CS250 VLSI Systems Design

Fall 2009

John Wawrzynek, Krste Asanovic'
with
John Lazzaro
and
Yunsup Lee (TA)

Why the heck is it CS250 and not EE250?

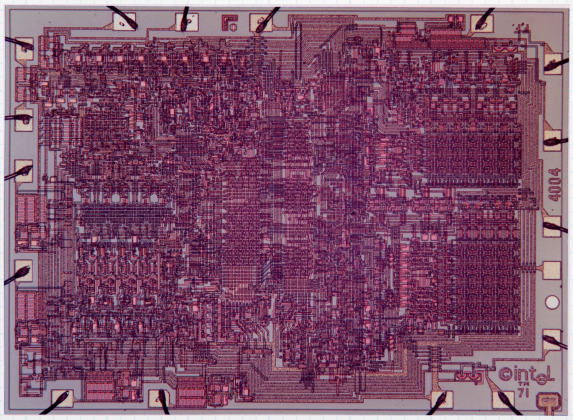
- We answer that with a course history (with a few embedded lessons).

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!

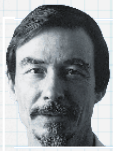


The Intel 4004 microprocessor, which was introduced in 1971. The 4004 contained 2300 transistors and performed 60,000 calculations per second. Courtesy: Intel.

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 technology; find out what it's telling you."

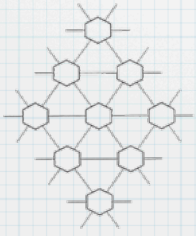
CS At Caltech

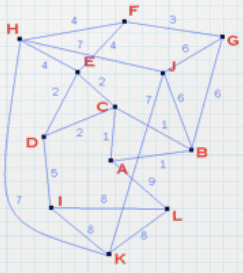
- Ivan Sutherland became founding head of the computer science division at CIT in 1974 (after leaving E&S)
- He and Mead teamed up to get the division off the ground making IC design (Integrated Systems) a key component of the research and teaching.
- My take:

  - 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 (OM1, OM2).

Pushing forward (1)

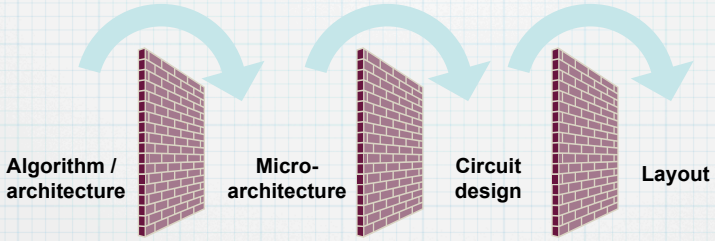
- The reality of integrated circuits:
  - Wires are expensive (area, delay, power), transistors are cheap.
  - Pre-ICs, the opposite was true.
- Therefore, plan the communication and the layout
  - Exploit locality, think about the "geometry" of the problem from the beginning. Choose algorithms/designs 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.

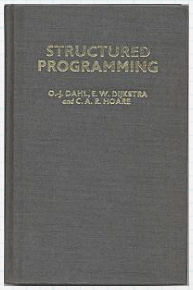
Pushing Forward (3)

- 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

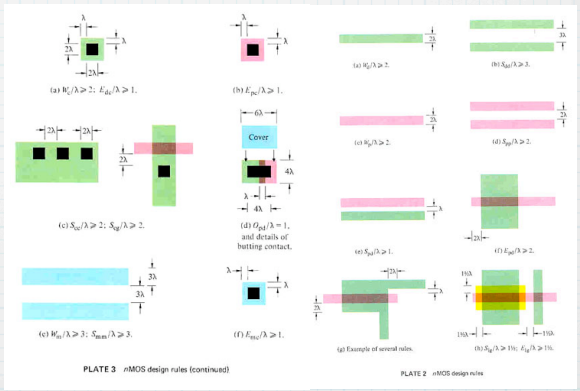
Ideas from Programming
(help manage complexity)

- "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



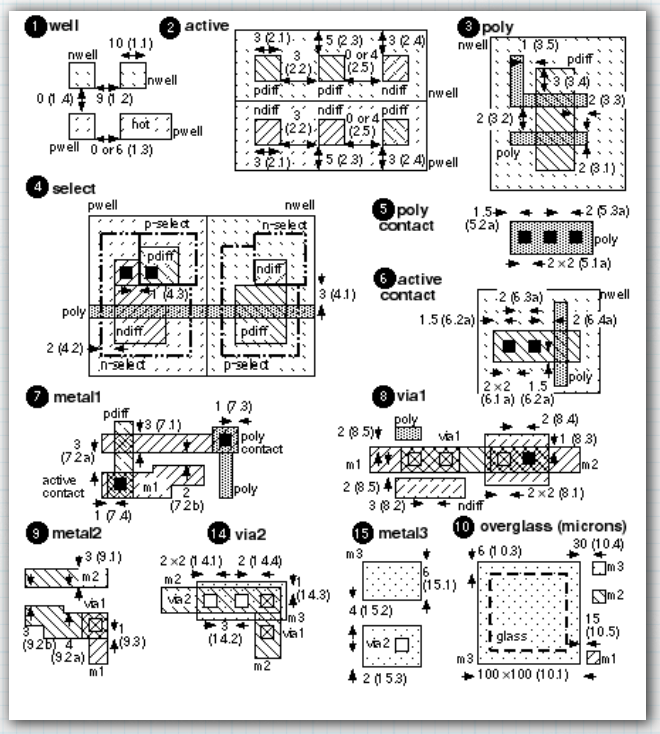
Design Representations (1)

- 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.



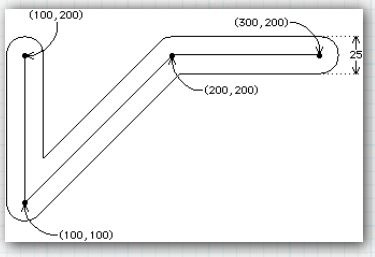
Scalable CMOS Design Rules

- 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 generate and parse.
  - Common sub-blocks could be reused from one design to the next (output pad drivers, etc.)



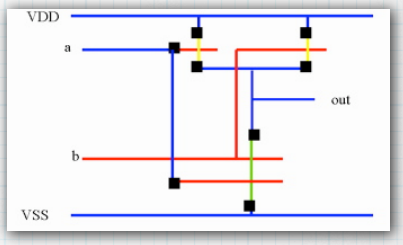
A sample CIF "wire" statement. The statement is:
`W25 100 200 100 100 200 200 300 200;`

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 to geometric design rules.



- 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)

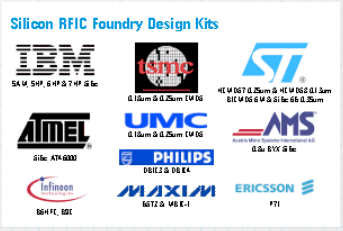
- For some time after CIF was invented. Layout was generated by hand, then typed in as a CIF file with a text editor.
- Layout compilers
  - Soon some designers started embedding CIF primitives in conventional programming languages: LISP, pascal, fortran, (later) C.
  - This allows designers to write 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:1 6
- Eventually, Cadence and Synopsys formed out of Berkeley.

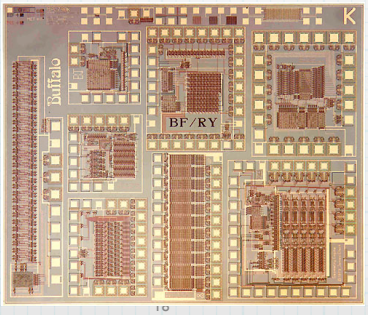
Silicon Foundries

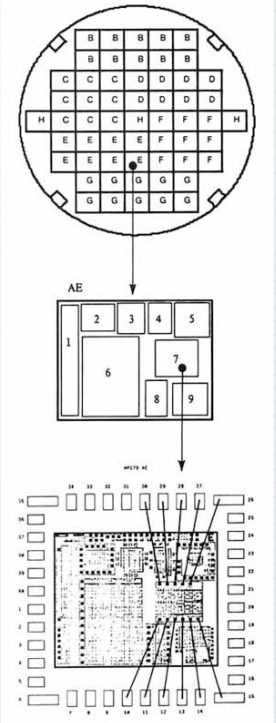
- Separate the designer from the fabricator: Modeled after the printing industry. (Very few authors actually own and run printing presses!)
- Simple standard geometric design rules where the key: these form the "contract" between the designer and manufacturer.
- Designer sends the layout (in CIF format), 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

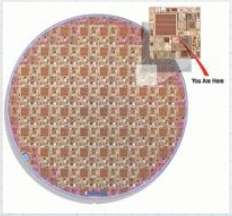
- Silicon processing is optimized for high-volume.
  - Large minimum order, high fixed-price (overhead), low per unit cost.
- While designing and characterizing new designs (prototyping), what is needed is low-volume low-cost production.
- Multi-project chips allowed multiple designers to share one set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





(MOS implementation Service)

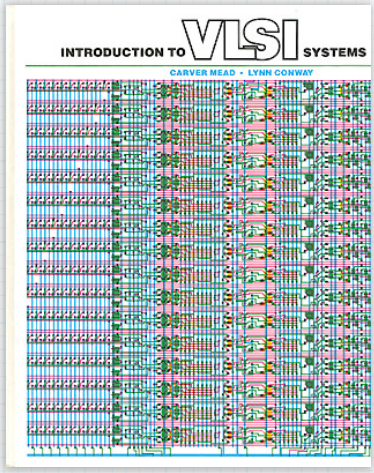
- For many years (1980-1996) fabrication was available (mostly in the form of MPCs) to US universities for free (paid by NSF and DARPA.
- Interestingly, DARPA originally saw this as a useful application of the ARPAnet (later to be known as the Internet). ARPA had invested to put this network together - 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, then 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.

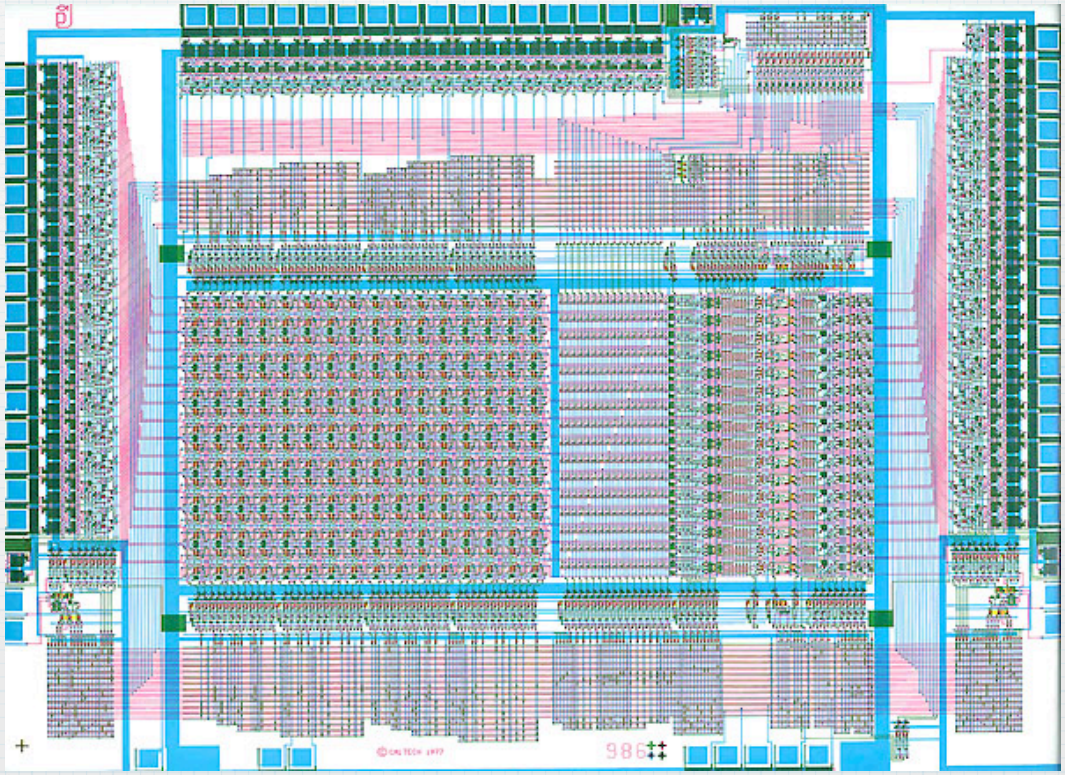
Education

- 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.

OM2



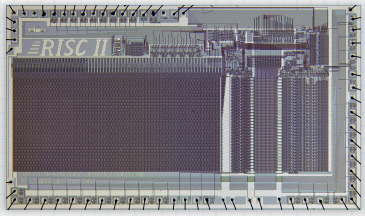
Spreading the Word

- 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 world.

At Berkeley (1)

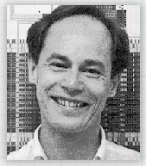
1980-1988: VLSI course continues to be taught by Professors Sequin, Patterson, & Katz.

~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.

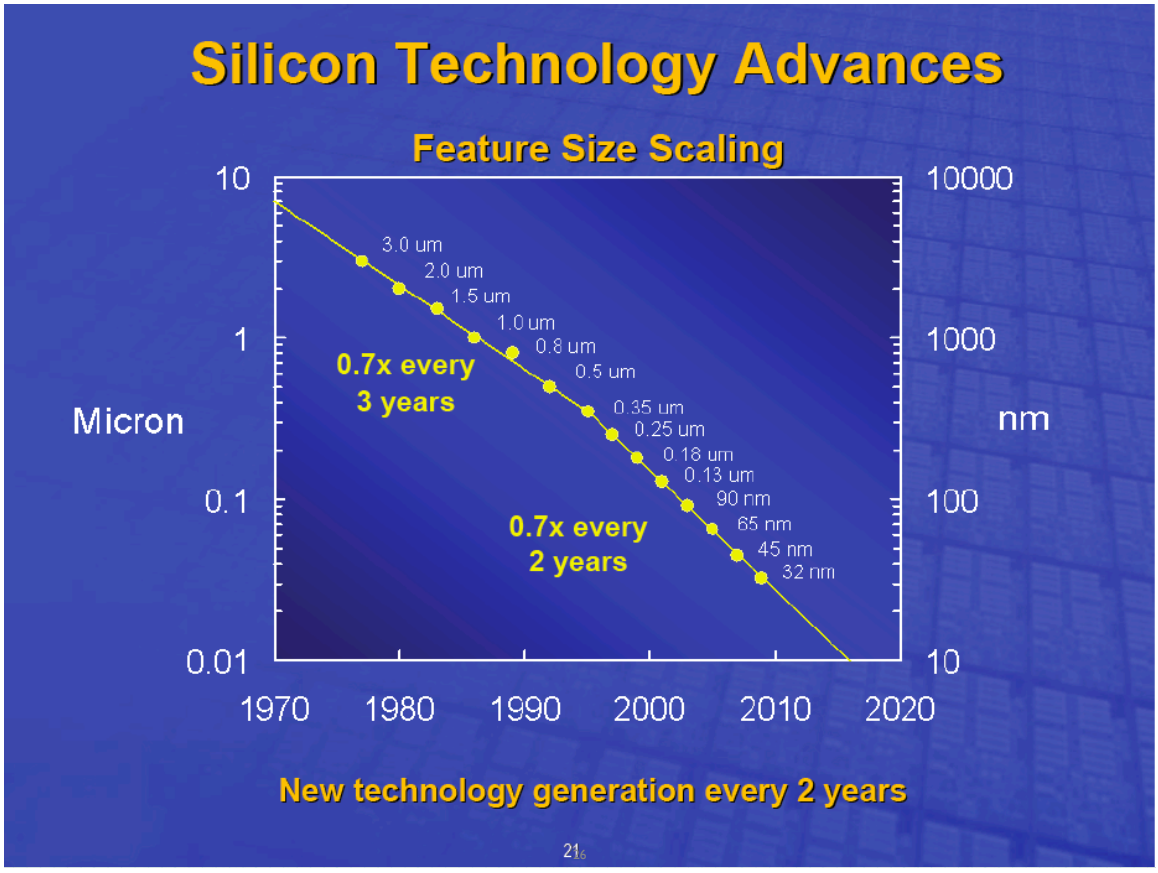
1988: Berkeley hires Caltech grad (student of C. Mead) to take over VLSI course. Offers course many times through the 90's.



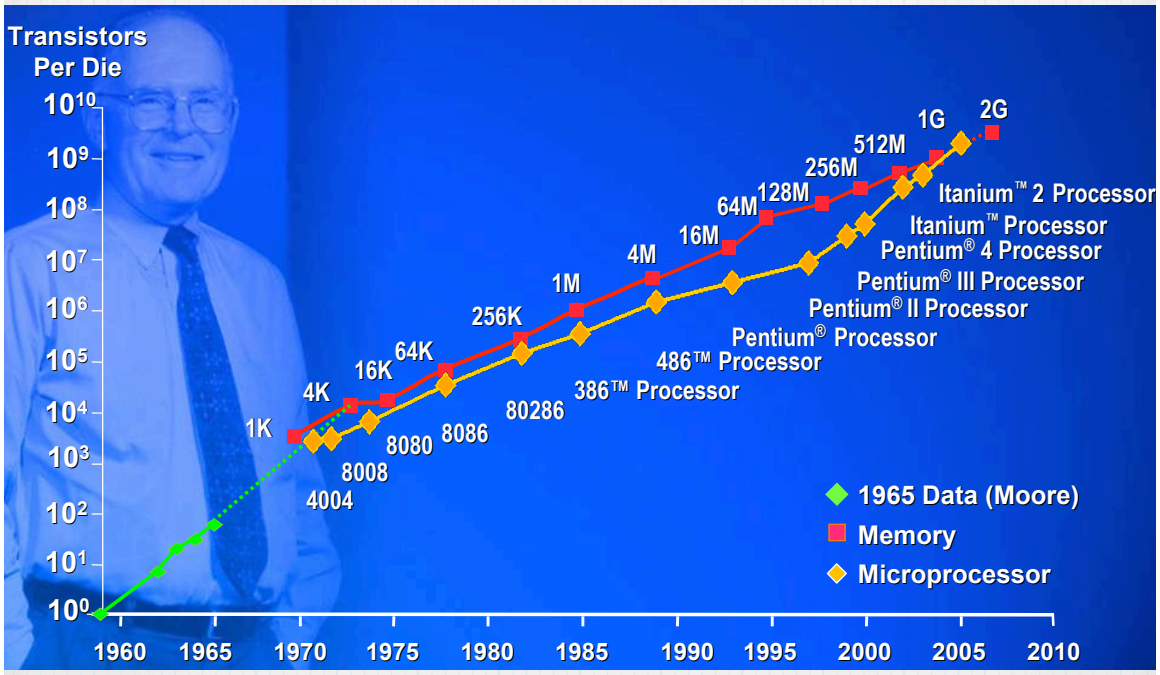
At Berkeley (2)

- 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 OOO, 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.
  - *A lot* has changed in 25 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?

So what *has* changed in 30 years?

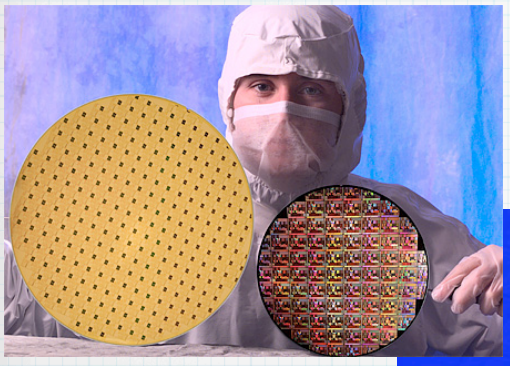


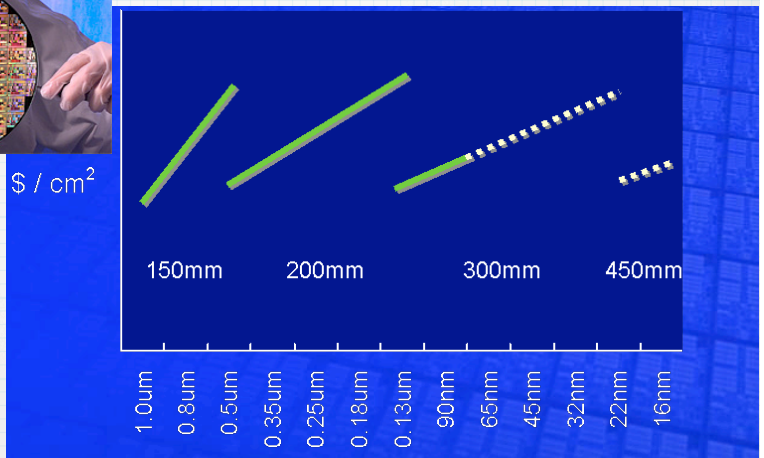
Moore's Law for CPUs and DRAMs



From: "Facing the Hot Chips Challenge Again", Bill Holt, Intel, presented at Hot Chips 17, 2005.

Secondary driver: Wafer size



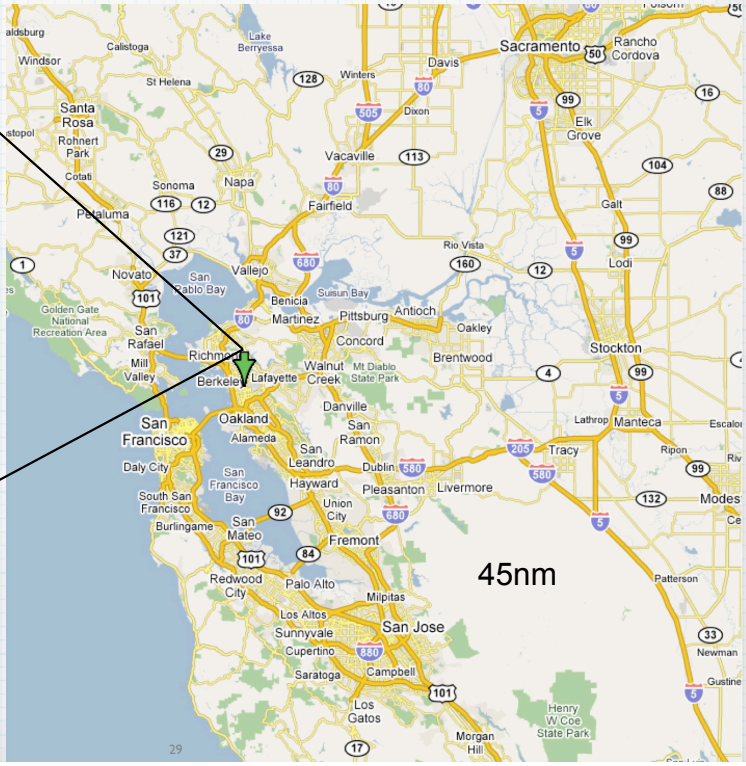


From: "Facing the Hot Chips Challenge Again", Bill Holt, Intel, presented at Hot Chips 17, 2005.

Wafer size conversions offset trend of increasing wafer processing cost

Processing advances

4μm



IC Technology Stuff (1)

- **Feature size:**

  then: ~4μm *now:* ~.045μm

- **Interconnect:**

  then: 2 layers *now:* ~10 layers, *then:* aluminum *now:* copper

- **Transistors:**

  then: planar MOSFET *now:* same

- **Layout and GDRs:**

  Essentially unchanged. More complex. Density and area-fill rules.

- **Circuits:**

  then: clocked static CMOS *now:* same (lots of crazy stuff in between)

Most CMOS circuits and layouts designed in 1980 would work if fabricated on today's IC process.

IC Technology Stuff (2)

- **Transistors:**

  then: near perfect switch *now:* leaky

- **Power consumption:**

  then: dynamic (switching) energy *now:* approaching 50% static leakage (back to the future - nMOS has similar problem)

- **New improved devices on the horizon: FinFETs**
- **Chip Input/Output**

  then: parameter pads *now:* often area pads

- **Lithographic Mask Costs:**

  then: few $k *now:* $M (full die, 65, 45nm)

IC Technology Stuff (3)

- **Device reliability:**

  then: devices nearly never fail *future (<65nm):* high soft and hard error rates

- **Process variations across die, die-to-die:**
  - Statistical variations in processing (wire widths/resitivity, transistor dimensions/strengths, doping inconsistencies) become apparent at smaller geometries.
  - Some circuits fast, others slow. Some high-power, some low.
  - Worst case design results in very bad overall performance.
- **Yield on leading edge processes dropping dramatically**
  - IBM quotes yields of 10 – 20% on Cell processor

Design Stuff

- Chip functionality:

  then: limited by area *now:* often limited by energy dissipation

- Design cost:

  now: design costs in $50M range for full-die custom designs (high percentage in verification)

- Implementation Alternatives: more alternatives that trade up-front design costs for per unit costs.
- FPGA compete aggressively with custom silicon

  then: most custom designs implemented at silicon level

  now: many more custom designs implemented with FPGAs

- Standard design abstraction:

  then: transistors circuits *now:* RTL in HDLs, standard "cores" and standard cells (higher productivity, somewhat less area/ energy efficient) -

Implementation Alternatives

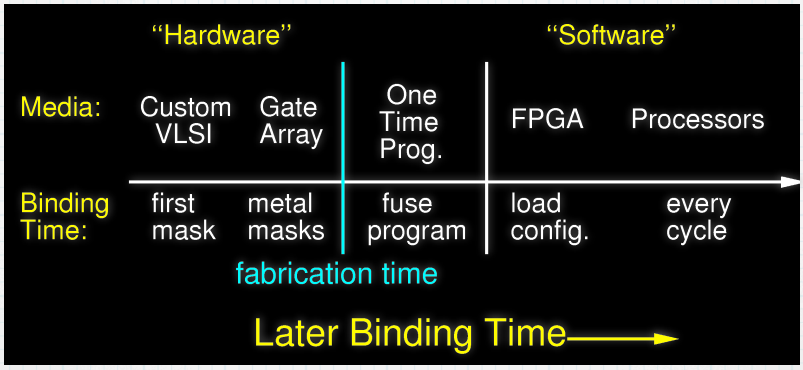
Full-custom:	All circuits/transistors layouts optimized for application.
Standard-cell:	Arrays of small function blocks (gates, FFs) automatically placed and routed.
Gate-array (structured ASIC):	Partially prefabricated wafers customized with metal layers or vias.
FPGA:	Prefabricated chips customized with loadable latches or fuses.
Microprocessor:	Instruction set interpreter customized through software.
Domain Specific Processor:	Special instruction set interpreters (ex: DSP, NP, GPU).

By "ASIC", most people mean "Standard-cell" based implementation.

*Wh*at are the important metrics of comparison?

The Important Distinction

- Instruction Binding Time
  - When do we decide what operation needs to be performed?



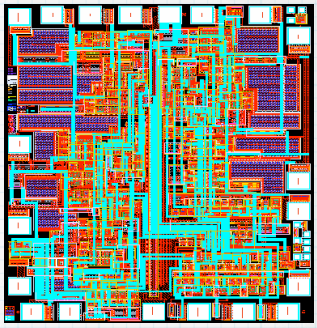
A. DeHon

- **General Principles**

Earlier the decision is bound, the less area, delay/energy required for the implementation.

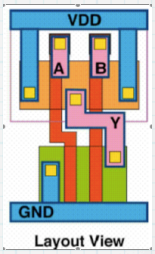
Later the decision is bound, the more flexible the device.

Full-Custom

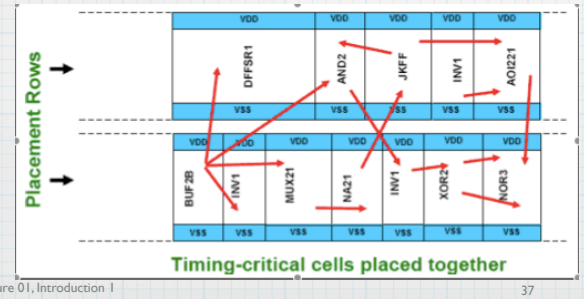


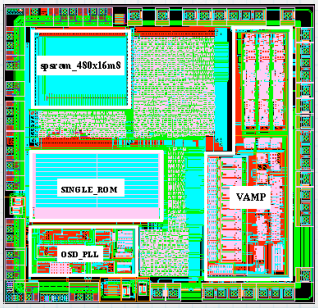
- Circuit styles and transistors are custom sized and drawn to optimize die, size, power, performance.
- High NRE (non-recurring engineering) costs
  - Time-consuming and error prone layout
- Optimizing for small die can result in low per unit costs, extreme-low-power, or extreme-high-performance.
- Common for analog design.
- Requires full set of custom masks.
- High NRE usually restricts use to high-volume applications/markets or highly-constrained and cost insensitive markets.

Standard-Cell*



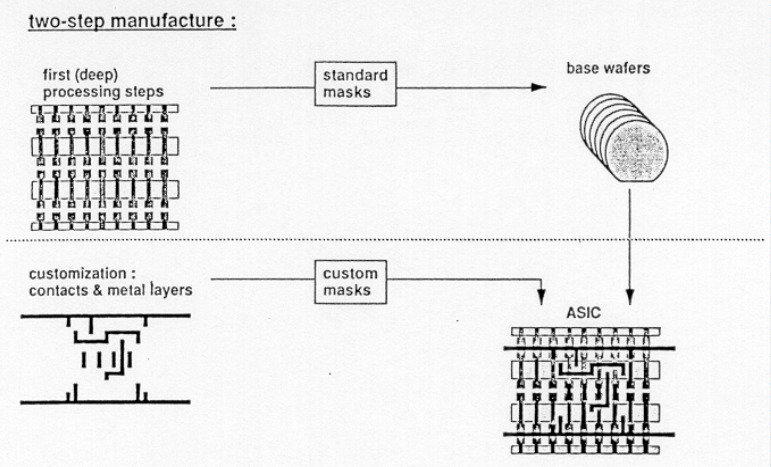
- Based around a set of pre-designed (and verified) cells
  - Ex: NANDs, NORs, Flip-Flops, counters, buffers, ...
- Each cell comes complete with:
  - layout (perhaps for different technology nodes and processes),
  - Simulation, delay, & power models.
- Chip layout is automatic, reducing NREs (usually no hand-layout).
- Requires full set of masks - nothing prefabricated.
- Non-optimal use of area and power, leading to higher per die costs than full-custom.
- Commonly used with other design implementation strategies (large blocks for memory, I/O blocks, etc.)





Gate Array

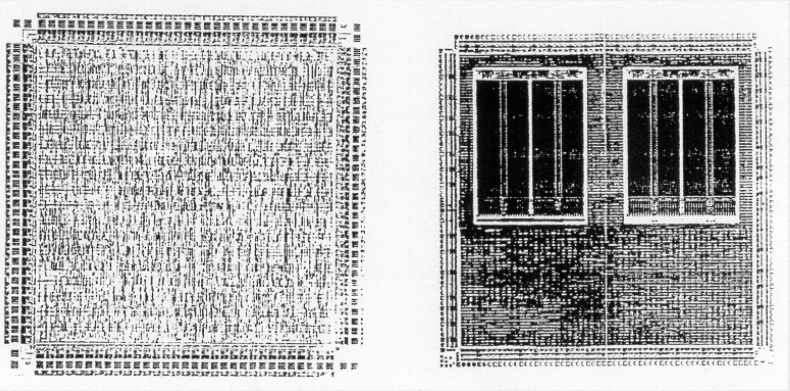
- Store prefabricated wafers of "active" & gate layers & local interconnect, comprising, primarily, rows of transistors. Customize as needed with "back-end" metal processing (contact cuts, metal wires). Could use a different factory.
- CAD software understands how to make gates, but also possible to customize at the transistor circuit level.



Gate Array

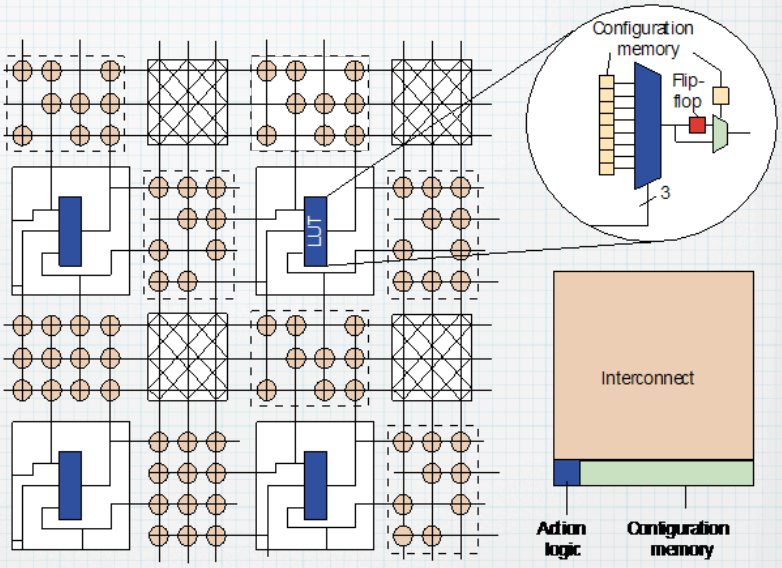
- Shifts large portion of design and mask NRE to vendor.
- Shorter design and processing times, reduced time to market.
- Highly structured layout with fixed size transistors leads to large sub-circuits (ex: Flip-flops) and higher per die costs.
- Memory arrays are particularly inefficient, so often prefabricated, also:

Sea-of-gates, structured ASIC, master-slice.



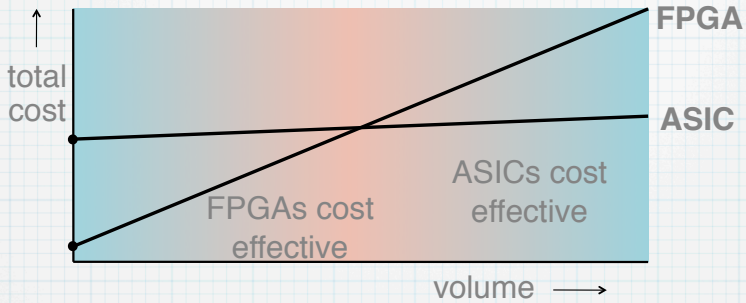
Field Programmable Gate Arrays

- Two-dimensional array of simple logic- and interconnection-blocks.
- Typical architecture: LUTs implement any function of n-inputs (n=3 in this case).
- Optional Flip-flop with each LUT.



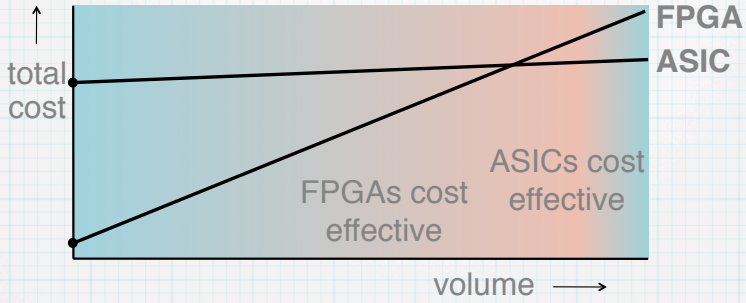
- Fuses, EPROM, or Static RAM cells are used to store the "configuration".
  - Here, it determines function implemented by LUT, selection of Flip-flop, and interconnection points.
- Many FPGAs include special circuits to accelerate adder carry-chain and many special cores: RAMs, MAC, Enet, PCI, SERDES, ...

Traditional FPGA versus ASIC argument



- ASIC: High NRE costs ($2M for 0.35um chip). Relatively Low cost per die.
- FPGAs: Very low NRE costs. Relatively low silicon efficiency ⇒ high cost per part.
- Cross-over volume from cost effective FPGA design to ASIC in the 10K range.

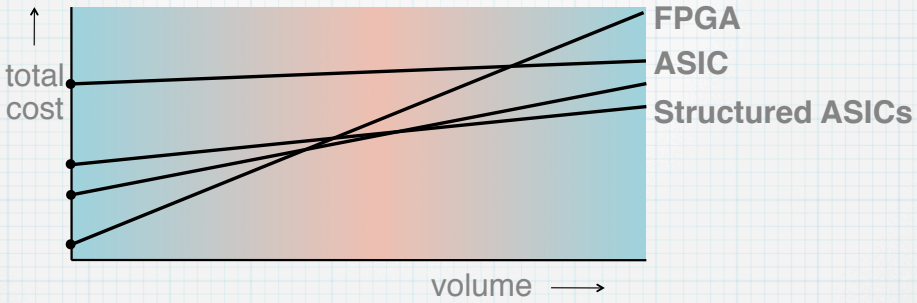
Cross-over Point is Moving Right



- ASIC: Increasing NRE costs ($40M for 90nm chip[1]) (verification, mask costs[2], etc.)
  - Fewer silicon designs becomes inevitable.
- FPGAs: Move in to fill the need, furthermore, FPGAs better able to follow Moore's Law, relatively cheaper to test.
- Cross-over volume now in the 100K range.

[1] Vahid Manian, VP manufacturing and operations, Broadcom Corp.
[2] Roger Minear, Agere Systems Inc, 30- 35- layer mask set ≈$650,000 for 130nm and $1.4M for 90nm.

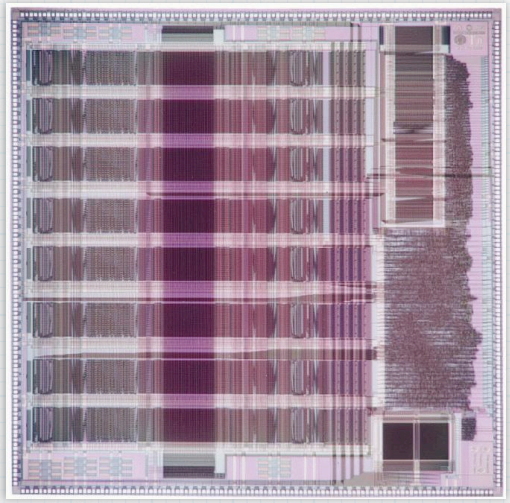
Post-fabrication Customization



- Gate Array like devices (structured ASICs) return to fill the gap. Post-fab customization with limited mask layers or direct-write e-beam.
  - Lower NREs than ASICs, more silicon efficiency than FPGAs.

Hybrids Chip Implementations Abound

- Ex: standard practice in microprocessors that data-paths are full-custom and control (instruction decode, pipeline control) in standard-cells. (Less common recently)

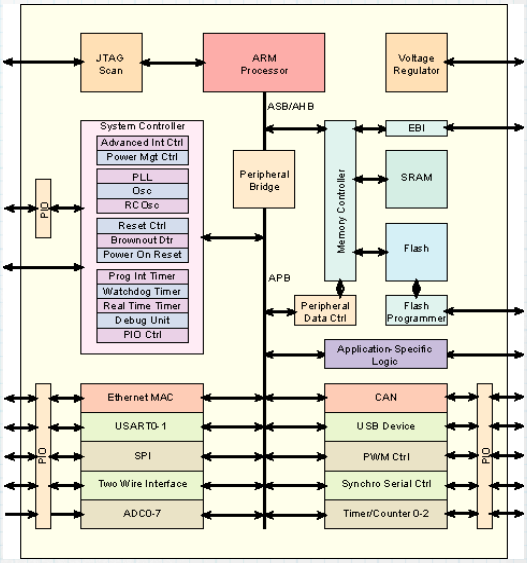


Control ("random") logic difficult to "regularize". Relatively small percentage of die area/power. Permits late binding of design changes.

Extra NAND or NOR gates were often added to control section, and some wafers left without metallization, to permit late design fixes through metal mask revisions (gate-array idea).

System-on-chip (SOC)

- Brings together: standard cell blocks, custom analog blocks, processor cores, memory blocks, embedded FPGAs, …
- Standardized on-chip buses (or hierarchical interconnect) permit "easy" integration of many blocks.
  - Ex: AMBA, Sonics, …
- "IP Block" business model: Hard- or soft-cores available from third party designers.
- ARM, inc. is the shining example. Hard- and "synthesizable" RISC processors.
- ARM and other companies provide, Ethernet, USB controllers, analog functions, memory blocks, …



- Pre-verified block designs, standard bus interfaces (or adapters) ease integration - lower NREs, shorten TTM.

SIP, SOP, MCM interesting alternatives.

Early '80's Design Methodology and Flow

- Schematic + Full-Custom Layout

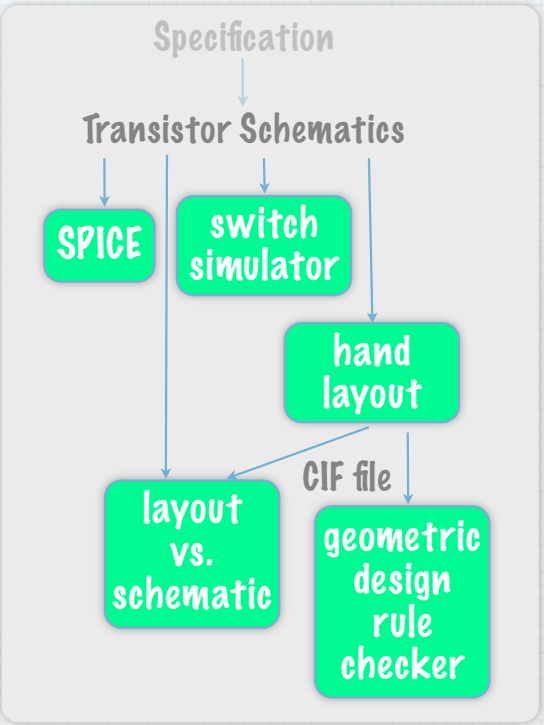
SPICE for critical path,

switch-level simulation for overall functionality,

hand layout,

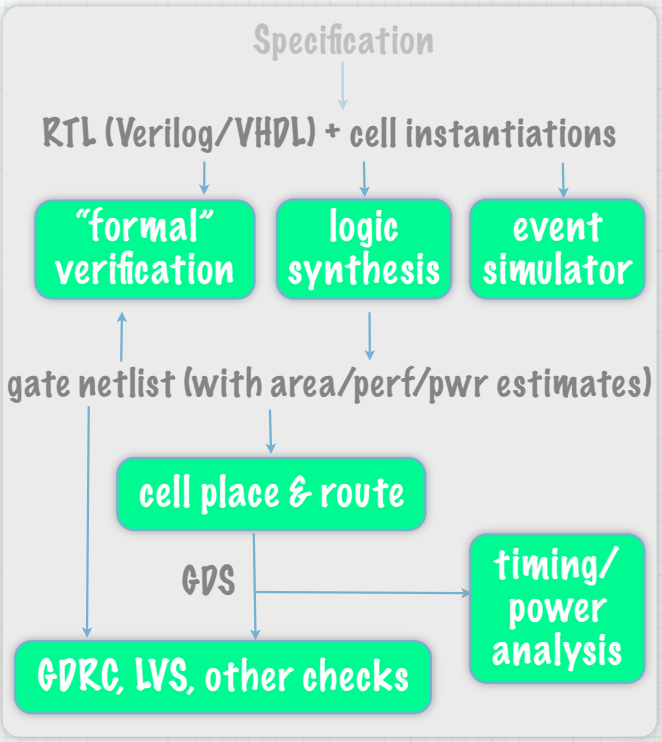
no power analysis,

layout verified with LVS and GDRC



Modern ASIC Methodology and Flow

- **RTL Synthesis Based**
  - HDL specifies design as combinational logic + state elements
  - Cell instantiations needed for blocks not inferred by synthesis (typically RAM)
  - Event simulation verifies RTL
  - "Formal" verification compares logical structure of gate netlist to RTL
  - Place & route generates layout
  - Timing and power checked statically
  - Layout verified with LVS and GDRC



Course Format (1)

The new CS250

- 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!)
- Entire class works on the pieces of a large advanced chip design.
  - Prototype of Parlab Infinicore architecture.
  - More details later.

Course Format (2)

- Most closely related courses:
  - CS 150 - undergraduate digital design. Prerequisite.
  - CS 152/252 Computer Architecture / Microarchitecture.
  - EE 141/242 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 7-8:
  - Lectures on fundamentals of "ASIC" design
  - Lab exercises to learn CAD tools
- Weeks 9-16:
  - Class project meetings/reviews
  - Paper reading/discussions
  - Group projects (3 people per group)
  - Guest speakers.
- Grading
  - 15% Paper Summaries and Class Discussion
  - 15% Labs
  - 70% Project

Course Details

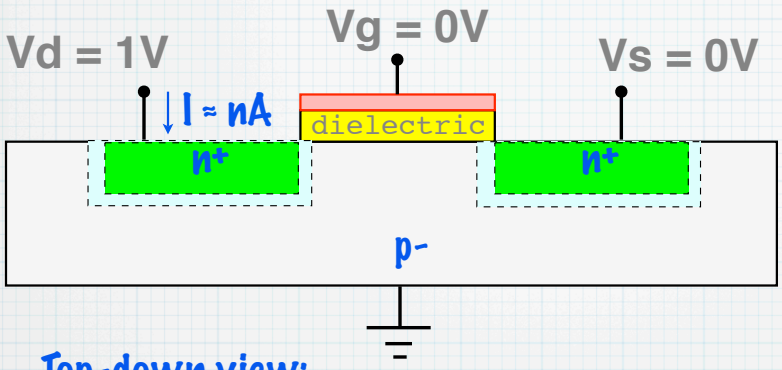
- Discussion section Fridays 2-3pm Fridays, 320 Soda.
  - Very important for tips on doing the labs and project
- First lab assignment out tomorrow, due Tuesday Sept 8 (1.5 weeks).
  - Lots of work, start early.
  - You will need to get an named instructional account.
- Generally, you get 4 "late days" for lab assignments.
  - No late days can be spent on the lab 1.
- Instructor office hours on the web.
  - Lazzaro by appointment.
- Enrollment
  - Undergrad: need to have taken cs150 (or equivalent) with B+ or better.
  - Grad: we assume you have taken undergraduate digital design. If not, see us for remedial materials.

End of Introduction

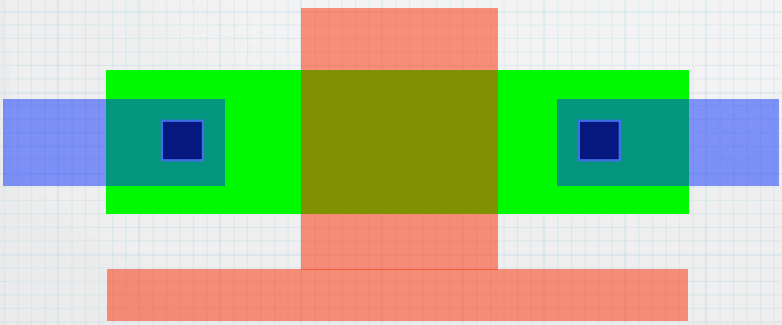
IC Fabrication and Layout Representation

"Mask" drawings sent to the fabrication facility to make the chips.

Mask set for an n-Fet (circa 1986)



Top-down view:



Masks

#1: n+ diffusion

#2: poly (gate)

#3: diff contact

#4: metal

Layers to do p-Fet not shown. Modern processes have 6 to 10 metal layers (or more) (in 1986: 2).

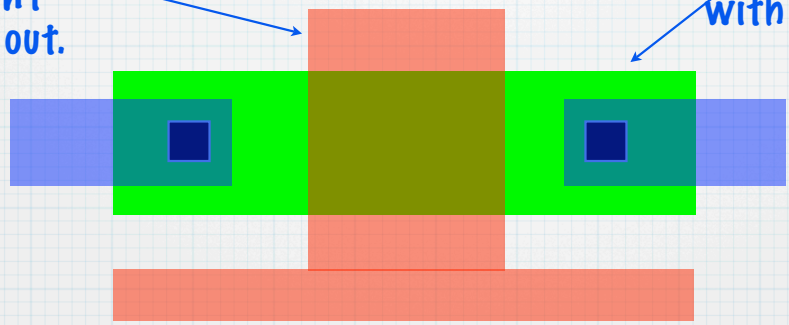
"Design rules" for masks, 1986 ...

Poly overhang. So that if masks are misaligned, channel doesn't short out.

Minimum gate length. So that the source and drain depletion regions do not meet!

length

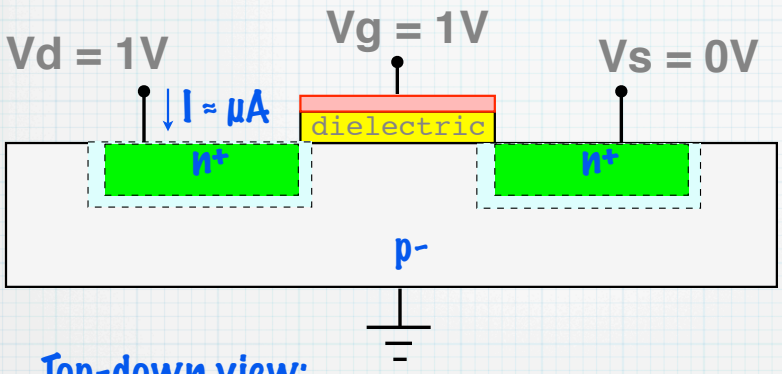
Metal rules: Contact separation from channel, one fixed contact size, overlap rules with metal, etc ...

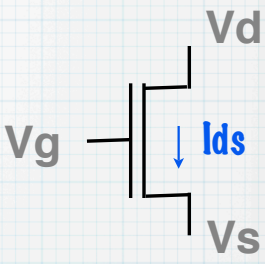


#1: n+ diffusion
#2: poly (gate)
#3: diff contact
#4: metal

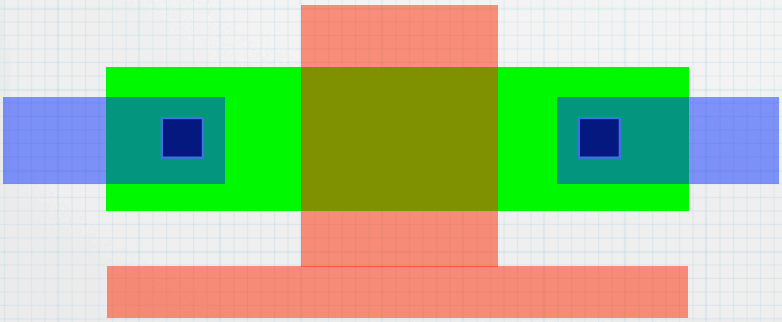
Fabrication

Mask set for an n-Fet ...





Top-down view:



Masks

#1: n+ diffusion

#2: poly (gate)

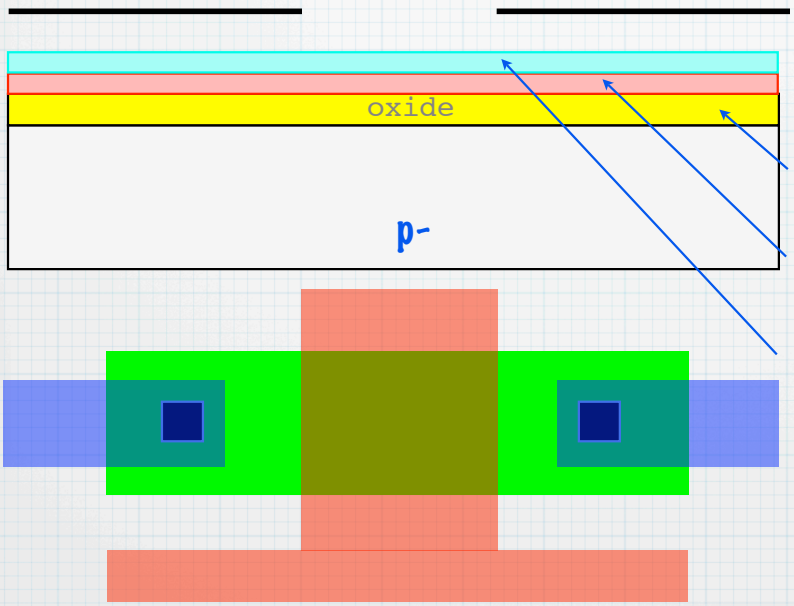
#3: diff contact

#4: metal

How does a fab use a mask set to make an IC?

Start with an un-doped wafer ...

UV hardens exposed resist. A wafer wash leaves only hard resist.



Steps

#1: dope wafer p-

#2: grow gate oxide

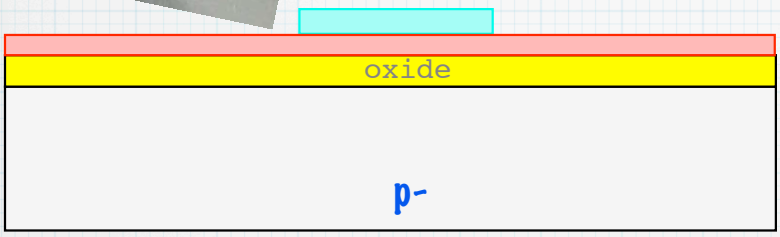
#3: deposit undoped polysilicon

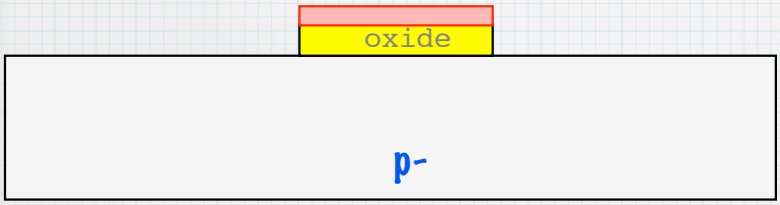
#4: spin on photoresist

#5: place positive poly mask and expose with UV.

Wet etch to remove unmasked ...

HF acid etches through poly and oxide, but not hardened resist.

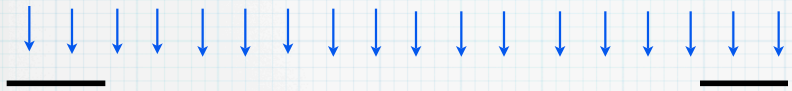




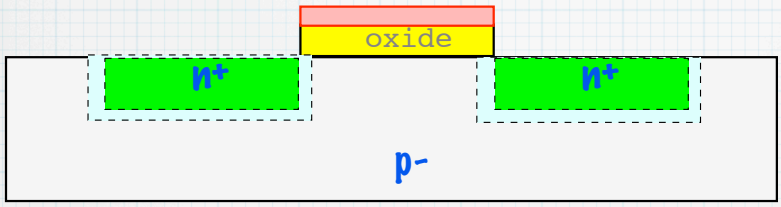
After etch and resist removal

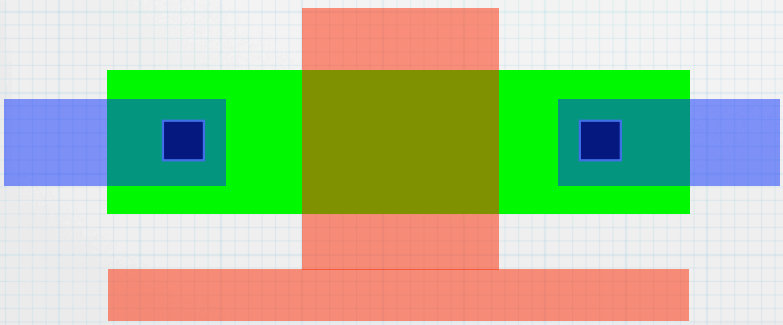
Use diffusion mask to implant n-type

accelerated donor atoms



Arsenic
As
74.922

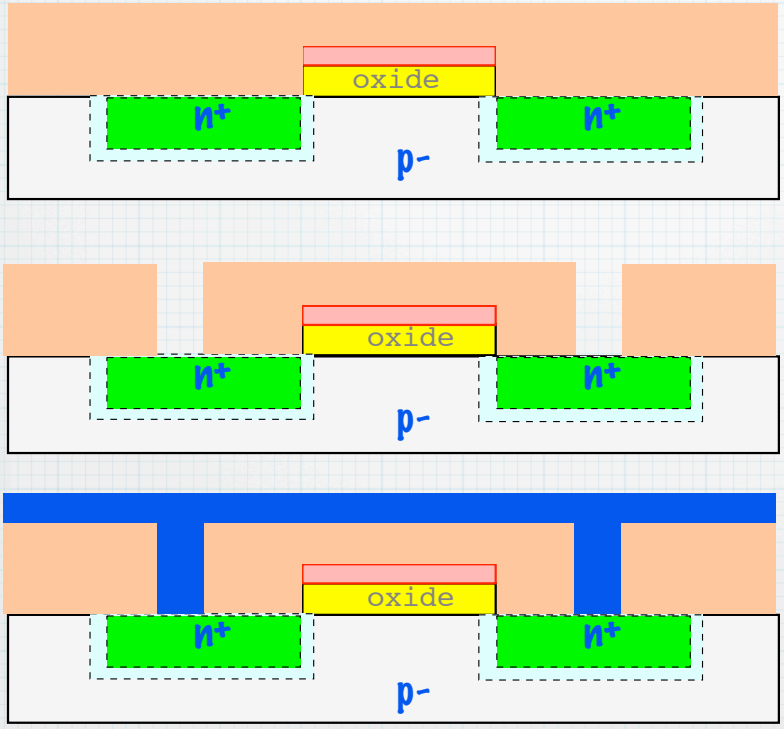




Notice how donor atoms are blocked by gate and do not enter channel.

Thus, the channel is "self-aligned", precise mask alignment is not needed!

Metallization completes device

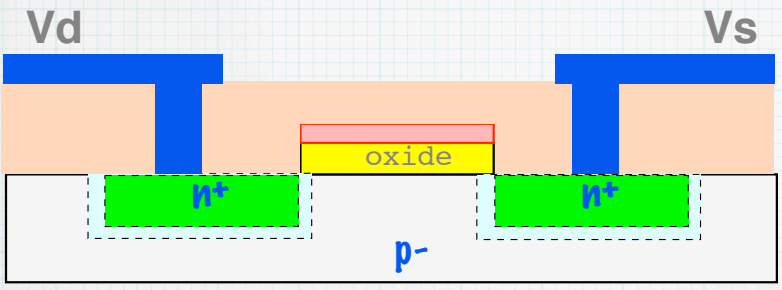


Grow a thick oxide on top of the wafer.

Mask and etch to make contact holes

Put a layer of metal on chip. Be sure to fill in the holes!

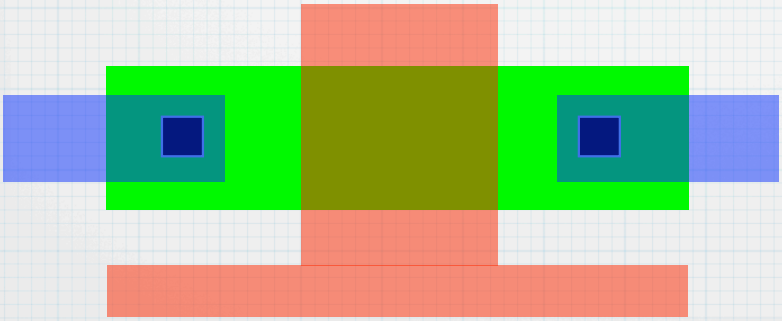
Final product ...



"The planar process"

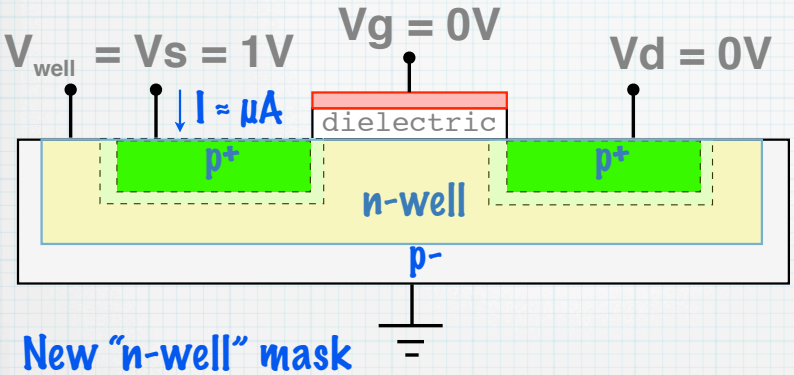
Jean Hoerni, Fairchild Semiconductor 1958

Top-down view:

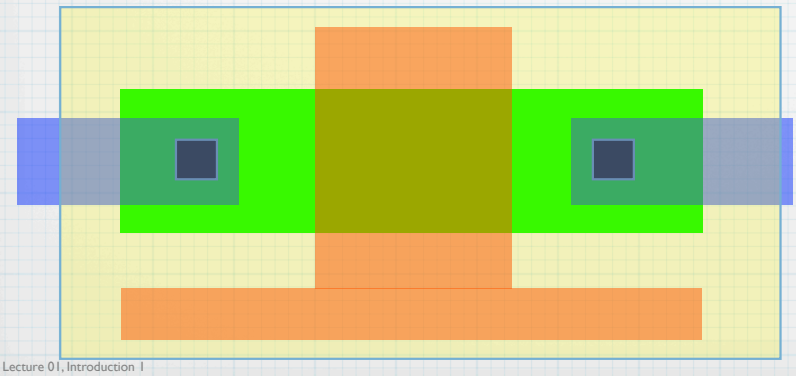


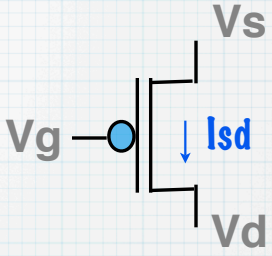
p-channel Transistors

p-Fet: Change polarity of everything



New "n-well" mask





"Mobility" of holes is slower than electrons.

p-Fets drive less current than n-Fets, all else being equal