# CS250
# VLSI Systems Design

Spring 2016

John Wawrzynek

with

Christopher Yarp (GSI)

# Why the heck is it CS250 and not EE250?

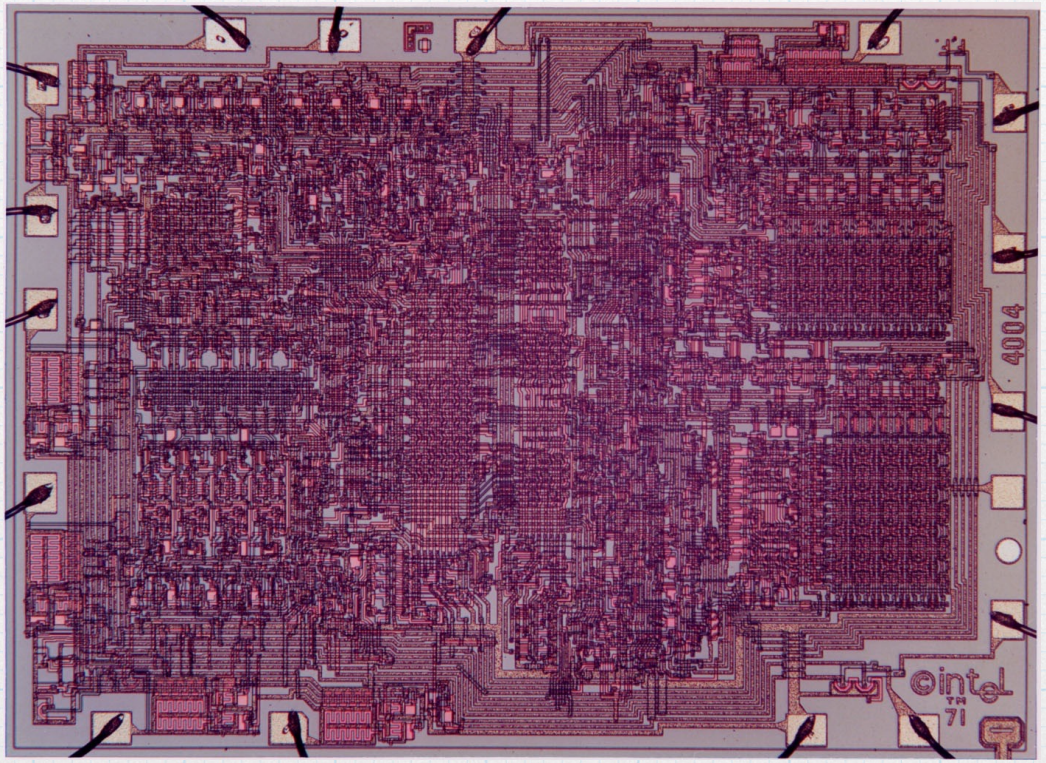
- We answer that with a course history (with a few embedded lessons).

*Warning: What follows is principally from memory. I've done my best to be accurate, but some errors or misinterpretations might exist.*

*Starts in 1958 with the invention of the Integrated Circuit independently by Robert Noyce (co-founder of Fairchild Semiconductor Corporation) and Jack Kilby (engineer at Texas Instruments).*

# IC Design in the 70's and early 80's

- Circuit design, layout, and processing tightly linked.
- Logic design and layout was "random"
- Chip design was the domain of industry (Fairchild, Intel, Texas Instruments, ...). These were IC processing companies. Those who controlled the physics controlled the creative agenda!




The Intel 4004 microprocessor, which was introduced in 1971. The 4004 contained 2300 transistors and performed 60,000 calculations per second. Courtesy: Intel.

Federico Faggin,
Ted Hoff,
Stan Mazor

Introduced to help sell memory chips!

# Meanwhile at Caltech...

- Carver Mead was designing and building prototype ICs (with help from his friends at Intel)
- His background was in physical electronics (invented several semiconductor devices such as the GaAs MESFET) but was deeply interested in the interaction of physical implementation and the higher level design of electronic systems:

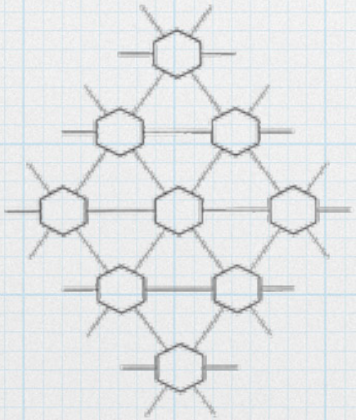
"Listen to the silicon; find out what it's telling you."

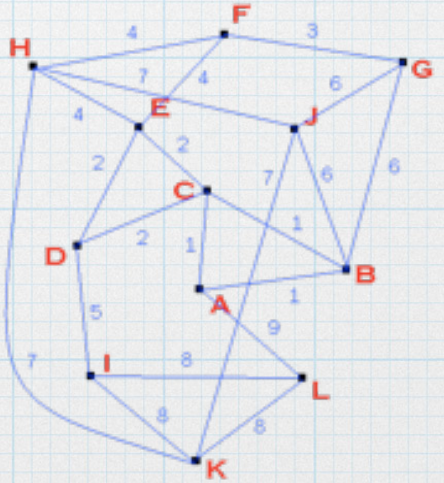
# CS At Caltech

- Ivan Sutherland became founding head of the computer science division at CIT in 1974 (after leaving E&S)
- He and Mead teamed up to get the division off the ground making IC design (Integrated Systems) a key component of the research and teaching.
- My take:
  - These two believed that IC design was at the heart of computer science because CS was largely about inventing and building computing devices.
  - The future of computing was integrated circuits:
  - Very flexible, "boundless" growth potential (was on an exponential grow curve with no end in sight!)
- Close to "pure thought" with few constraints and "nasty realities"
- The potential of "LSI" was not going to be reached with the status quo in industry.
- Worked together over the next 6 years to establish the faculty, industrial ties, curriculum, research projects with silicon structures as a key component.
- They set off to build their own machines (0M1, 0M2).

# Pushing forward (1)

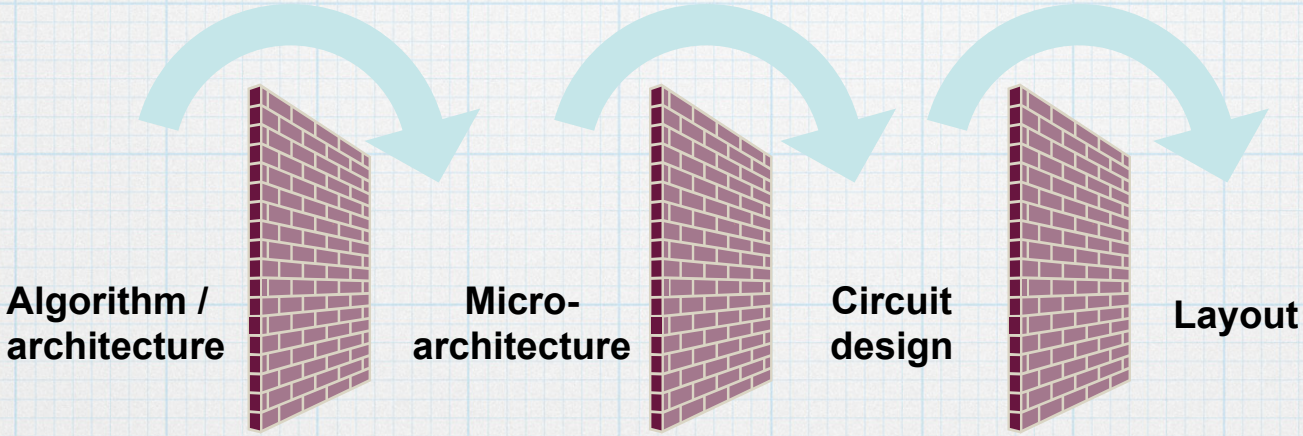
- The reality of integrated circuits:
  - Wires are expensive (area, delay, power), transistors are cheap.
  - Pre-ICs, the opposite was true.
- Therefore, plan the communication and the layout
  - Exploit locality, think about the "geometry" of the problem from the beginning. Choose algorithms/designs accordingly.
  - Algorithms/designs represented as communication graphs in a large number of dimensions, not a good idea.





# Pushing Forward (2)

- Put IC design expertise into the hands of those best qualified to take advantage of its potential:
- Those with intimate knowledge of computation and algorithms: computer scientists!
- Traditionally, IC design had been stratified:




- Emergence of the "tall thin designer". Spans all levels of the design and implementation stack.
- Would lead to more successful innovation and highly optimized designs.

# Pushing Forward (3)

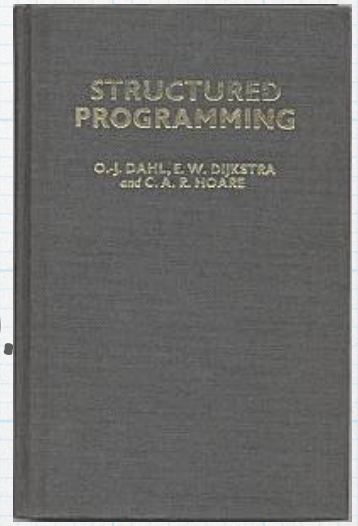
- How to enable system architects:
  - Managing the complexity was the key challenge. Manipulating multiple levels of design complexity was difficult and projected to get much worse looking forward (remember Moore's Law).
  - Providing universal access to IC fabrication.
- Solutions:

1. Ideas from software
2. New design representations
3. Computer aided design tools
4. Silicon "foundries"
5. Education

} All linked

# Ideas from Programming
## (help manage complexity)

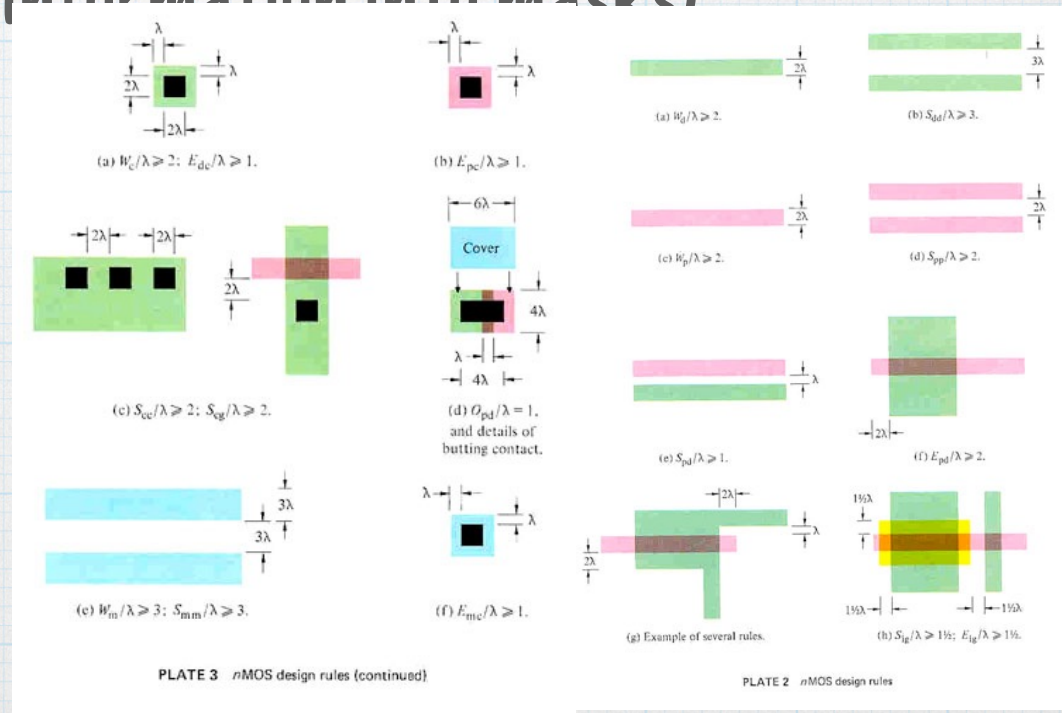
- "Structured Programming" was getting popular (Dijstra, el. al.)
  - No goto statements
  - Block organization.
  - Use of hierarchy, abstraction (sub-routines).




- "Structured Design" for ICs:
  - Exploit regularity and symmetry
  - Use and reuse common sub-blocks (flip-flops, gates, arithmetic, etc.)
  - Represent designs hierarchically

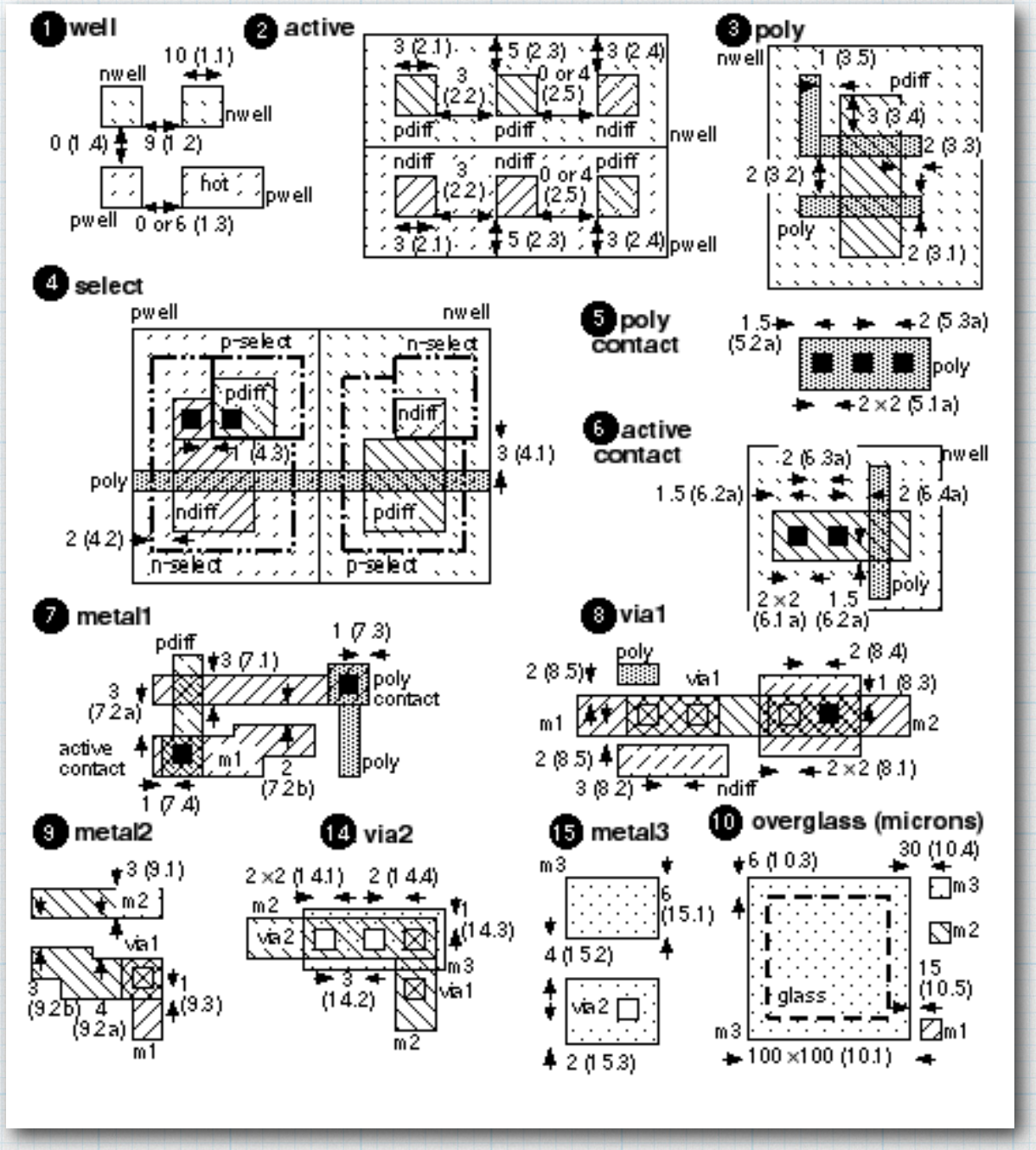
# Design Representations (1)

- Previously, to generate the mask information for fabrication, the designed needed intimate knowledge of the manufacturing process. Even once this knowledge was distilled to a set of "Geometric Design Rules", this set of rules was voluminous with many special cases.
- Mead and associates come up with a much simplified set of design rules (single page description). A sort of "API" or abstraction of the process (back end processing could automatically convert this information into masks)
- Sufficiently small set that designers could memorize.
- Sufficiently abstract to allow process engineers to shrink the process and preserve existing layouts.
- Process resolution becomes a "parameter", λ.



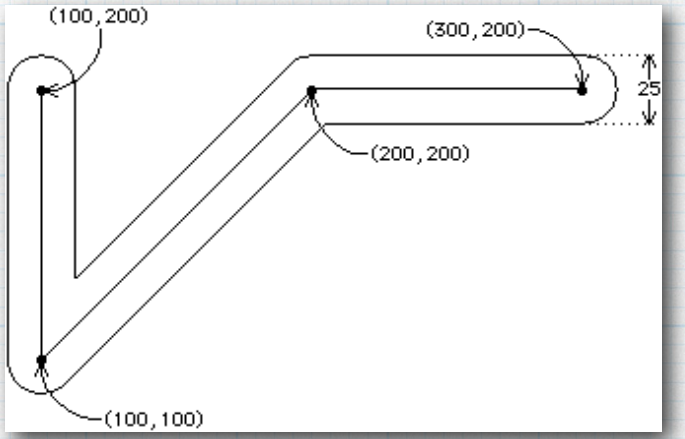
# Scalable CMOS Design Rules

- Created with the transition from nMOS to CMOS (a much nicer technology), around 1985.
- Little changed over the years.



# Design Representations (2)

- Caltech Intermediate Form (CIF)
- Capture layout information, needed to generate masks and process.
- ASCII text file with geometric primitives and hierarchical definitions.
  - Simple and human readable.
  - Easy to generate and parse.
  - Common sub-blocks could be reused from one design to the next (output pad drivers, etc.)




A sample CIF "wire" statement. The statement is:

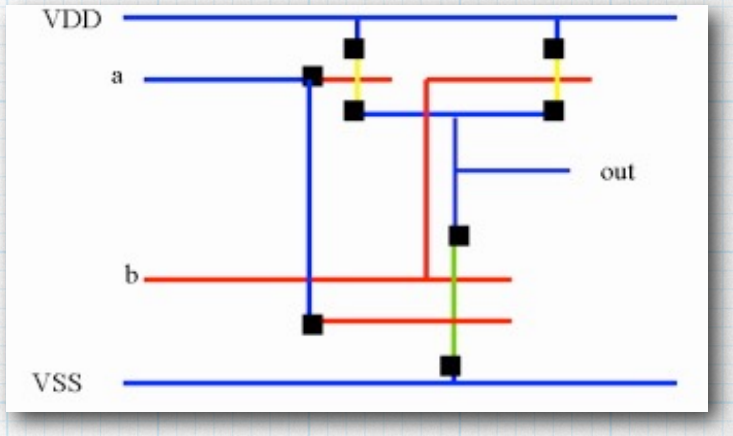
```
W25 100 200 100 100 200 200 300 200;
```

# Design Representations (3)

- Previously, designed were represented by hand drawings. Then masks where made by transferring drawings to rubylith.
  - Base layer of heavy transparent dimensionally stable Mylar. A thin film of deep red cellophane-like material covers the base layer. Patterns formed by cutting (often by hand) the transparent covering.
- Using an electronic format (CIF) meant:
  - Layouts easily stored and transmitted
  - Written to tape and transferred to manufacturer (tape out).
  - Transmitted over the network (new idea back then).
  - Software could automatically check for layout errors.
  - Generated from a program - huge idea.

# Design Representations (3)

- "Simplified" approach extended upward.
- "Sticks" diagrams for layout:
  - Simultaneously captures circuit topology and geometry.
  - Back end tool "fleshes out" real geometry and compacts according




- For functional circuit descriptions, transistors as "switches".
- Simple RC-based and "tau" timing models (later lead to "logical effort")
- Standard simple circuits for common functions. Previously, designers had many tricks, and many alternative circuits.

# Computer Aided Design (1)

- Several advances lead to the development of interactive tools for generating layout:
- Computer based layout representation (CIF).
- Advances in computer graphics (thanks to Ivan Sutherland and friends) and display devices.
- Personal "workstation" (Xerox Alto - Chuck Thacker). "Back room" computers didn't have the necessary bandwidth to the display.
- ICARUS (first such system?)
- Berkeley version - MAGIC

# Computer Aided Design (2)

- For some time after CIF was invented. Layout was generated by hand, then typed in as a CIF file with a text editor.
- Layout compilers
  - Soon some designers started embedding CIF primitives in conventional programming languages: LISP, pascal, fortran, (later) C.
  - This allows designers to write programs that generated layout. Such programs could be parameterized:

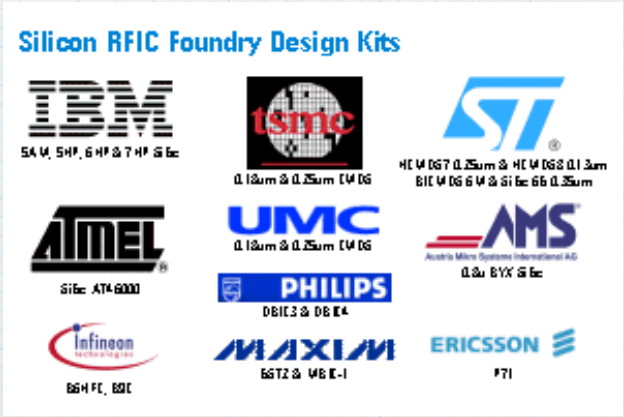
```
define GENERATE_RAM(rows, columns) {
   for I from 1 to rows
      for J from 1 to columns (GENERATE_BITCELL)}
GENERATE_RAM(128, 32);
```

- Lead to circuit/layout generation from higher level descriptions:
  - Bristle-blocks (first "silicon compiler", Dave Johanssen). Generated processor architectures (datapaths) from high level specification
  - Elements: adder, regfile, I/O block, ... Width:16
- Eventually, Cadence and Synopsys formed out of Berkeley.

# Silicon Foundries

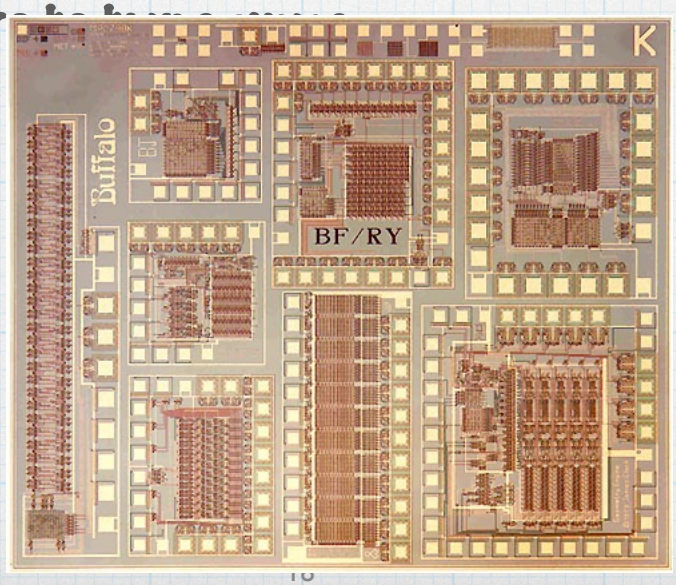
- Separate the designer from the fabricator: Modeled after the printing industry. (Very few authors actually own and run printing presses!)
- Simple standard geometric design rules were the key: these form the "contract" between the designer and manufacturer.
- Designer sends the layout (in CIF format), foundry manufactures the chip and send back. Designer promises not to violate the design rules. Foundry promises to accurately follow layout.

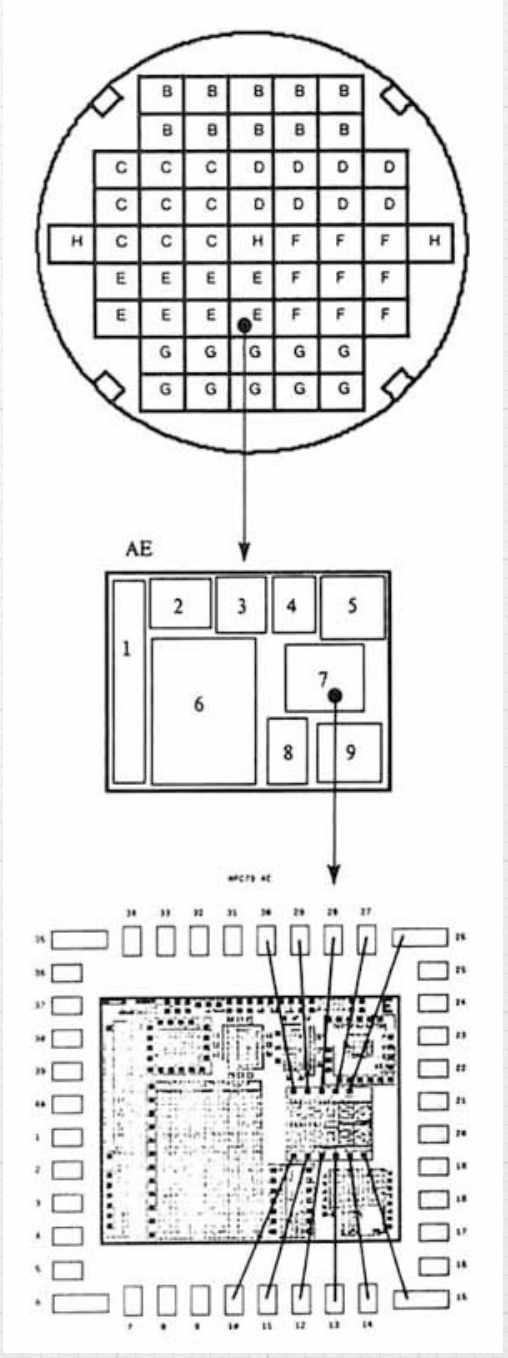
- A scalable model for the industry:
  - IC fab is expensive and complex
  - Amortizes the expense over many designers (batch processing with deep queues help).
  - Designers and companies not held back by need to develop and maintain large expensive factories.
  - "fabless" semiconductor companies - lots of these and very few foundries.

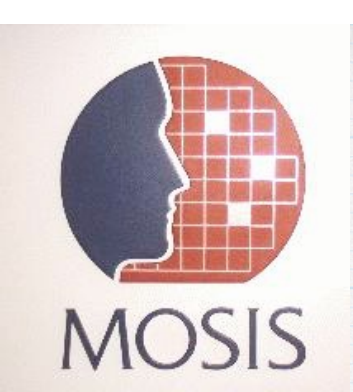


# Multi-project Chips

- Silicon processing is optimized for high-volume.
  - Large minimum order, high fixed-price (overhead), low per unit cost.
- While designing and characterizing new designs (prototyping), what is needed is low-volume low-cost production.
- Multi-project chips allowed multiple designers to share one set of masks, a set of wafer. Brings cost of production down to levels appropriate for pr

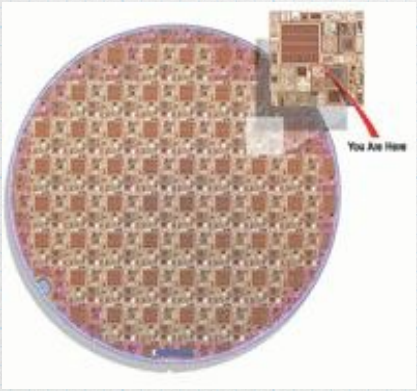




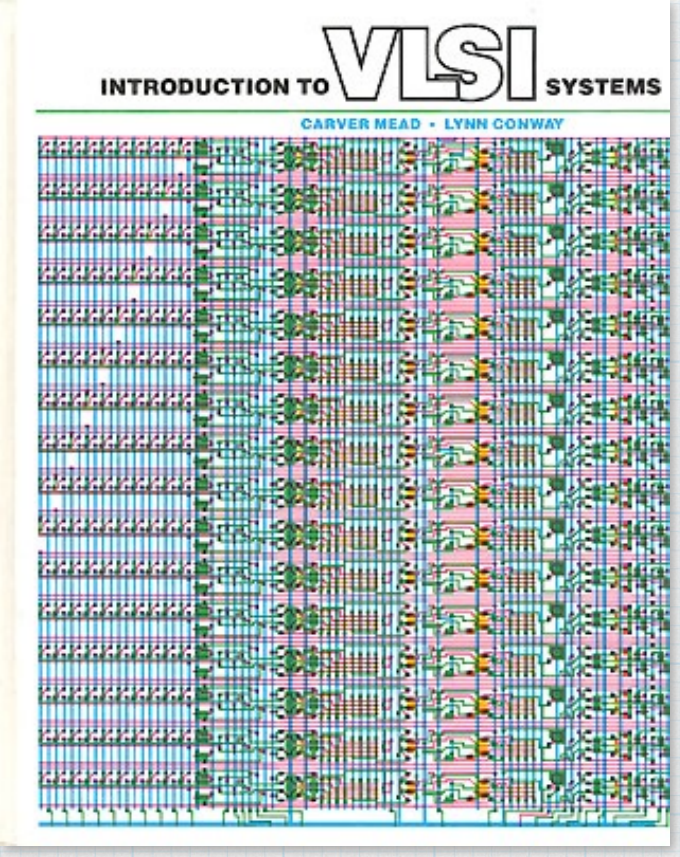

# (MOS implementation Service)

- For many years (1980-1996) fabrication was available (mostly in the form of MPCs) to US universities for free (paid by NSF and DARPA.
- Interestingly, DARPA originally saw this as a useful application of the ARPAnet (later to be known as the Internet). ARPA had invested to put this network together - world-wide-web and email hadn't happened yet, so ARPA was looking for a way to justify their investment.
- The MOSIS project at USC/ISI collected designs from around the country. Designs were FTPed to MOSIS, they brokered their manufacturing with silicon foundries.
- Become THE way to do projects in classes (like CS250) and research.
- Over 50,000 designs prototyped for universities, industry, and government agencies.
- Continues today, subsidized by paying customers, with spare space offered for free to universities.



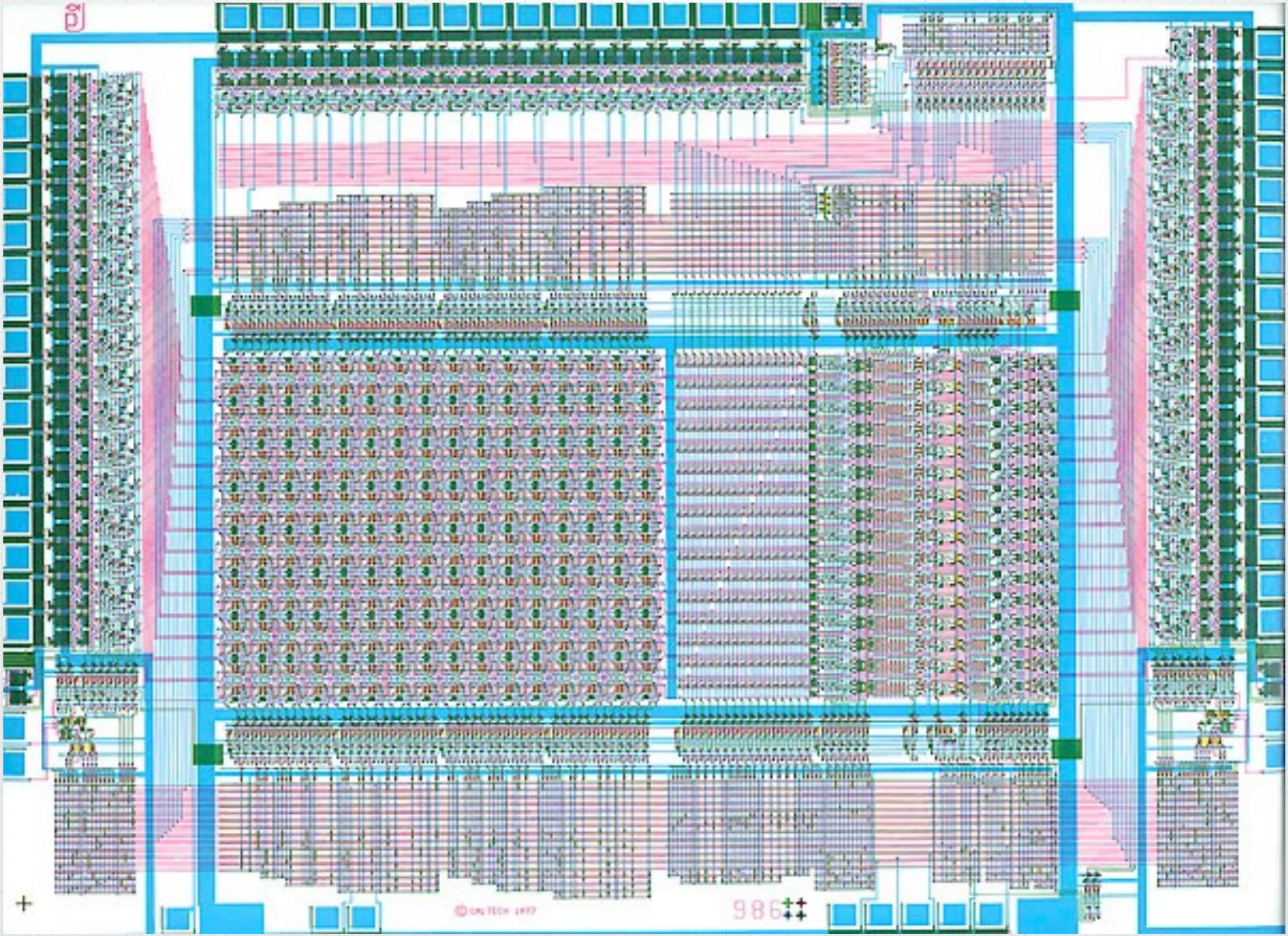
# Education

- The new simple design representations made it easy to teach and learn (even for computer scientists - remember the original targets)
- Text book by Carver Mead and Lynn Conway, 1980.




- Presented elegant clear treatment of physics, processing, circuits, and design methodology for nMOS chips.
- Continued as the standard text, even long after CMOS supplanted nMOS (sadly never revised).
- Key to its success was the large design example
- OM2 design becomes the model for all microprocessor designs.

# 0M2



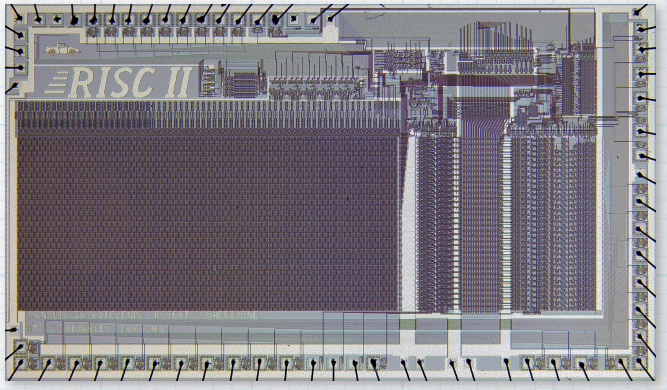
# Spreading the Word

- Limited printing (of chapters 1-3) were used as course notes in 1977 by Mead at Caltech and Carlo Sequin at UC Berkeley.
- Chapters 1-5 1978 by Ivan Sutherland and Amr Mohsen at Caltech, by Bob Sproull at CMU, Frohman-Bentchkowsky at Hebrew University, Jerusalem, and by Fred Rosenberger at Washington University.
- Prepublication of entire book, in fall of 1978, in courses at Caltech and UC Berkeley, and by Kent Smith at the University of Utah, and by Lynn Conway, while visiting MIT.
- Within a few years, this seminal text was adopted for chip design courses at over 100 universities throughout the

# At Berkeley (1)

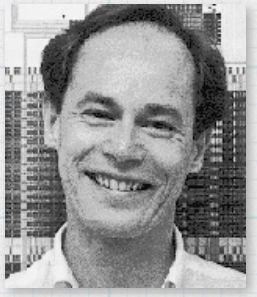
1980-1988: VLSI course continues to be taught by Professors Sequin, Patterson, & Katz.

~1985: Students in advanced version of the course with Sequin and Patterson, design first two RISC processors. Working closely with designers, Prof. Ousterhout develops MAGIC IC design tools.




Late 80's. Patterson returns to architecture focus, Katz to OS/Networking, Sequin to graphics.

1988: Berkeley hires Caltech grad (student of C. Mead) to take over VLSI course. Offers course many times through the 90's.



# At Berkeley (2)

- Through the 1990's ...
- EE141, EE241 develop to cover much of the same material (processing, CMOS devices, circuits, sub-systems) however, 250 continues to be a practical hands-on, experience-with-real-CAD-tools, design-a-real-chip course.
- VLSI chips start to grow in complexity past practical limits of university 1-semester projects (super-scalar 000, etc.).
- Late 90's. Academic teaching/design/research focus shifts to FPGAs. Much shorter "turn-around" time. FPGAs get large and practical for wide range of applications.
- 1999: Most recent CS250 offering as a design course.
- Spring 2007: Offered as a survey course, no design project.
- 2009: Asanovic, Wawrzynek, Lazzaro revive CS250 as a design course. Focus on Verilog synthesis and design space exploration. Strong connections to research agenda.

– *A lot* has changed in 30 years! Many new challenges/opportunities on the way!

- What of the Mead/Sutherland methodology and ideas from 1980 still apply?
- Is there a new more appropriate methodology for the modern era

# Course Format (1)

*The new CS250 (as of Fall 2009)*

- As with course from the '80s, VLSI design for system architects.
  - Focus on common ASIC design methodology:
    - RTL synthesis and standard cell implementation. No transistor level layout.
- Back to a "design centric course". Learn by doing.
  - Requires a lot of infrastructure set up (thanks to Yunsup Lee, Brian Zimmer, Brian Richards)
- Entire class works implementing RISC processors.
  - Many variations on a theme.
  - More details later.

# Course Format (2)

- Most closely related courses:
  - EECS 151/251A - digital design. Prerequisite.
  - CS 152/252 Computer Architecture / Microarchitecture.
  - EE 242 Transistor level circuits and layout.
  - EE 244 Computer Aided Design of ICs (CAD algorithms)

**Course Theme:**
**How do we get the best design results from the standard design flow using tradeoffs in area/performance/energy and exploring microarchitectural alternatives.**

# Course Structure

- Check Website Calendar/Info for details
- Weeks 1-5:
  - Lectures on fundamentals of "ASIC" design
  - Lab exercises to learn CAD tools
- Weeks 6-14:
  - Project related activities
    - Project group presentation (proposal, progress, final report)
      - "private project meetings" : instructors meet in private with groups
- Grading: 5% Class Participation, 25% Labs, 70% Project
- Please, no Laptop/iPad/handheld use in class. We will have a short break midway in each class so you can catchup on email, etc.

# Some Important Dates (tentative)

- Lab 0 Due: Jan 26 (Tu)
- Lab 1 Due: Jan 28 (Th)
- Lab 2 Due: Feb 4 (Th)
- Project Proposal Due: Feb 11 (Th)
- Lab 3 Due: Feb 18 (Th)
- Oral Project Proposals: Feb 23/25 (Tu/Th)
- Lab 4 Due: Mar 3 (Th)
- Oral Project Status: Apr 5/7 (Tu/Th)
- Project Final Presentations: May 3/5 (Tu/Th RRR wk)
- Final Project Report: May 10 (Tu Exam wk)
- These are all hard deadlines, so please budget your time accordingly. Total of 4 late days for labs.

# More Course Details

- Discussion section TBD. Please complete the doodle poll.
  - Very important for tips on doing the labs and project
- You will need to get a named instructional account to log onto our servers installed with the CAD tools.
- Piazza for all Q/A, announcements, etc., check website.
- Instructor office hours on the web.
- Enrollment
  - Undergrad: need to have taken CS150 or EECS151 (or equivalent) with B+ or better.
  - Grad: we assume you have taken undergraduate digital design. If not, see us for remedial materials.
- Design Language
  - For all, we assume Verilog/VHDL experience.
  - However, we will be introducing you to our local hardware design language, call Chisel (under construction.)

# Project Details

- Project groups of 2 people.
- Leverage the existing RISC ISA (RISC-V) processor with co-processor interface:
  - Explore one or more micro-architectural variations to improve performance or energy efficiency (e.g. super pipelined, multi-threading)
  - System Level optimizations:
    - Domain specific accelerators (e.g. Crypto Engine)
    - Chip multi-processor (many-core RISC-V)
  - Other ideas are welcome. Make a good case, and do necessary background work.
- Generate a set of VLSI implementations performing a design space exploration determining the Pareto optimal points in the performance, area, and energy efficiency space.
- Many more details in a few weeks.

# End of Introduction part 1