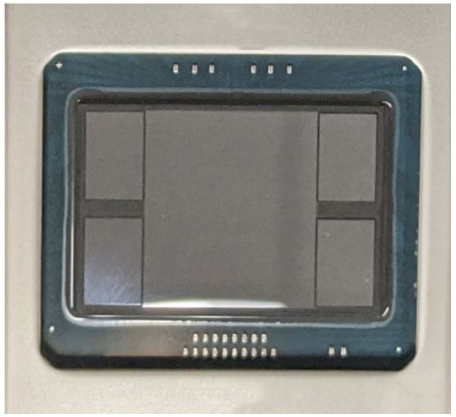
EECS151/251A : Introduction to Digital Design and ICs

Lecture 3 – Design Metrics

In April 2023, Google announced that TPU V4 has been deployed in production since 2020. At HotChips'23 they presented additional details about the design

7nm process

- Optimized for training, superset of TPU v4i:
  - Two TPUv4i Tensorcores
  - 2X HBM of TPUv4i
  - 3D vs. 2D torus
- 275 peak TFLOPS
  - BF16 with FP32 accumulation
  - Also supports int8 like TPUv4i
- Typical power ~200W



N. Jouppi, A. Swing, HotChips'23

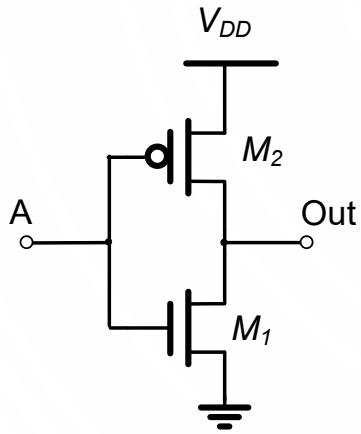
Review

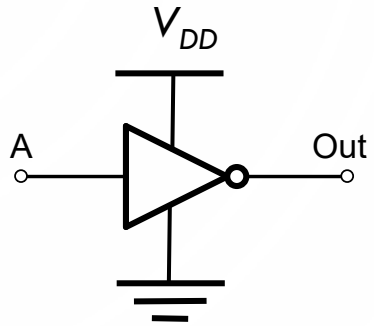
- Design and verification have their distinct phases in project development
  - Specification, architecture, RTL, physical design, test
- Boolean logic
  - Reviewed combinational and sequential logic
  - Implemented in CMOS

Design Metrics: Robustness

What Makes Circuits Digital?

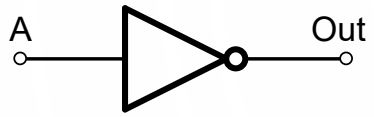
- Chips are noisy
- Supply noise will appear at the output of the logic gate

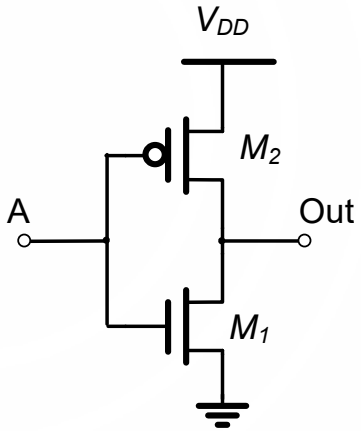




- The following logic gate should still interpret its inputs as 0s and 1s
- This necessary property is called "Restoration" or "Regeneration"
- A lot of money was spent in the past to unsuccessfully make logic out of non-regererative gates
  - Some of emerging CMOS replacements don't have gain…

Beneath the Digital Abstraction

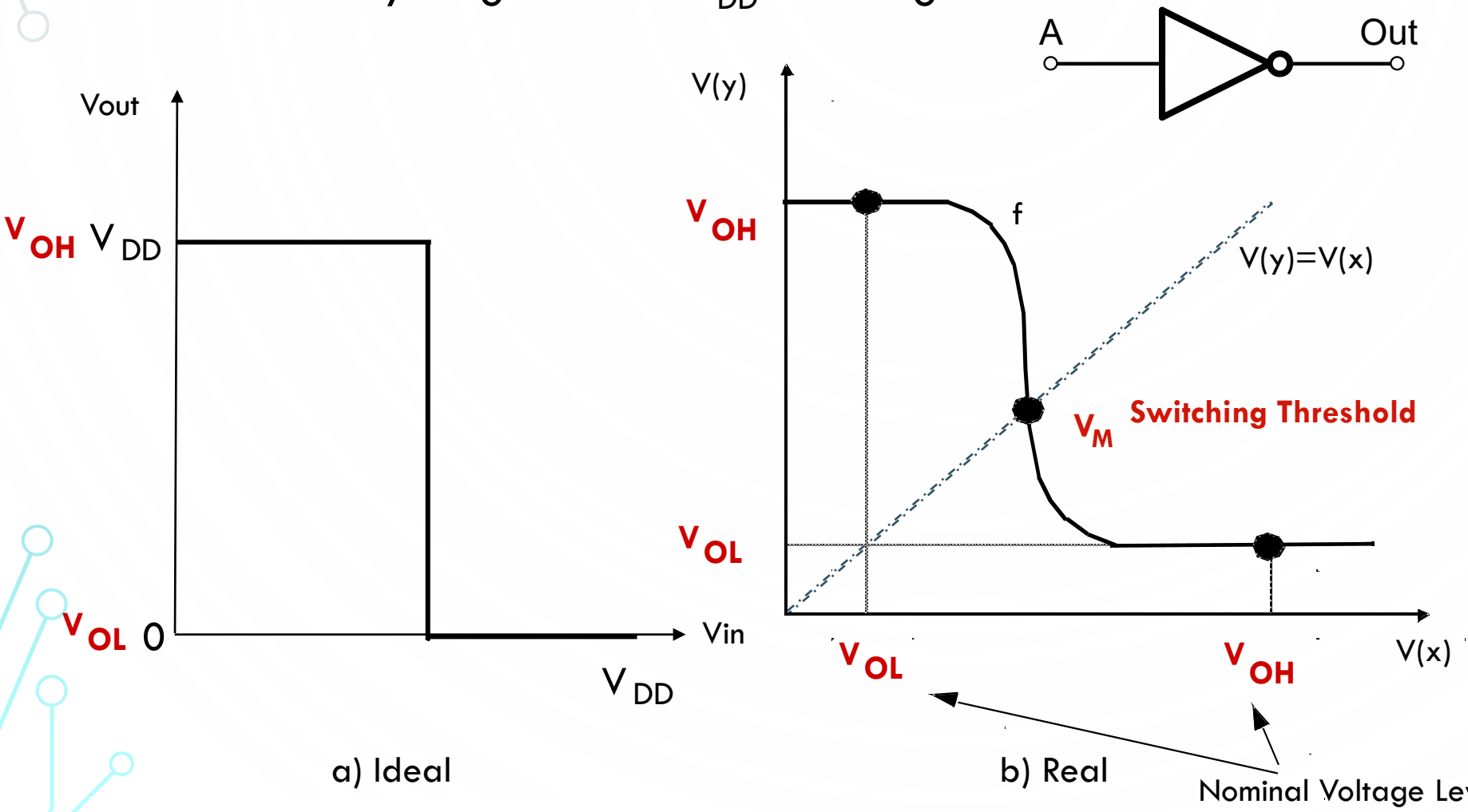




- Logic levels:
  - Mapping a continuous voltage onto a discrete binary logic variable
  - Low (0): $[0, V_L]$
  - High (1): $[V_H, V_{DD}]$
  - V_L V_H: nominal voltage levels

Voltage Transfer Characteristic

- A gate should interpret everything that is close to 0V as a logic 0
  - And everything close to V_{DD} as a logic 1

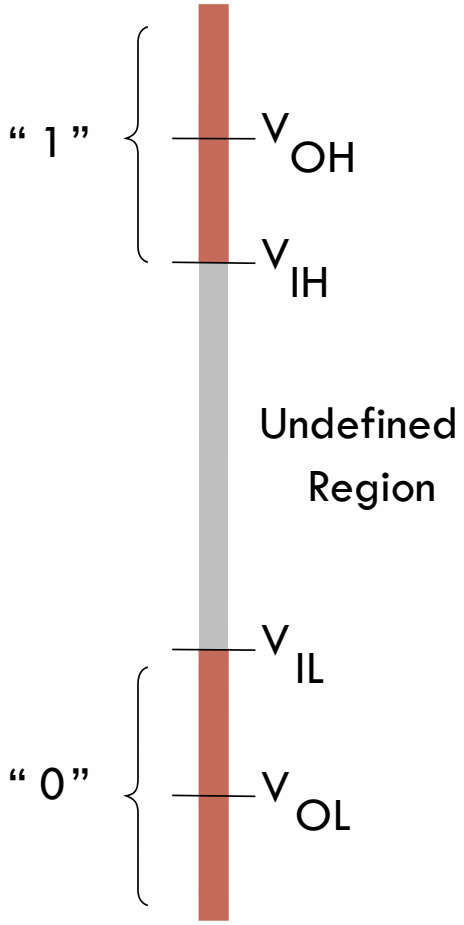


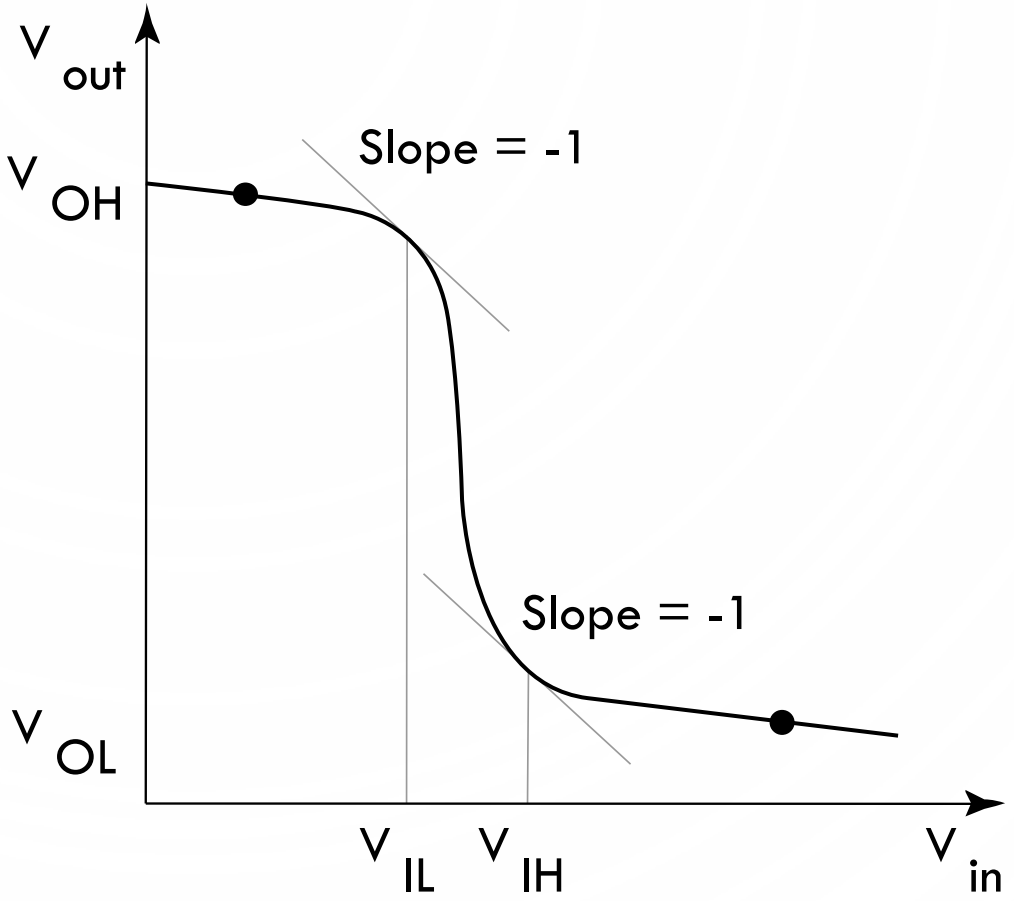
$V_{OH} = f(V_{OL})$
$V_{OL} = f(V_{OH})$
$V_M = f(V_M)$

In CMOS:

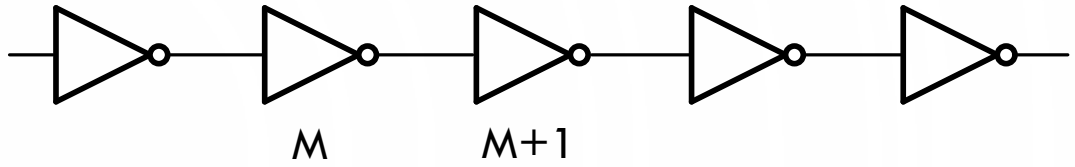
$V_{OH} = V_{DD}$
$V_{OL} = 0$
$V_M \sim V_{DD}/2$

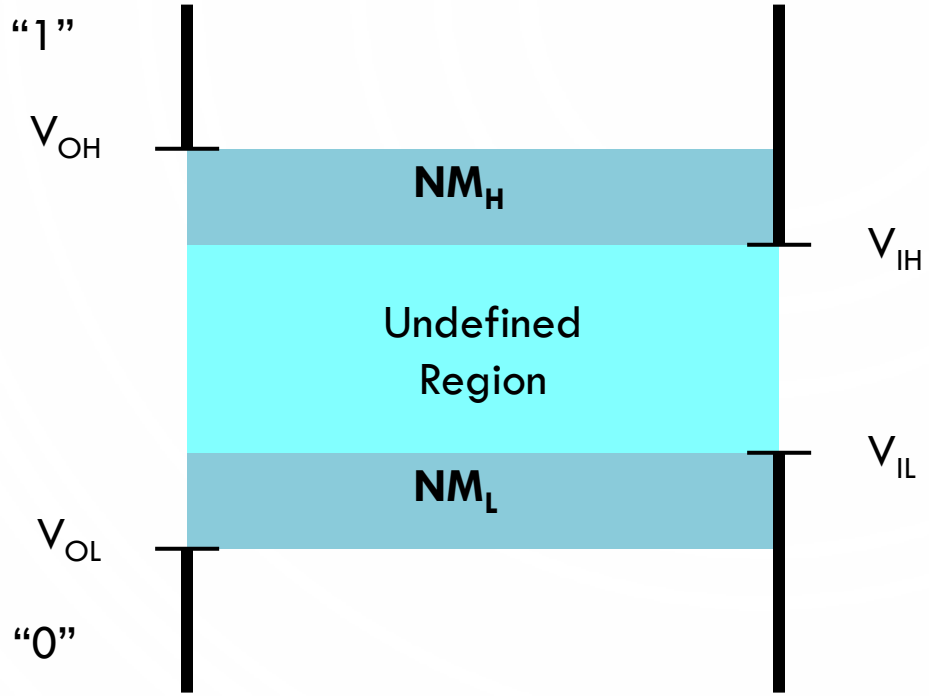
Mapping Between Analog Voltages and Digital Signals





Definition of Noise Margins





Noise margin high:
$NM_H = V_{OH} - V_{IH}$

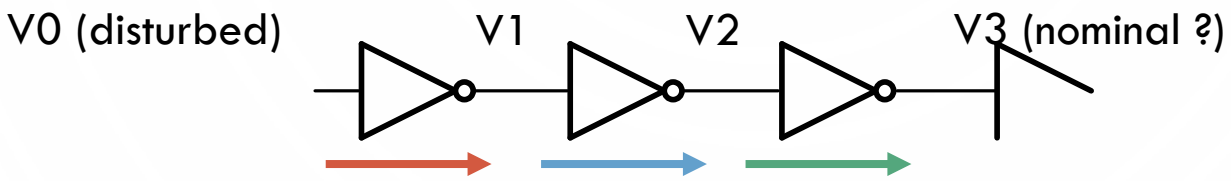
Noise margin low:
$NM_L = V_{IL} - V_{OL}$

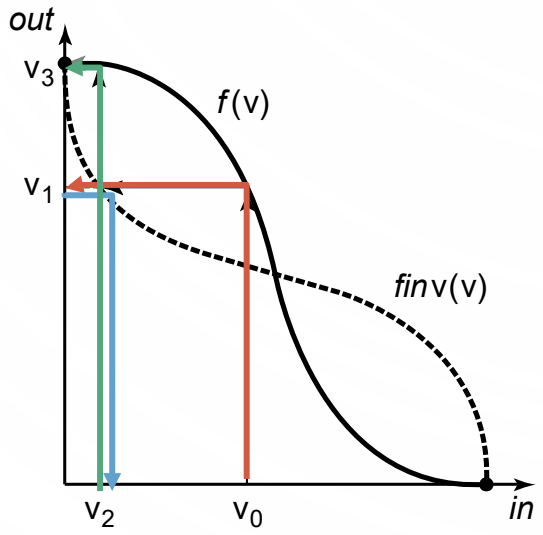
Gate Output (Stage M) → Gate Input (Stage M+1)

The amount of **noise** that could be added to **a worst-case output** so that the signal can still be interpreted correctly as **a valid input** to the next gate.

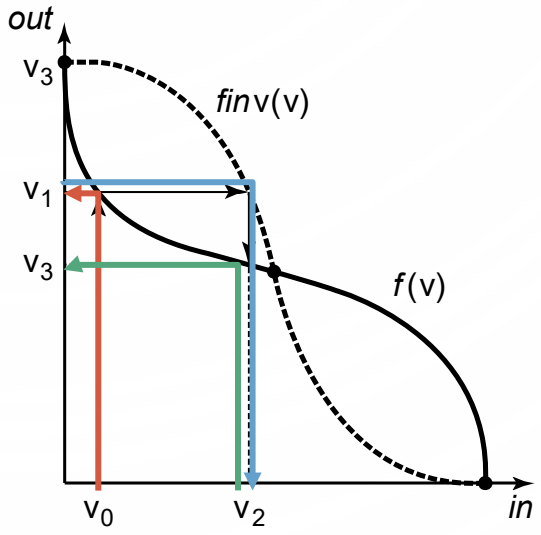
Regenerative Property

- Ensures that a **disturbed** signal gradually **regenerates** one of the **nominal voltage levels** after passing through a few logical stages.
  - Look for a sharp transition in voltage transfer characteristics.





Regenerative gate

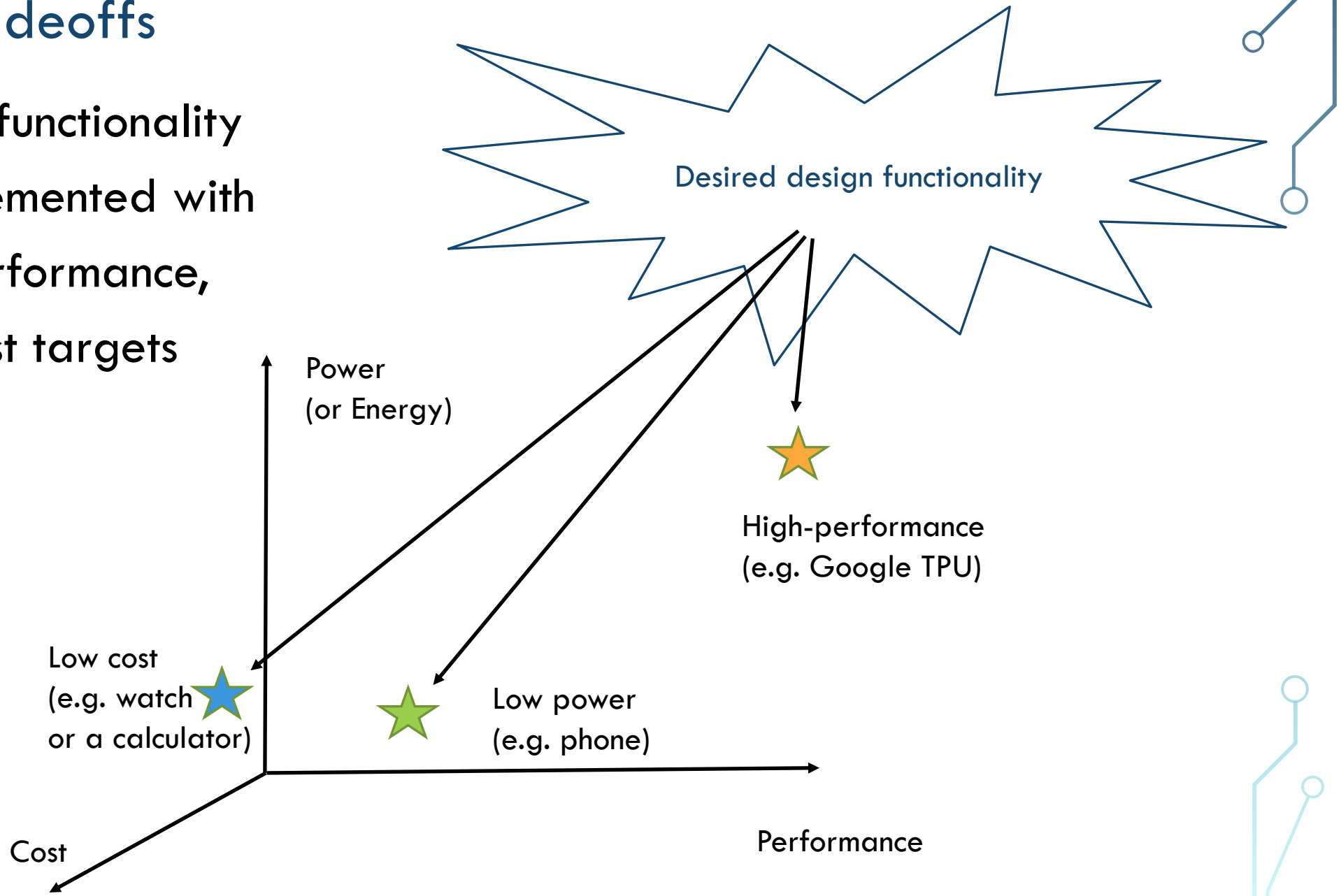


Non-regenerative gate

Design Metrics: Performance

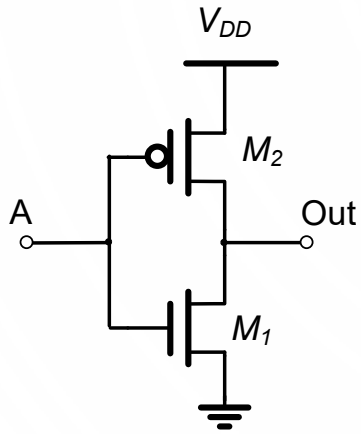
Design Tradeoffs

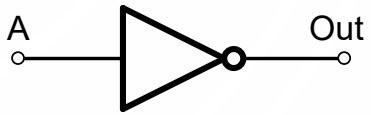
- The desired functionality can be implemented with different performance, power or cost targets



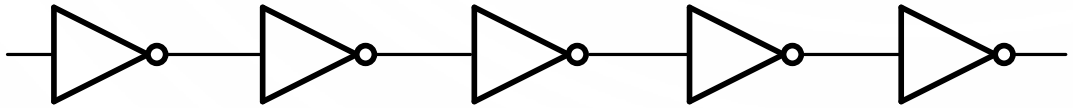
Digital Logic Delay

- Changes at the inputs do not instantaneously appear at the outputs
  - There are finite resistances and capacitances in each gate…

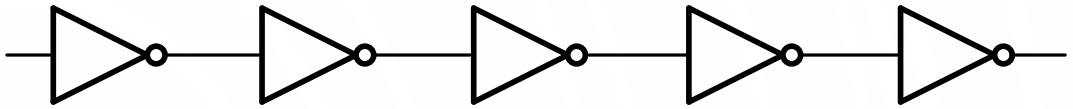




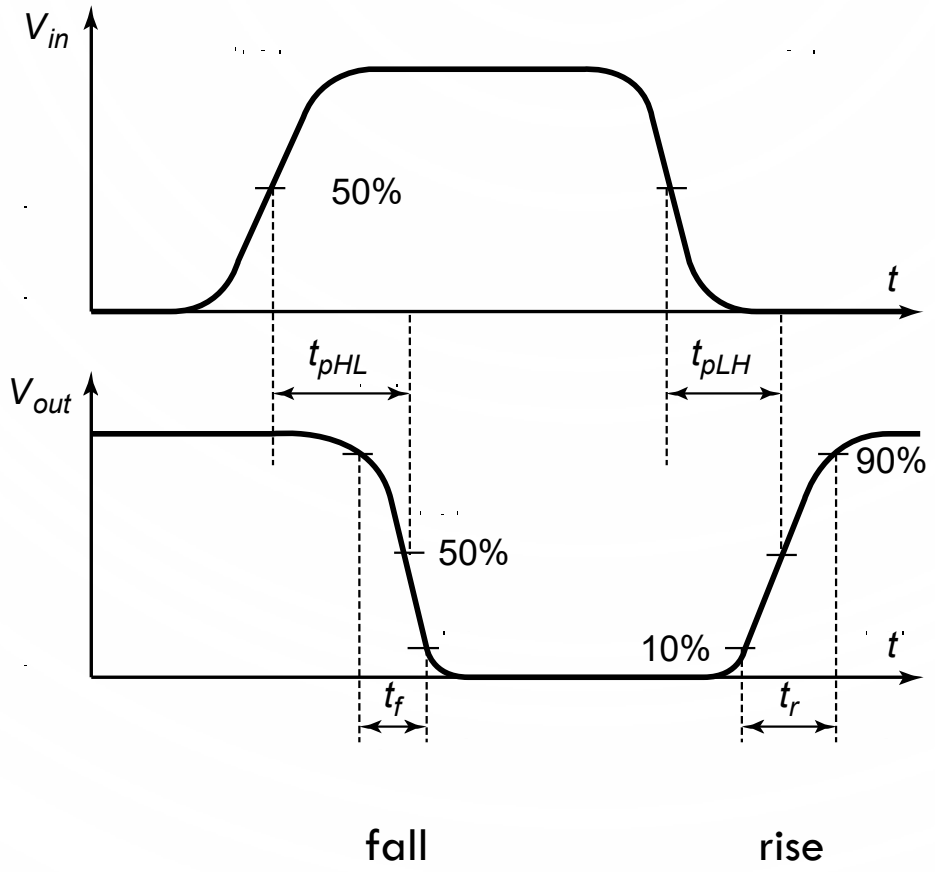
- Propagation through a chain of gates



Delay Definitions

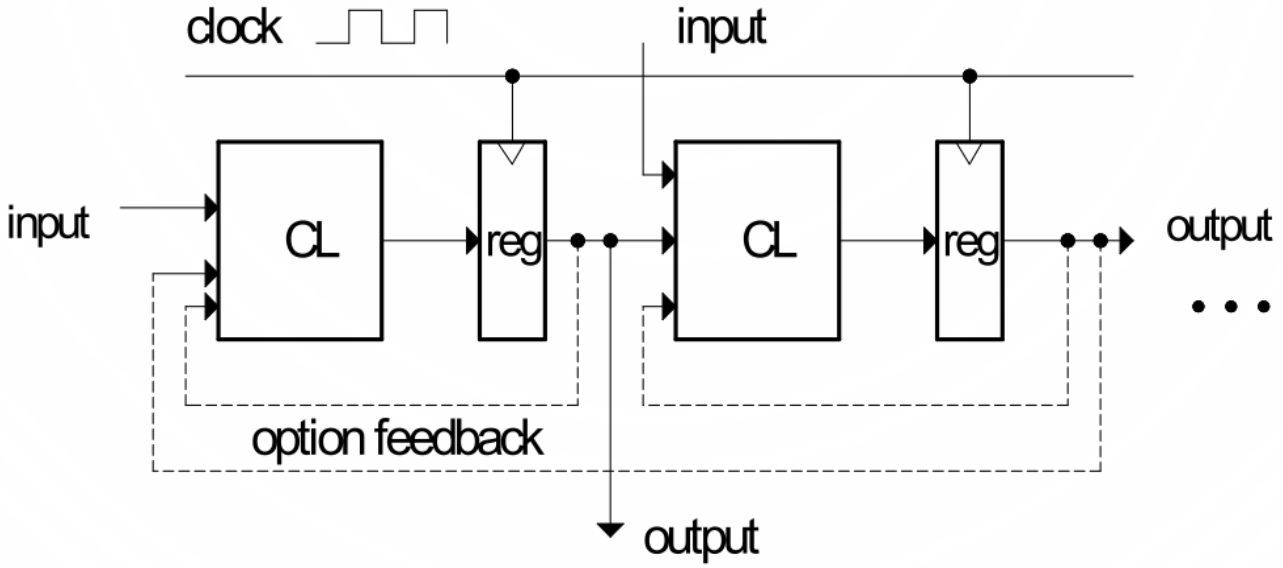


- Delay calculations need to be additive
  - Calculate the delay from the same point in the waveform



Digital Logic Timing

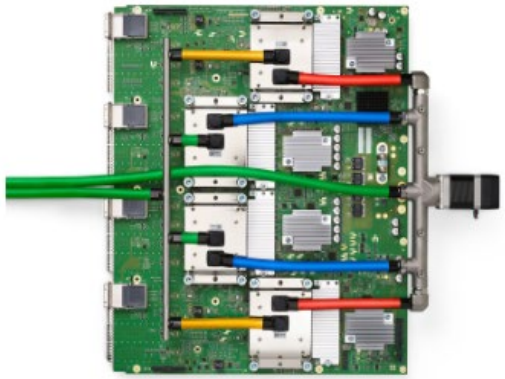
- The longest propagation delay through CL blocks sets the maximum clock frequency



- To increase clock rate:
  - Find the longest path
  - Make it faster

Performance

- Throughput
  - Number of tasks performed in a unit of time (operations per second)
  - E.g. Google TPUv4 chip performs 275 TFLOPS (10^{12} floating-point operations per second, where a floating point operation is BFLOAT16)
  - Watch out for 'op' definitions – can be a 1-b ADD or a double-precision FP add (or more complex task)
  - Peak vs. average throughput
- Latency
  - How long does a task take from start to finish
  - E.g. facial recognition on a phone takes 10's of ms
  - Sometime expressed in terms of clock cycles
  - Average vs. 'tail' latency

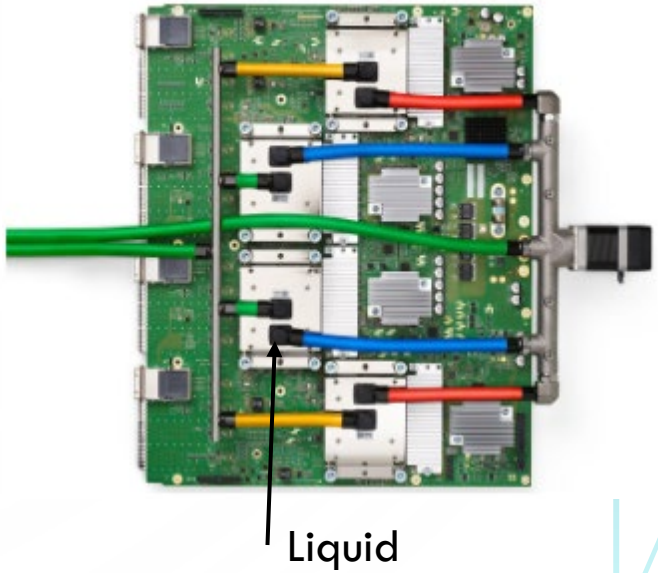


Design Metrics: Energy and Power

Energy and Power

- Energy (in joules (J))
  - Needed to perform a task
  - Add two numbers or fetch a datum from memory
    - (or fetch two numbers, add them and store in memory)
  - Active and standby
  - Battery stores certain amount of energy (in Ws = J or Wh)
  - That is what utility charges for (in kWh)
  - Energy footprint matters for the environment!
- Power (in watts (W))
  - Energy dissipated in time (W = J/s)
  - Sets cooling requirements
    - Heat spreader, size of a heat sink, forced air, liquid, …

TPU v4. ~200W



Design Metrics: Cost

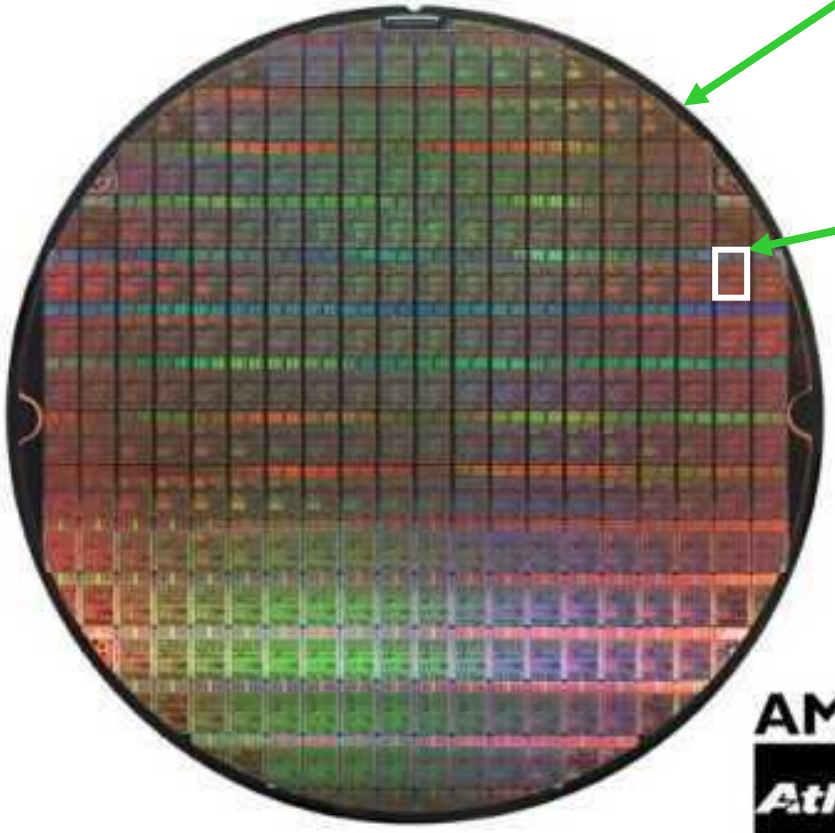
Cost

- Non-recurring engineering (NRE) costs
- Cost to develop a design (product)
  - Amortized over all units shipped
  - E.g. $20M in development adds $.20 to each of 100M units
- Recurring costs
  - Cost to manufacture, test and package a unit
  - Processed wafer cost is ~10k (around 16nm node) which yields:
    - 1 Cerebras wafer-scale chip
    - 50-100 large FPGAs or GPUs
    - 500 laptop CPUs
    - >1000 cell phone SoCs

$$\text{cost per IC} = \text{variable cost per IC} + \frac{\text{fixed cost}}{\text{volume}}$$

$$\text{variable cost} = \frac{\text{cost of die} + \text{cost of die test} + \text{cost of packaging}}{\text{final test yield}}$$

Die Cost



$$\text{cost of die} = \frac{\text{cost of wafer}}{\text{dies per wafer} * \text{die yield}}$$

AMD Athlon

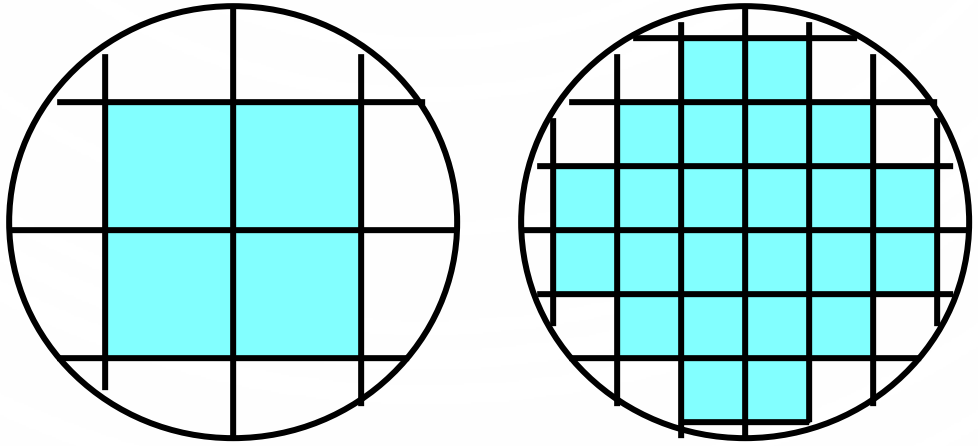
From: http://www.amd.com

Yield

$$Y = \frac{\text{No. of good chips per wafer}}{\text{Total number of chips per wafer}} \times 100\%$$

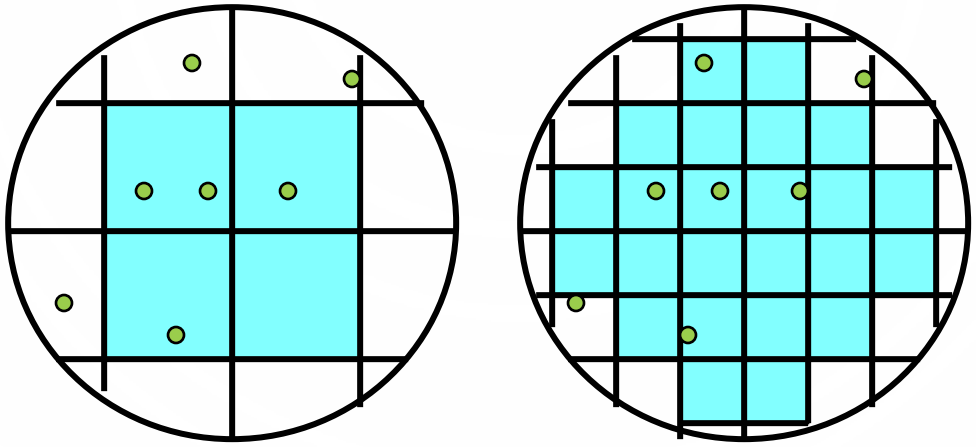
$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per wafer} \times \text{Die yield}}$$

$$\text{Dies per wafer} = \frac{\pi \times (\text{wafer diameter}/2)^2}{\text{die area}} - \frac{\pi \times \text{wafer diameter}}{\sqrt{2 \times \text{die area}}}$$



Defects

Yield = 0.25



Yield = 0.76

$$\text{die yield} = \left(1 + \frac{\text{defects per unit area} \times \text{die area}}{\alpha}\right)^{-\alpha}$$

α is approximately in the range 0.5-5

If α =3, die cost $= f(\text{die area})^4$

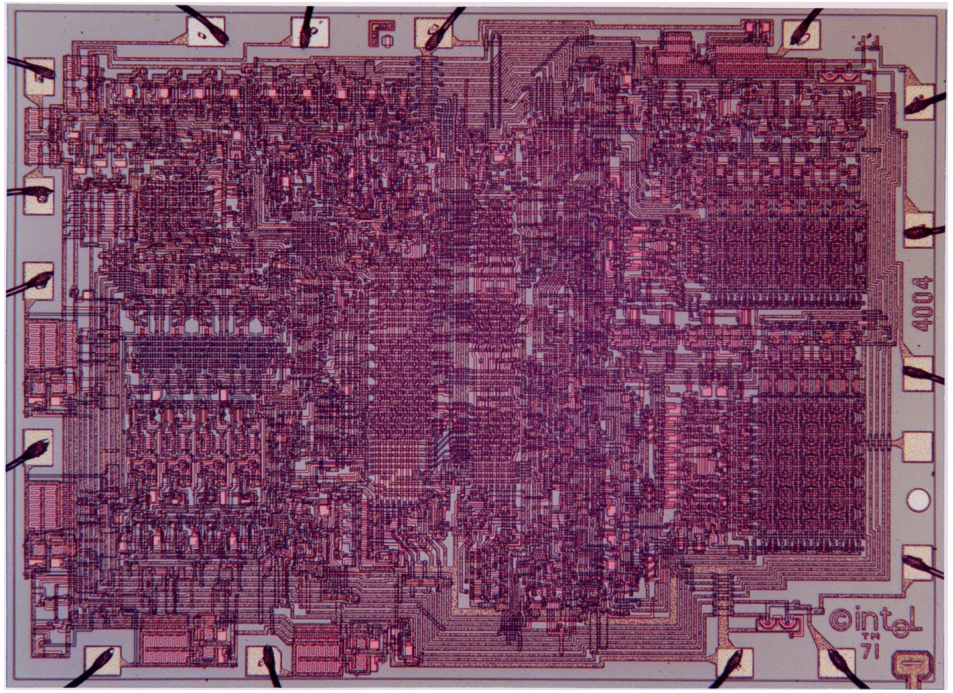
Administrivia

- Bora's office hour today at 1pm
  - Regular office hour on Mondays at 10am
- Lab 1 posted, please start it before coming to the lab session
- Lab 2 is more involved
  - Be prepared
  - Verilog primer
- Homework 1 posted today, due next Friday
  - Start early

How Did We Get Here?

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

- Circuit design, layout, and processing tightly linked.
- Logic design and layout was all done by-hand
- Chip design was typically done by vertically integrated companies, who designed and fabricated their own chips
  - Fabs weren't that expensive back then



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!

Early Days in Academia

12

ELECTRONIC NEWS, MONDAY, OCTOBER 23, 1961

2 New Labs at California U. To Aid Integrated Circuitry

By RICHARD BUSBACK
Special to Electronic News

BERKELEY, Calif.—Research in semiconductor integrated circuitry by the Electronics Research Laboratory of the University of California, here, will be enhanced next Spring with the addition of two laboratories, Prof. D. O. Pederson, director of ERL, said last week.

He predicted that "substantially more than the current 10 per cent of ERL's total effort will be devoted to this area," when the new semi-conductor integrated circuits fabrication laboratory and electron beam microscopy facility begin operating.

Professor Pederson was interviewed following a two-day closed meeting which reviewed for the supporting agencies, ERL's research efforts during the past year. He noted that during the next two years ERL will study the basic circuit functions which can be realized in integrated circuits and will try to understand the necessary interactions of properties needed to achieve, among others, oscillators and flip-flops.

In trying to achieve a circuit function without recourse to separate components, ERL will use the scanning electron beam microscope to inspect and study what goes on in an integrated circuit.

To absorb various integrated circuit technologies, Prof. Pederson said ERL has worked with, among others, Westinghouse Electric Corp., Youngwood, Pa., Motorola Corp.s Semiconductor division, Phoenix, and Fairchild Semiconductor, Mountain View, Calif.

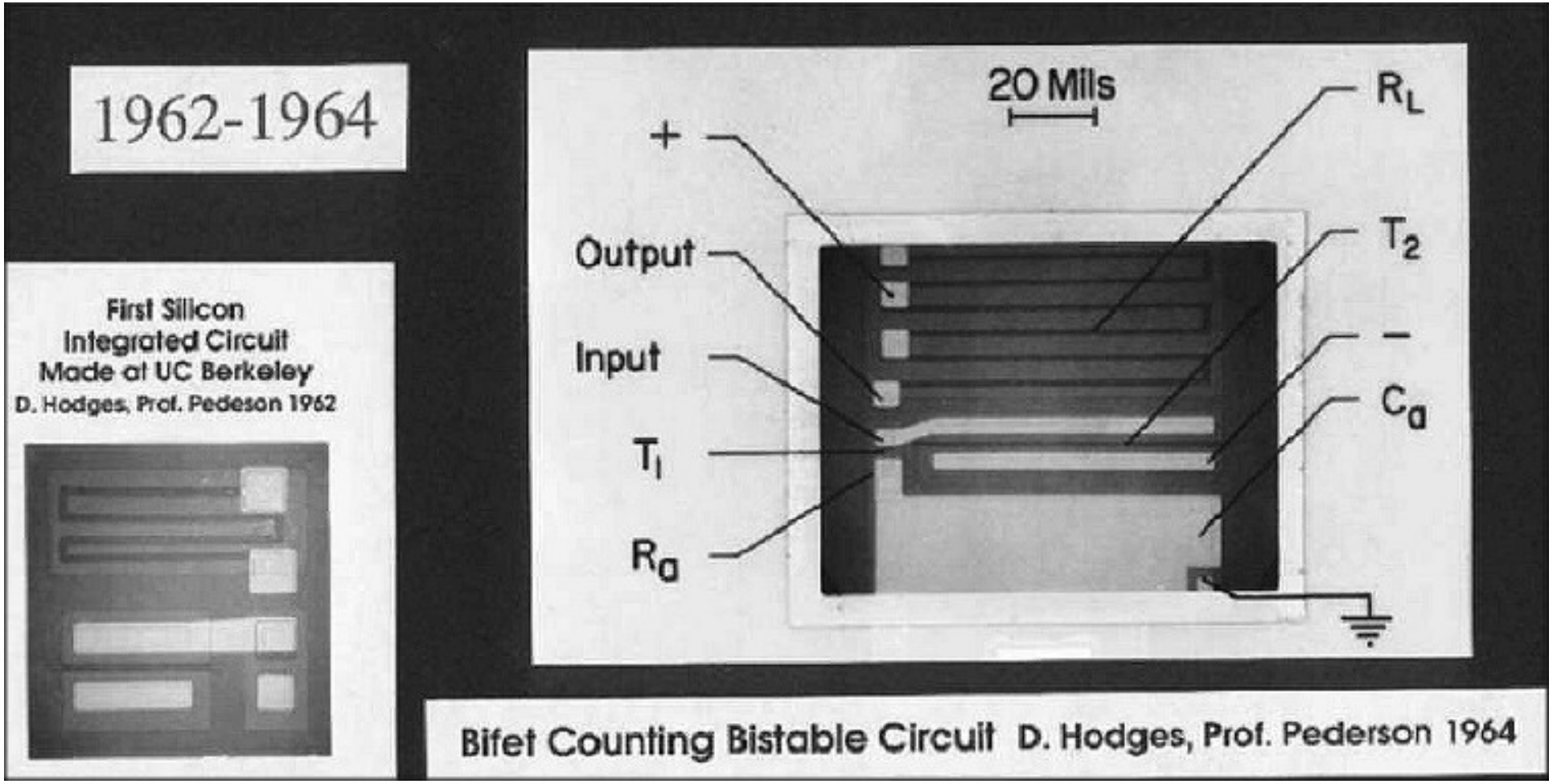
Funds for research in integrated circuitry during the past 18 months have come from the Aeronautical Systems division, Air Force Systems Command, Wright-Patterson AFB, Dayton, O., and a tri-service contract. The latter was administered jointly by the Office of Naval Research, the Air Force Office of Scientific Research and the Department of the Army Research Office and Signal Corps.

Prof Pederson said a high temperature plasma containment machine is expected to be installed shortly after Jan. 1.

Basic research in hot plasmas without regard to obtaining power from the plasma is considered along with systems theory and building interpretive computer languages as the other major research areas of interest, Prof. Pederson said. Of the 150 projects at ERL, about 25 per cent concern systems theory. About 50 per cent of the projects are concerned with experimental and analytical research in physical electronics and magnetics, he said.

The first ever university Microlab was built at UCB in 1962

First ICs at Berkeley



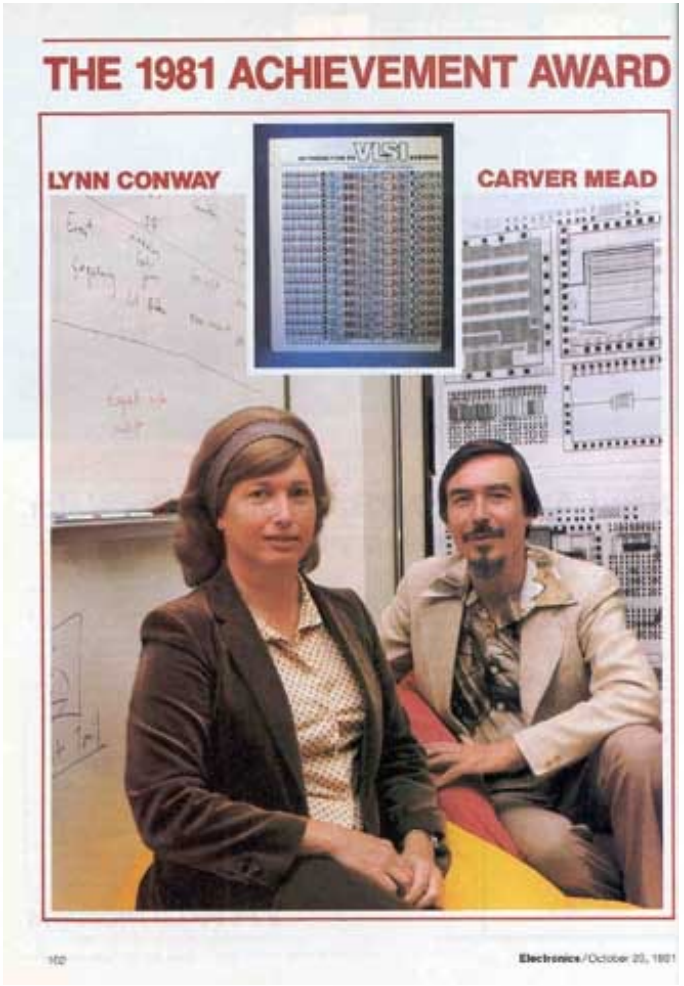
Early Design Practice

- Initially, designs 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.

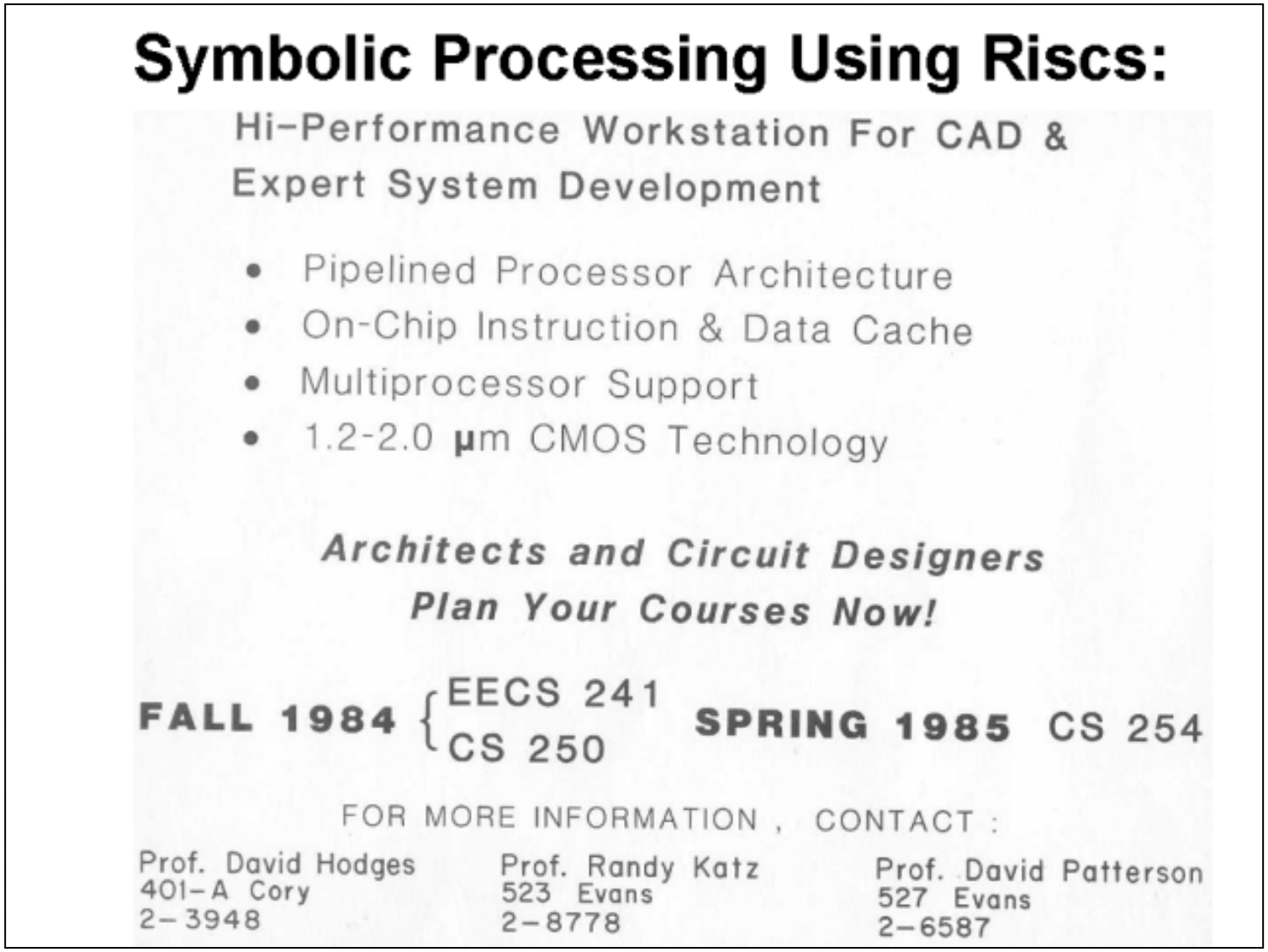
- Later transition to an electronic format (CIF, GDS) meant: Layouts easily be stored and transmitted. Written to tape and transferred to manufacturer (tape-out).Transmitted over the network (new idea back then). Software could automatically check for layout errors. Generated from a program - **huge idea.**

Gray and McCreary in Cory Hall

VLSI Design Revolution



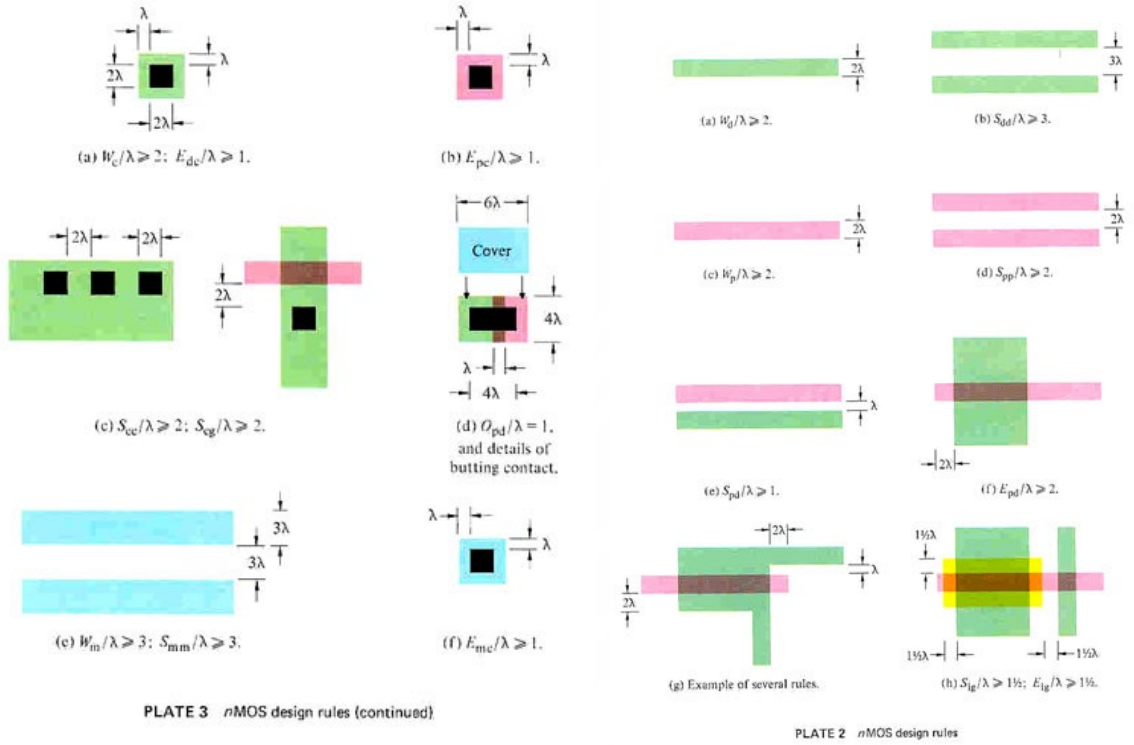
Berkeley Courses



Geometric Design Rules

- Early on, to generate the mask information for fabrication, the designer 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.
- Academics (C. Mead, L. Conway and others) came up with a much simplified set of design rules (single page description)

  - 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.



Key Development: 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, foundry manufactures the chip and send back.

- A scalable model for the industry: IC fab is expensive and complex. Amortizes the expense over many designers. 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.

TSMC, Global Foundries, Intel, UMC, Samsung, SMIC, …

Computer Aided Design (1)

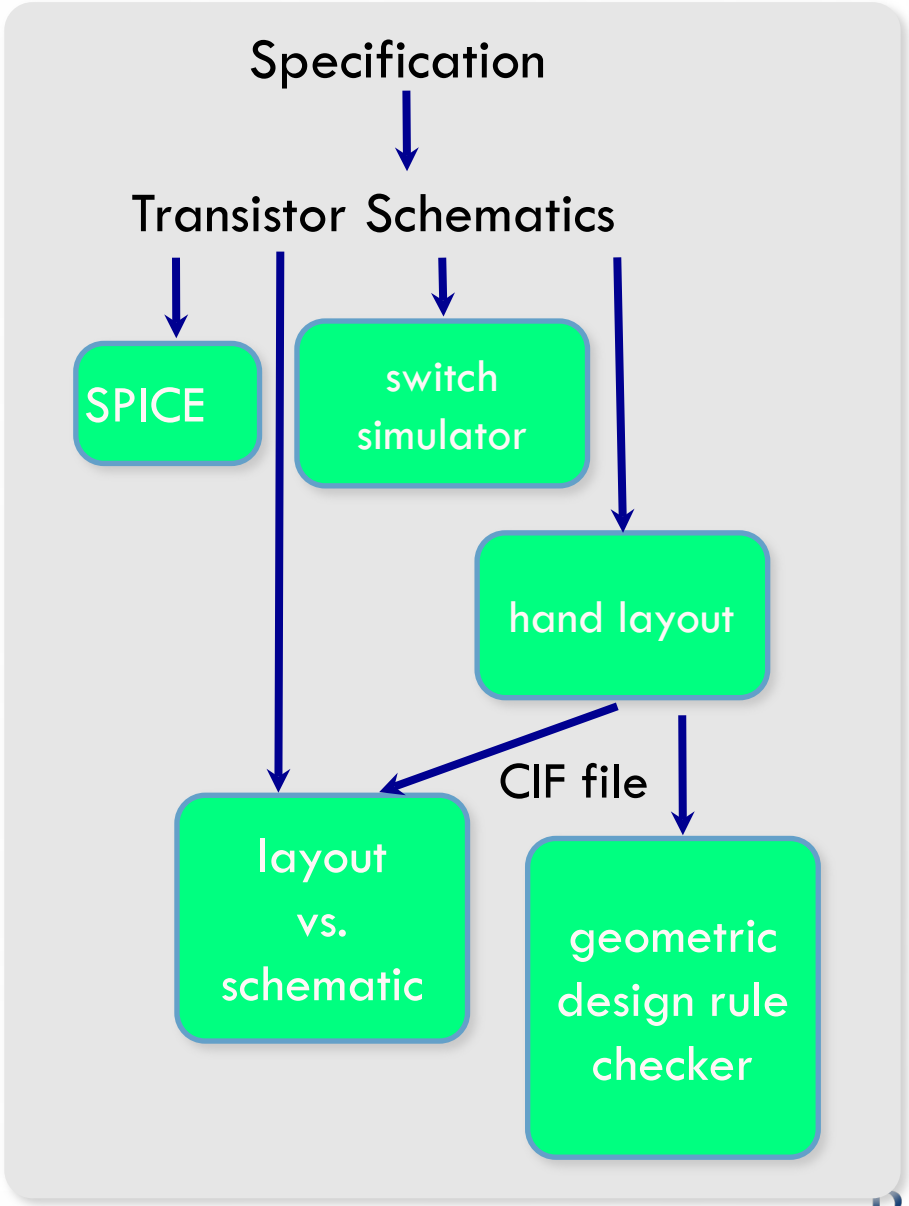
Several advances lead to the development of interactive tools for generating layout:

- Computer based layout representation (CIF, GDS).
- Advances in computer graphics (Ivan Sutherland) and display devices.
- Personal "workstation" (Xerox Alto - Chuck Thacker). "Back room" computers didn't have the necessary bandwidth to the display.
- Berkeley version - MAGIC

Early '80's Design Methodology and Flow

Schematic + Full-Custom Layout

- SPICE for timing,
- Switch-level simulation for overall functionality,
- Hand layout,
- No power analysis,
- Layout verified with geometric design rule checker (DRC) and later also layout versus schematic (LVS) checkers



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, 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.
- Eventually, Cadence and Synopsys (formed out of Berkeley) built tools to generate layout from HDLs.

Implementation Alternatives

Implementation Alternative Summary

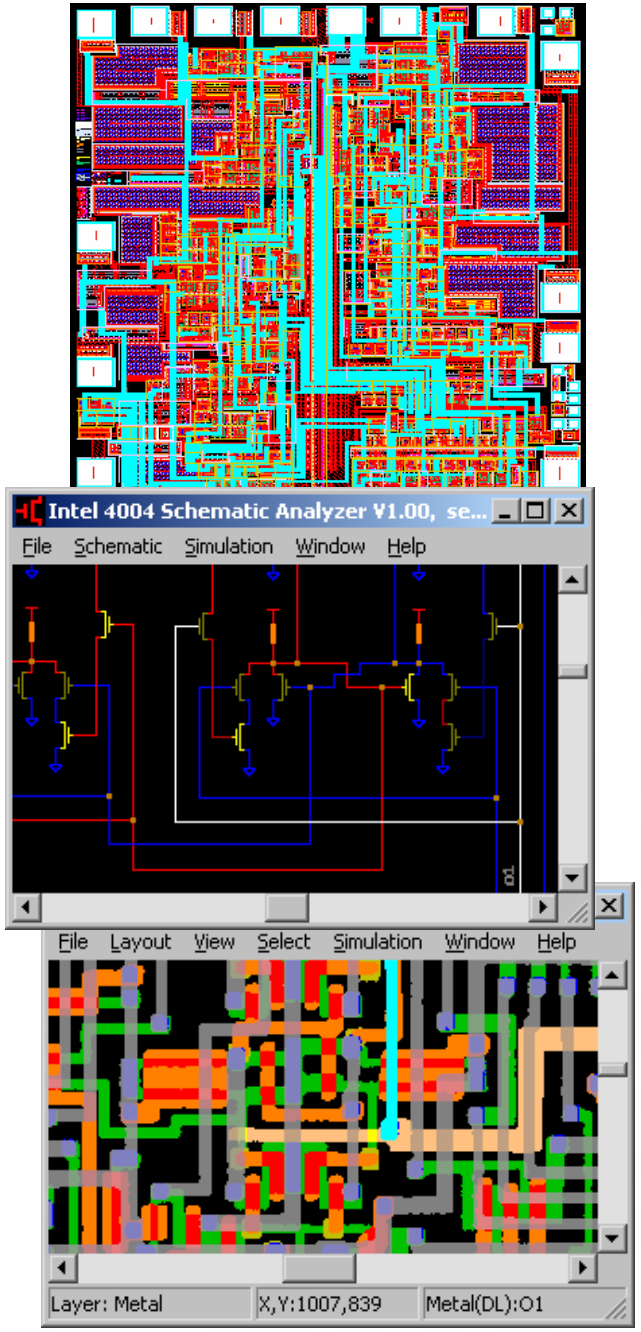
Full-custom:	Every transistors layout hand-drawn and optimized.
Standard-cell:	Logic gates and “macros” automatically placed and routed.
Gate-array (structured ASIC):	Partially prefabricated wafers with arrays of transistors customized with metal layers or vias.
FPGA:	Prefabricated chips that can be customized “in the field”
Microprocessor:	Instruction set interpreter customized through software.
Domain-specific processor:	Special instruction set interpreters (ex: DSP, NPU, GPU).

These days, “ASIC” almost always means standard-cell design.

Full-Custom

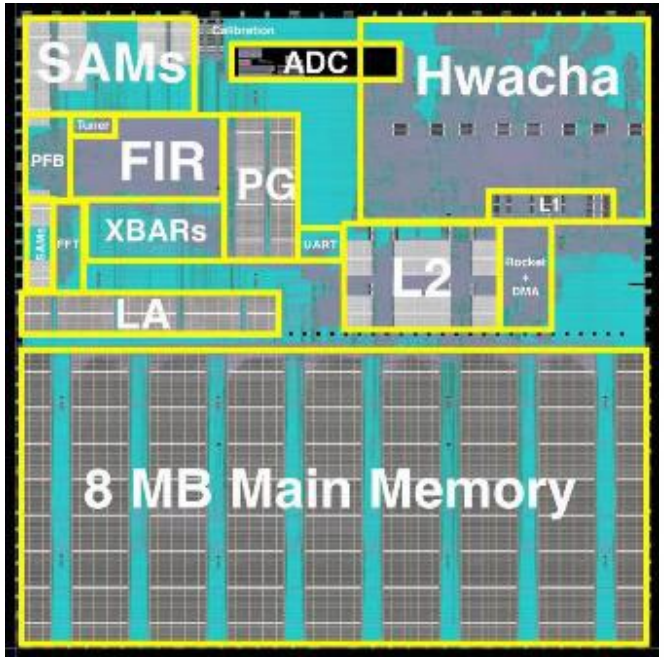
- Circuit styles and transistors are custom sized and drawn to optimize performance and power.
- High NRE (non-recurring engineering) costs
  - Time-consuming layout iterations

- Common today for **analog design.**
- In digital world – for **I/Os, memories** and **arrays.**

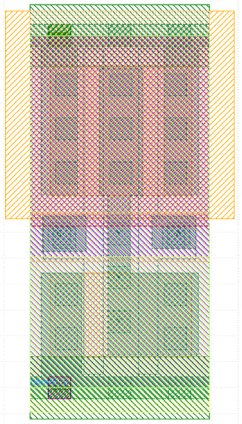


Standard Cells

- Library of logic gates (1000-2000)
  - Combinational: NANDs, NORs, … Sequential: Flip-flops, latches, …
- Each cell comes complete with:
  - layout, schematic, symbol
  - Logic (Verilog), timing, power models.
- Chip layout is automatic, reducing NREs (usually no hand-layout).
- Memories, I/O and analog components complete the ASIC



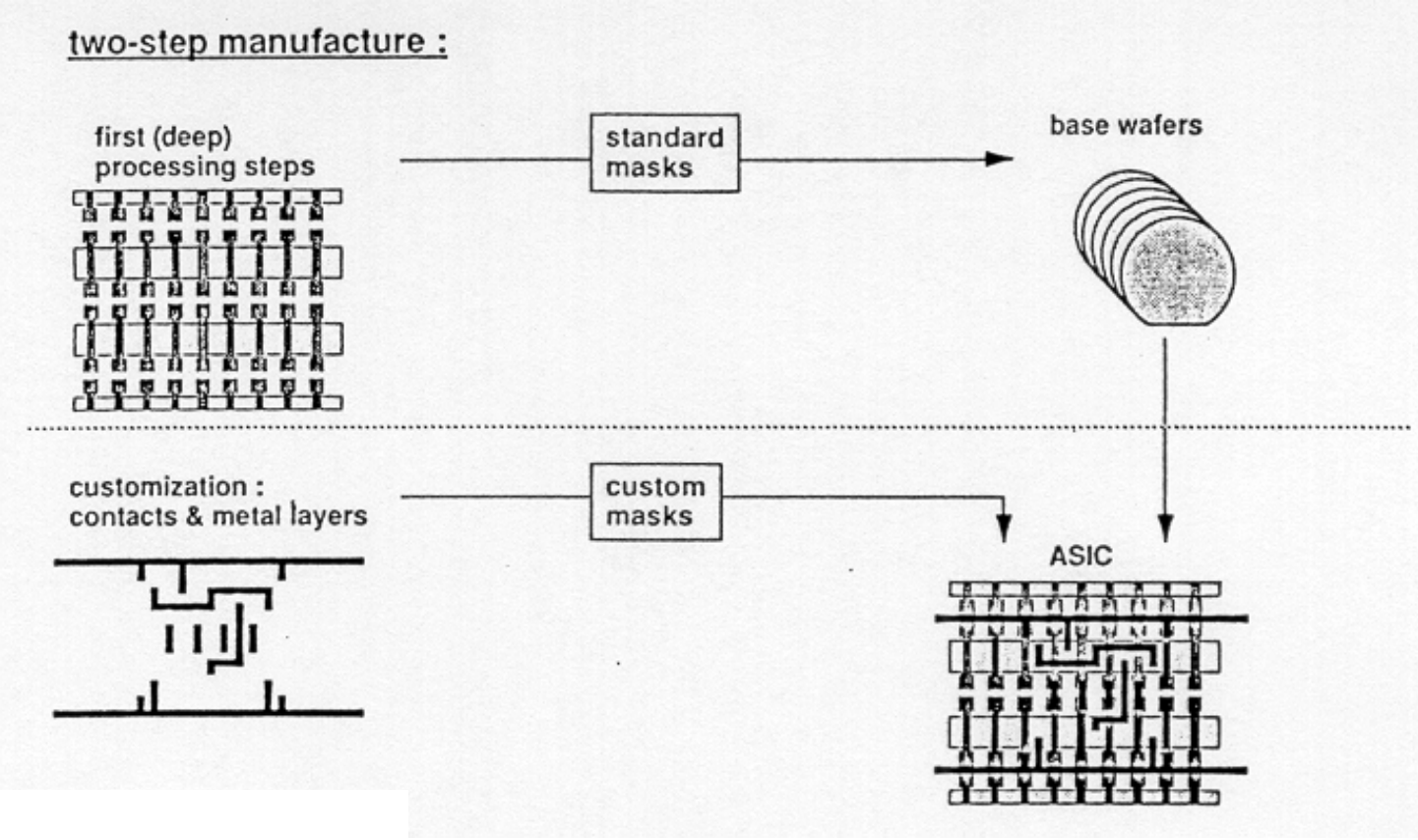
NAND2



SkyWater 130nm process

Gate Array

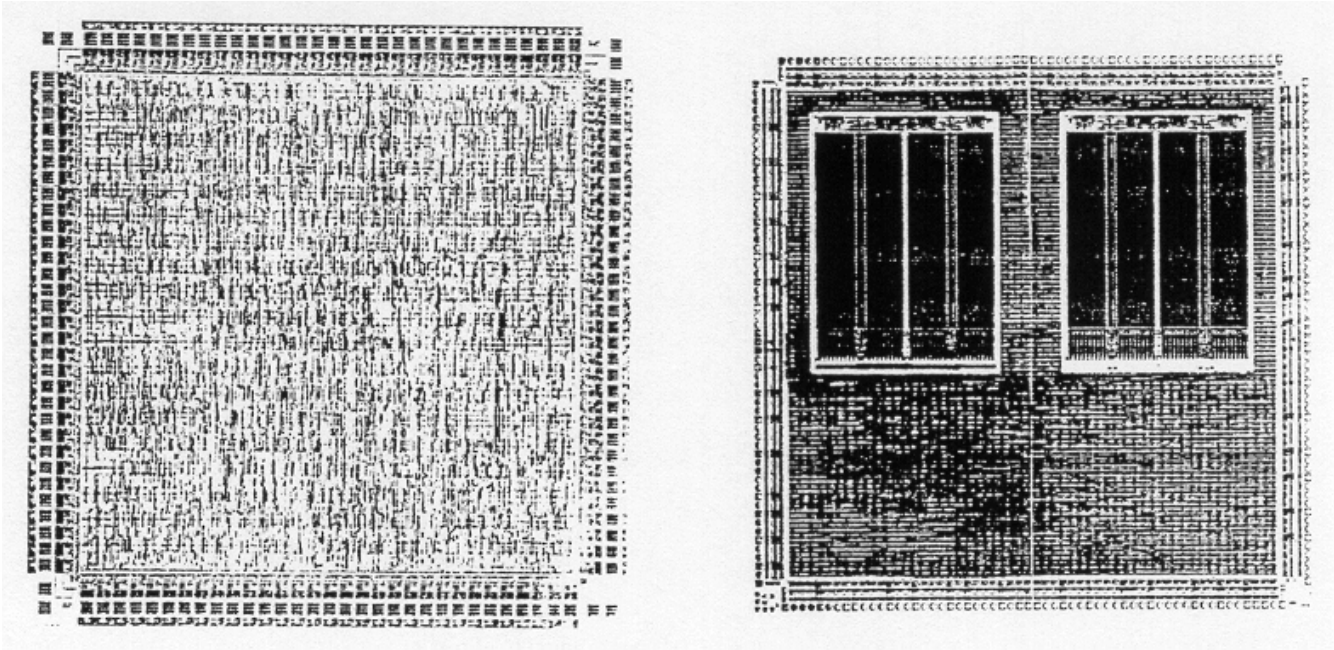
- Prefabricated wafers of, rows of transistors. Customize as needed with "back-end" metal processing (contact cuts, metal wires). Could use a different factory.
- CAD software understands how to make gates and registers.



Gate Array

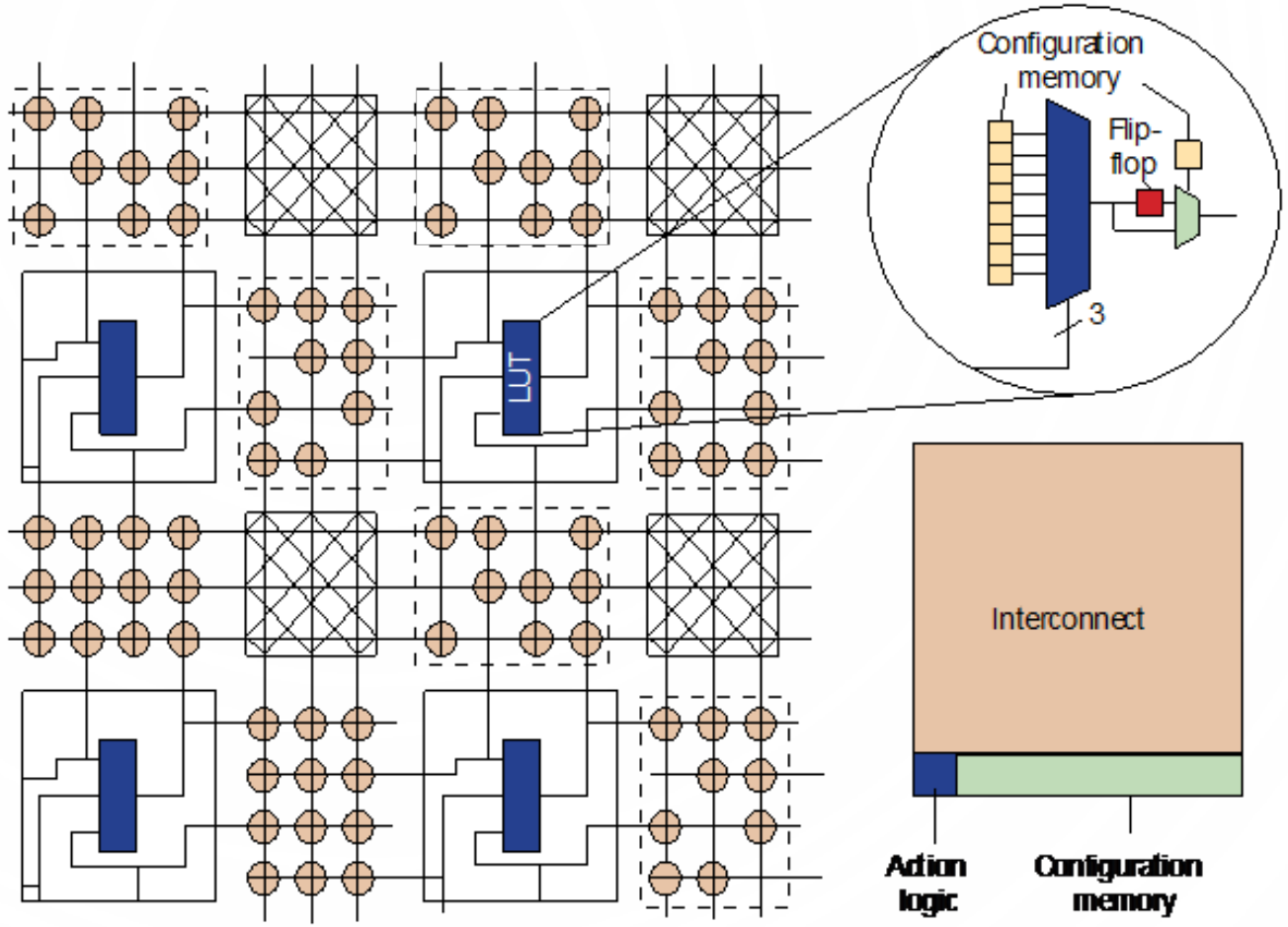
- Shifts large portion of design and mask NRE to vendor.
- Shorter design and processing times, reduced time to market for user.
- Highly structured layout with fixed size transistors leads to large sub-circuits (ex: Flip-flops) and higher per die costs.
- Memory arrays are particularly inefficient,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

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



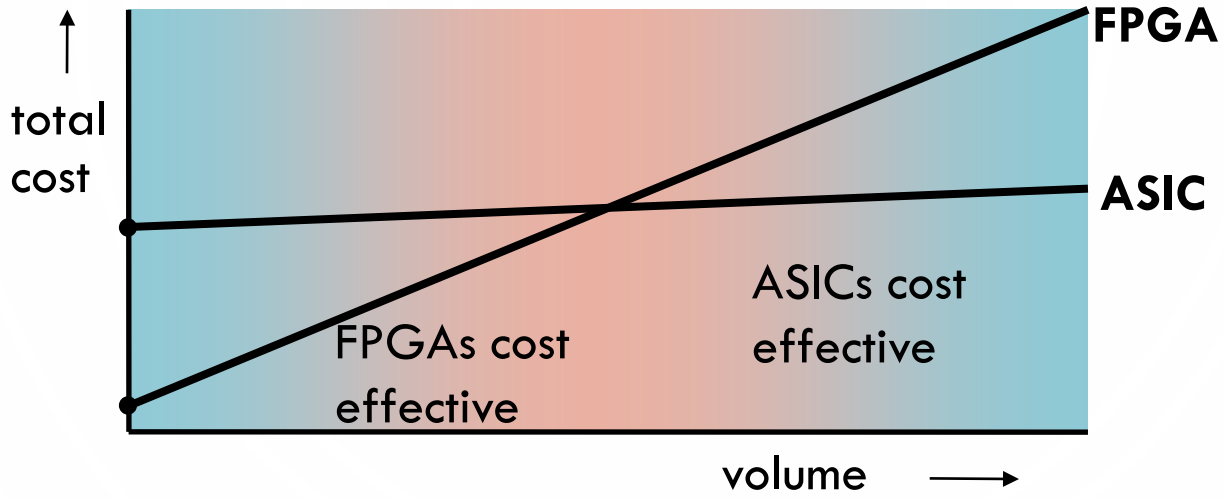
Field Programmable Gate Arrays (FPGA)

- Two-dimensional array of simple logic- and interconnection-blocks.
- Typical architecture: Look-up-tables (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, Ethernetnet, USB, PCIe, CPUs, ...

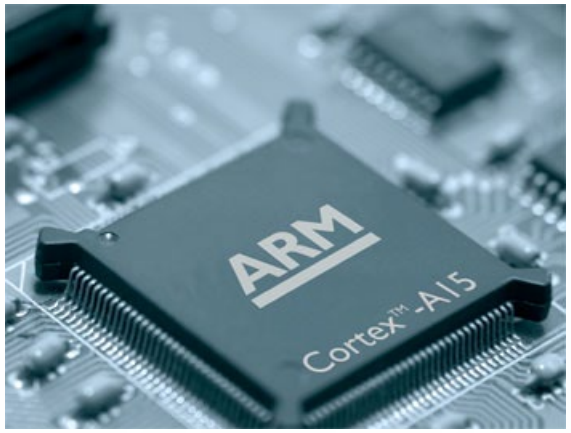
FPGA versus ASIC



- **ASIC:** Higher NRE costs (10's of $M). Relatively Low cost per die (10's of $ or less).
- **FPGAs:** Low NRE costs. Relatively low silicon efficiency ⇒ high cost per part (> 10's of $ to 1000's of $).
- **Cross-over volume** from cost effective FPGA design to ASIC was often in the 100K range.
- **ASICs often have >10x higher performance and 10x lower power for the same application.**

Microprocessors

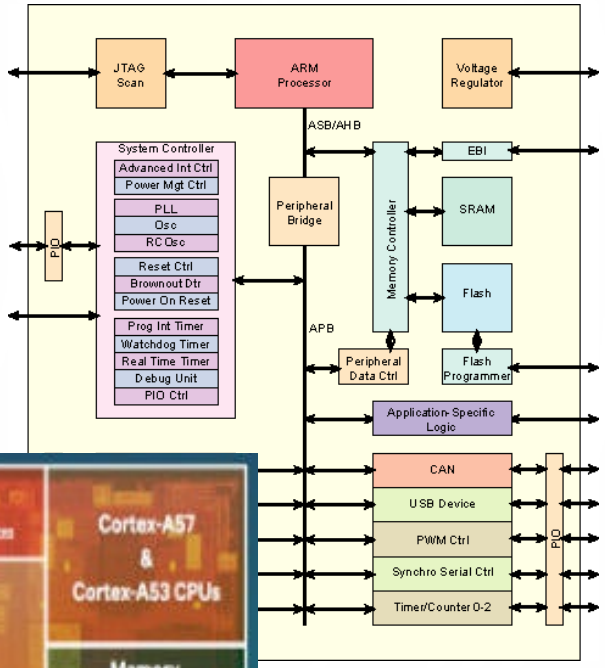
- Where relatively low performance and/or high flexibility is needed, a viable implementation alternative:
  - Software implements desired function
  - "Microcontroller", often with built in nonvolatile program memory and used as single function.
- Two "abstraction" levels:
  - Instruction Set Architecture (ISA)
  - "Synthesizable" RTL model ("soft core", available in HDL)
- Their implementation can be both ASIC or FPGA



§	Assembler
	ADD{cond}{S} Rd, Rn, <Operand2>
	ADC{cond}{S} Rd, Rn, <Operand2>
5E	QADD{cond} Rd, Rm, Rn
5E	QDADD{cond} Rd, Rm, Rn
	SUB{cond}{S} Rd, Rn, <Operand2>
	SBC{cond}{S} Rd, Rn, <Operand2>
	RSB{cond}{S} Rd, Rn, <Operand2>
	RSC{cond}{S} Rd, Rn, <Operand2>
5E	QSUB{cond} Rd, Rm, Rn
5E	QDSUB{cond} Rd, Rm, Rn
2	MUL{cond}{S} Rd, Rm, Rs
2	MLA{cond}{S} Rd, Rm, Rs, Rn
M	UMULL{cond}{S} RdLo, RdHi, Rm, Rs
M	UMLAL{cond}{S} RdLo, RdHi, Rm, Rs
6	UMAAL{cond} RdLo, RdHi, Rm, Rs

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: AXI, …
- "IP Block" business model: Hard- or soft-cores available from third party vendors.
- ARM, inc. is the example. Hard- and "synthesizable" processors.
- ARM and other companies provide, Ethernet, USB controllers, analog functions, memory blocks, …



Qualcomm Snapdragon

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

ASIC Design

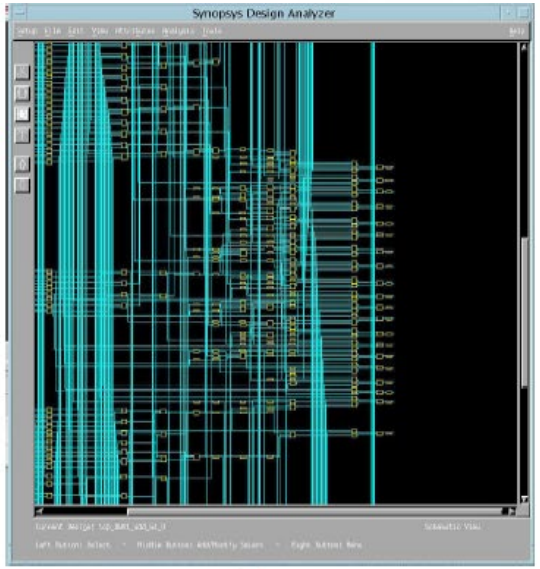
Verilog to ASIC Layout Flow

- "Push-button" approach

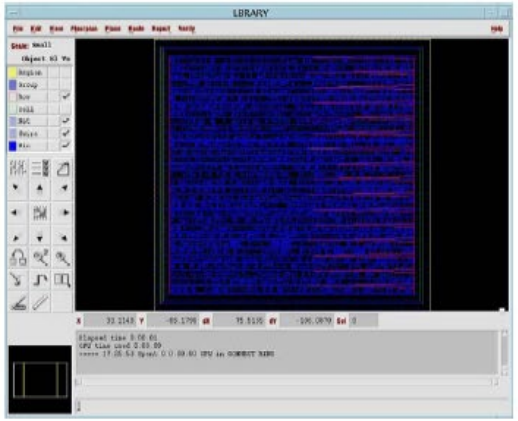
```
module adder64 (a, b, sum);
  input  [63:0] a, b;
  output [63:0] sum;

  assign sum = a + b;
endmodule
```

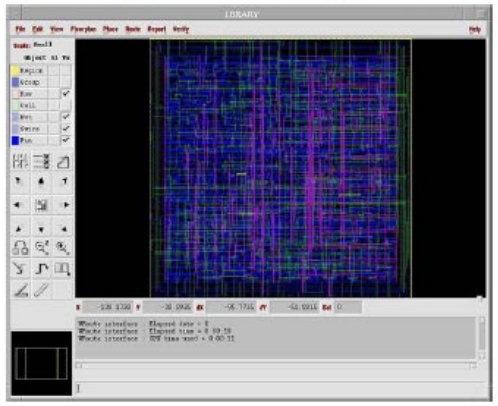
After Synthesis

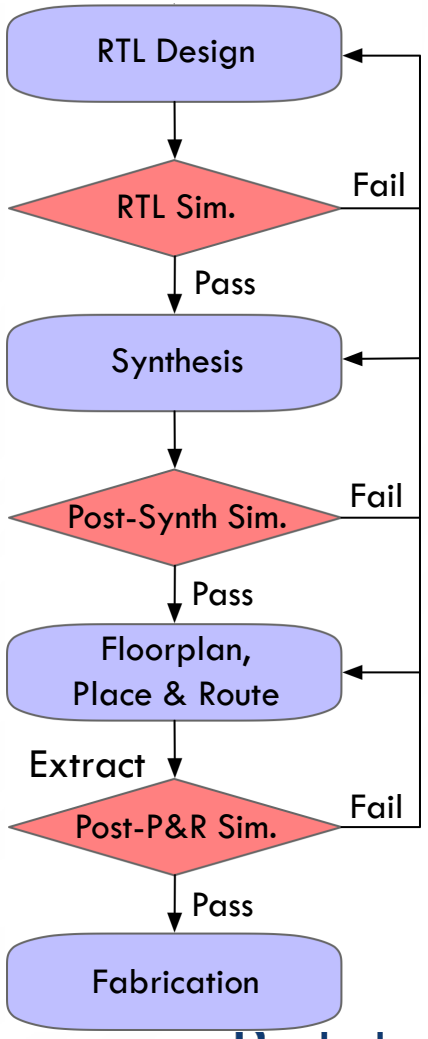


After Placement

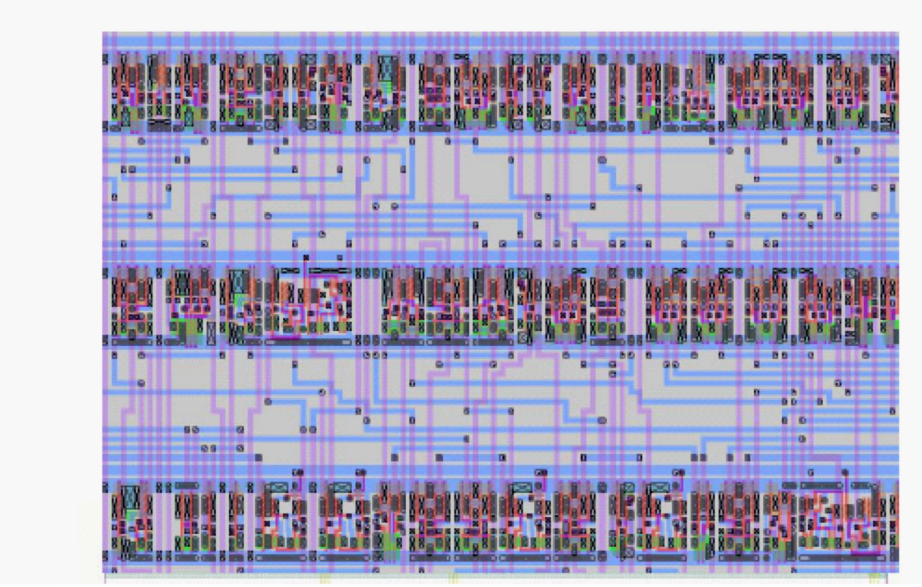


After Routing

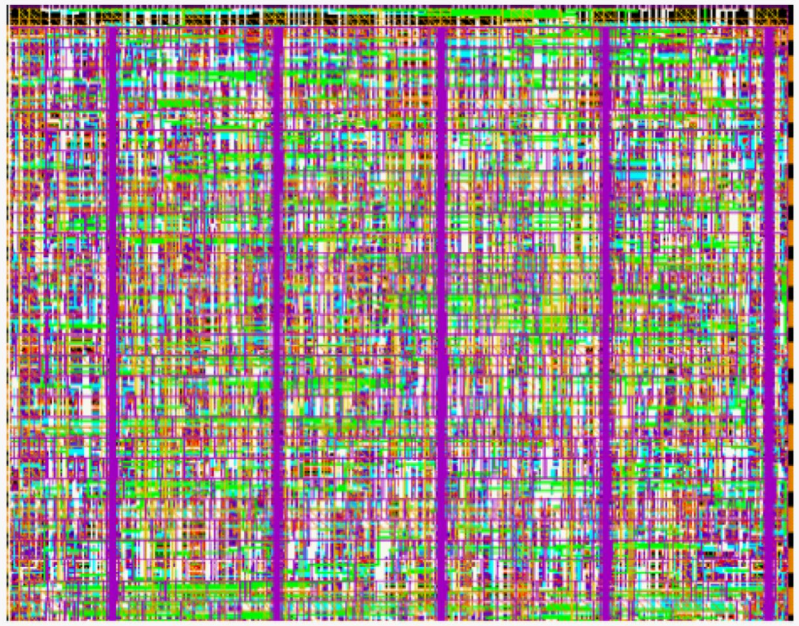




Standard cell layout methodology



Old times: 1μm, 2-metal process



Modern sub-100nm process
"Transistors are free things that fit under wires"

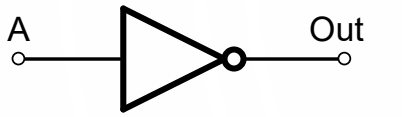
- With limited # metal layers, dedicated routing channels were needed
- Currently area dominated by wires

Standard Cells (ASAP7)

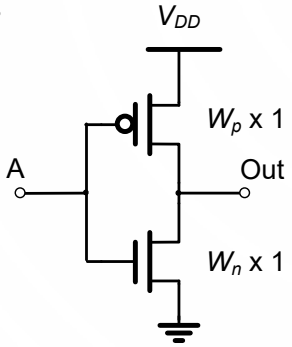
Name

1x Inverter

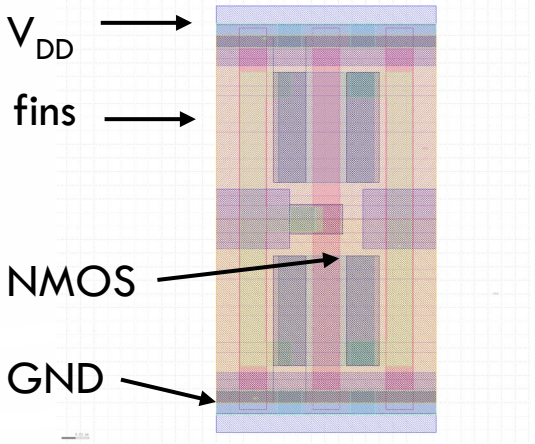
Symbol



Schematic



Layout (gds)



INVX1_ASAP7_75T_R

asap7sc7p5t_24_INVBUF_RVT_TT Cell Library: Process , Voltage 0.70, Temp 25.00

Truth Table

INPUT	OUTPUT
A	Y
0	1
1	0

Footprint

Cell Name	Area
INVx1_ASAP7_75t_R	0.69984

Pin Capacitance Information

Cell Name	Pin Cap(ff)	Max Cap(ff)
	A	Y
INVx1_ASAP7_75t_R	0.43869	46.08000

Leakage Information

Cell Name	Leakage(pW)		
	Min.	Avg	Max.
INVx1_ASAP7_75t_R	0.00000	51.15440	53.32100

Delay Information

Delay(ps) to Y rising :

Cell Name	Timing Arc(Dir)	Delay(ps)		
		Min	Mid	Max
INVx1_ASAP7_75t_R	A->Y (FR)	5.53739	36.47100	284.32000

Verilog for logic sim

```
`celldefine
module INVx1_ASAP7_75t_R (Y, A);
      output Y;
      input A;

      // Function
      not (Y, A);

      // Timing
      specify
            (A => Y) = 0;
      endspecify
endmodule
`endcelldefine
```

Netlist for Spice sim

```
.subckt PM_BUFX10_ASAP7_75T_R%A 2 7 10 13 15 23 27 VSS
c21 27 VSS 0.00191995f $X=0.081 $Y=0.135
c22 23 VSS 0.0244917f $X=0.0565 $Y=0.1325
c23 13 VSS 0.00510508f $X=0.135 $Y=0.135
c24 10 VSS 0.0605852f $X=0.135 $Y=0.0675
c25 2 VSS 0.0649407f $X=0.081 $Y=0.0675
r26 23 27 1.66358 $w=1.8e-08 $l=2.45e-08 $layer=M1
$thickness=3.6e-08 $X=0.0565
...
```

.lib for synthesis

```
cell (INVx11_ASAP7_75t_R) {
   area : 3.03264;
   cell_leakage_power : 562.699;
   pg_pin (VDD) {
     pg_type : primary_power;
     voltage_name : "VDD";
   }
   pg_pin (VSS) {
     pg_type : primary_ground;
     voltage_name : "VSS";
   }
   leakage_power () {
     value : 538.867;
     when : "(A * !Y)";
     related_pg_pin : VDD;
   }
...
```

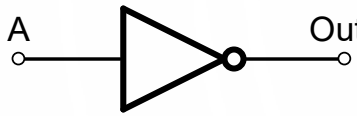
+ .lef for place and route, and other files for LVS and DRC

Standard Cells (SkyWater 130nm)

Name

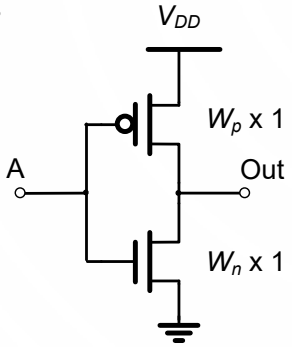
Symbol

1x Inverter

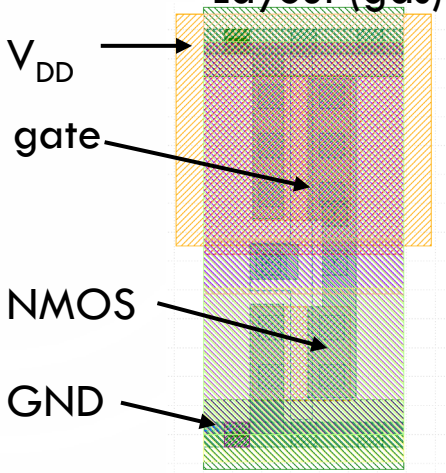


sky130_fd_sc_hd__inv_1

Schematic



Layout (gds)



.lib for synthesis

```
{  "area": 3.7536,
"cell_footprint":
"sky130_fd_sc_hd__inv",
"cell_leakage_power": 7.628985,
"driver_waveform_fall": "ramp",
"driver_waveform_rise": "ramp",
"leakage_power": [    {
"value": 14.5579575,      "when":
"A"    },    {      "value":
0.7000129,      "when": "!A"    }
],  "pg_pin,VGND": {
"pg_type": "primary_ground",
"related_bias_pin": "VPB",
"voltage_name": "VGND"  },
"pg_pin,VNB": {    "pg_type": …
```

Verilog for logic sim

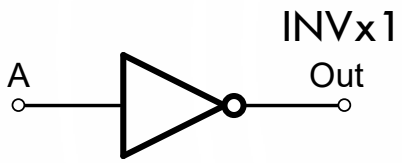
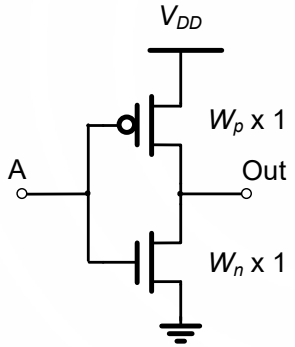
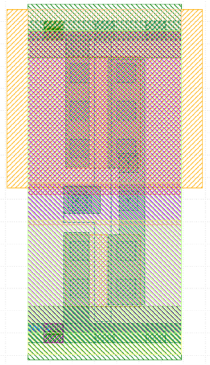
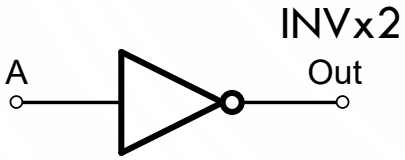
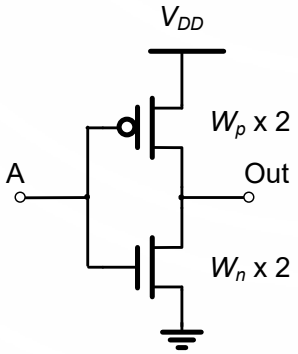
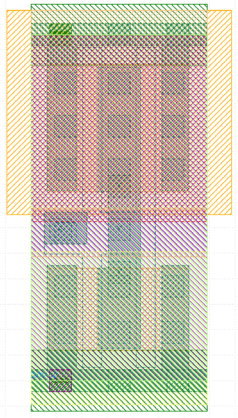
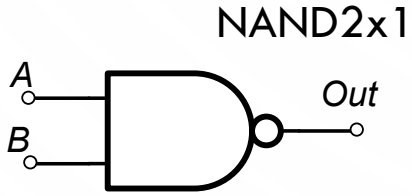
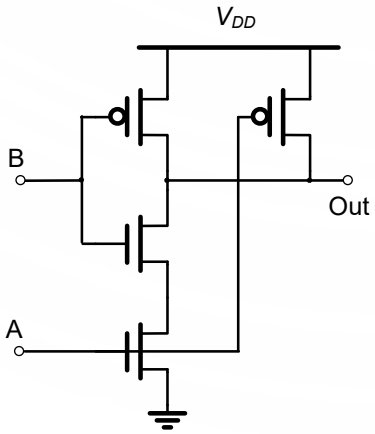
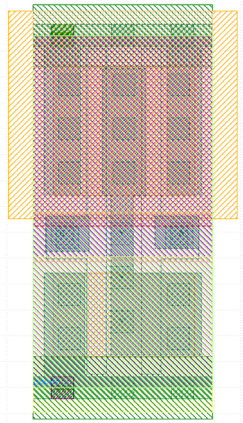
```
`celldefinemodule
sky130_fd_sc_hd__inv_1 (
    Y   ,
    A   ,
    VPWR,
    VGND,
    VPB ,
    VNB);
    output Y   ;
    input  A   ;
    input  VPWR;
    input  VGND;
    input  VPB ;
    input  VNB ;
sky130_fd_sc_hd__inv base (
.Y(Y),         .A(A),
.VPWR(VPWR),         .VGND(VGND),
.VPB(VPB),         .VNB(VNB)
);endmodule`endcelldefine
```

Netlist for Spice sim

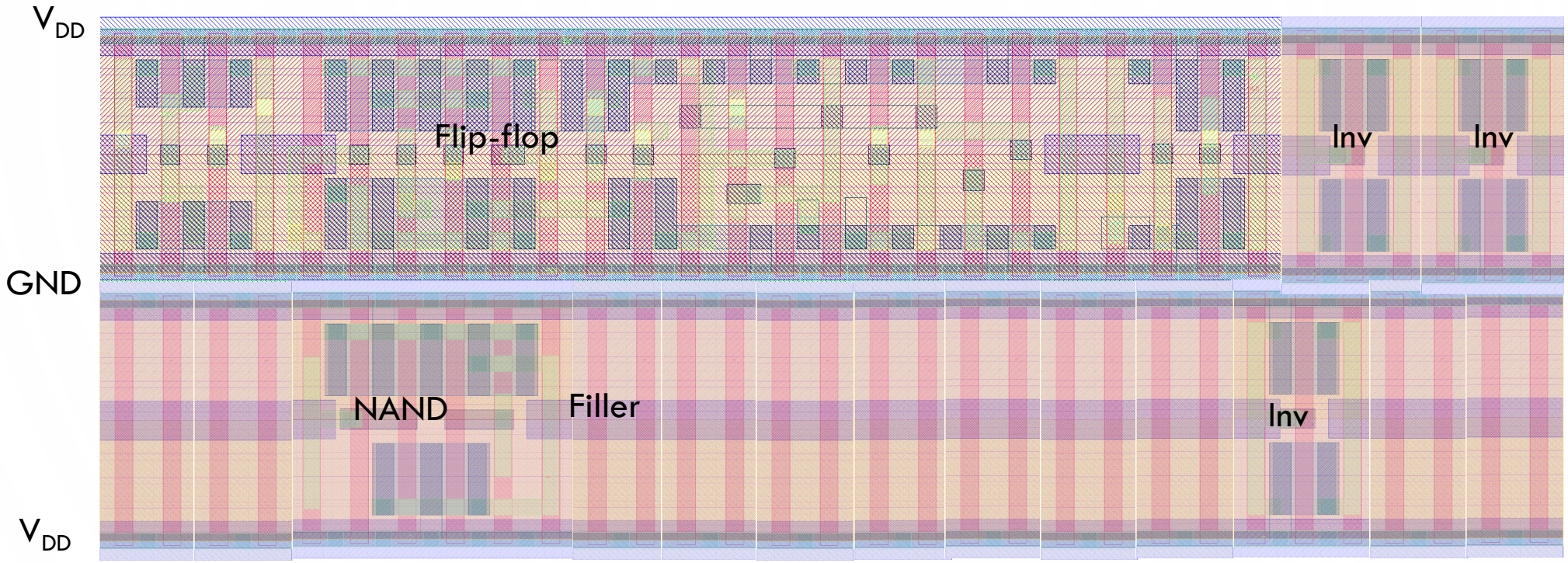
```
.subckt sky130_fd_sc_hd__inv_1 A VGND VNB VPB VPWR YX0 VGND A Y
VNB sky130_fd_pr__nfet_01v8 w=650000u l=150000uX1 VPWR A Y VPB
sky130_fd_pr__pfet_01v8_hvt w=1e+06u l=150000u.ends
```

+ .lef for place and route, and other files

Standard Cells

Name	Symbol	Schematic	Layout
1x Inverter			
2x Inverter			
1x NAND2			

Standard Cell Placement

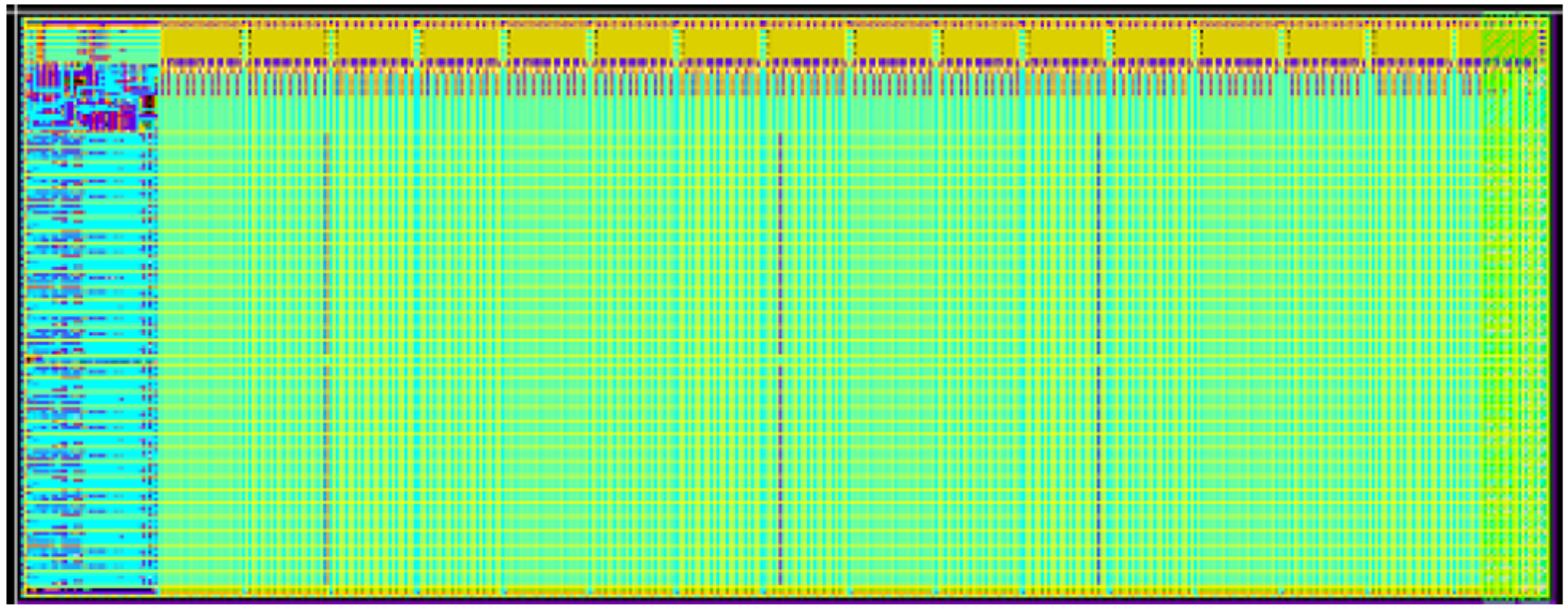


Standard cells abut and flip

Router connects inputs and outputs

Macro modules

256×32 (or 8192 bit) SRAM Generated by hard-macro module generator



- Generate highly regular structures (entire memories, multipliers, etc.) with a few lines of code
- Verilog models for memories automatically generated based on size

Quiz

True or false?

a) FPGAs are usually faster than ASICs

b) Verilog syntax is different for FPGAs and ASICs

c) One cannot change the FPGA function in a deployed product

www.yellkey.com/suddenly

```
abc
1.FFF
2.FFT
3.FTF
4.FTT
5.TFF
6.TFT
7.TTF
8.TTT
```

Hardware Description Languages

Hardware Description Languages

Verilog:

- Simple C-like syntax for structural and behavior hardware constructs
- Mature set of commercial tools for synthesis and simulation
- Used in EECS 151 / 251A

VHDL:

- Semantically very close to Verilog
- More syntactic overhead
- Extensive type system for "synthesis time" checking

System Verilog:

- Enhances Verilog with strong typing along with other additions to support verification

BlueSpec:

- Invented by Prof. Arvind at MIT
- Originally built within the Haskell programming language
- Now available commercially: bluespec.edu

Chisel:

- Developed at UC Berkeley
- Used in CS152, CS250
- Available at: chisel.eecs.berkeley.edu

Verilog: Brief History

- Originated at Automated Integrated Design Systems (renamed Gateway) in 1985. Acquired by Cadence in 1989.
- Invented as simulation language. Synthesis was an afterthought. Many of the basic techniques for synthesis were developed at Berkeley in the 80's and applied commercially in the 90's.
- Around the same time as the origin of Verilog, the US Department of Defense developed VHDL (A double acronym! VSIC (Very High-Speed Integrated Circuit) HDL). Because it was in the public domain it began to grow in popularity.
- Afraid of losing market share, Cadence opened Verilog to the public in 1990.
- An IEEE working group was established in 1993, and ratified IEEE Standard 1394 (Verilog) in 1995. We use IEEE Std 1364-2005.
- Verilog is the language of choice of Silicon Valley companies, initially because of high-quality tool support and its similarity to C-language syntax.
- VHDL is still popular within the government, in Europe and Japan, and some Universities.
- Most major CAD frameworks now support both.

Summary

- The design process involves traversing the abstraction layers of specification, modeling, architecture, RTL design and physical implementation
- Targets are processors, FPGAs or ASICs
- Automated design flows help manage the complexity
- Optimize for performance, energy and cost