SYNOPSYS®
Predictable Success

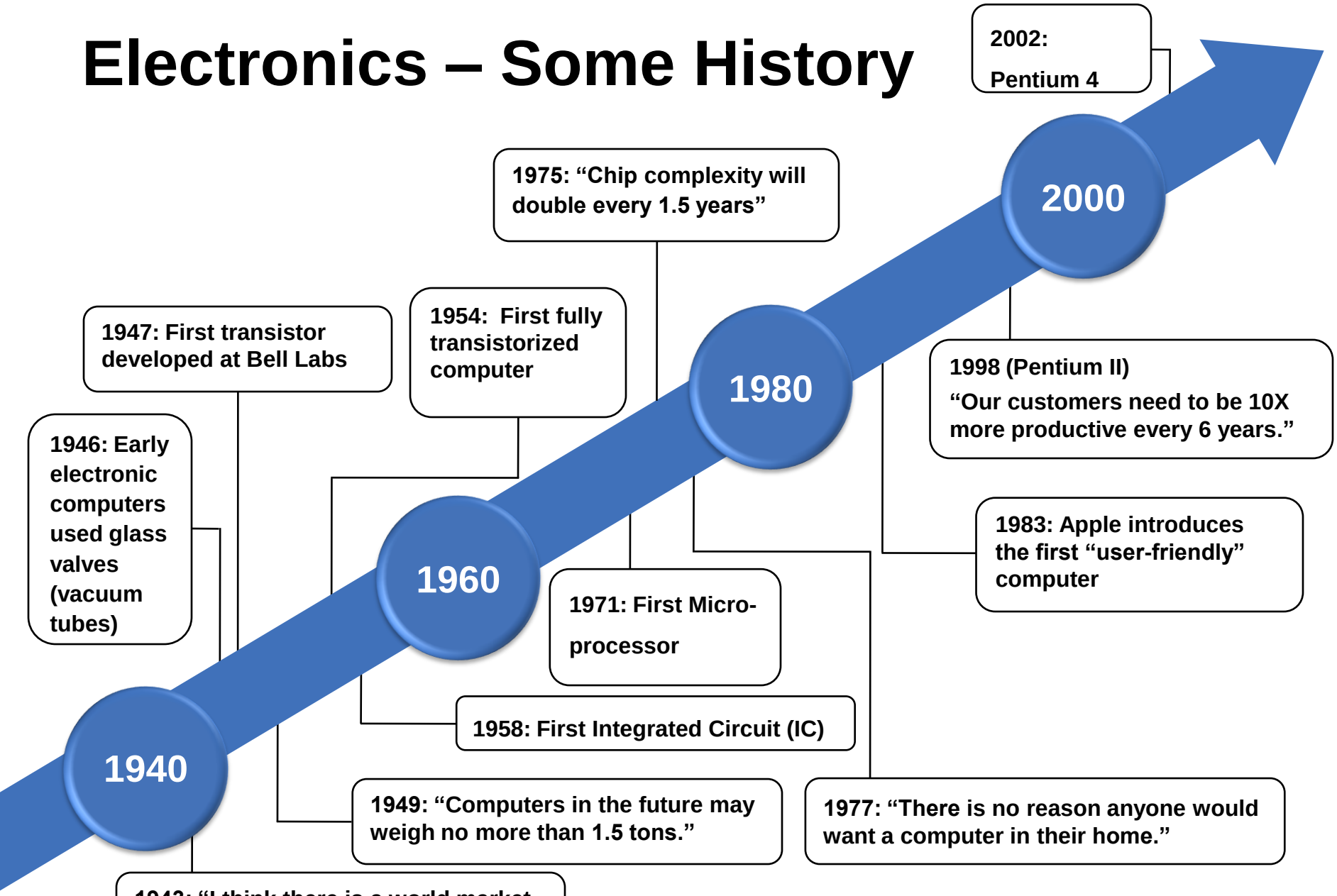
Digital Integrated Circuits Design Flow

Victor Grimblatt
R&D Director
Marzo 2011

Agenda

- Introduction
- CMOS Technology
- ASIC Design Flow

Electronics – Some History



Let's Make a Lot of Money!

- Semiconductor chips are pervasive
- Semiconductor industry is about $256.3B*
- EDA industry is about $4.6B*
  - Annual revenue:
    - Synopsys 1.37B*
    - Cadence 1.16B*
    - Mentor 0.81B*
    - Magma 128M*
  - Market share*: S – 30%, M – 18%, C – 19%
- Market drivers include:
  - Time-to-market
  - Global competition
  - Lower costs
  - Technology: smaller, faster, denser

Productivity increases 10X every 6 years!

* 2009

Smaller is More

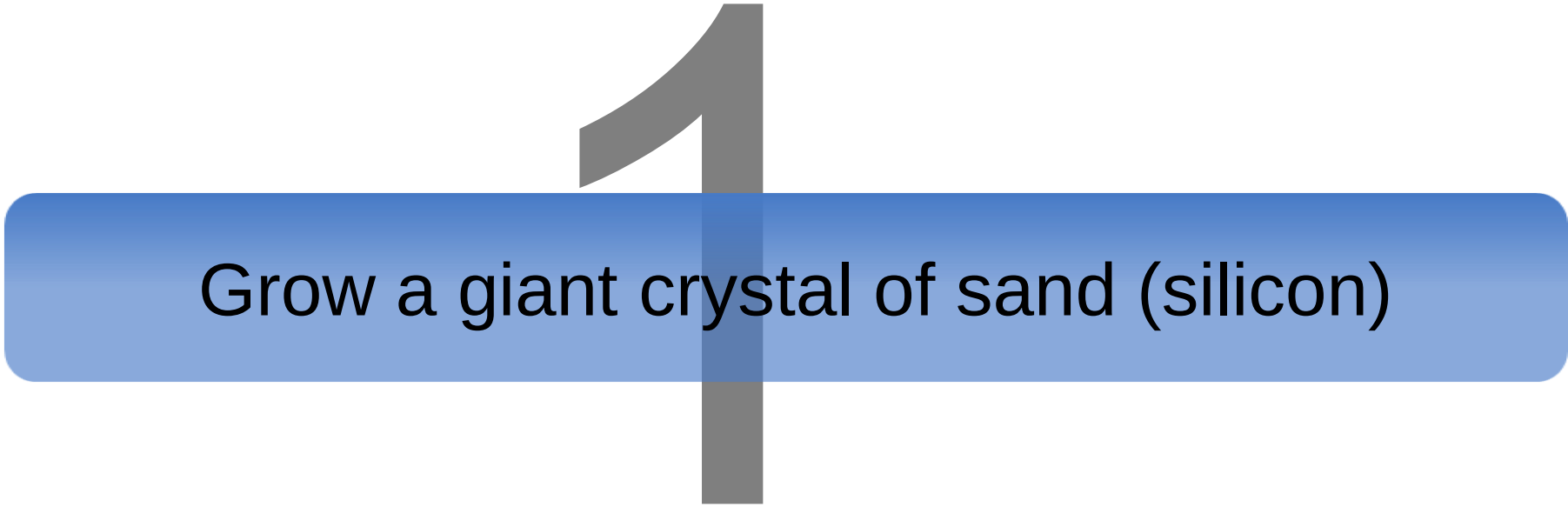
How can you put 100 million switches on a chip the size of your fingernail?

How Do You Make a Chip?

14

steps

How Do You Make a Chip?



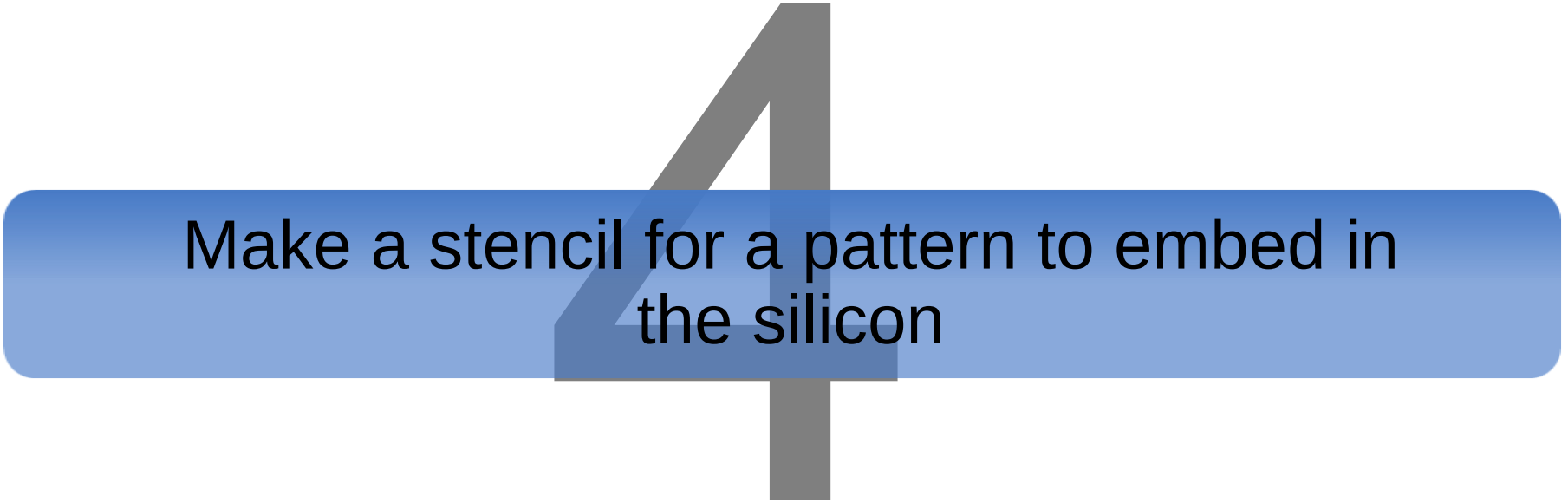
How Do You Make a Chip?

2

Slice it up into round wafers & polish them

How Do You Make a Chip?

How Do You Make a Chip?



How Do You Make a Chip?

5

Shrink the stencil and shine a light through it

How Do You Make a Chip?

6

Dip the wafer in acid to etch away the soft parts

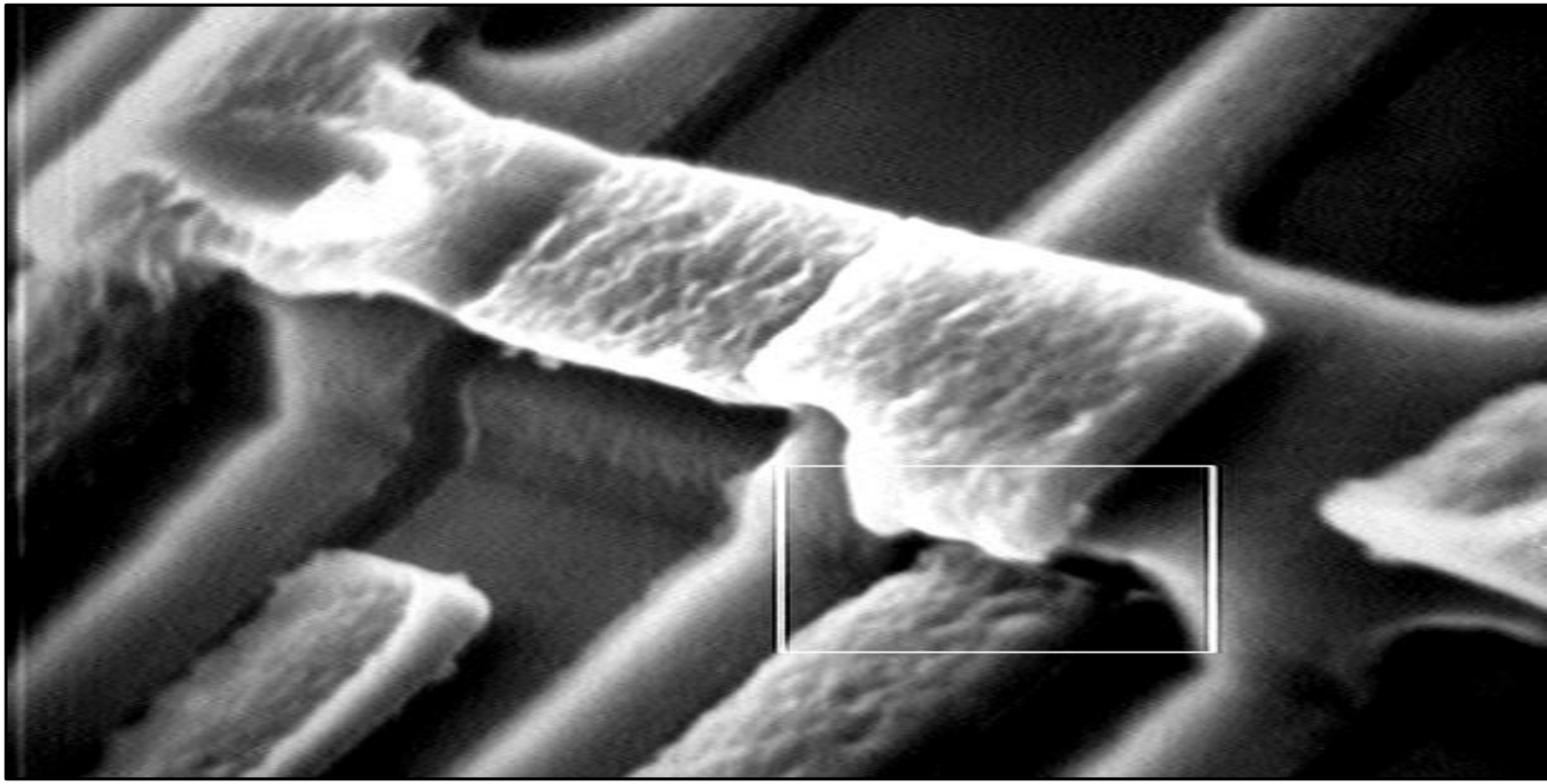
How Do You Make a Chip?

7

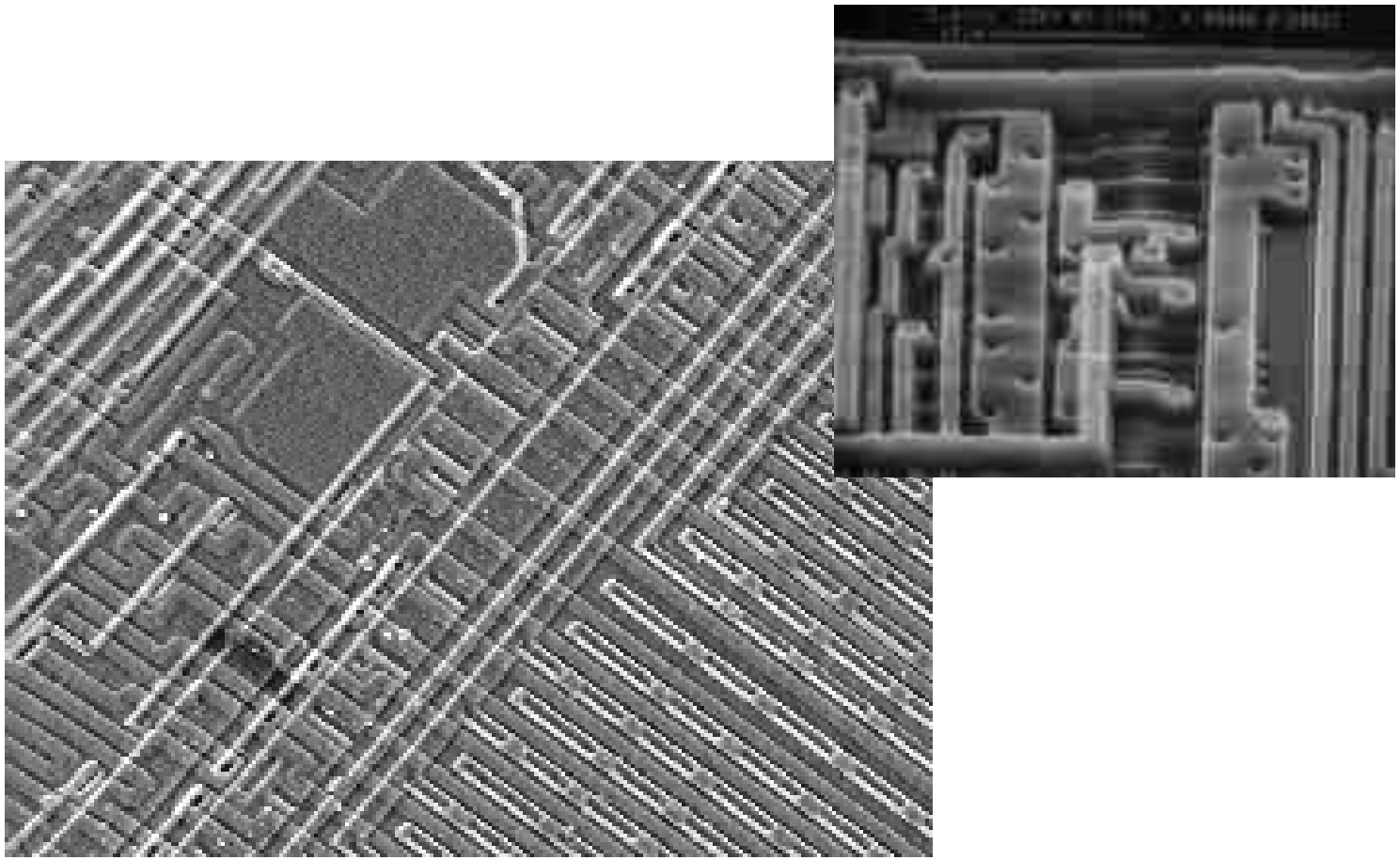
Repeat steps 3 - 6 many times, producing layers of patterns etched into the wafer

➡ **(cool term: Photolithography)**

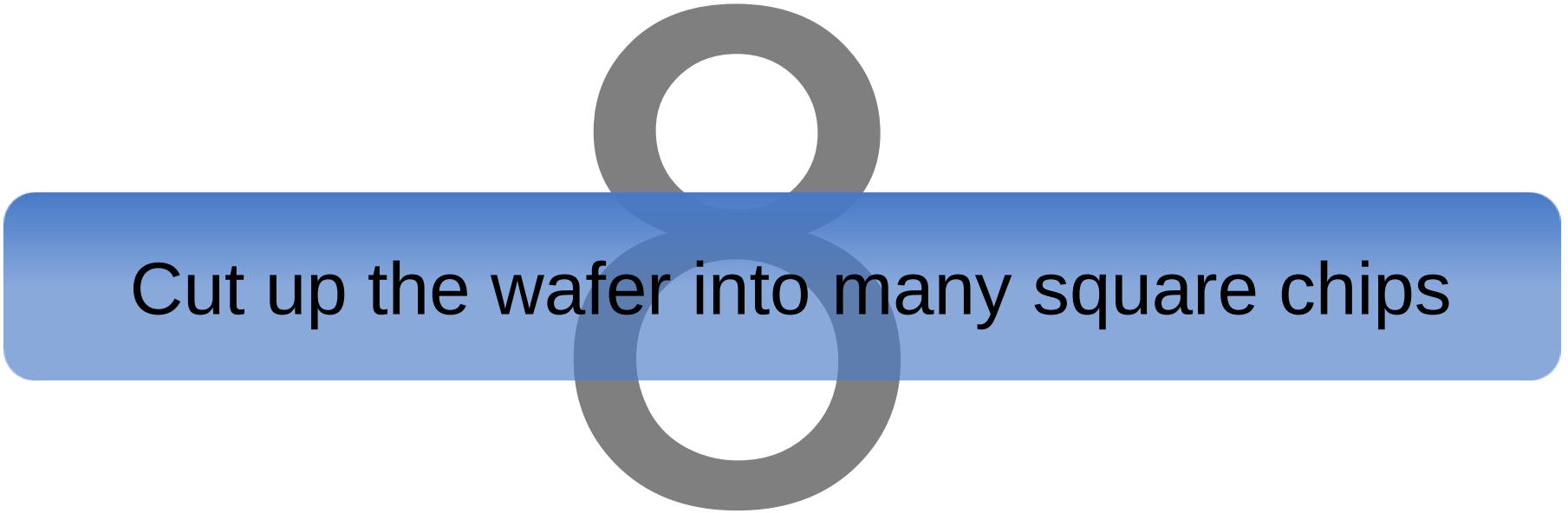
Under the Microscope



Look at the Layers



How Do You Make a Chip?



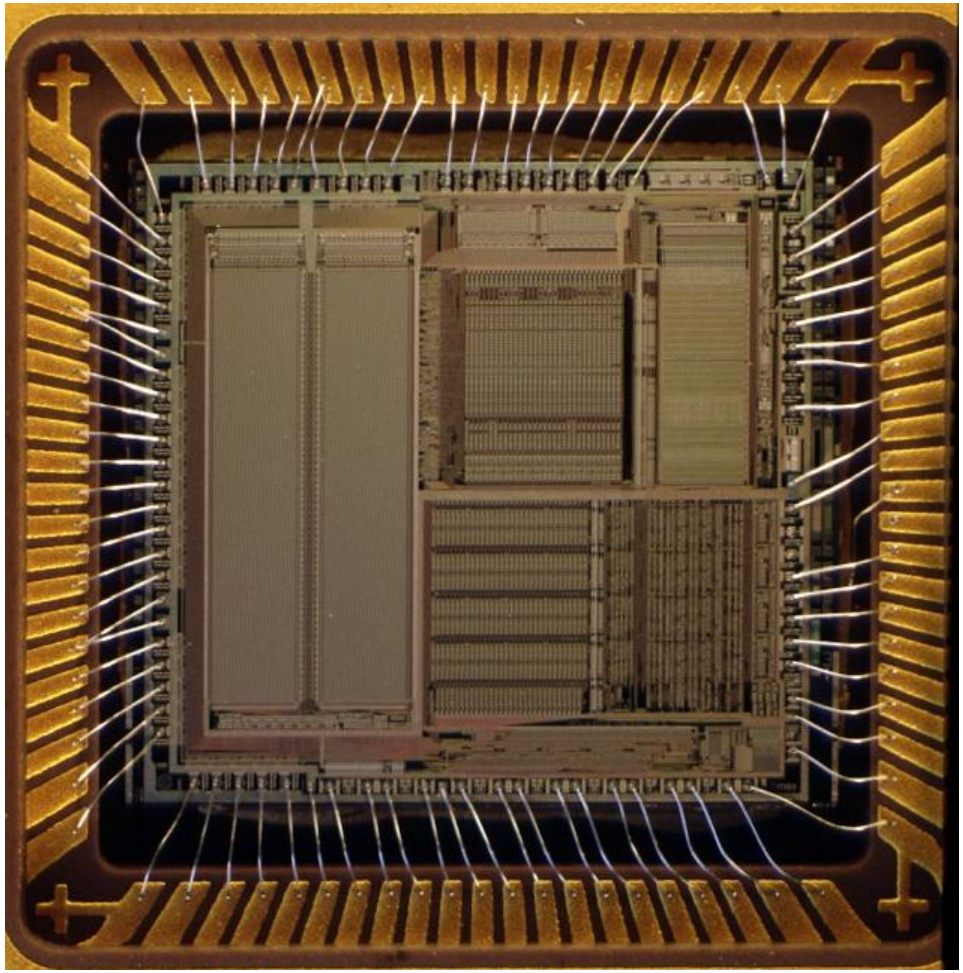
How Do You Make a Chip?

How Do You Make a Chip?

10

Connect the chip to the pins
of the package with tiny gold wires

Chip Connects to the Outside World



How Do You Make a Chip?

11

Put the chip on a tester machine
and run a test

How Do You Make a Chip?

12

Throw away the chips that fail the test!

How Do You Make a Chip?

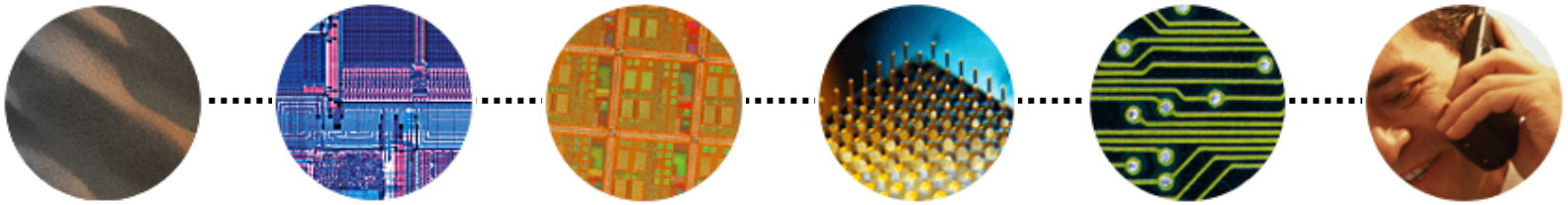
13

Assemble different kinds of chips onto a board

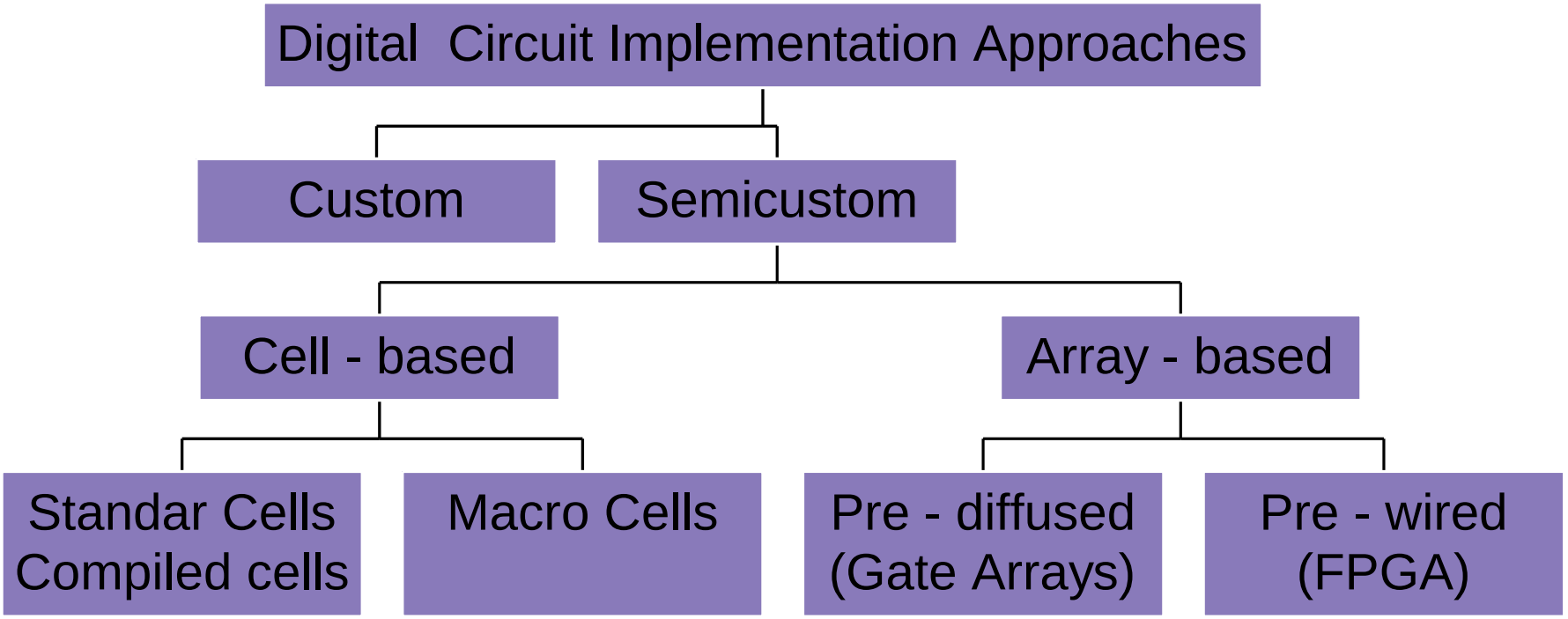
How Do You Make a Chip?

14

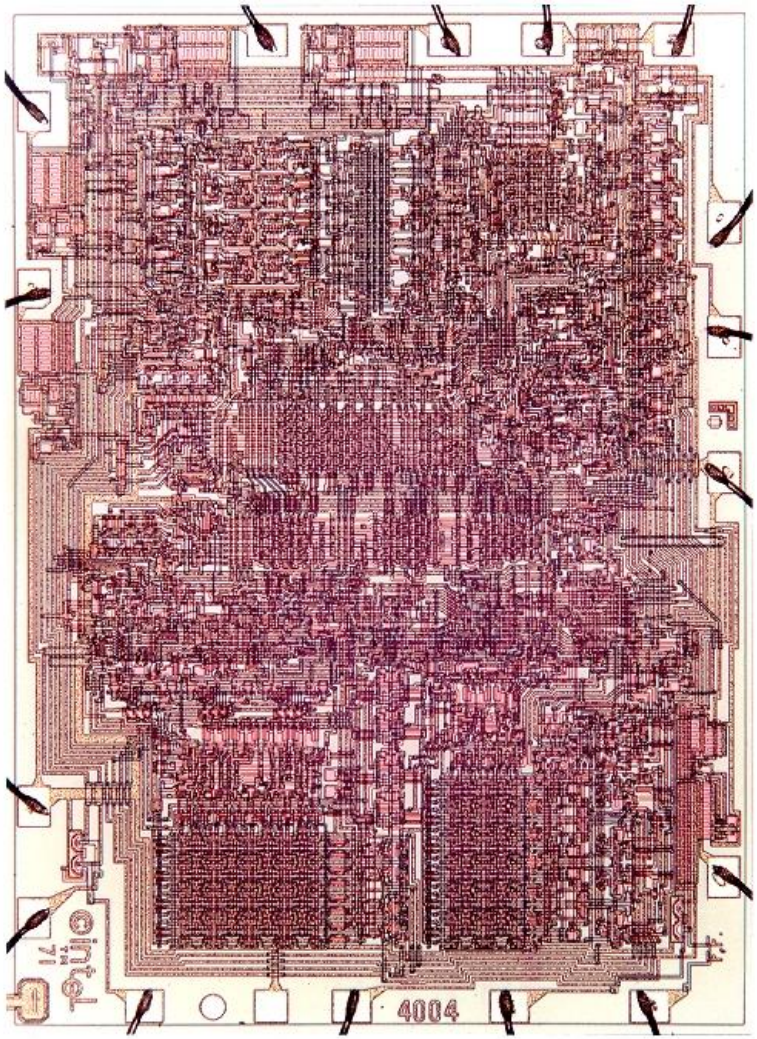
Install the board into a phone, computer...



Design Styles



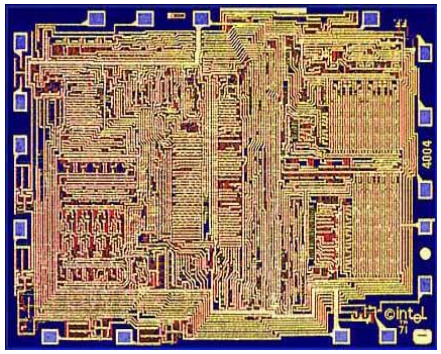
The Custom Approach



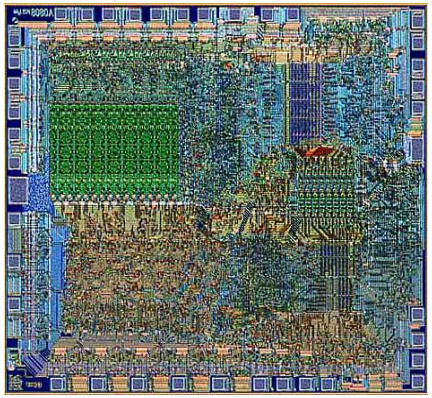
Intel 4004

Courtesy Intel

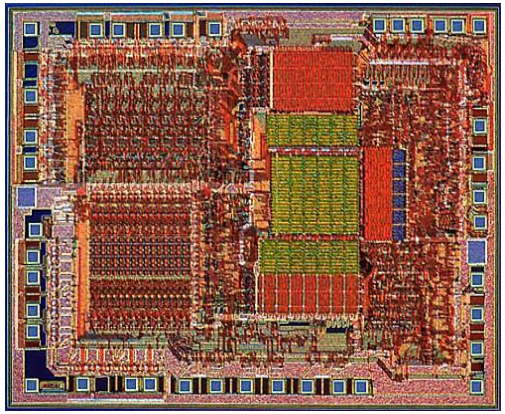
Transition to Automation and Regular Structure



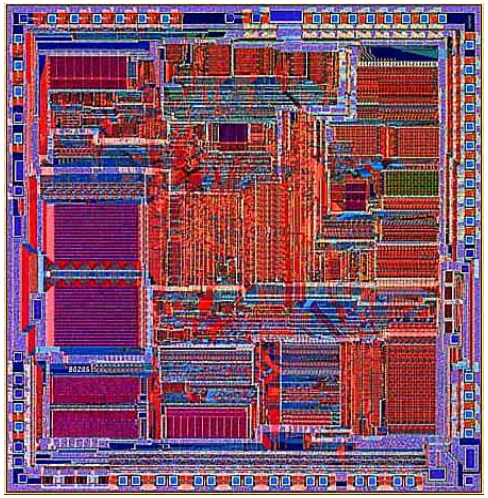
Intel 4004 (‘71)



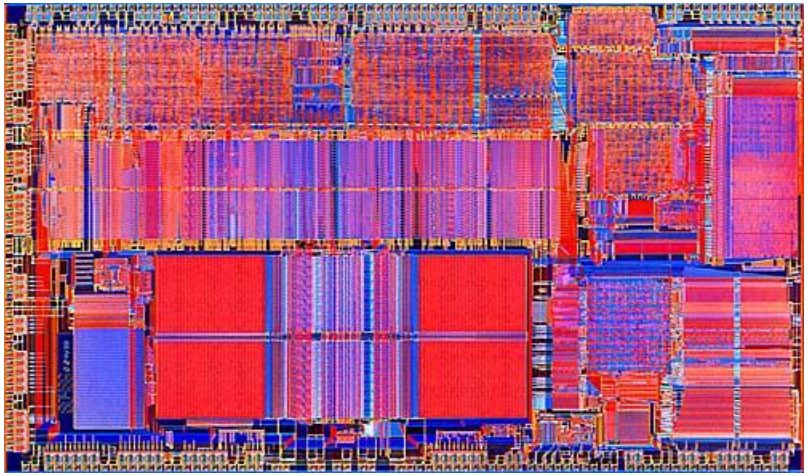
Intel 8080



Intel 8085



Intel 8286



Intel 8486

Courtesy Intel

Types of Circuits

- ASIC (Application Specific Integrated Circuit)
- General Purpose Integrated Circuits

ASIC

- A chip designed to perform a particular operation
- Generally not software programmable to perform different tasks
- Often has an embedded CPU
- May be implemented in FPGA

Examples of ASIC

- Video processor to decode or encode MPEG-2 digital TV signals
- Encryption processor for security
- Many examples of graphical chips
- Network processor for managing packets, traffic flow, etc.

Full Custom ASICs

- Every cell and transistor is designed by hand from scratch
- Only way to design analog portions of ASICs
- Highest performance but longest design time
- Full set of masks required for fabrication

Standard Cell ASICs (semi custom)

- Designer uses predesigned logic cells (e.g. AND, NAND)
- Designers save time, money, and reduce risk
- Standards cells can be optimized individually
- Standard cells library is designed using full custom
- Some standar cells, such as RAM and ROM, and some datapath cells (e.g. multiplier) are tiled together to create macrocells

Gate Array ASICs

- Transistors are predefined in the silicon wafer
- Predefined pattern of transistors is called “base array”
- Smallest element is called “base cell”
- Base cell layout is the same for each logic cell
- Only interconnection between cells and inside the cell is customized
- Slower than cell based designs but implementation time is faster (less time in factory)

Agenda

- Introduction
- CMOS Technology
- ASIC Design Flow

CMOS Technology

- Complementary Metal Oxide Semiconductor
- Consists of NMOS and PMOS transistors
- MOS (Metal Oxide Semiconductor) Field Effect transistor

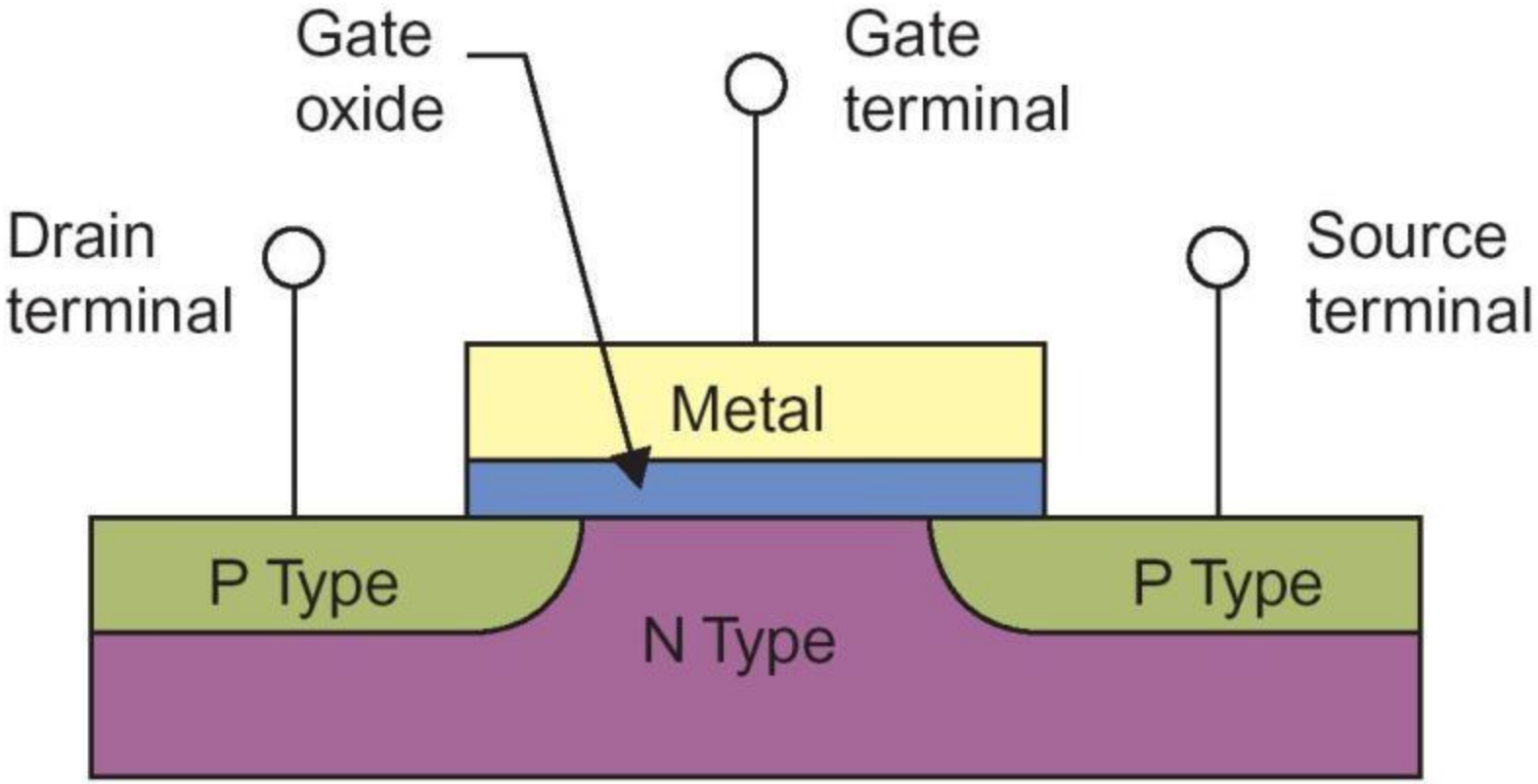
MOS Transistor

- Basic element in the design
- Voltage controlled device
- It is formed by layers
  - Semiconductor layer
  - Silicon dioxide layer
  - Metal layer
- Consists in three regions
  - Source
  - Drain
  - Gate

MOS Transistor

- Source and drain are quite similar
- Source is the node which acts as the source of charge carriers
- Charge carriers leave the source and travel to the drain
- Source is the more negative terminal in N channel MOS, it is the more positive terminal in P channel MOS
- Area under the gate is called the “channel”

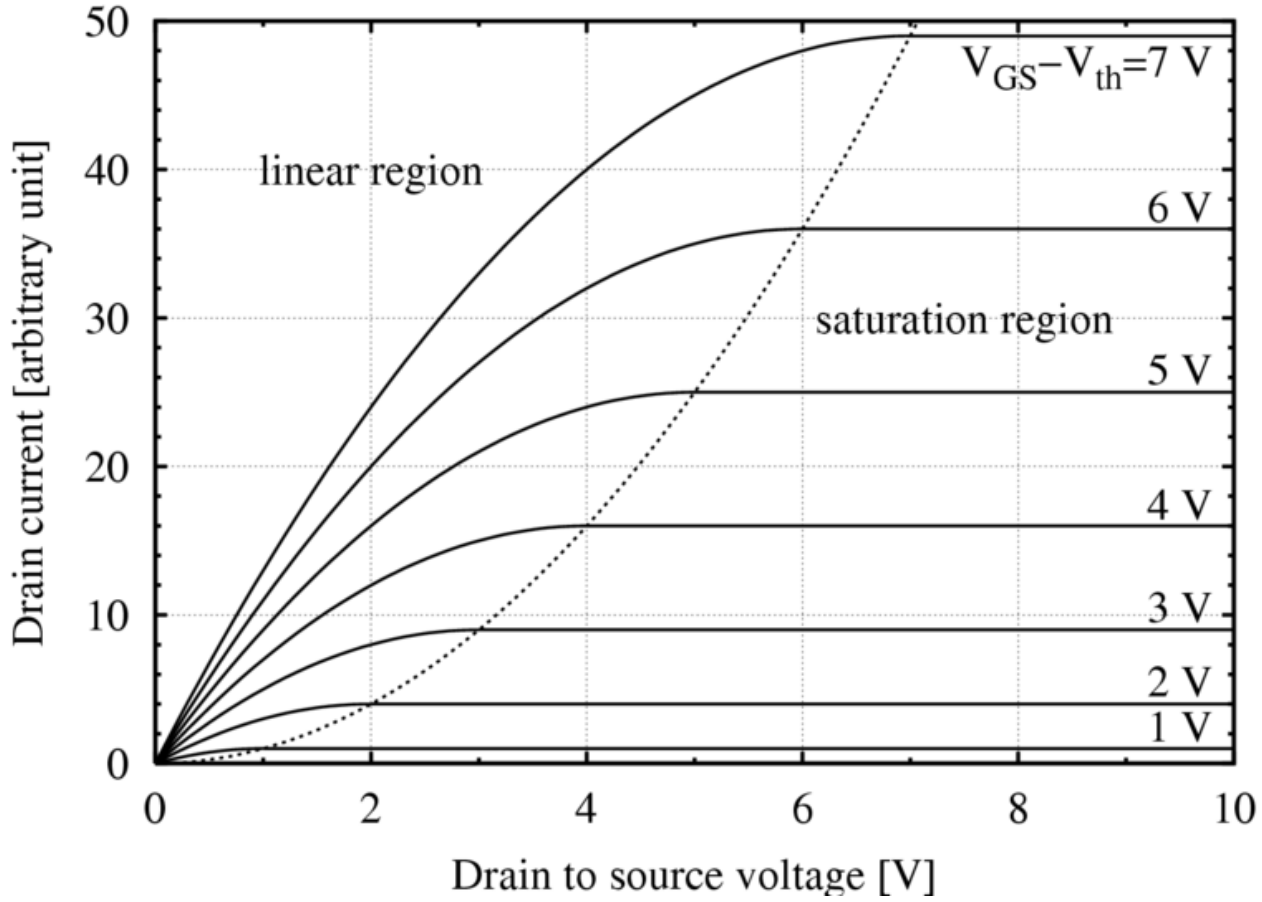
MOS Transistor



MOS Transistor

- Needs some kind of voltage initially for the channel to form
- When the channel is not yet formed, the transistor is in the “cut off region”
- Threshold Voltage: voltage at which the transistor stars conducting (channel begin to form)
- Transistor in “linear region” at threshold voltage
- When no more charge carriers go from the source to the drain, the transistor is into the “saturation region”

MOS Transistor



CMOS Technology

- Made up of both NMOS and CMOS transistors
- CMOS logic devices are the most common devices used today in IC design
  - High density
  - Low power
  - High operating clock speed
  - Ease of implementation at the transistor level
- Only one driver but gates can drive as many gates as possible
- Gates output always drives another CMOS gate input

CMOS Inverter

- Requires one PMOS and one NMOS transistors
- NMOS provides the switch connection to ground when the input is logic high → Output load is discharged and output is driven to logic “0”
- PMOS transistor provides the connection to power supply (V_{DD}) when the input is logic low → Output load is charged and output is driven to logic “1”

CMOS Technology

- “Holes” are the charge carriers for PMOS transistors
- Electrons are the charge carriers for NMOS transistors
- Mobility of electrons is 2 times than that of “holes”
- Output rise and fall time is different between PMOS and NMOS transistors
- To adjust time PMOS W/L ratio is about twice NMOS W/L ratio

CMOS Technology

- L is always constant in a standard cell library
- W changes to have different drive strengths
- Resistance is proportional to L/W → increasing the width decreases resistance

Power Dissipation

- Big percentage is due to the charging and discharging of capacitors
- Low power design are techniques used to reduce power dissipation

Sources of Power Dissipation

- Dynamic Switching Power: Due to charge and discharge of capacitances
  - Low to high output transition draws energy from the power supply
  - High to low transition dissipates energy stored in CMOS transistor
  - Total power drawn = load capacitance*V_{DD}^2*f
- Short Circuit Current: Occurs when the rise/fall time at the input of the gate is larger than the output rise/fall time

Sources of Power Dissipation

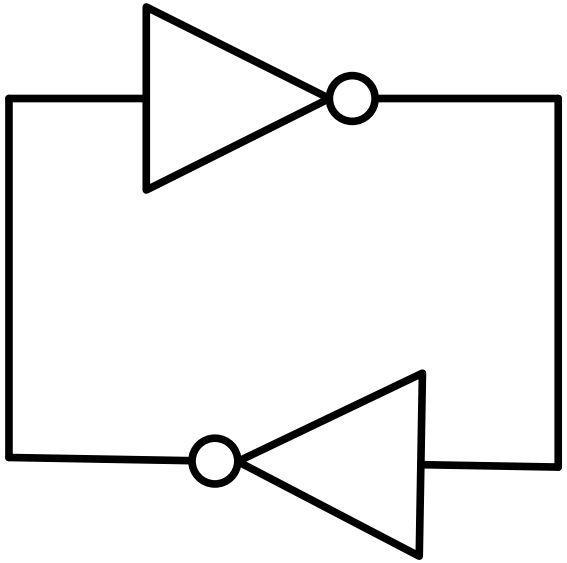
- Leakage Current Power: Caused by 2 reasons
  - Reverse-Bias diode leakage on transistor drains. It happens when one transistor is off, and the active one charges up/down the drain using the bulk potential of the other transistor
  - Sub-Threshold leakage through the channel to an “OFF” transistor/device

Transmission Gate

- Consists of a PMOS transistor connected in parallel to a NMOS transistor
- Transmits the value at the input to the output
- PMOS transmits a strong 1
- NMOS transmits a strong 0
- Advantages
  - Better characteristics than a switch
  - Resistance of the circuit is reduced (transistors in parallel)

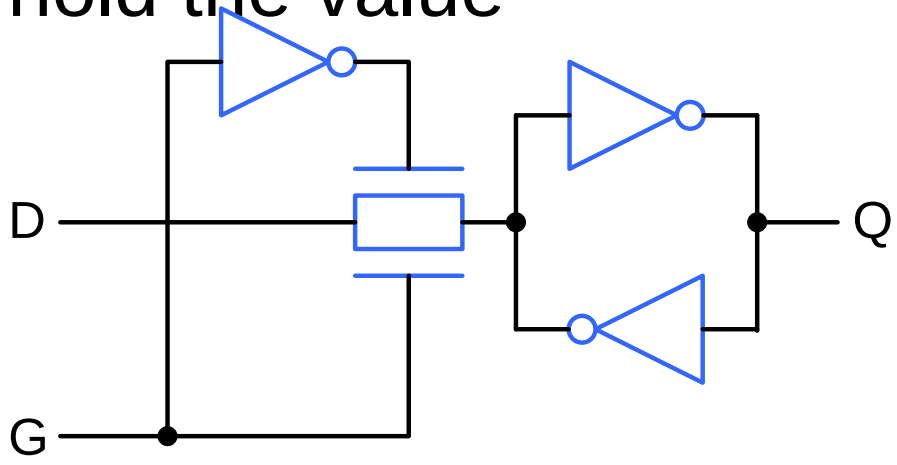
Sequential Element

- Stores a logic value by having a feedback loop
- There are two types
  - Latches
  - Flip Flops



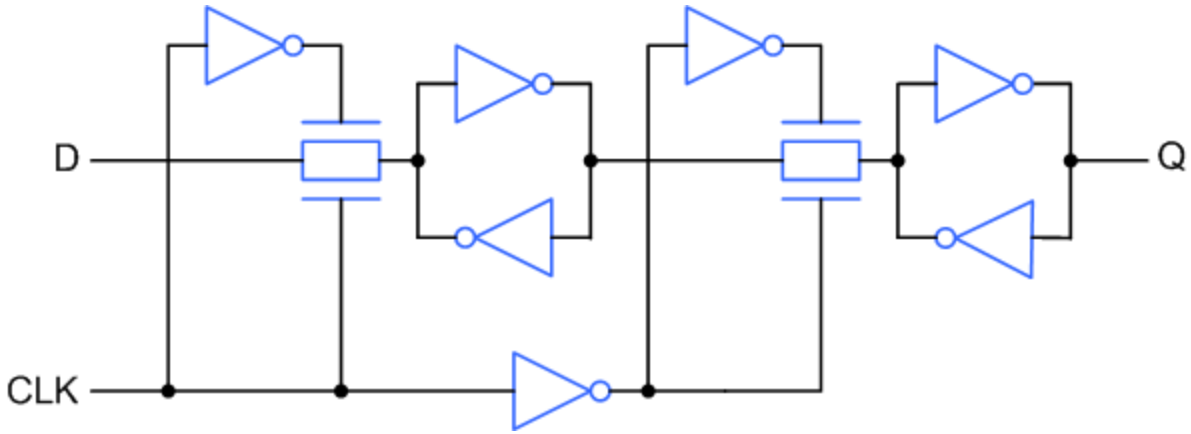
Latches

- Asynchronous element
- When G is high, transmission gate is switched on and input D passes to the output
- When G is low, transmission gate is off an inverters hold the value



Flip Flops

- Synchronous element
- FFs are constructed with 2 latches in series
- First latch is called Master, the second one is called Slave
- Inverted clock is fed to the slave latch transmission gate



Agenda

- Introduction
- CMOS Technology
- ASIC Design Flow

Design Objectives

- Speed
- Area
- Power
- Time to market

How Do You Design a Chip?

Today's chips have hundreds of thousands of gates

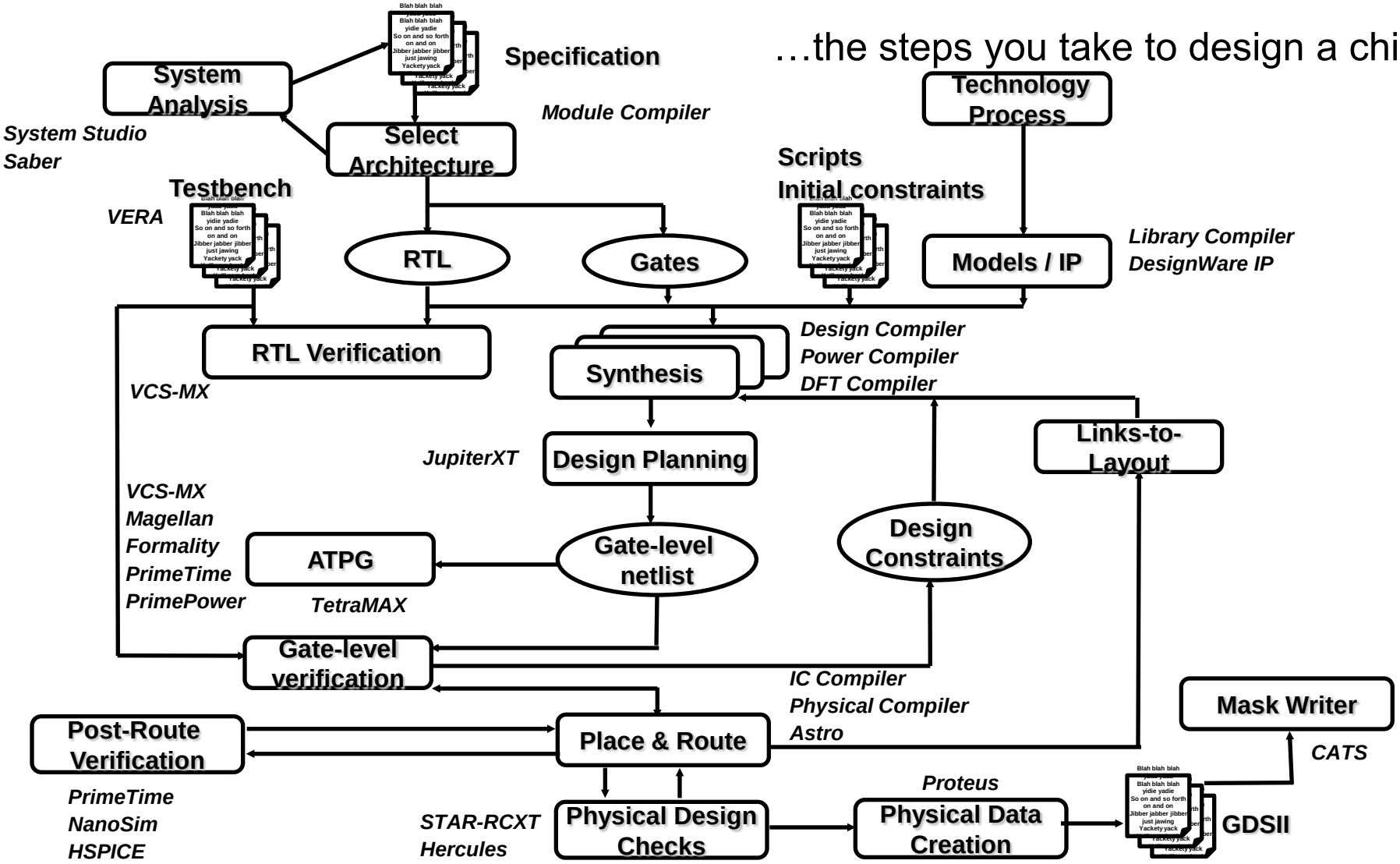
Designing a chip is the task of figuring out how to make the chip do what you want it to do

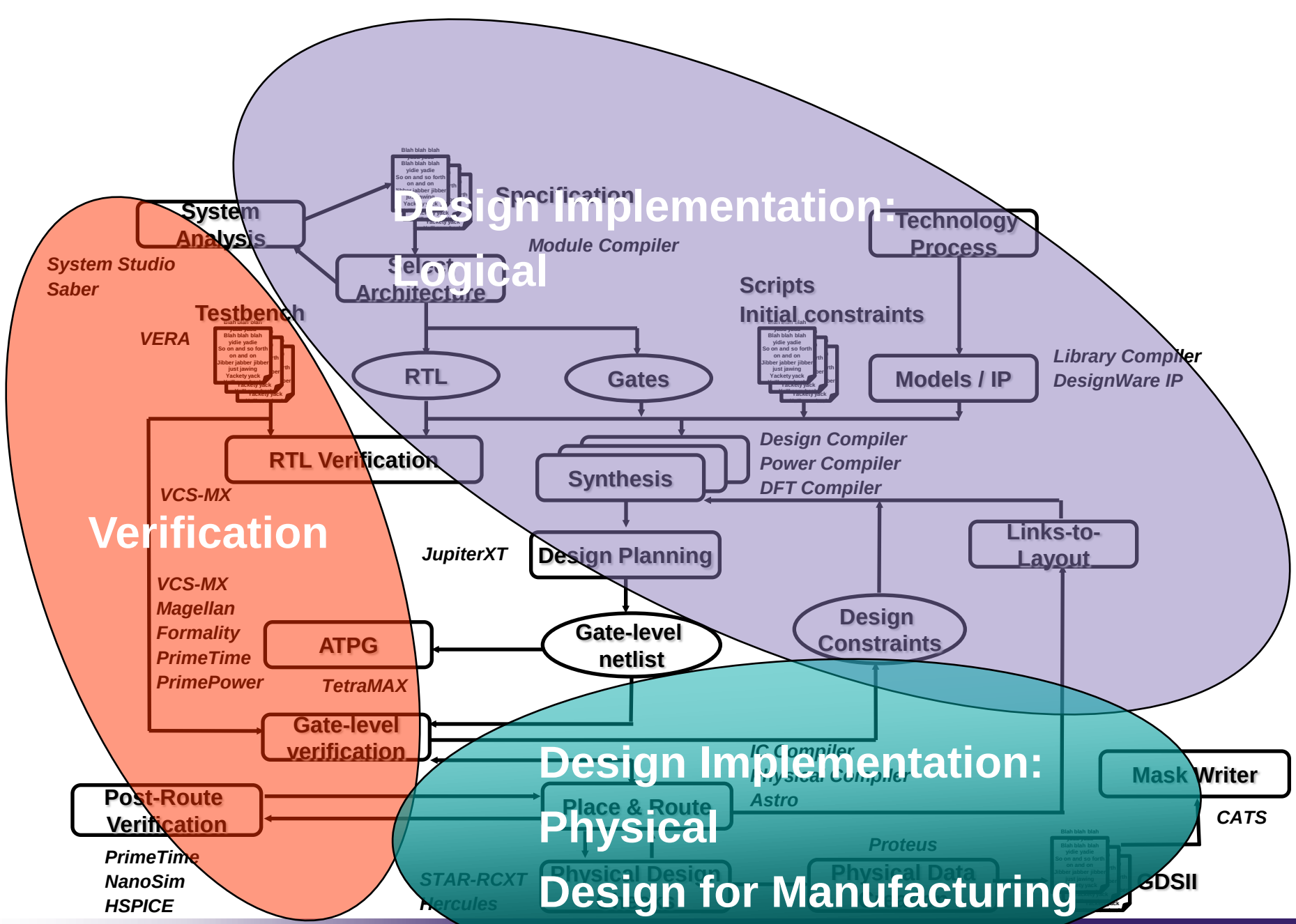
EDA=Electronic Design Automation

Without automation, this task would be impossible

What's a Design Flow?

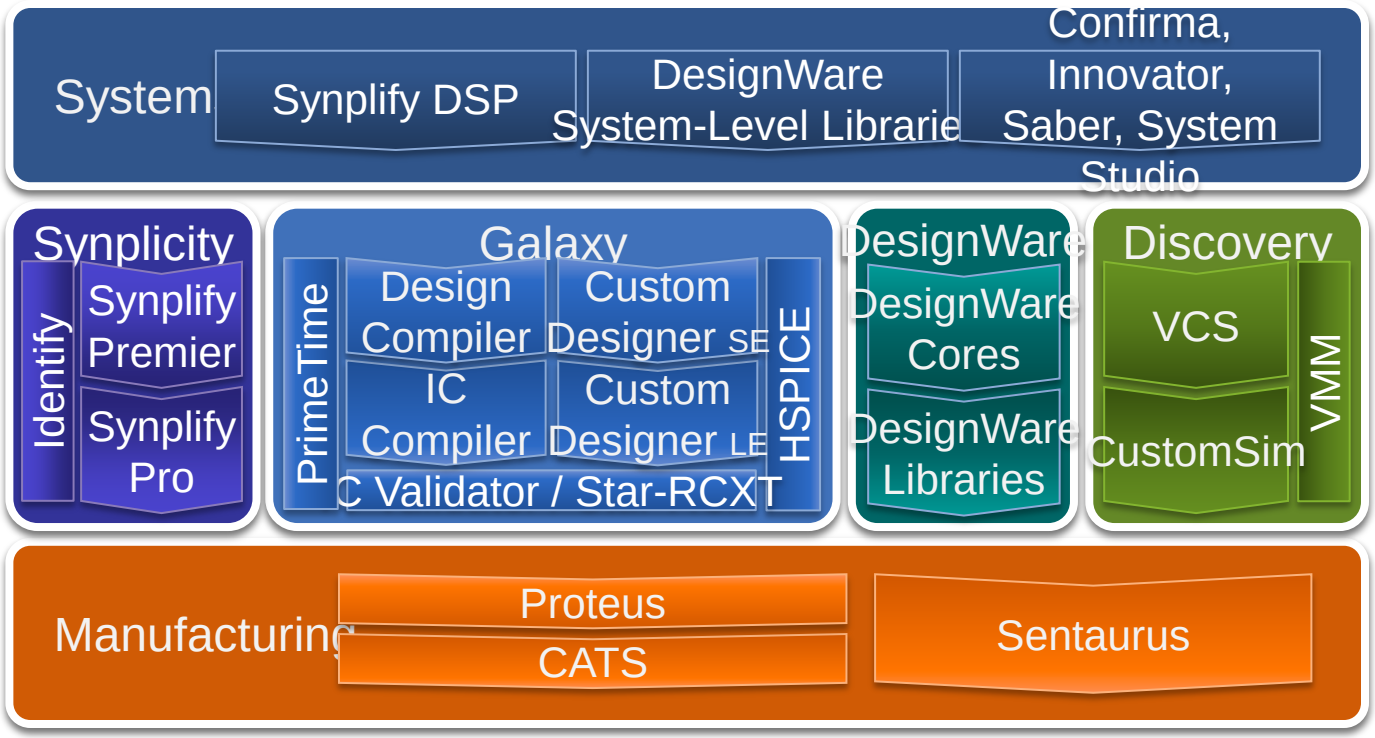
…the steps you take to design a chip!



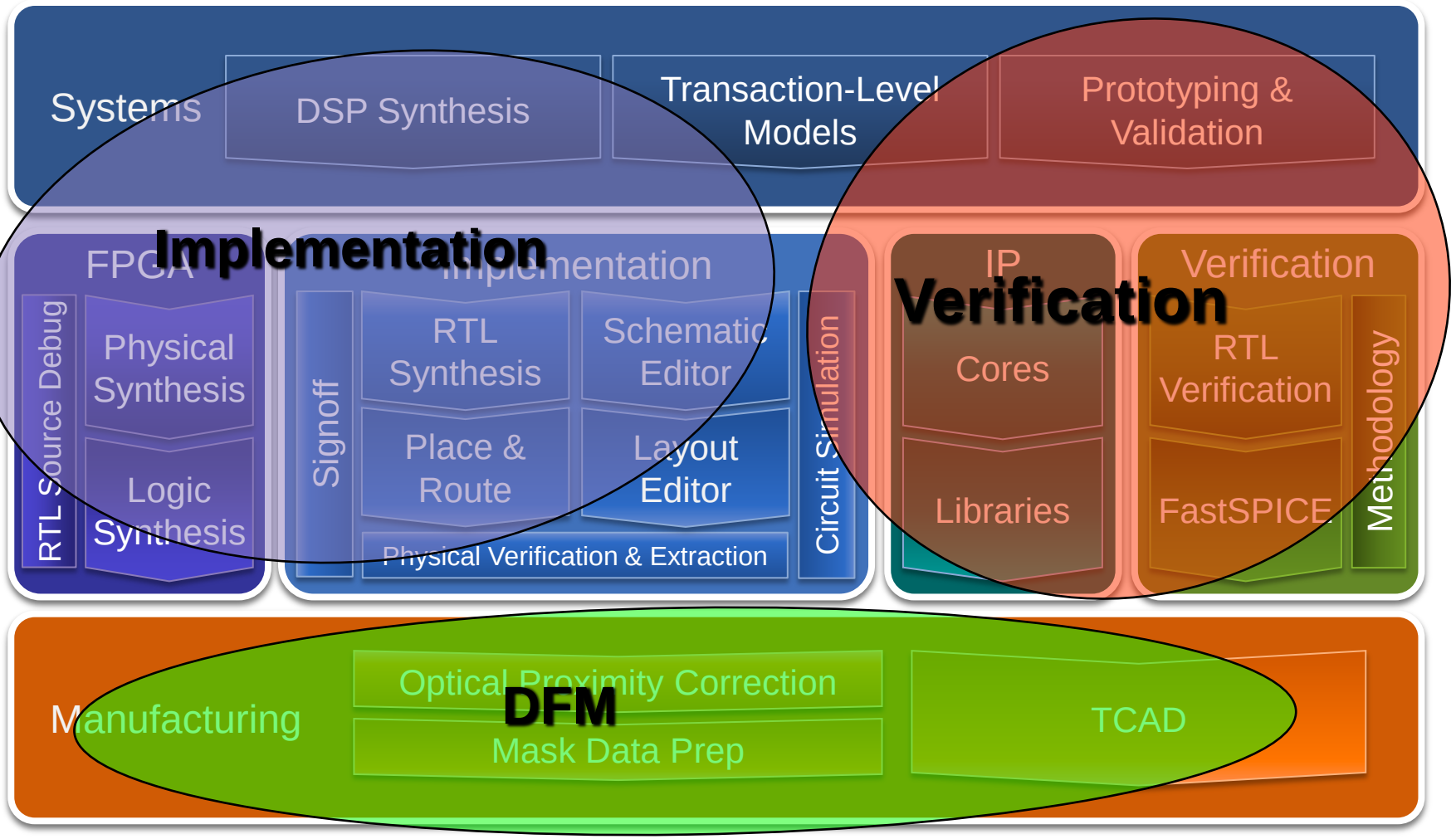


What's a Platform?

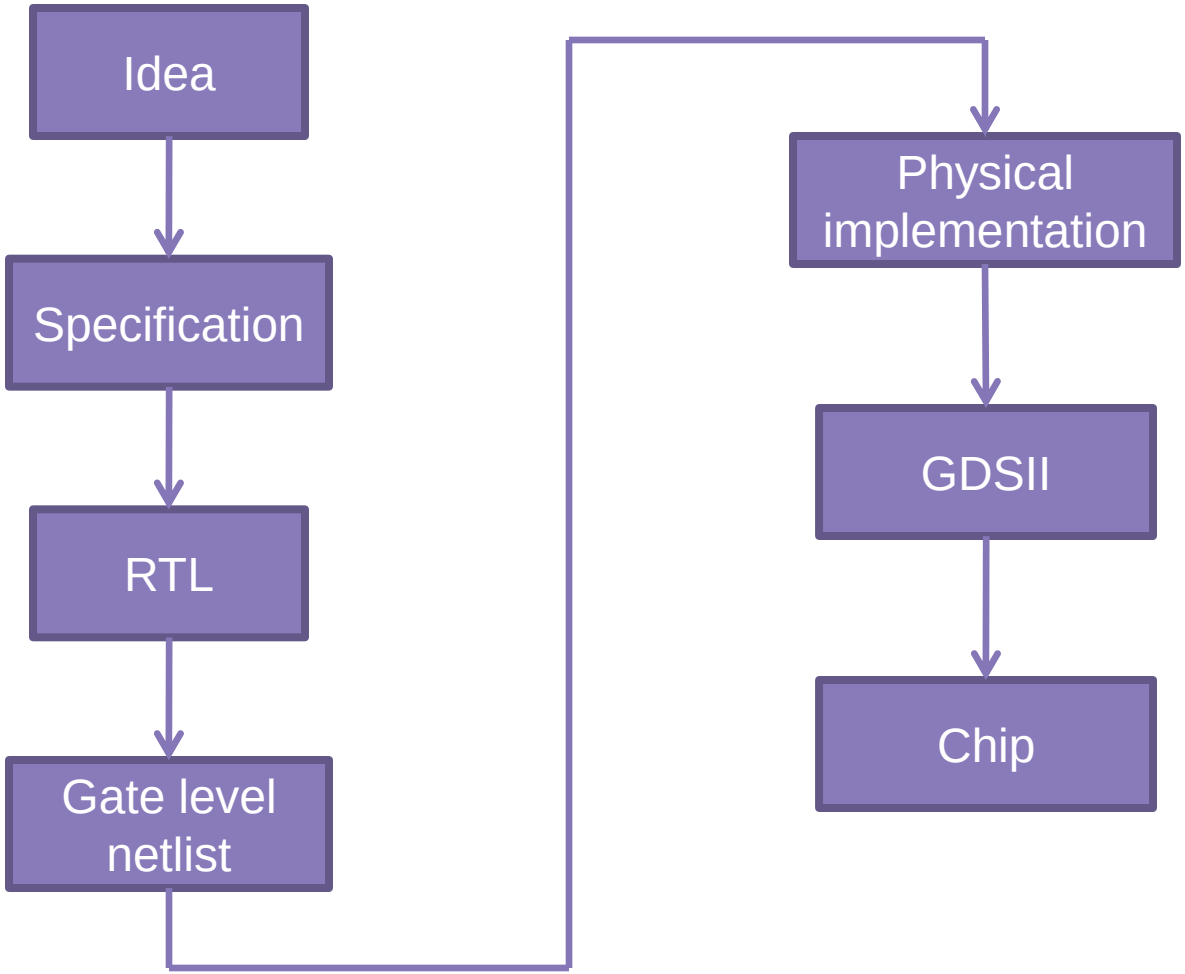
- A collection of EDA tools that work together well
- Not to be confused with a compute platform
  or platform-based design!



Product Areas



Simple ASIC Design Flow



A Simple Design Flow – 7 Steps

1. "Spec" your chip
2. Generate the gates
3. Make the chip testable

Design Implementation: *Logic*

4. Ensure the gates will work

Verification

5. Layout your chip
6. Double-check your layout
7. Turn your design into silicon

Design Implementation: *Physical*
Design for Manufacturing

“Spec” Your Chip

- Describe what you want your chip to do
- Write a “spec” - use a language like SystemVerilog or VHDL

A spec for a cell phone ringer might look like this:

if incoming_call AND line_is_available then RING;

“Spec” Your Chip

Today’s complex chip designs can have a million lines of specification created by a team of 100 people working for 6 months

“Spec” Your Chip

- You can buy a ready-made spec
  - *example: DesignWare Library*
- Give the spec to the computer to begin automation
  - *example: (V)HDL Compiler*

Specification

- Goals and constraints of the design
- Functionality (what the chip will do)
- Performance (e.g speed and power)
- Technology constraints (e.g. size and space)
- Fabrication technology

Structural and Functional Description

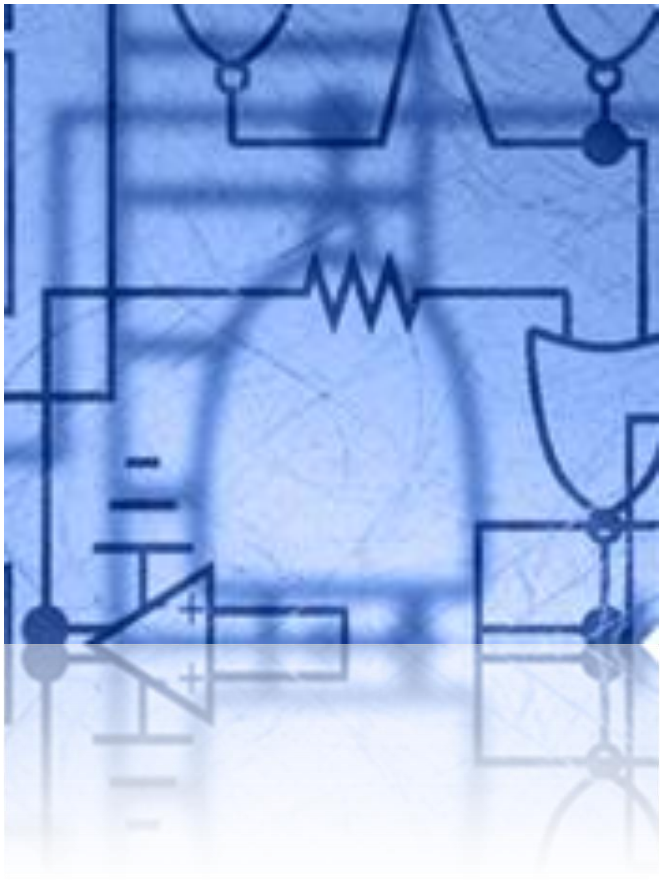
- Decide the circuit architecture (structure)
  - RISC/CISC
  - ALU
  - Pipelining
  - Etc.
- Breaks the system into several sub systems
- Functionality of sub systems should match the specification
- Sub systems need to be implemented through logic representation (boolean expressions), FSM, combinational logic, sequential logic, schematics, etc. → Logic Design

Register Transfer Level (RTL)

- Describes the several sub systems
- Should match the functional description
- Verilog or VHDL (Hardware description Language – HDL)
- HDlL: Language used to describe a digital system
- Verification is performed at this stage to ensure that RTL matches the idea

Generate the Gates

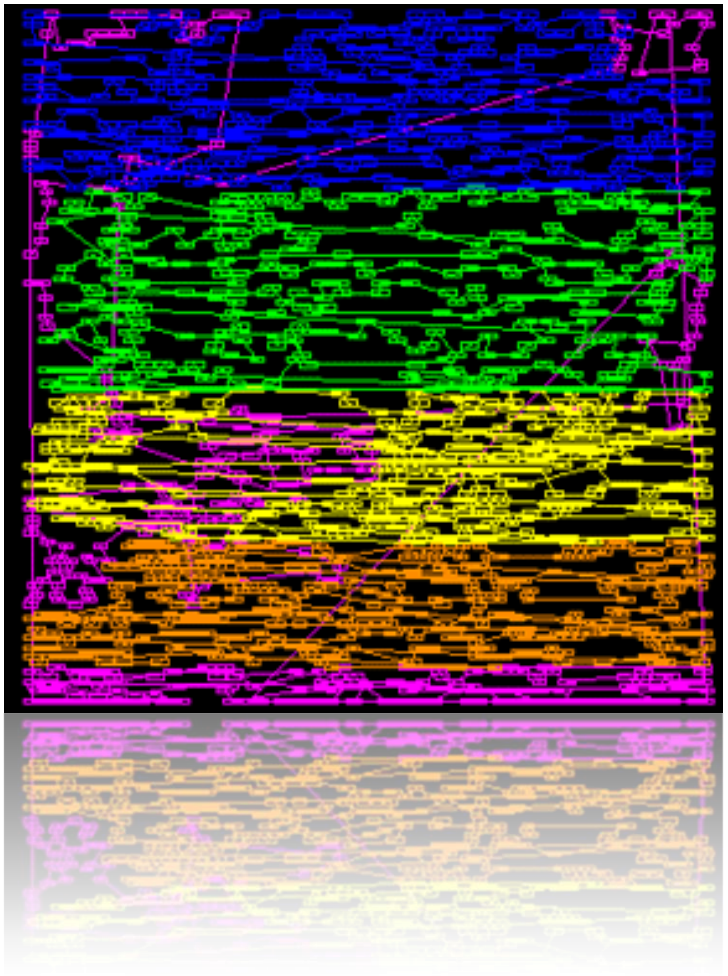
- Figure out the detailed logic gates
- Use a computer program (EDA tool)
- Synthesis
  - *example: Design Compiler, Physical Compiler, IC Compiler*
- Save the logic gates to use in a future design
  - *example: DesignWare Library*



Make the Chip Testable

- Help the manufacturer to find defects on the chip
- Add special gates that send information out of the chip
- Use EDA tools! Test Synthesis

  - example: DFT Compiler, DFT MAX



Gate Level Netlist

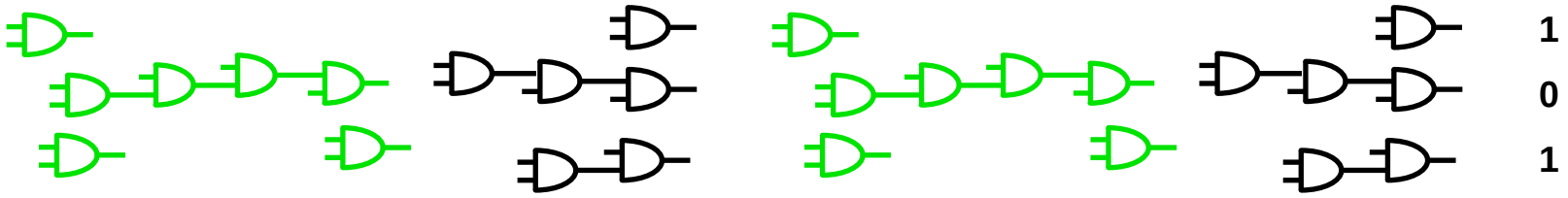
- Generated from the conversion of RTL into an optimized gate level netlist
- Done by synthesis tools
- Synthesis tool takes an RTL description and a standard cell library and produces a gate level netlist
- Considers constraints such as timing, area, testability, and power
- The result is a completely structural description with only standard cells at the leaves of the design
- This level is also verified by simulation or formal verification

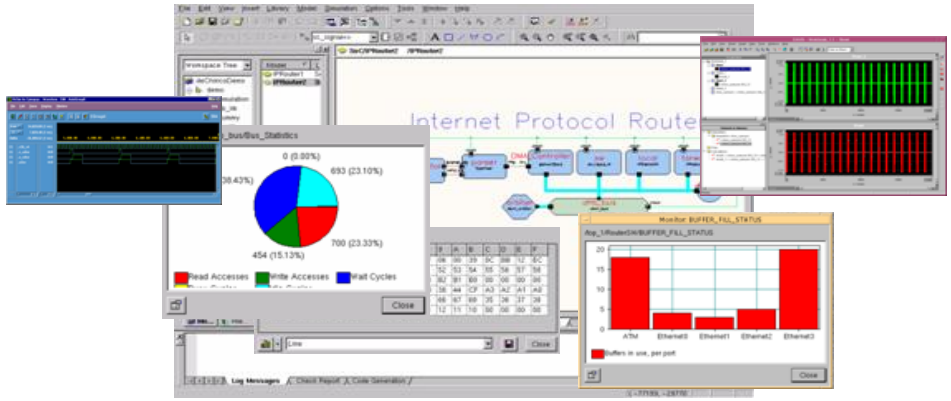
Ensure Gates Will Work

Verify that the logic gates will do what you want, when you want them to, with EDA tools

- Simulation, Timing Analysis, Testbench Generation

- example: VCS-MX, PrimeTime, VERA





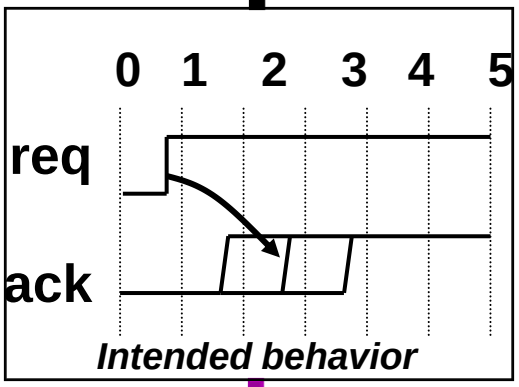
Ensure Gates Will Work

Put statements in the design description

- Assertion-based verification = “Smart Verification”

Assertion

```
assert property (@(posedge clk) $rose(req)|-> ##[1:3] $rose(ack));
```

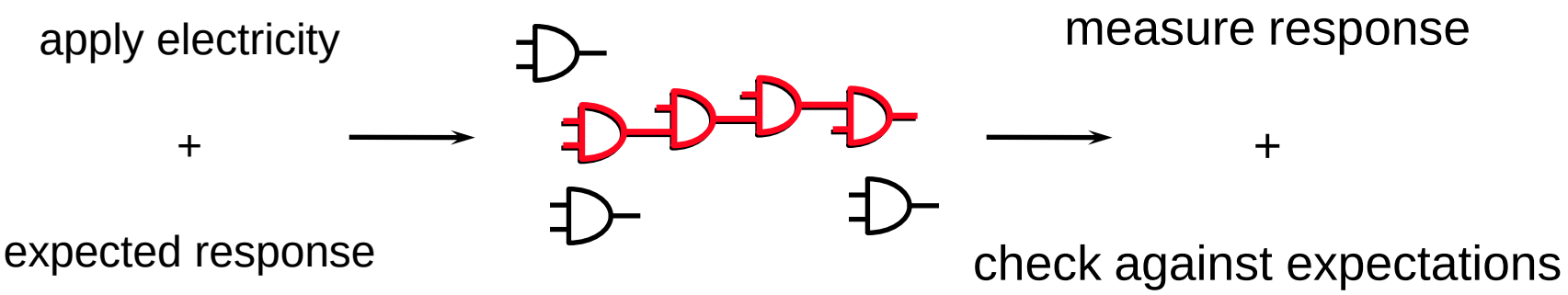


Old way

```
always @(posedge req)
  begin
     repeat (1) @(posedge clk);
     fork: pos_pos
        begin
           @(posedge ack)
           $display("Assertion Success",$time);
             disable pos_pos;
        end
       begin
          repeat (2) @(posedge clk);
          $display("Assertion Failure",$time);
          disable pos_pos;
       end
     join
  end // always
```

Ensure Gates Will Work

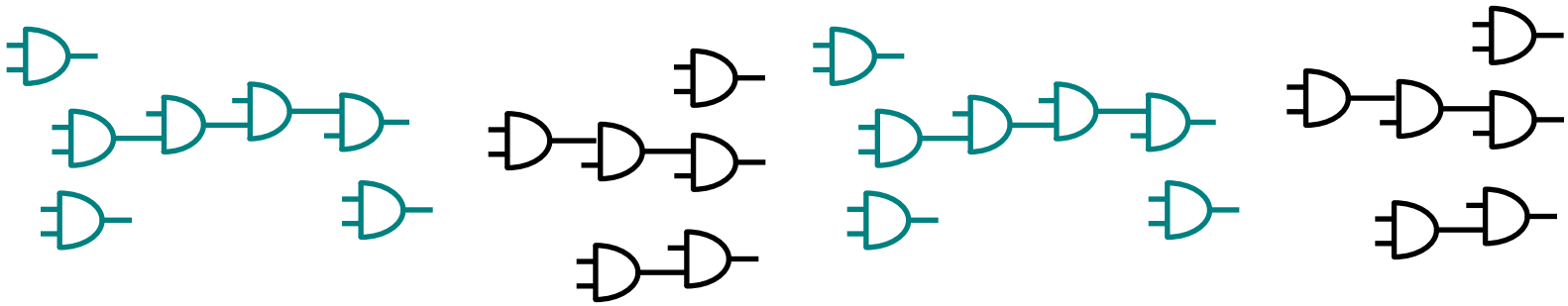
- Create a test program
- For manufacturer to throw out defective chips
- Need millions of combinations of electrical stimuli
- Use EDA tools! Automatic Test Program Generation
  - *example: TetraMAX*



Ensure Modifications Will Work

- Tweak the whole thing to make it better
  - Faster (speed), smaller (area), less battery (power)
  - Use EDA tools! Optimization

    - example: Design Compiler, Power Compiler, Physical Compiler



Make sure it's still the same design - Formal Verification

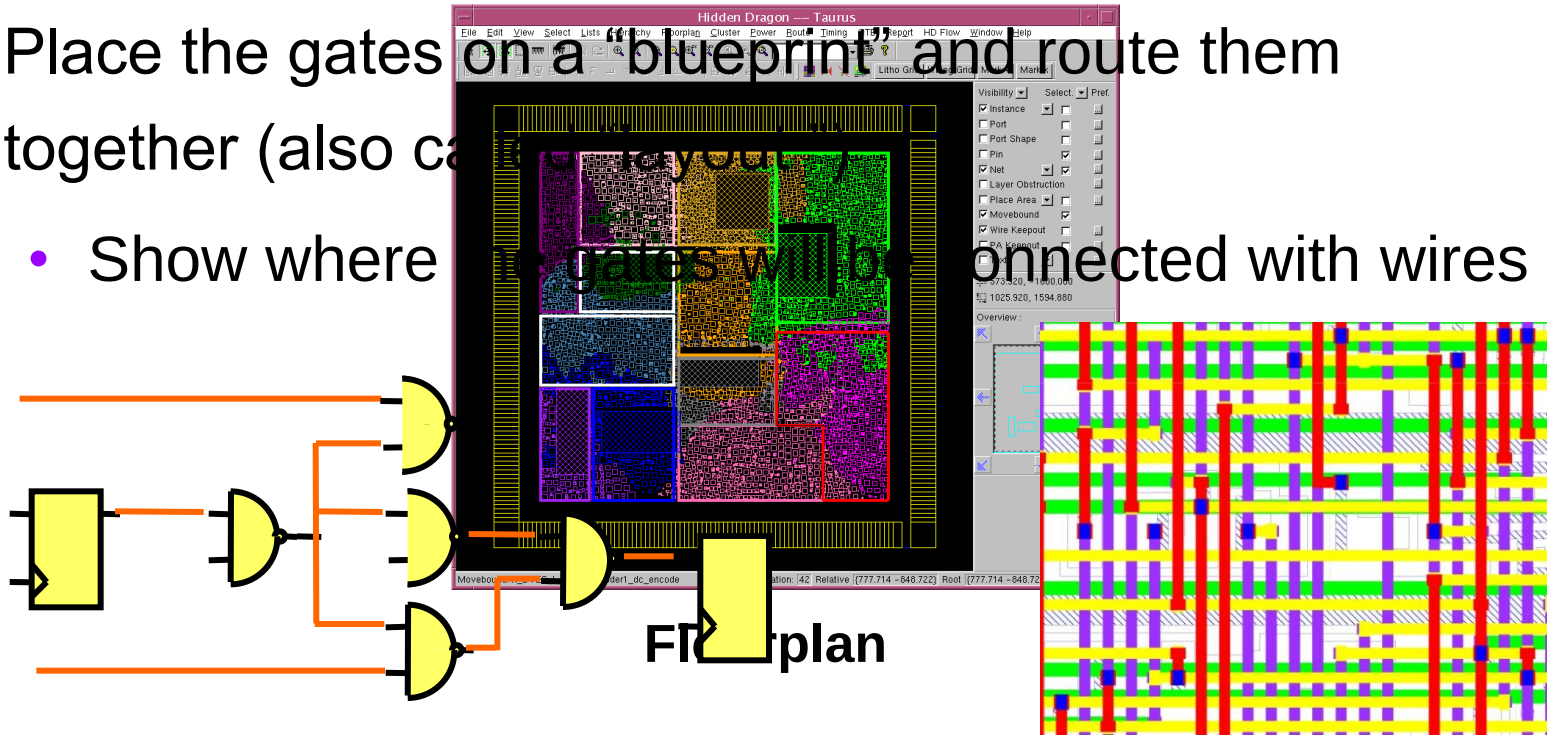
- example: Formality

Physical Implementation

- Gate level netlist is converted into geometeric representation
- It corresponds to the layout of the design
- Layout is designed according to design rules specified in the library
- Consists in three sub steps
  - Floor planning
  - Placement
  - Routing
- Produces a GDSII file

“Blueprint” Your Chip

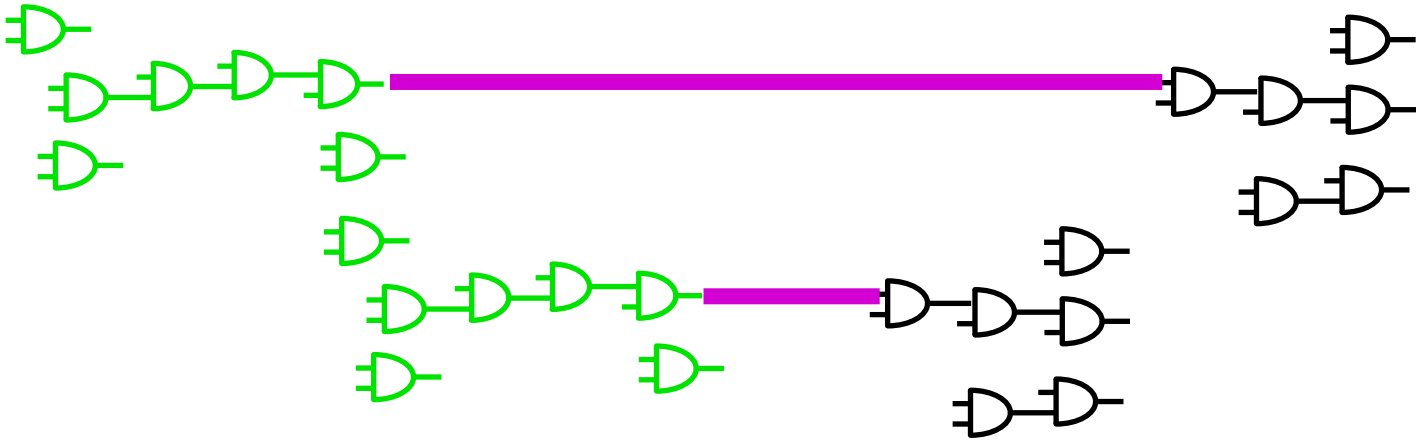
- Design planning: create a “floorplan” so the gates will go where you want
- Place the gates on a “blueprint” and route them together (also ca
  - Show where
  - onnected with wires



- example: Jupiter-XT, Physical Compiler, IC Compiler, Astro

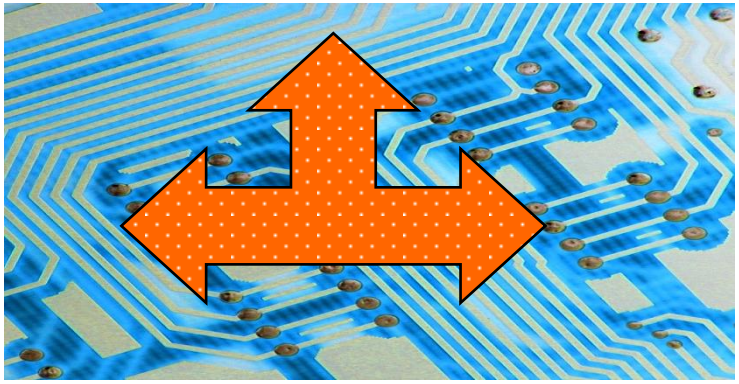
Double-Check Your Blueprint

- Length of the connecting wires will change how fast the chip will run
- Simulate it again - Post-route (post-layout) verification
  - *example: VCS-MX, NanoSim, HSPICE*

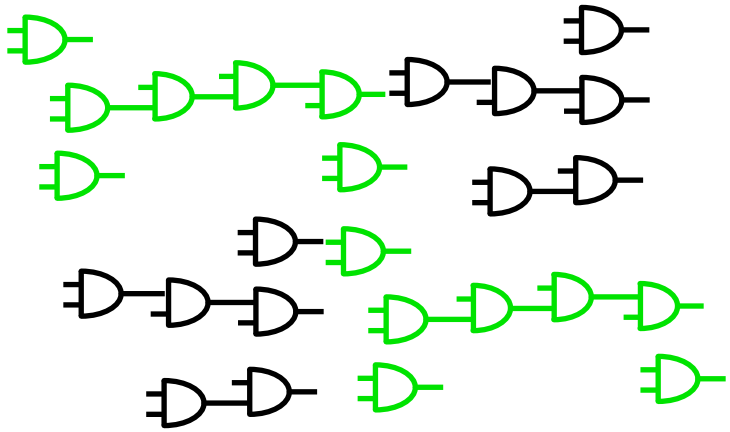


Double-Check Your Blueprint

- Make sure “what you see is what you get”
  - Compare what you designed to what’s in your layout
  - Layout versus Schematic (LVS)
- Follow the manufacturer’s rules
  - Perform Design Rule Checks (DRC)



- example: Star-RCXT, Hercules



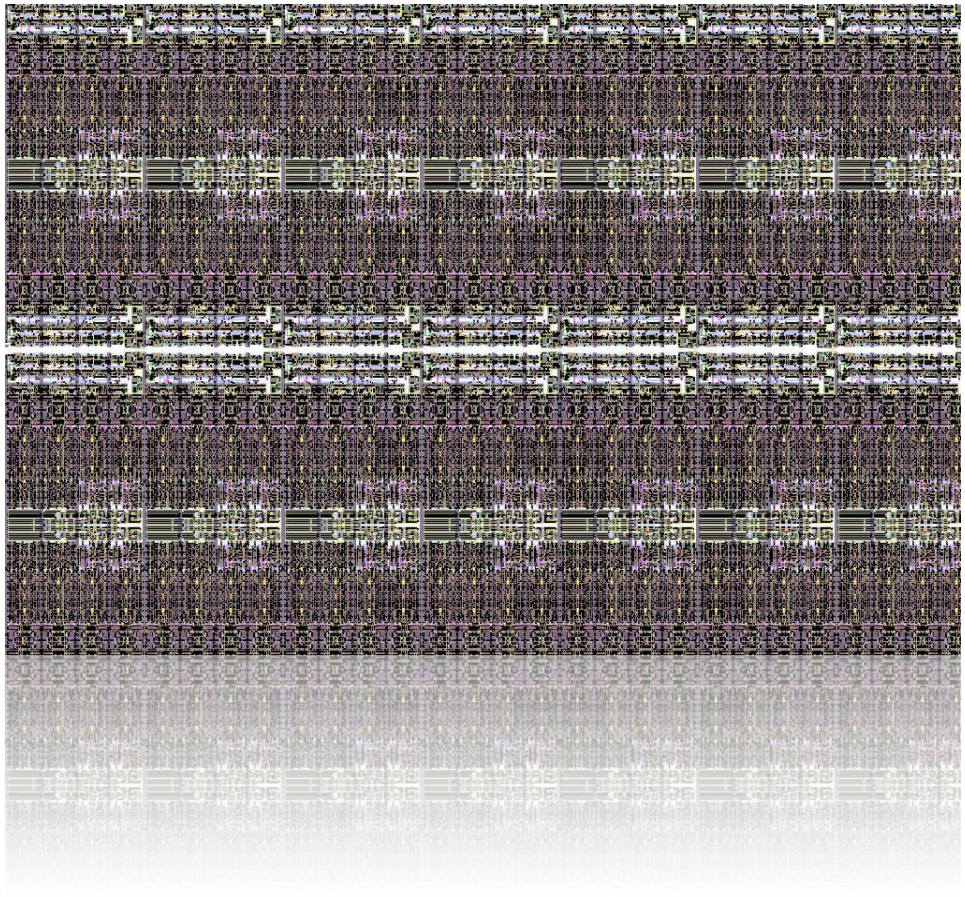
GDSII

- File used by the foundry to fabricate the ASIC
- Physical verification is performed to verify if the layout is designed according the rules

Turn Your Design Into Sand

- Produce the layout data: GDSII

  - example: IC Compiler



GDSII – Text View

STRNAME EXAMPLE	02000200 60000201 1C000300 02000600 .....`.... 000000
BOUNDARY	01000E00 02000200 60002500 01000E00 .....%.`..... 000010
LAYER	42494C45 4C504D41 58450602 12002500 .%....EXAMPLELIB 000020
1	413E0503 14000300 02220600 59524152 RARY..".....>A 000030
DATATYPE	1C00545A 9BA02FB8 4439EEA7 C64B3789 ZK../.9D'.ZT..7K 00004

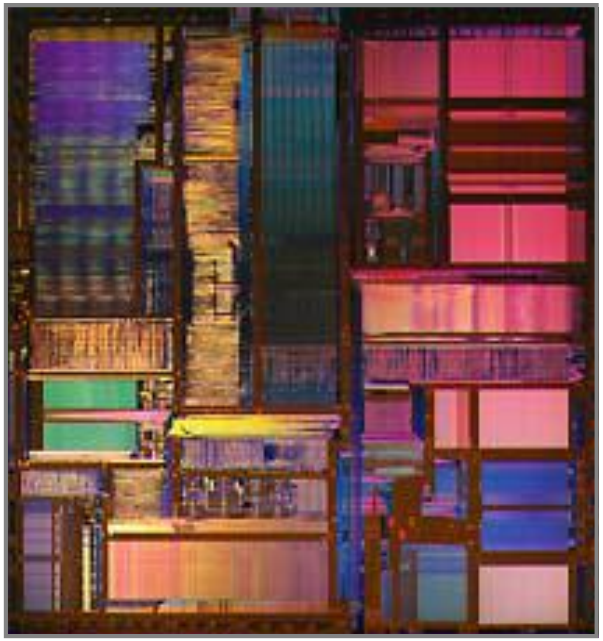
Files are usually 20 to 30 gigabytes in size, but after you correct the image, the size can reach a 150 gigabytes - that's 150 billion bytes or over a trillion bits!

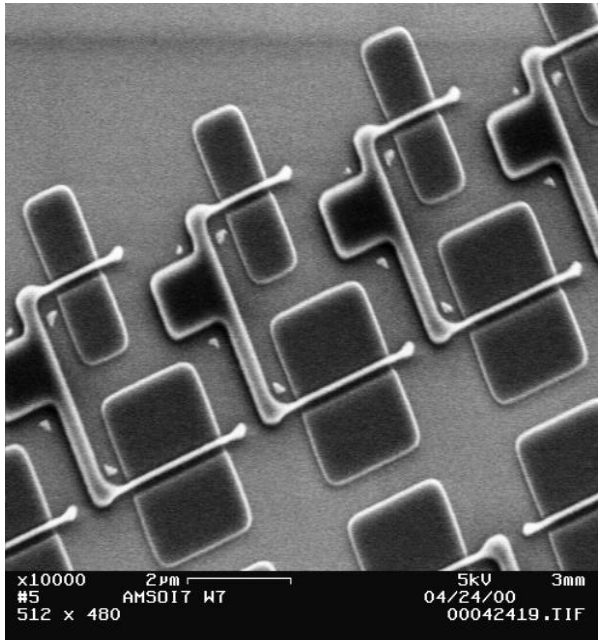
-10000	
-10000	
-10000	
10000	
ENDEL	
ENDSTR	

Turn Your Design Into Sand

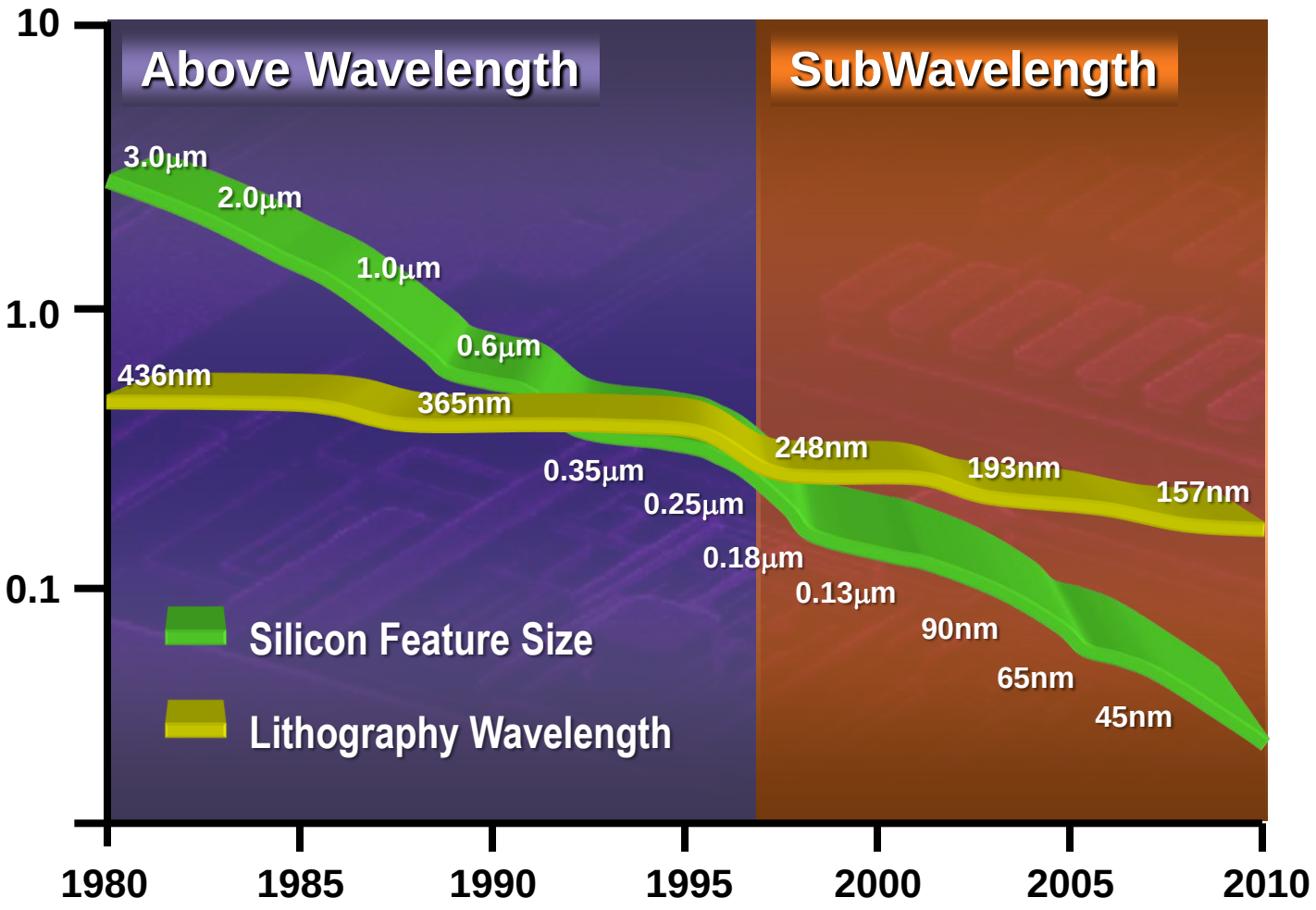
Correct the image

- *example: Proteus, Progen, IC Workbench, SiVL/LRC, iN-Phase, CATS*

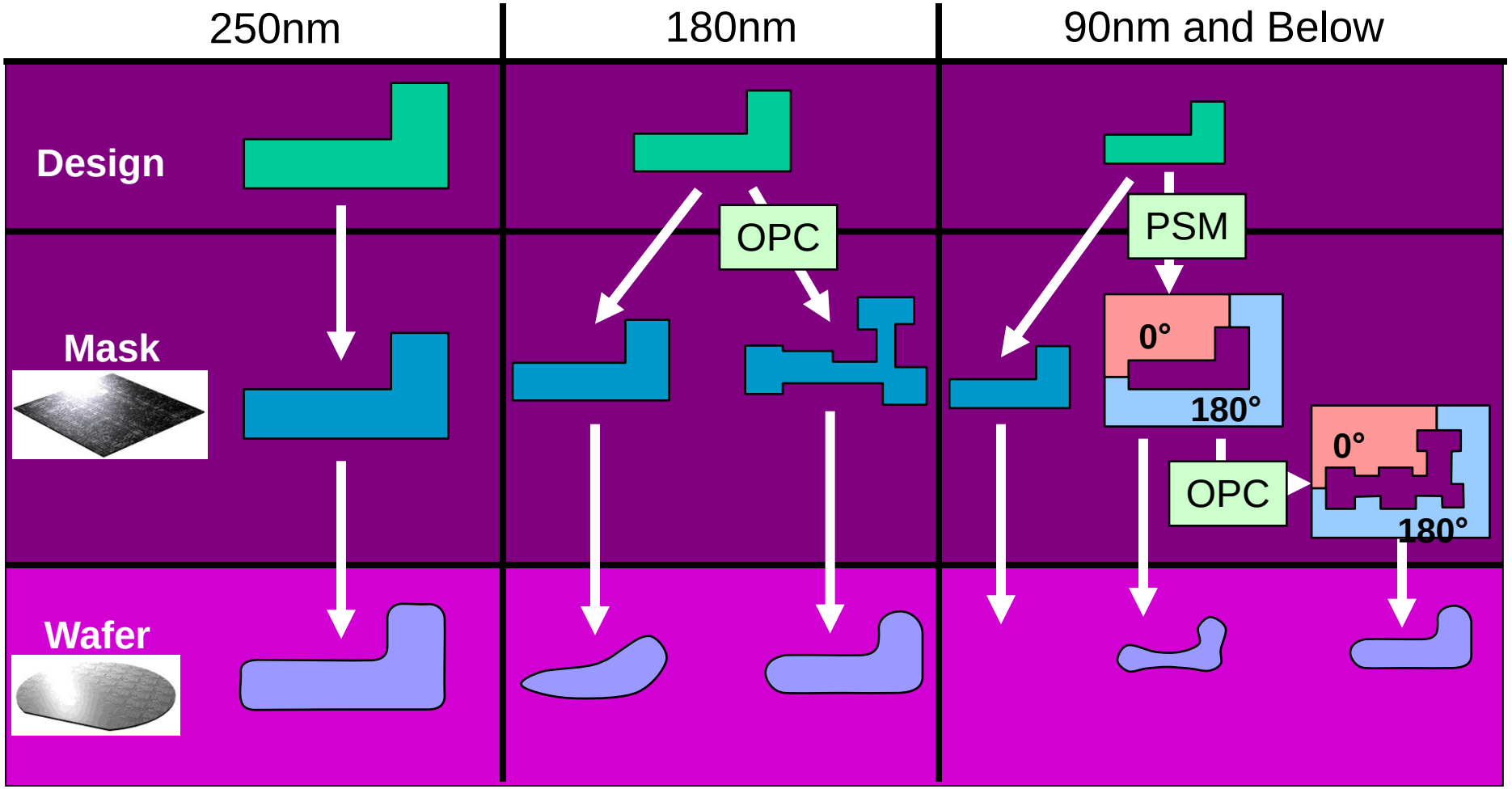




Subwavelength Lithography Challenge

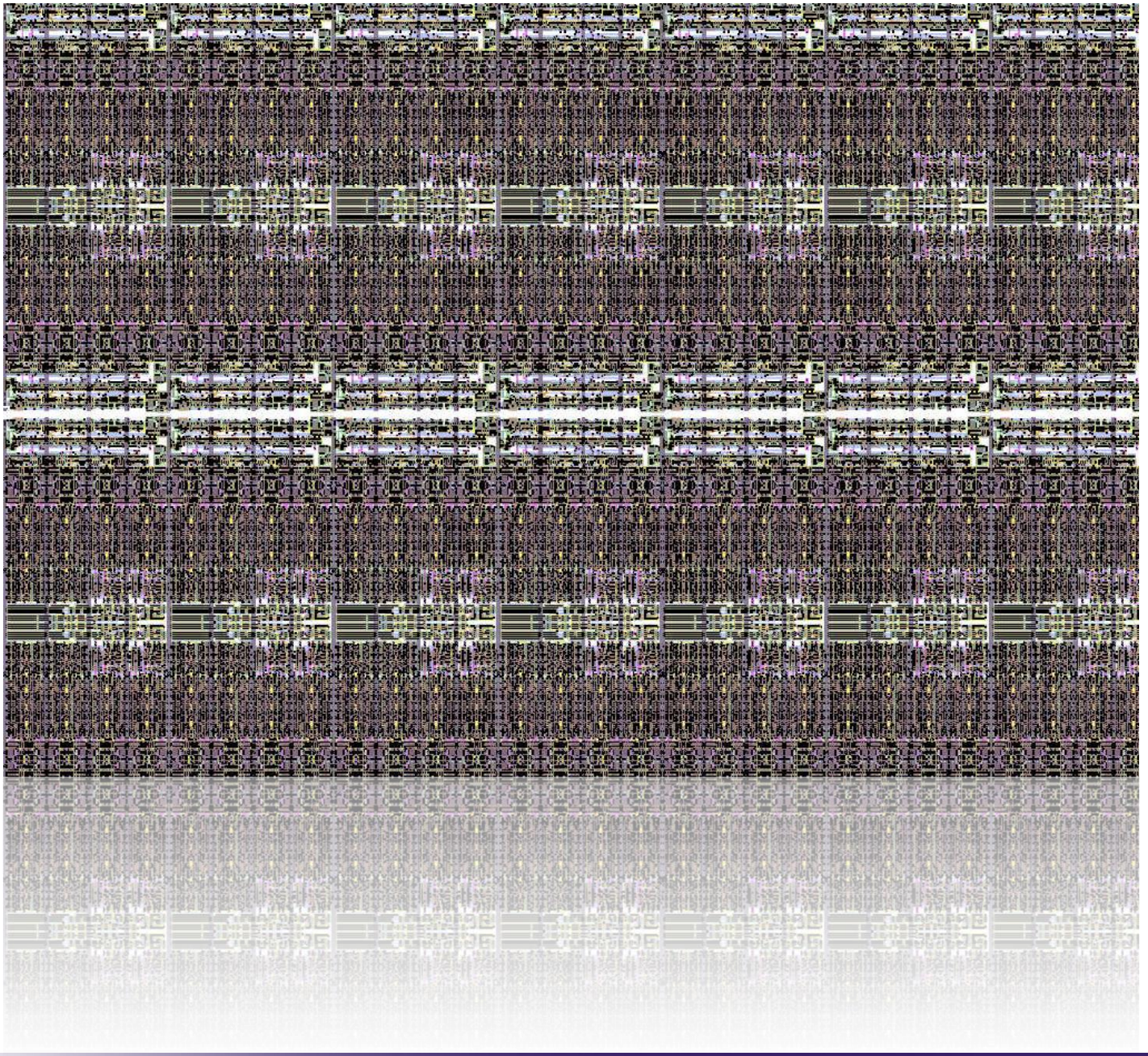


Correcting the Image



OPC = Optical Proximity Correction

PSM = Phase Shift Mask



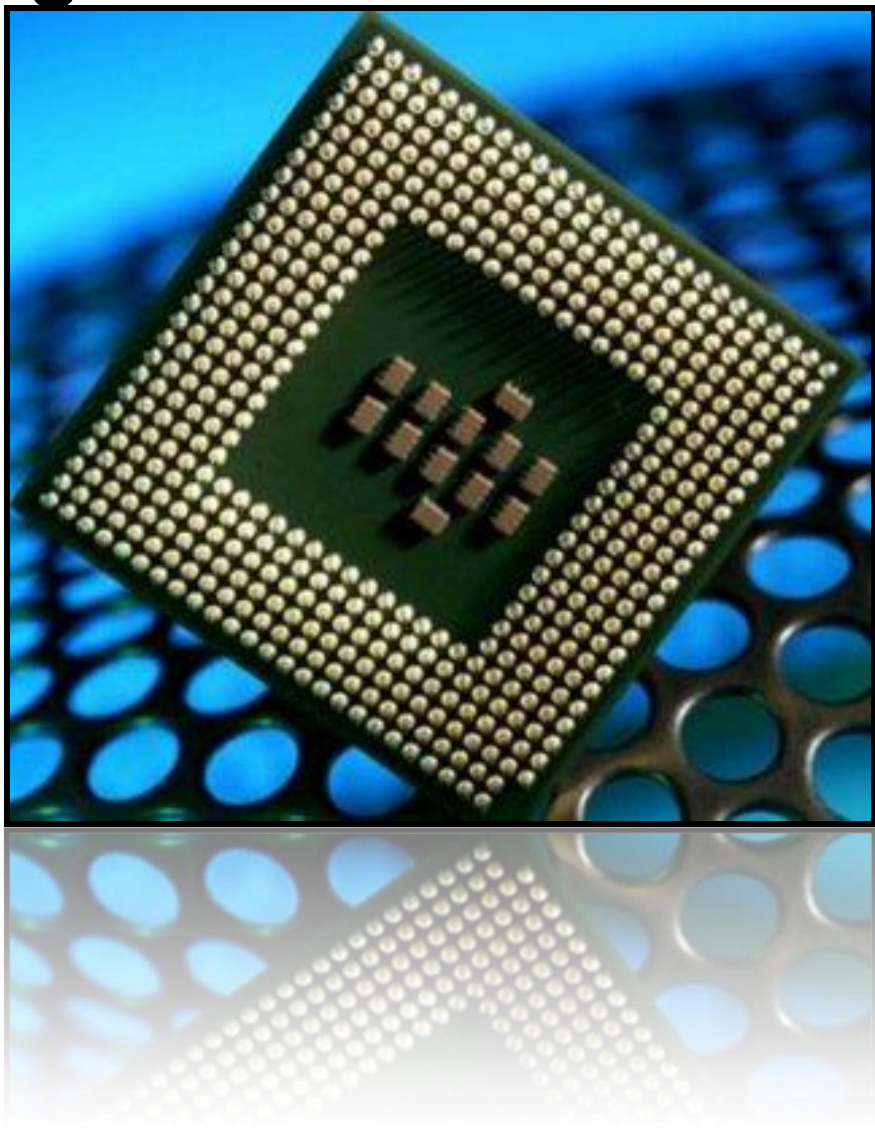
Step 7. Turn Your Design Into Sand

Take a deep breath

and “sign off”!

Your semiconductor vendor will:

- Rerun your design
- Generate a database for manufacturing
- Perform photolithography
- Put your chip into a package
- Run your test program
- Deliver your finished chips



Make an Electronic Product

Put the chips together

on a board and into your

phone, computer…

Better yet,

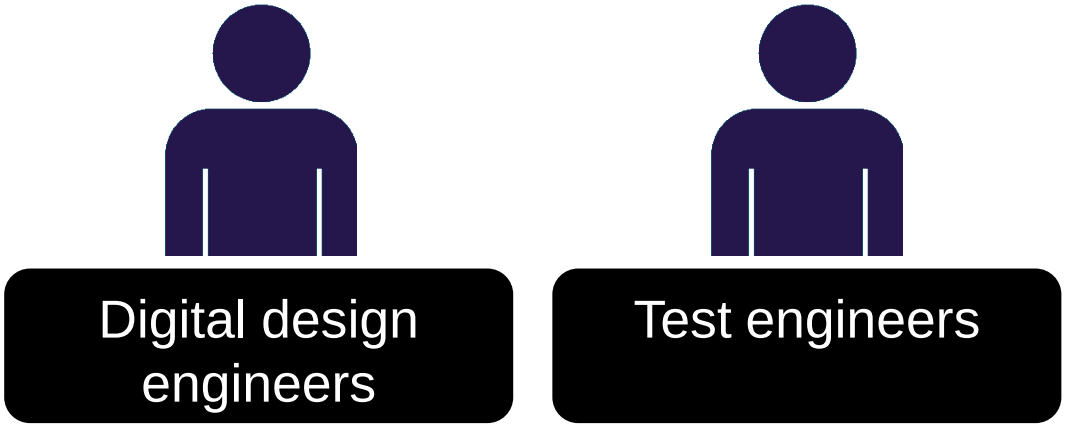
have someone else

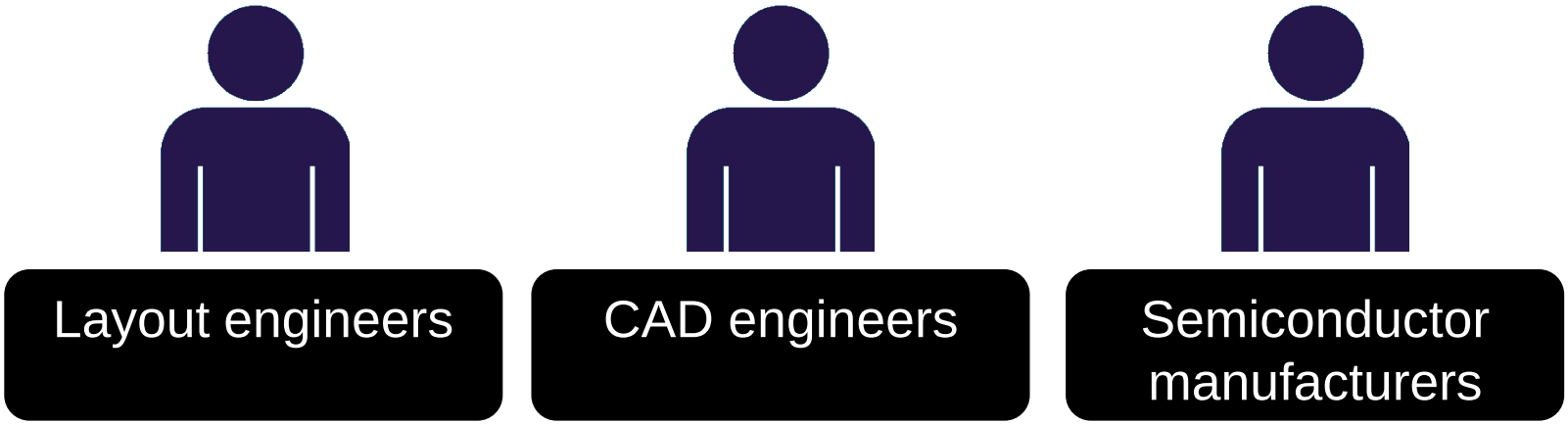
do it for you

Now you're

ready to

go to market!





who are the **players**?

Predictable Success