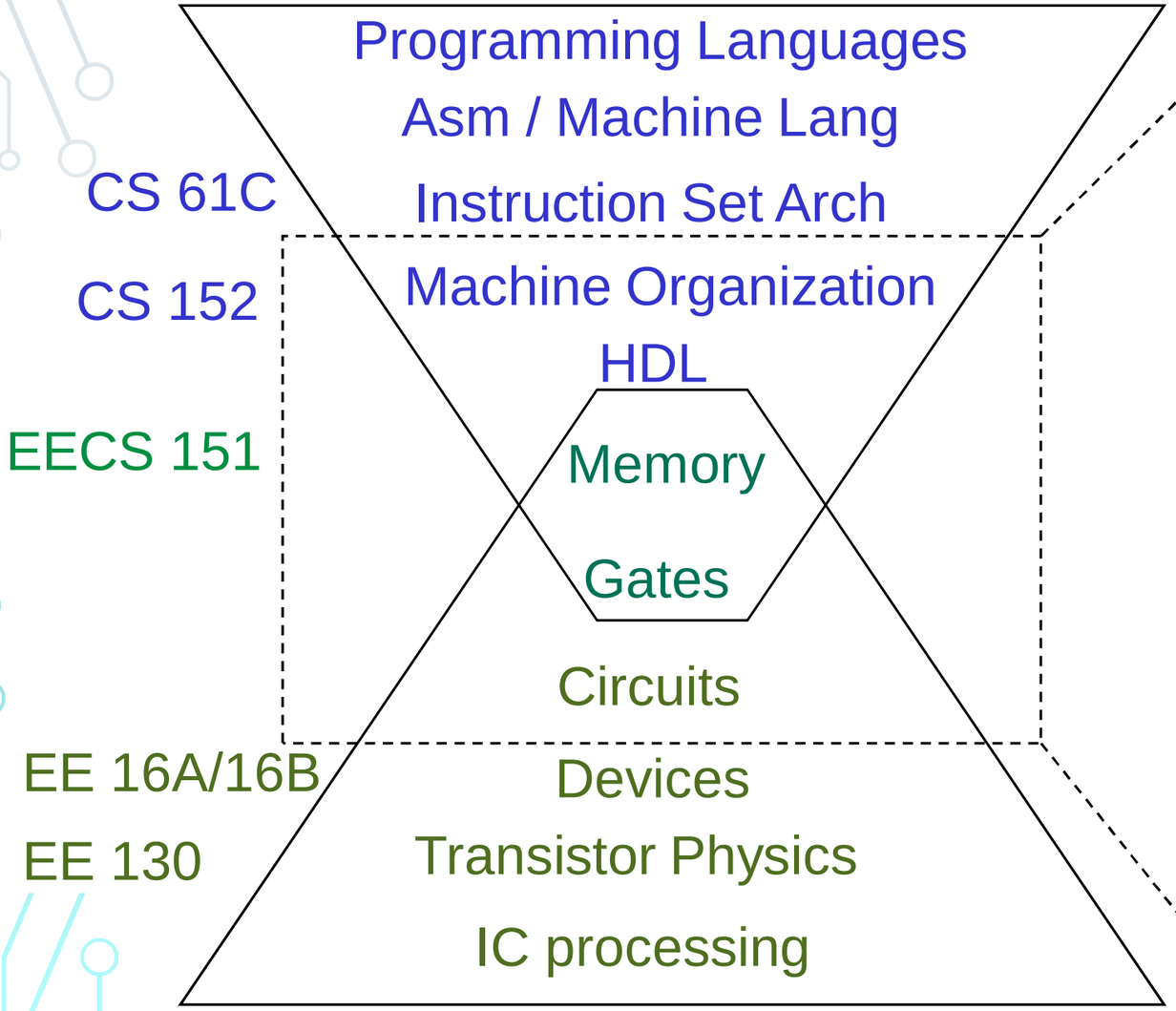
Class Goals and Expected Outcomes

What This Class is All About?

- **Introduction to digital integrated circuit and system engineering**
  - Key concepts needed to be a good digital system designer
  - Discover you own creativity!
- **Learn models that allow reasoning about design behavior**
  - Manage design complexity through abstraction and understanding of tools
  - Allow analysis and optimization of the circuit's performance, power, cost, etc.
- **Learn how to make sure your circuit and system works**
  - *There are way more ways to mess up a chip than to get it right.*

Digital design is not twitch.com!
Learn by doing!

Course Focus



Deep Digital Design Experience

Fundamentals of Boolean Logic

Synchronous Circuits

Finite State Machines

Timing & Clocking

Device Technology & Implications

Controller Design

Arithmetic Units

Memories

Testing, Debugging

Hardware Architecture

Hardware Design Languages (HDL)

Design Flow (CAD)

Prerequisites

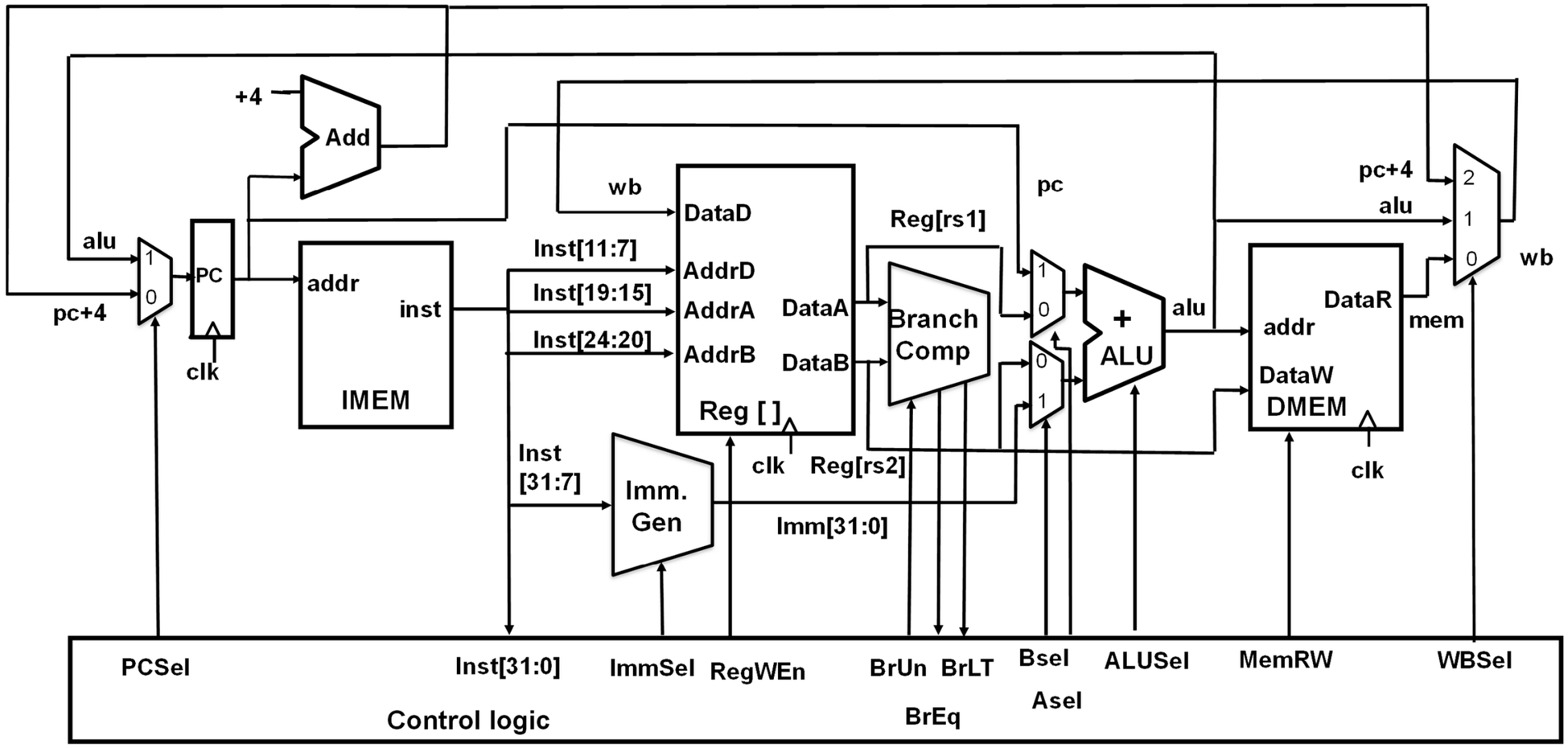
- CS61C
  - C, Boolean logic, RISC-V ISA
  - We will review combinational and sequential logic and RISC-V datapath, pipelining (and go much more in depth)
- EE16A/B
  - Digital gates, RC networks
  - We will review transistor operation and design of CMOS logic

Possible Course Sequences

- $\left\{\begin{array}{l}\text{CS 61C}\\ \text{EE 16A/B}\end{array}\right\} \rightarrow \text{EECS 151} \rightarrow \text{CS152} \rightarrow \ldots$
- $\left\{\begin{array}{l}\text{CS 61C}\\ \text{EE 16A/B}\end{array}\right\} \rightarrow \text{EECS 151} \rightarrow \text{CS152} \rightarrow \ldots$

- With 151/152 background should be able to take any graduate-level course in architecture/digital systems
- EECS 151 + EE 140 is a springboard into integrated circuits

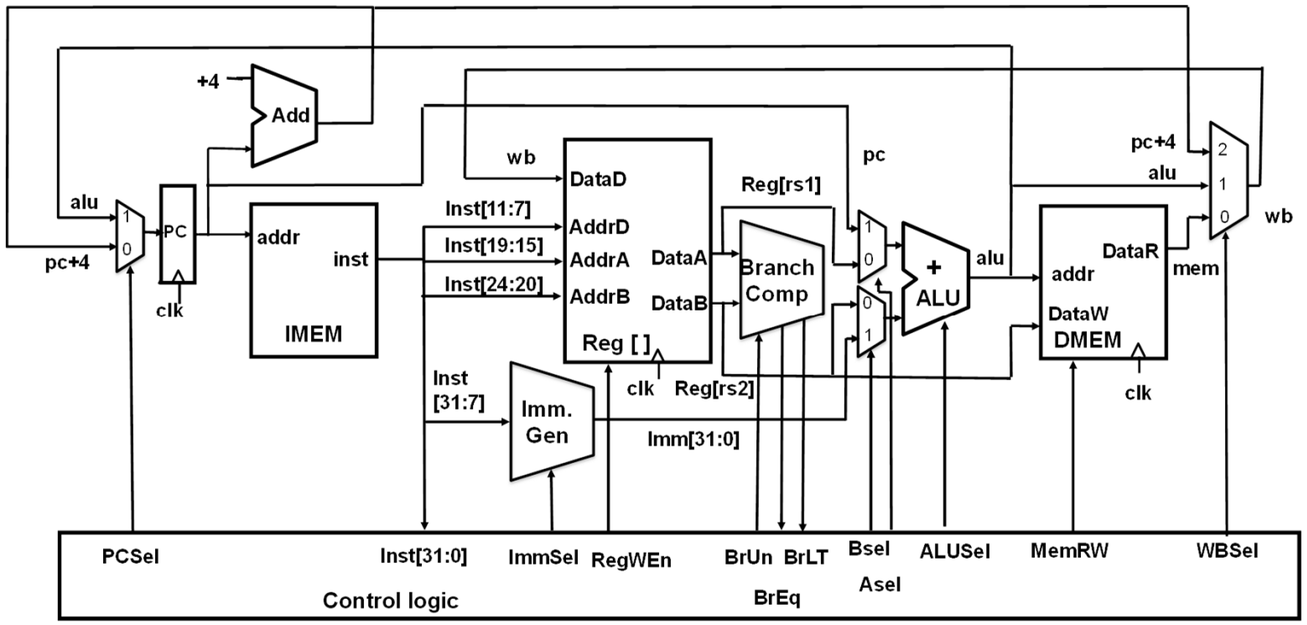
CS61C Background – RV32I

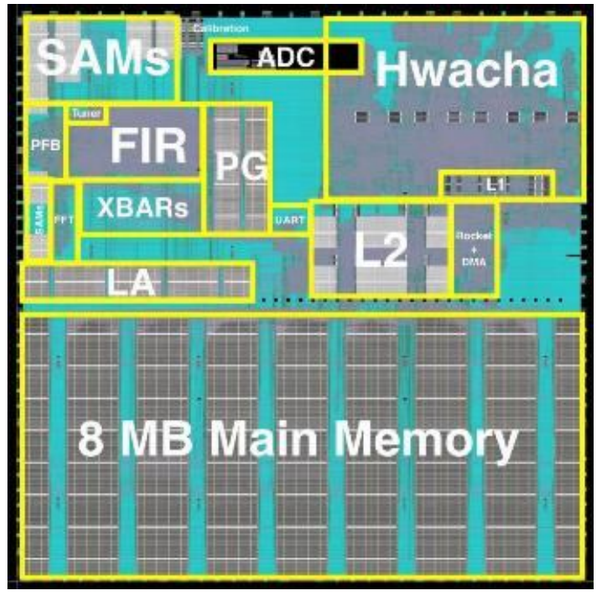


- Don’t worry about details – we will rebuild it and make it work!

At the end of EECS 151

- Should be able to build a complex digital system





Berkeley chip in the next issue
of IEEE Journal of Solid-State Circuits

Quiz: How Best to Represent -12.75?

a) 2s Complement (but shift binary pt)

b) Bias (but shift binary pt)

c) Combination of 2 encodings

d) Combination of 3 encodings

e) We can't

Shifting binary point means "divide number by some power of 2. E.g., $11_{10} = 1011.0_2$ so $(11/4)_{10} = 2.75_{10} = 10.110_2$

Administrativia

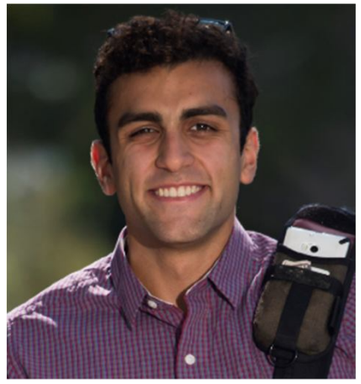
EECS151/251A Crew

Professor Borivoje Nikolić (Bora)
509 Cory Hall
bora@berkeley.edu
Office Hours:
M 11-12AM

Professor Sophia Shao
570 Cory Hall
ysshao@berkeley.edu
Office Hours:
W 11-12AM

Vighnesh Iyer
Discussion Sections, FPGA Labs
Office Hours:
M 11am-12pm

Rebekah Zhao
FPGA Labs
Discussions
Office Hours:
Tu 11am-12pm



Ryan Kaveh
FPGA Labs
Discussions
Office Hours:
Th 4-5pm

Cem Yacin
Discussion Sections, ASIC Labs
Office Hours:
F 2-3pm

All TA office hours held in 125 Cory.

Reader: TBD

Course Information

- Basic Source of Information, class website:

  http://inst.eecs.berkeley.edu/~eecs151/fa19/

  - Lecture notes and video modules
  - Assignments and solutions
  - Lab and project information
  - Exams
  - Piazza Discussion Forum
  - Many other goodies …

Print only what you need: Save

Class Organization

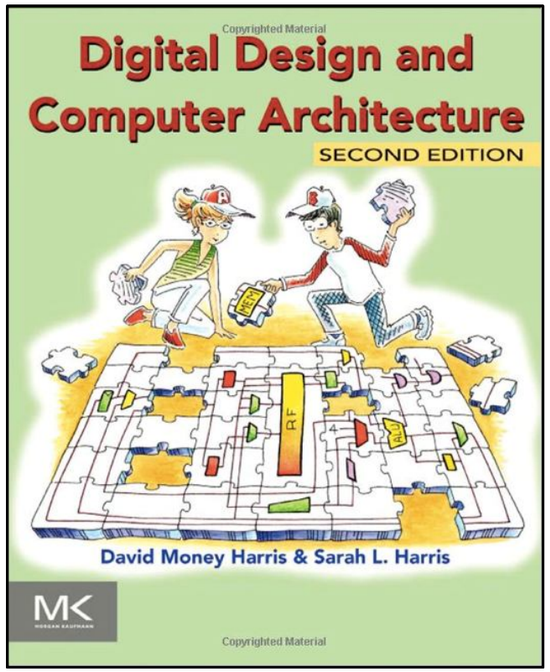
- Lectures
- Discussion sessions (F 10-11am, M 1-2pm)
- Office hours (6 hours per week!)
- Problem Sets (~1 per week)
- Labs – FPGA or ASIC (or both)
- Design project
- 2 Midterms + 1 Final

Lectures

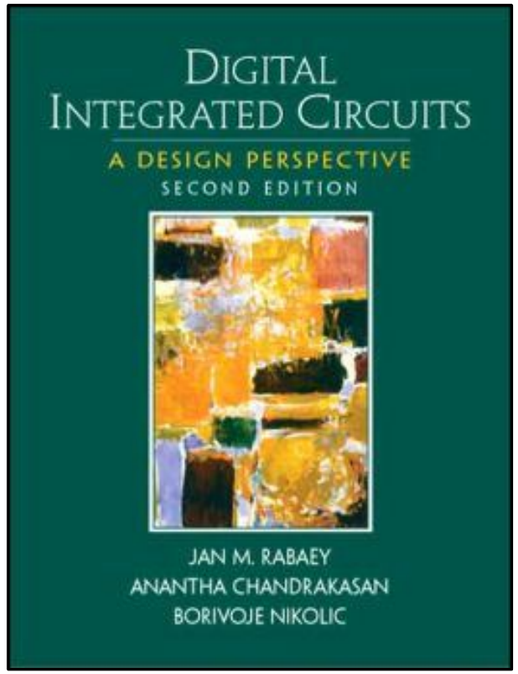
- Slides available on website before the lecture
- Lectures are webcast!
  - But please do come to lectures!
  - We like interactive lectures!

Class Textbooks

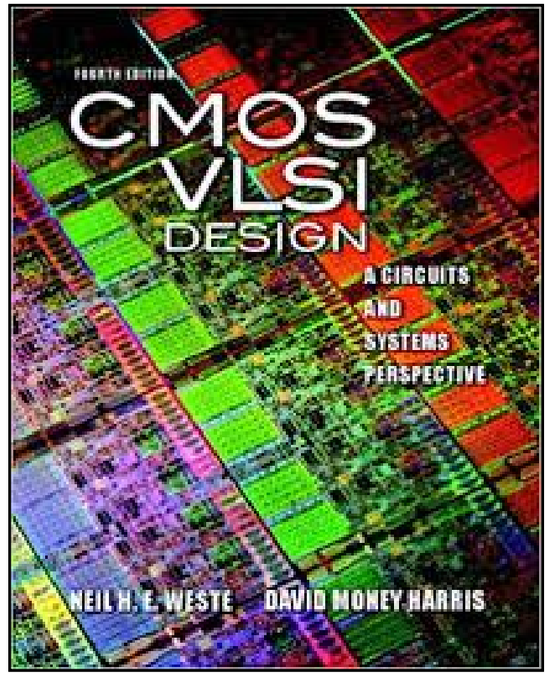
No Required Book this semester



Recommended (previously required)



Recommended (previously required)



Useful

- Useful LA lab reference (EE151/251LA):
  - Erik Brunvand: Digital VLSI Chip Design with Cadence and Synopsys CAD Tools

Discussion Sessions

- Start this week!
- Review of important concepts from lecture (remember - no textbook)
- Help with problem sets

Problem Sets

- Approximately 10 over the course of the semester (one per week)
- Posted on Thursday, due on Friday 11:59pm, 8 days later
- Essential to understanding of the material
- Ok to discuss with colleagues but need to turn in your own work / write-up
- Late turn-in: 20% point deduction per day, except with documented medical excuse
- Solutions posted the week after due date

Labs

- Choose either FPGA or ASIC or both
- 7 FPGA / 7 ASIC lab exercises, done solo
  - Lab report (check off) due by next lab session
- Design project lasts 7 weeks, done with partner
  - Project demo/interview RRR week
  - Project report due RRR week
- All Labs held in 125 Cory
  - ASIC: F 11-2PM
  - FPGA: Th 5-8PM, M 8-11am Tu 8-11AM
- All Labs start this week!

Midterm and Final

- Midterms scheduled in evening
- Review session in advance
    - Midterms:
        - Wed. Oct. 2, 8-10pm (tentative)
        - Wed. Nov. 6, 8-10pm (tentative)
- Final: Th, Dec 19, 3-6PM

All exams are closed book – with one double sided 8.5x11 sheet of notes

Course Information

- For interactions between faculty, GSIs and fellow students – we are using Piazza

  For fastest response **post your questions on Piazza**.

(make sure to register ASAP if you don't want to miss any of the action)

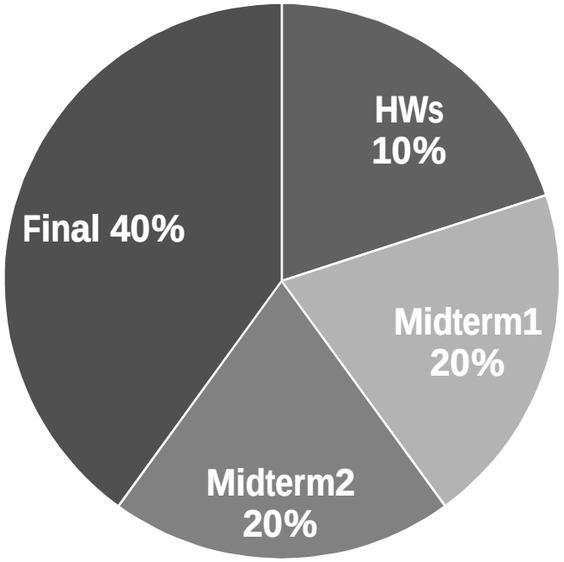
http://piazza.com/berkeley/fall2019/eecs151251afall2019

Cheating Policy

- Details of our cheating policy on the class web site. Please read it and ask questions.
- If you turn in someone else's work as if it were your own, you are guilty of cheating. This includes problem sets, answers on exams, lab exercise checks, project design, and any required course turn-in material.
- Also, if you knowingly aid in cheating, you are guilty.
- However, it is okay to discuss with others lab exercises and the project (obviously, okay to work with project partner). Okay to discuss homework with others. But everyone must turn in their own work.
- Do not post your work on public repositories like github (private o.k.)
- If we catch you cheating, you will get **negative points** on the assignment: It is better to not do the work than to cheat!
  If it is a midterm exam, final exam, or final project, you get an F in the class.
- All cases of cheating reported to the office of student conduct.

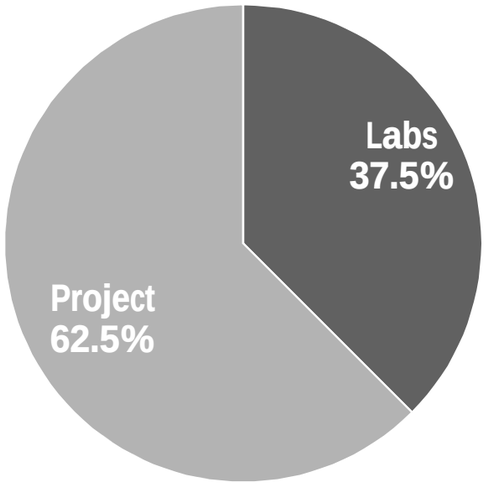
Grading Breakdown

EECS 151/251A Lecture



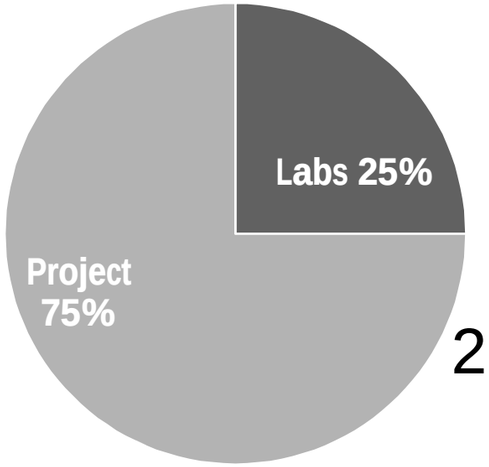
3 units

LA – ASIC



2 units

LB – FPGA



2 units

Tips on How to Get a Good Grade

The lecture material is not the most challenging part of the course.

- You should be able to understand everything as we go along.
- Do not fall behind in lecture and tell yourself you “will figure it out later from the notes or book”.
- Notes will be online before the lecture (usually the night before). Study them before class.
- Ask questions in class and stay involved in the class - that will help you understand. Come to office hours to check your understanding or to ask questions.
- Complete all the homework problems - even the difficult ones.
- The exams will test your depth of knowledge. You need to understand the material well enough to apply it in new situations.

You need to enroll in both the lab and the course.

- Take the labs very seriously. They are an integral part of the course.
- Choose your project partner carefully. Your best friend may not be the best choice!
- Most important (this comes from 30+ years of hardware design experience):
  - Be well organized and neat with homework, labs, project.
  - In lab, add complexity a little bit at a time - always have a working design.
  - Don’t be afraid to throw away your design and start fresh.

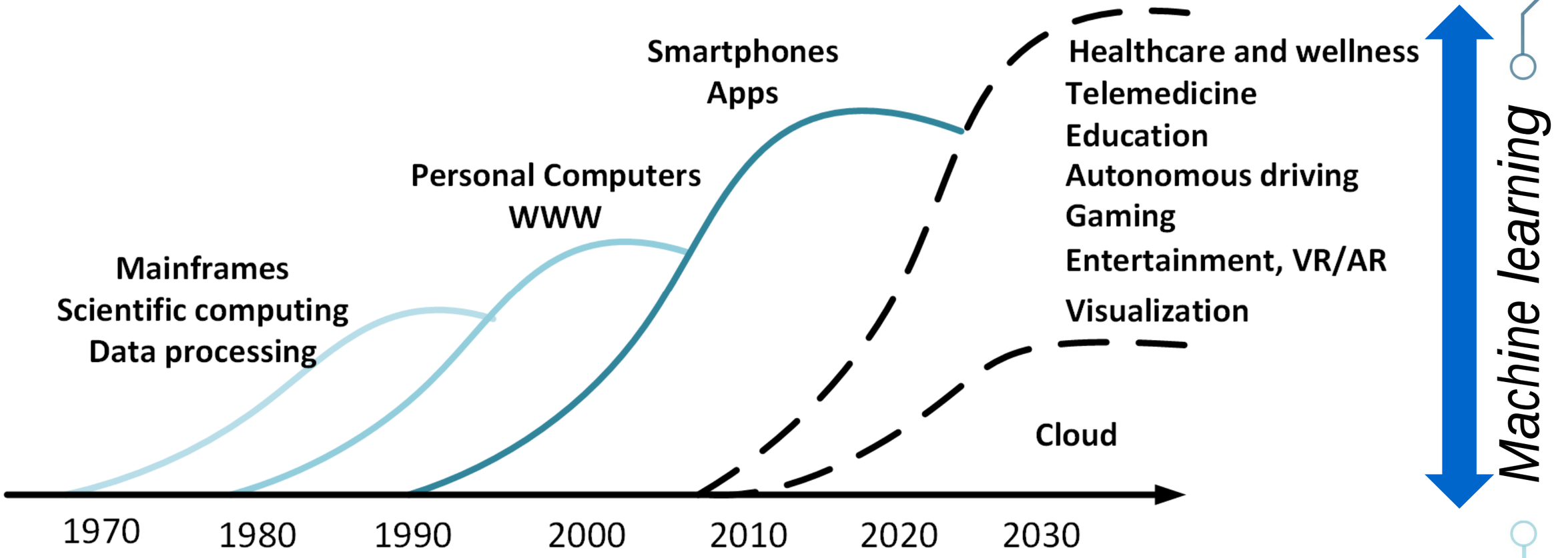
Getting Started

- Discussions and labs start this week
- HW 1 assigned later this week
- Register on Piazza as soon as possible
- Register for your EECS151 class account at inst.eecs.berkeley.edu/webacct
- If you are registering through concurrent enrollment:
  - See us in person this week

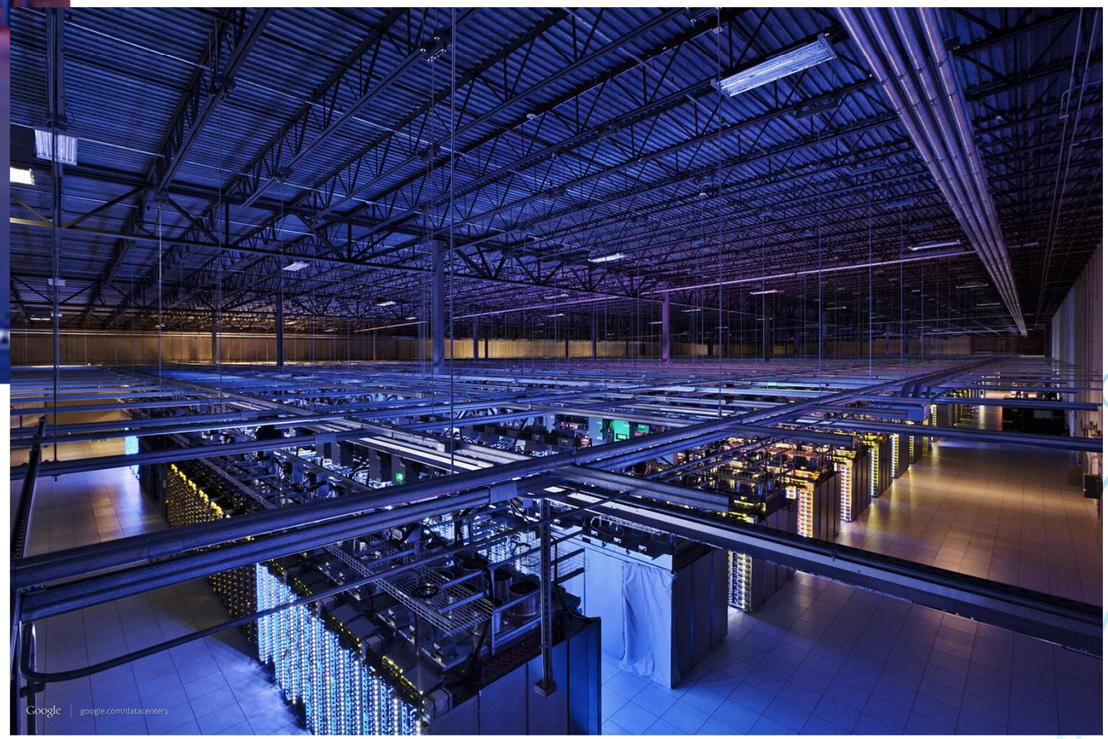
Digital Integrated Circuits and Systems

Past, Present and Future

Diversifying Applications



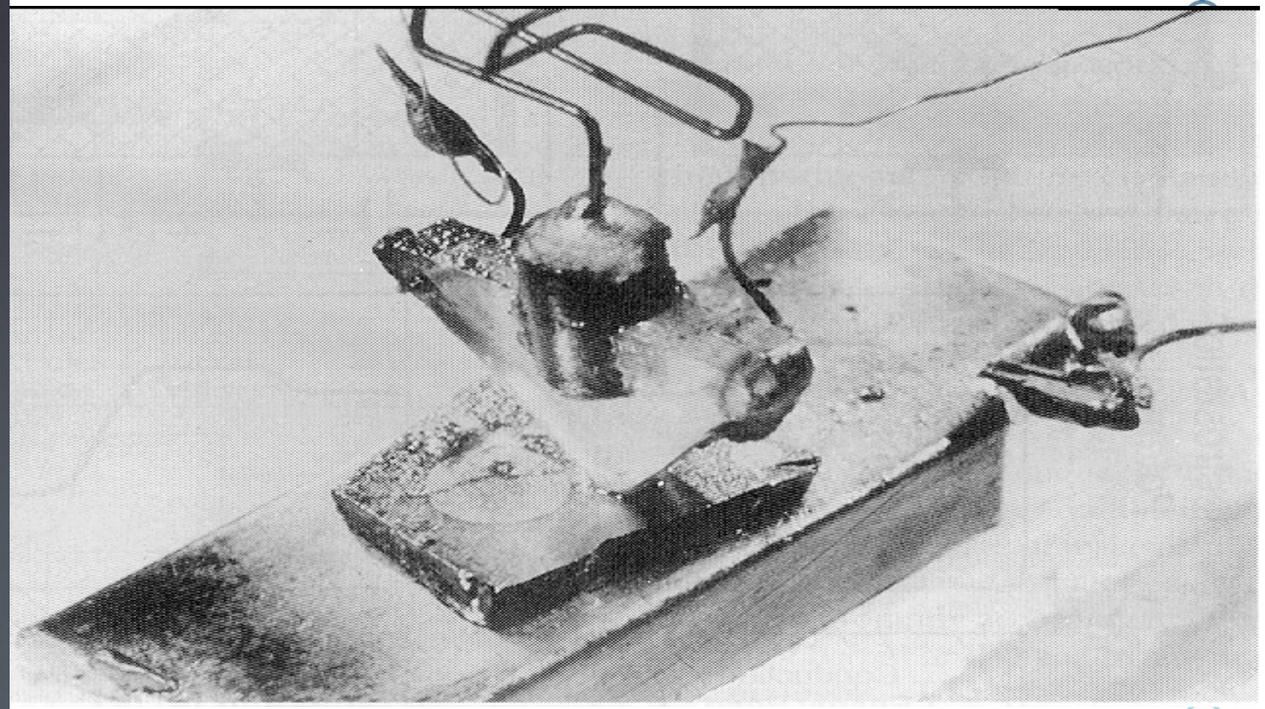
Remember: My Other Computer …



- It is being customized as well!

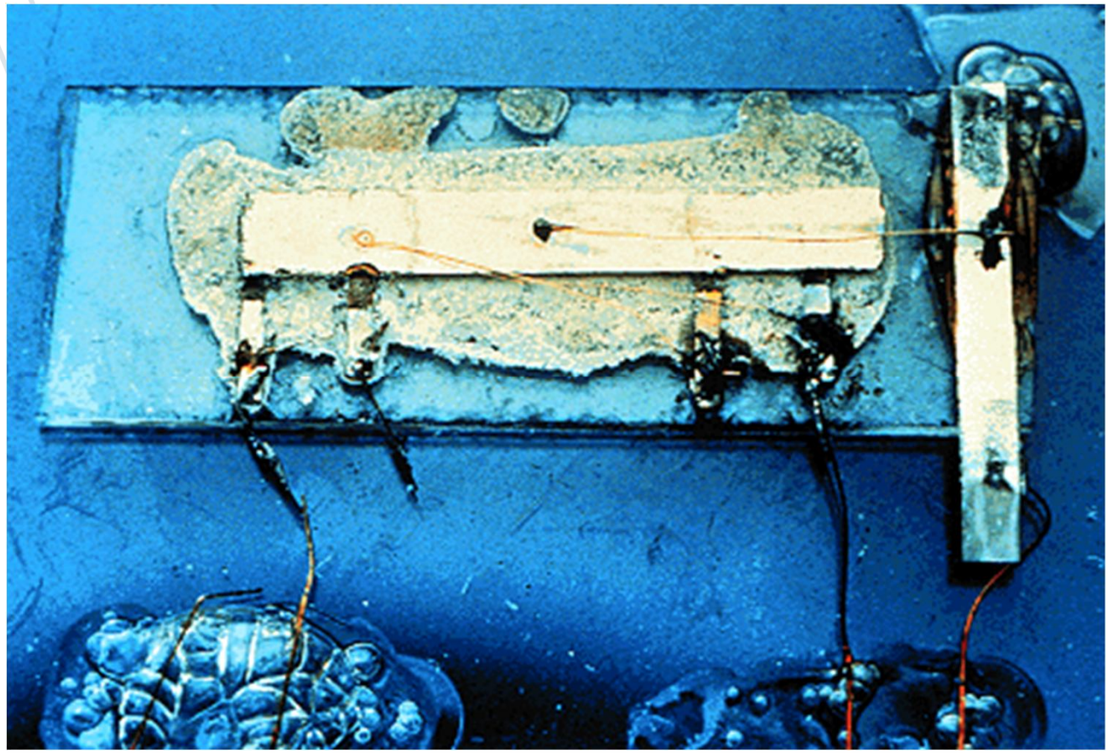
How Did All This Arise?

The Transistor Revolution



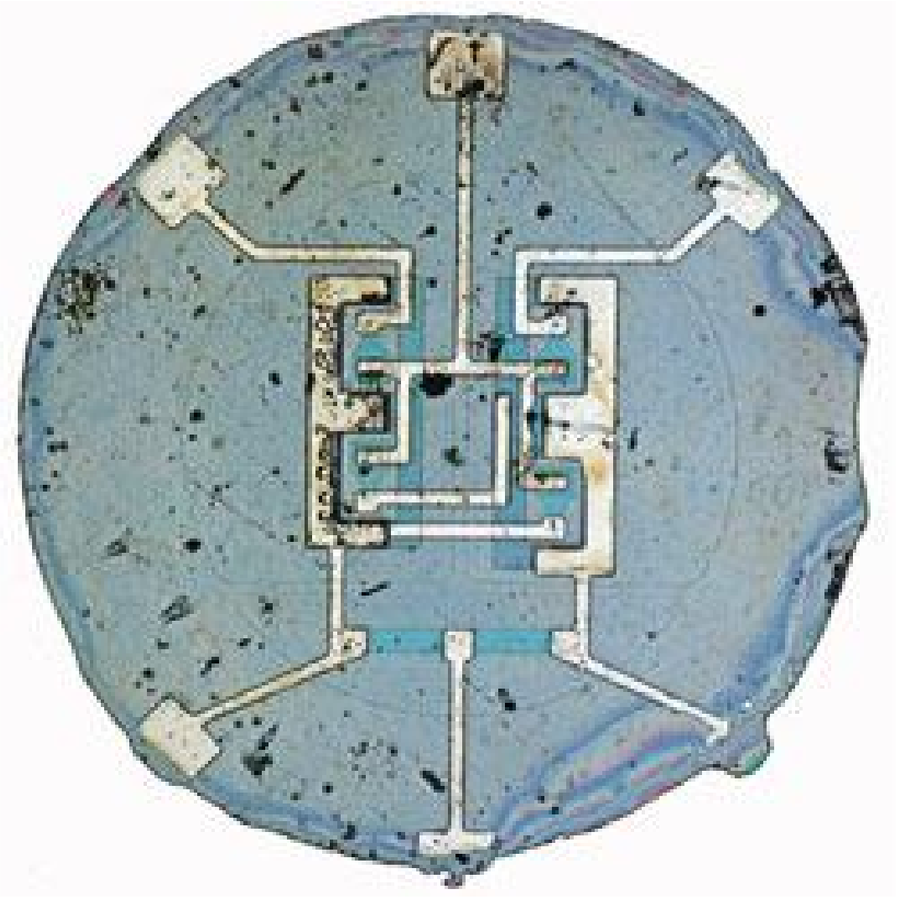
First transistor
Bell Labs, Dec 1947

First Integrated Circuits (1958-59)



Jack Kilby, Texas Instruments

Bob Noyce, Fairchild



Moore's Law

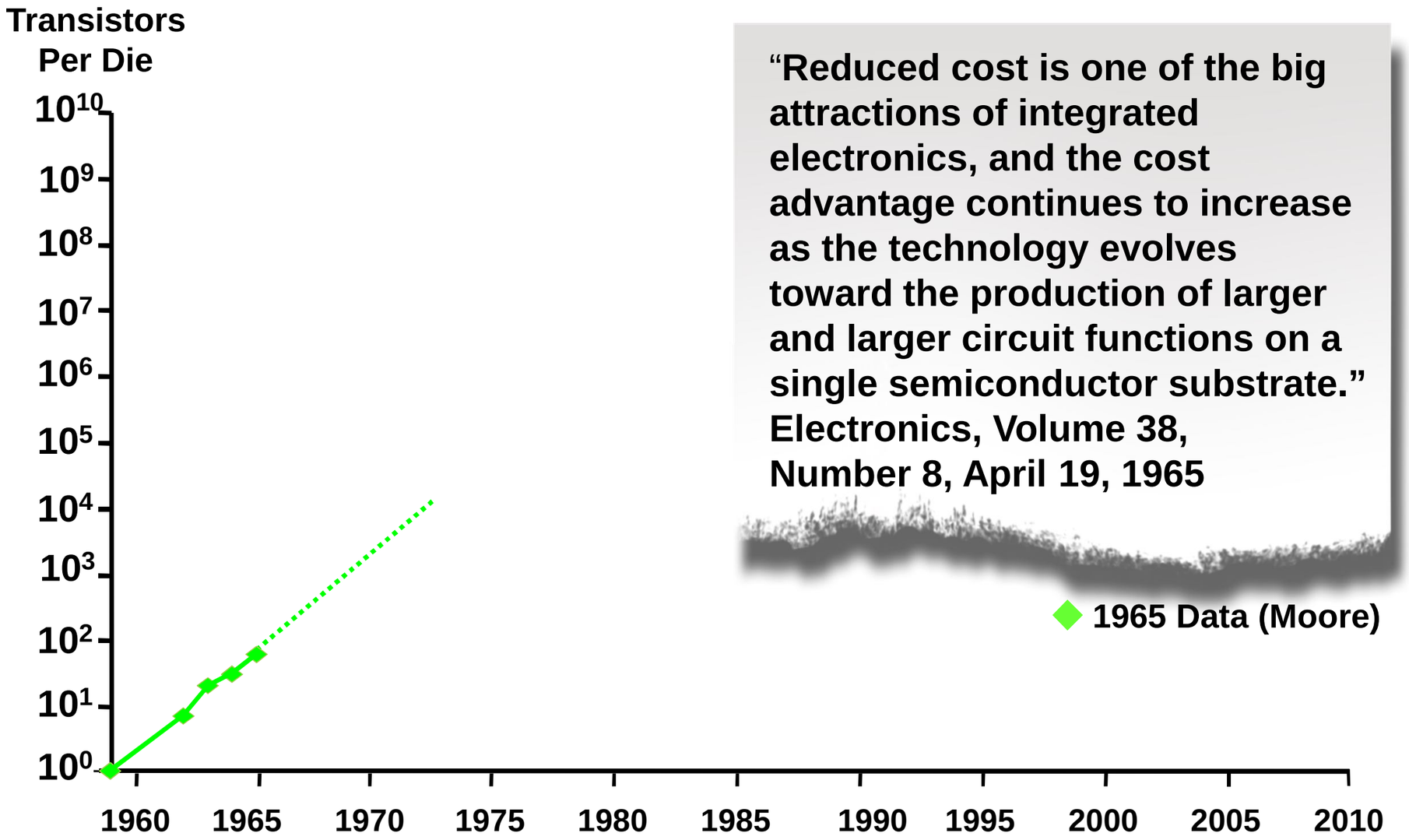
- In 1965, Gordon Moore noted that the number of transistors on a chip doubled every 12 months.
- He made a prediction that semiconductor technology will double its effectiveness every ~~12~~ months

~~18~~

24

"The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term, this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000."
Gordon Moore, Cramming more Components onto Integrated Circuits, (1965).

Moore's Law - 1965

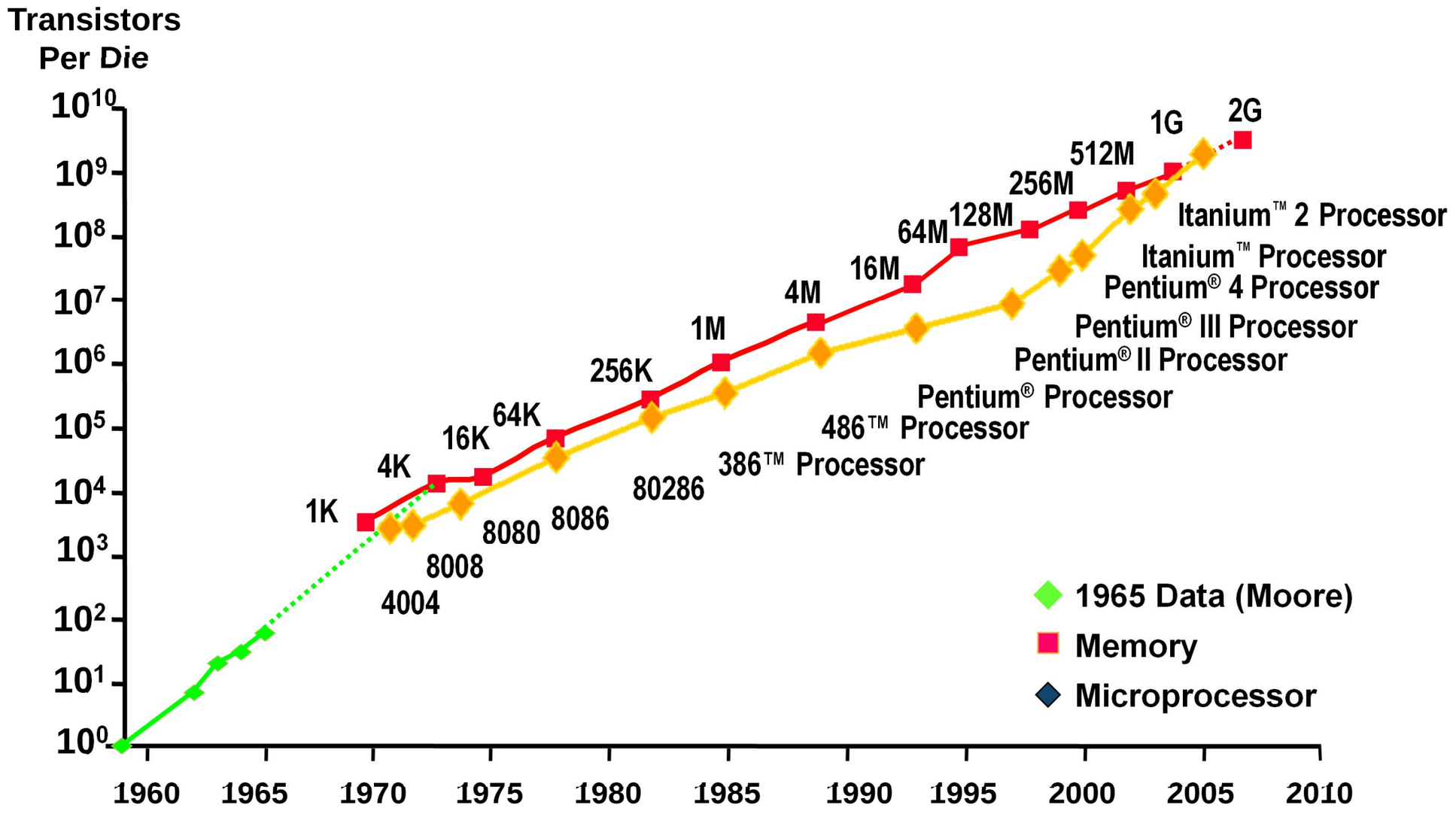


"Reduced cost is one of the big attractions of integrated electronics, and the cost advantage continues to increase as the technology evolves toward the production of larger and larger circuit functions on a single semiconductor substrate."
Electronics, Volume 38, Number 8, April 19, 1965

Graph from S.Chou, ISSCC'2005

Source: Intel

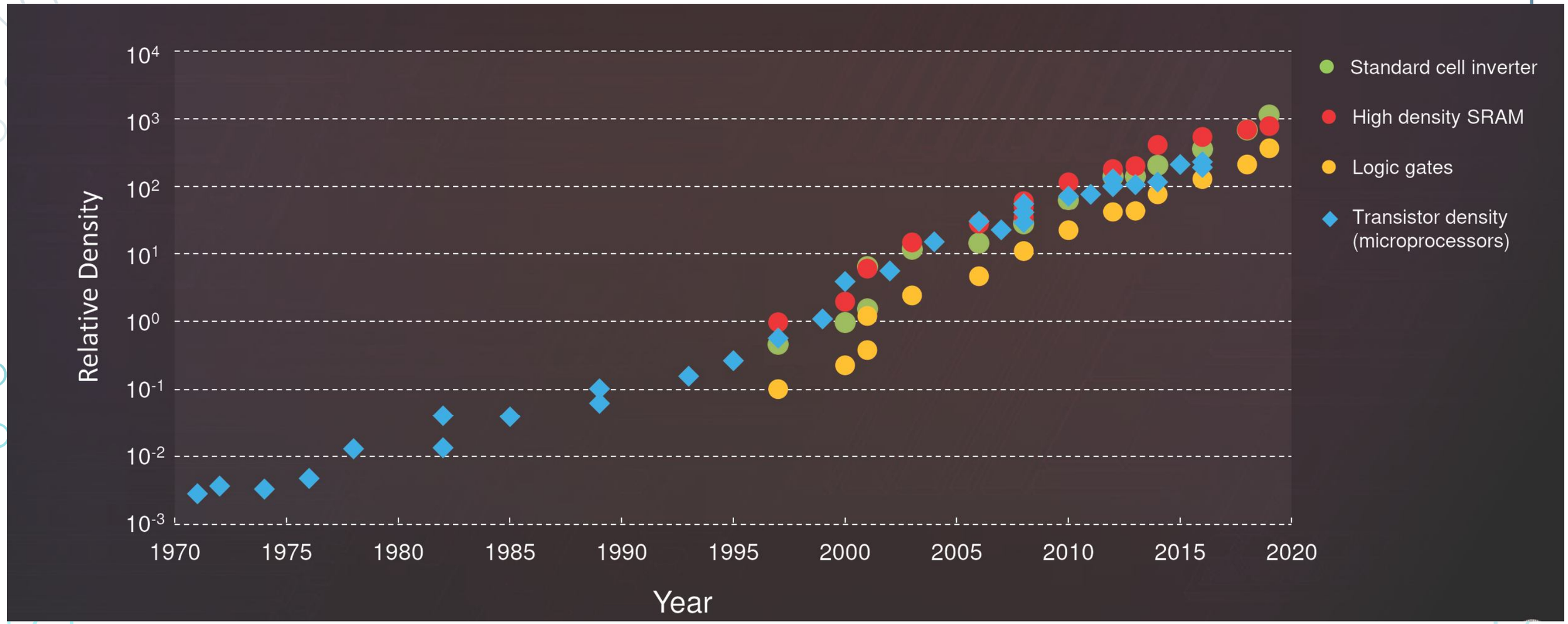
Moore’s Law - 2005



Graph from S.Chou, ISSCC’2005

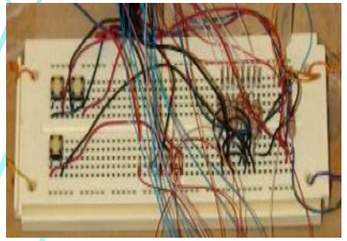
Source: Intel

Transistor Counts

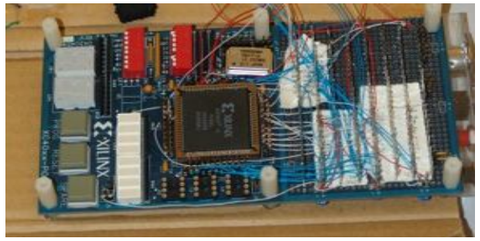


H.-S. P. Wong, HotChips, August 2019.

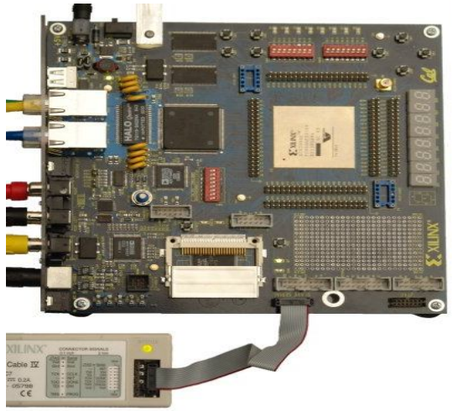
CS150/EECS151 Project Complexity



1980 Pong game
10's of logic gates

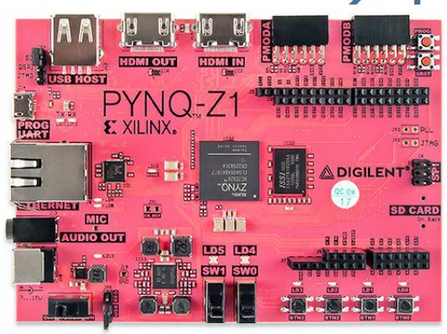


1995 MIDI synthesizer
1000's of logic gates



2000-2010 eTV tuner
10K's logic gates

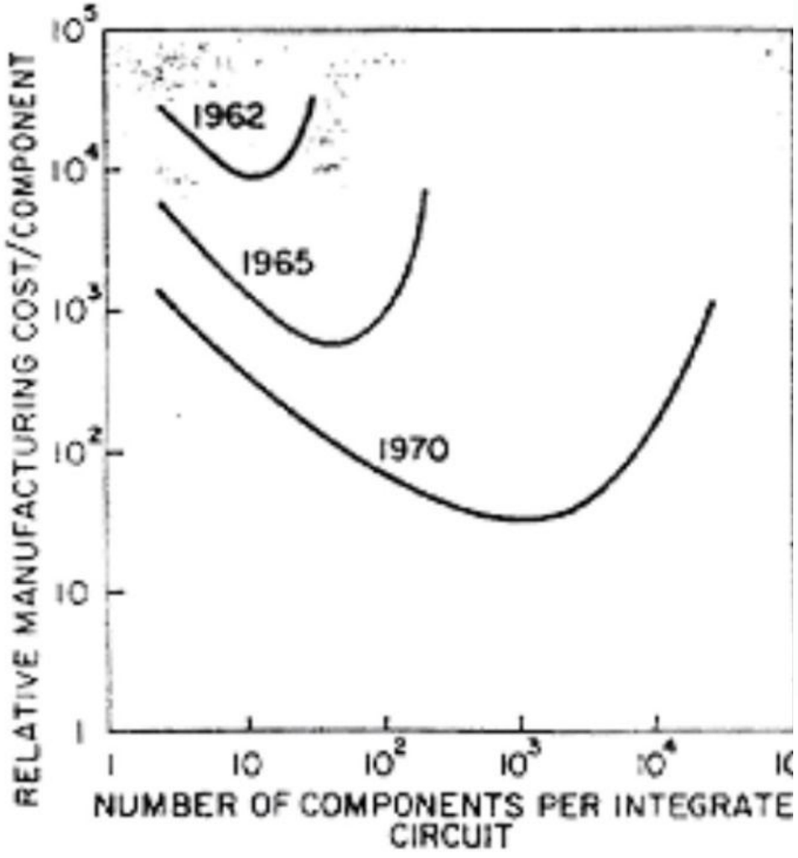
2010-2017 MIPS CPU or BYO
1M logic gates

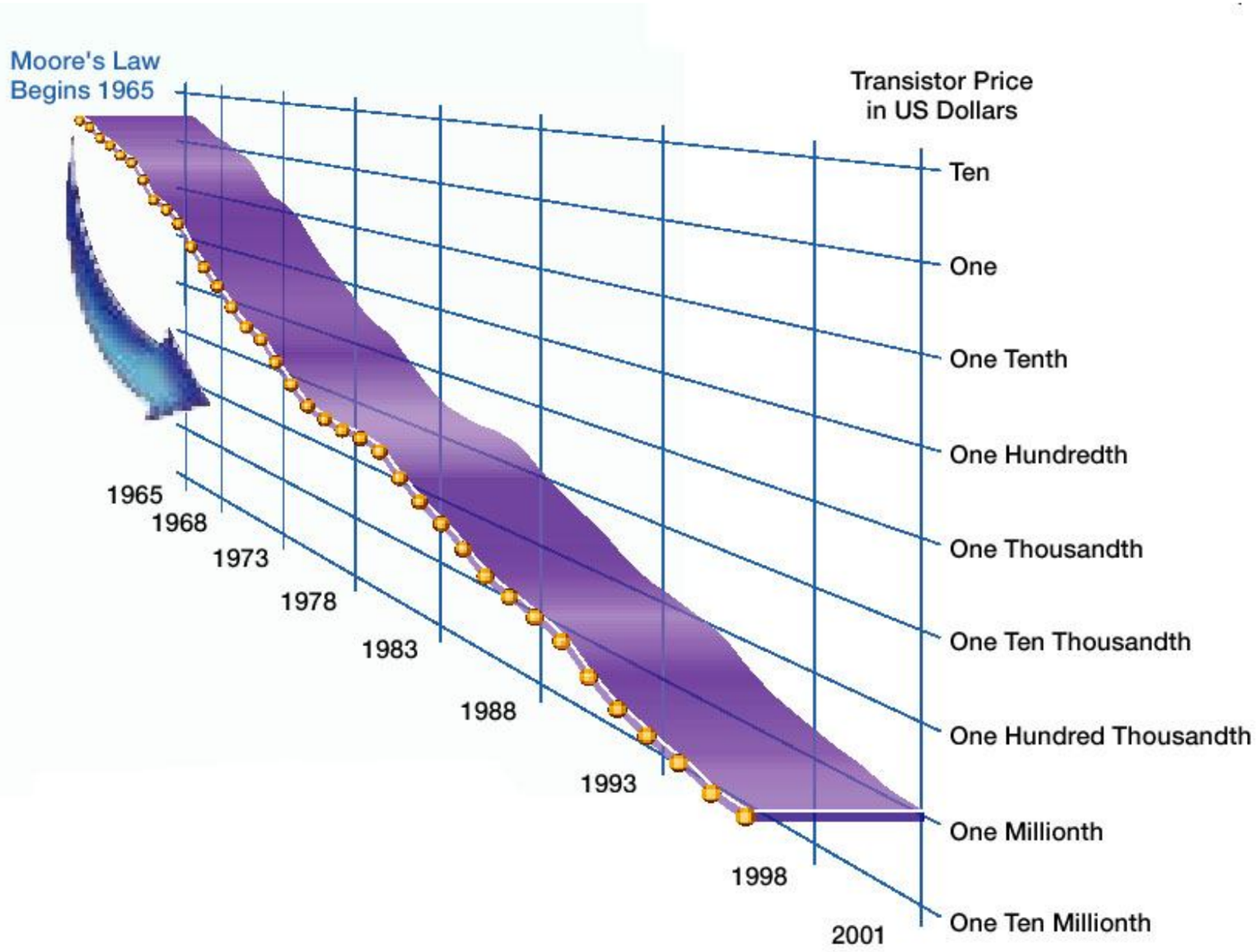


2018 MIPS CPU
Programmable SOC: dua
core ARM, 85K logic cells,
220 MACC

The Key: Cost

Moore's 1965 paper:





Moore's Law ends when cost reduction stops

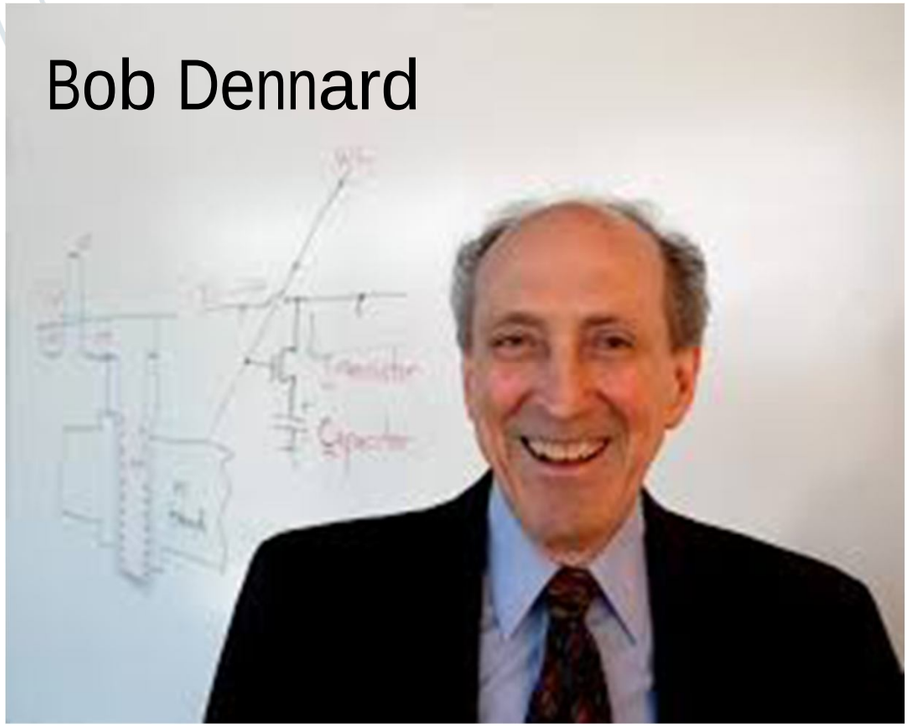
Recent Cost Trend

L. *Su, HotChips, August 2019.*

Cost nearly doubled!

The Other Trends

Dennard Scaling (1974)



- Voltages (and currents) should be scaled proportionally to the dimensions of the transistor
- If so, delay and power should scale

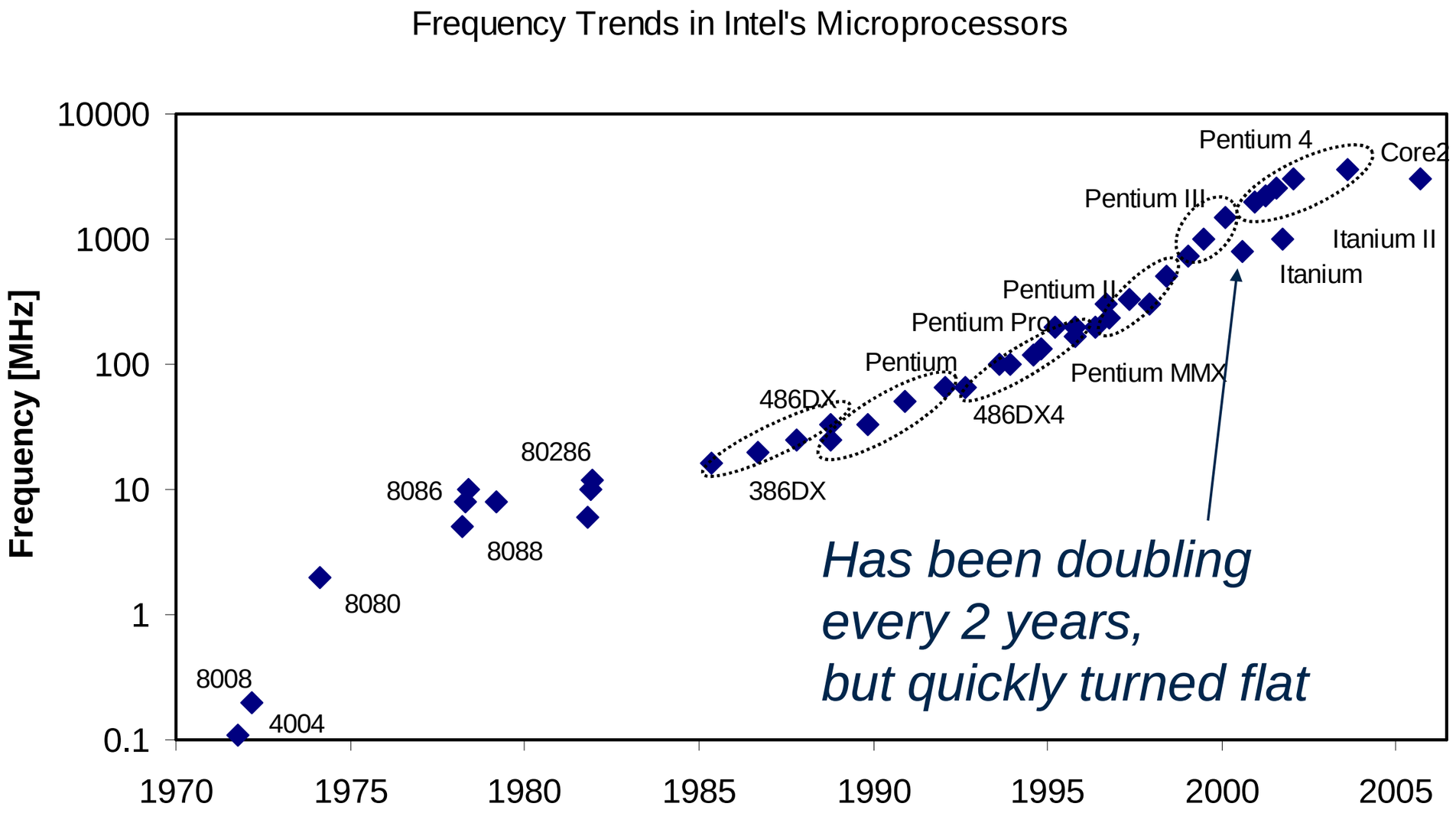
$$Delay \approx C \cdot V / I_{avg}$$

$$P \approx C \cdot V^2 / Delay$$

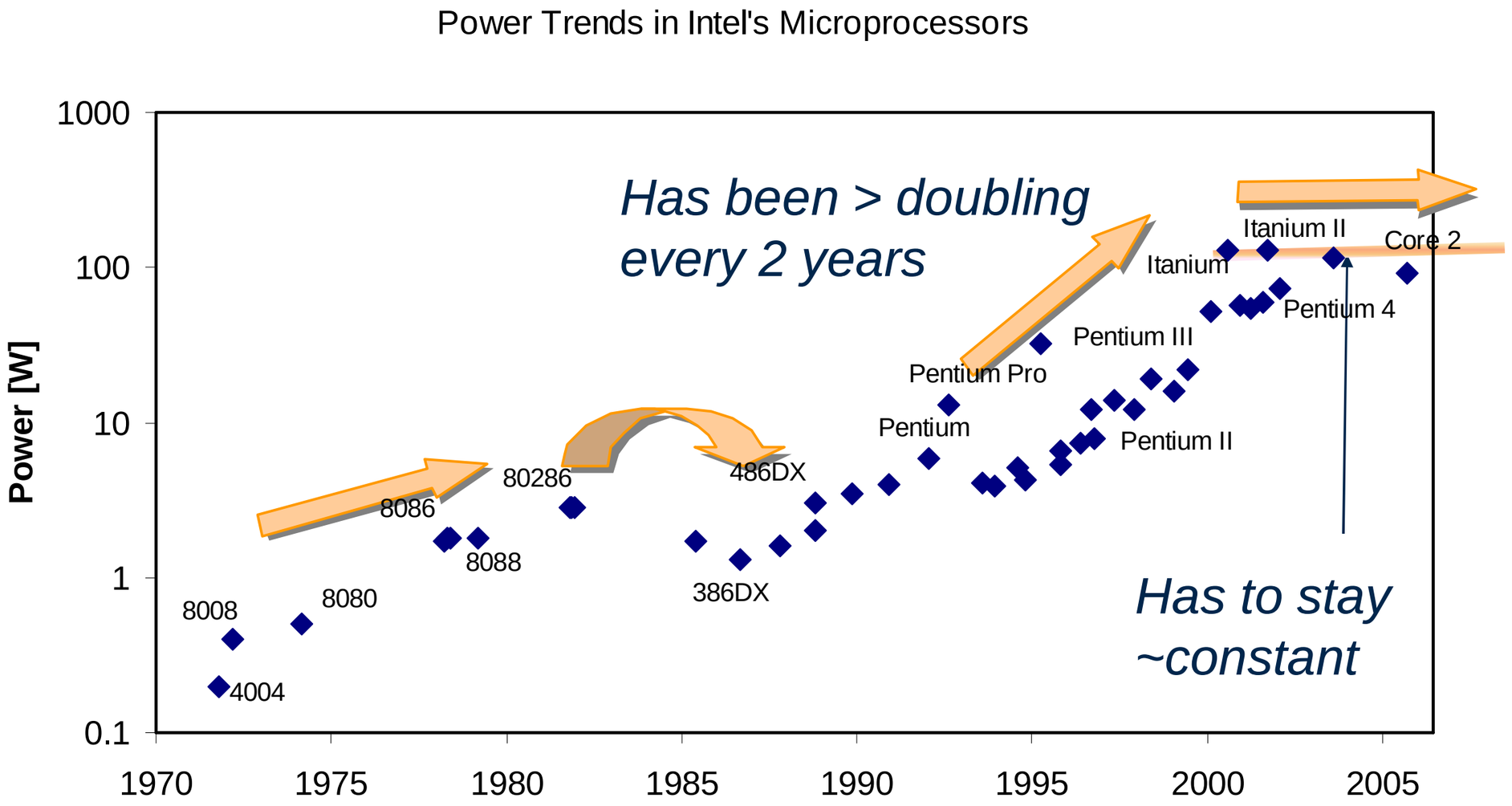
- And, in theory, power *density* constant!

R.H. Dennard, F. Gaensslen, H.-N. Yu, L. Rideout, E. Bassous, A. LeBlanc, Andre "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE Journal of Solid State Circuits*. SC-9 (5), 1974.

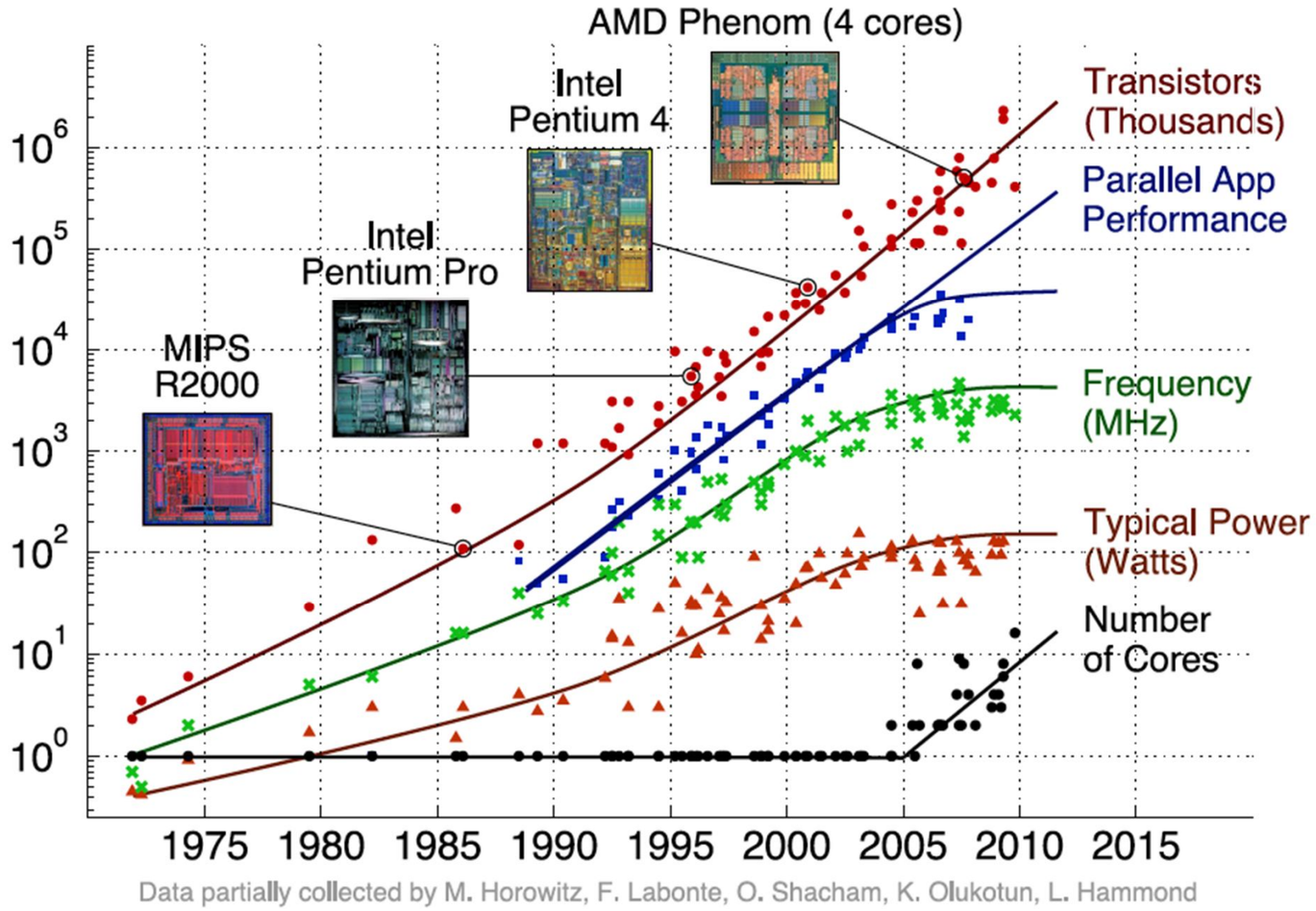
Frequency


Frequency Trends in Intel's Microprocessors
Frequency [MHz]
10000
1000
100
10
1
0.1
1970
1975
1980
1985
1990
1995
2000
2005
8008
4004
8080
8086
8088
80286
386DX
486DX
486DX4
Pentium
Pentium Pro
Pentium II
Pentium III
Pentium 4
Pentium MMX
Itanium
Itanium II
Core2
Has been doubling
every 2 years,
but quickly turned flat

Power Dissipation



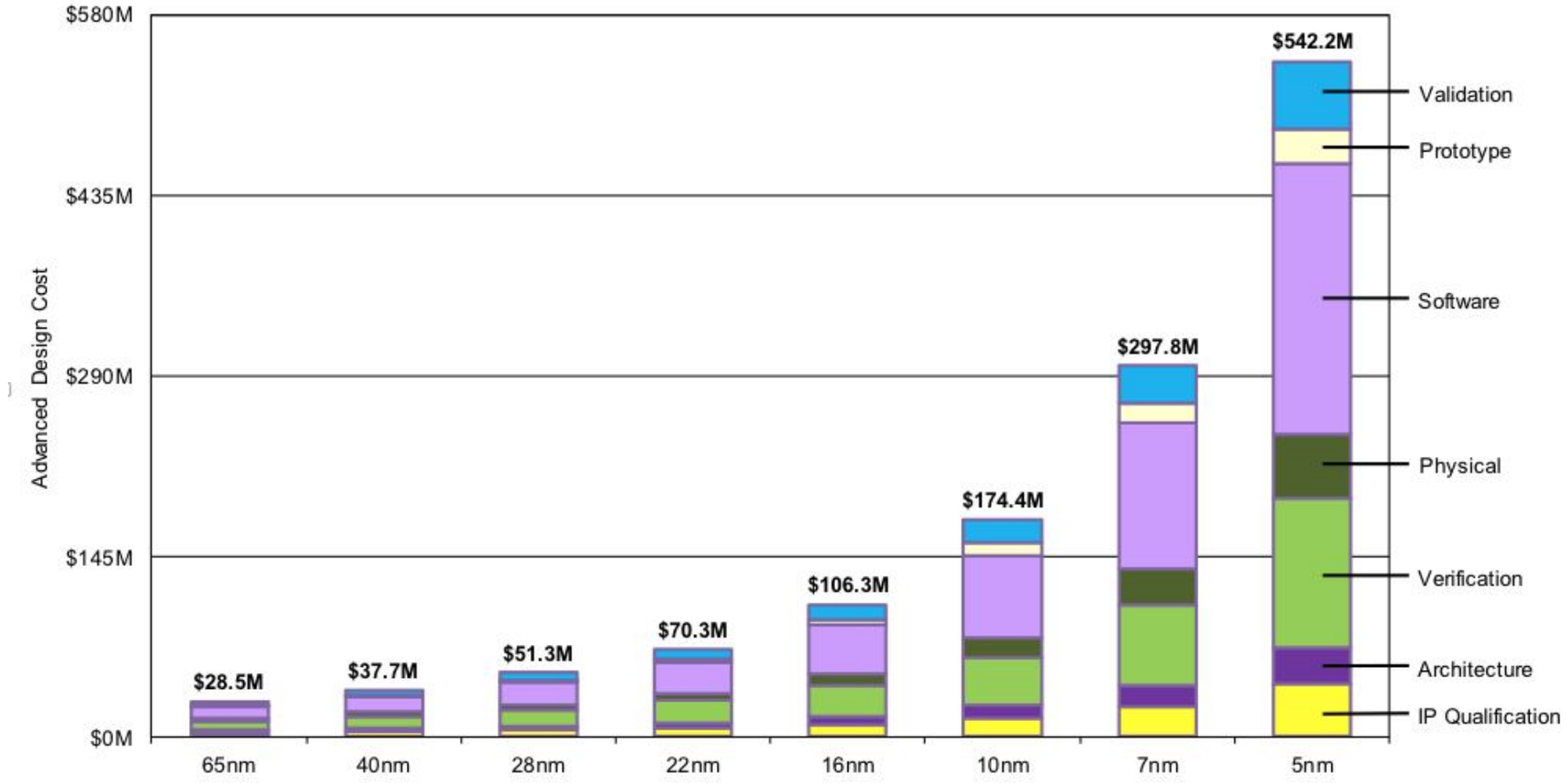
Power and Performance Trends



- What do we do next?

The Other Demon: Complexity and Design Costs

Cost Of Developing New Products



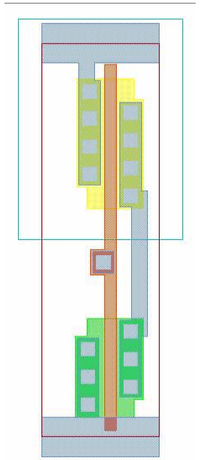
- These are non-recurring (NRE) costs, need to be amortized over the lifetime of a product
- We will attempt to dismantle this…

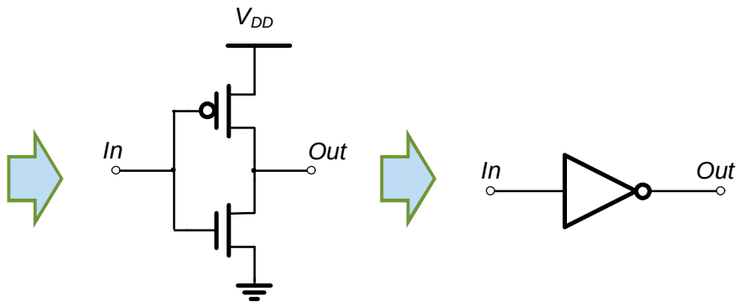
Solution

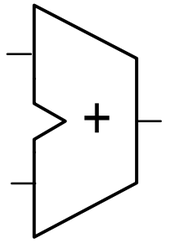
- Software faces the complexity issues
- Solutions: Increase abstraction level, improve reuse, rely on open source, be agile…
- Apply to hardware design

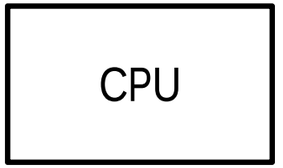
Abstraction

- How to design a Pong game
  - Hand layout
  - Gate-level design (semi custom)
  - Synthesis, place and route
  - HLS, HDL
  - “Computer, design a pong game”



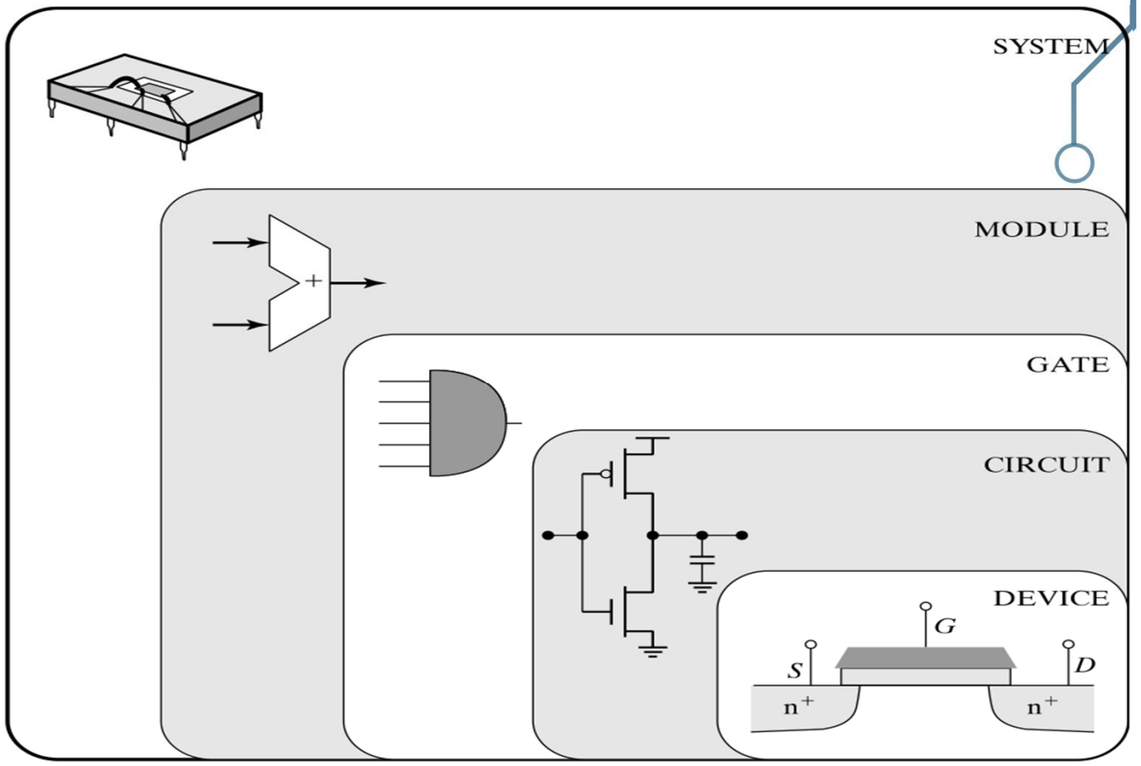


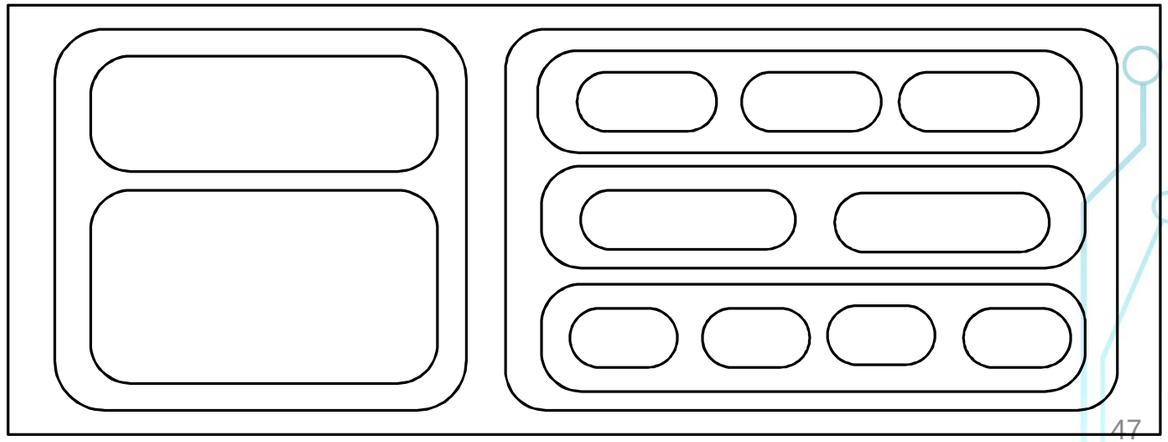




Hierarchy in Designs – Complexity Control

- Design Abstraction
  - Hide details and reduce number of things to handle at any time
- Modular design
  - Divide and conquer
  - Simplifies implementation and debugging





Digital Design: What's it all about?

Given a functional description and performance, cost, *&* power constraints, come up with an implementation using a set of primitives.

- How do we learn how to do this?
    1. Learn about the primitives and how to use them.
    2. Learn about design representations.
    3. Learn formal methods and tools to manipulate the representations.
    4. Look at design examples.
    5. Use trial and error - CAD tools and prototyping. Practice!
- Digital design is a bit an art as well as a science. The creative spirit is critical in combining primitive elements & other components in new ways to achieve a desired function.
- However, unlike art, we have objective measures of a design:

Performance Cost Power