A bird’s eye view on chip projects

3+ good reasons

1 main question

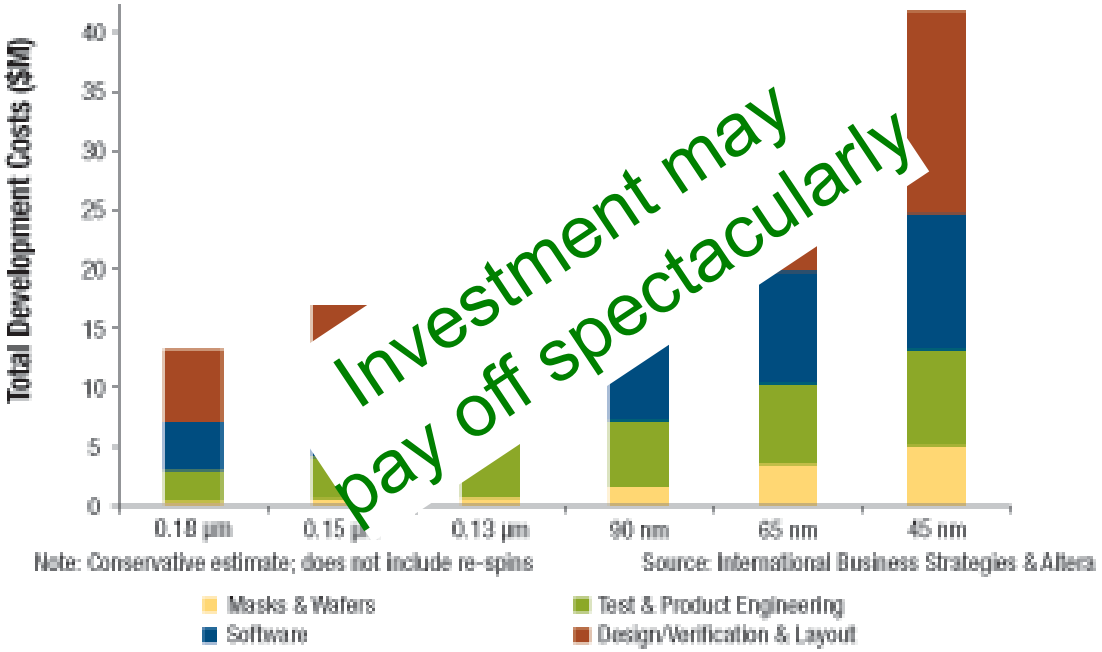
The ASIC Flow

ASIC – Application Specific Integrated Circuit
A chip that is produced solely tailored for you

- Smaller
- Cheaper
- Harder to copy
- Less power/energy/cooling
- More integration ...

The wafer is cut (“diced”) into many pieces, each containing one copy of the circuit.
Each of these pieces is called a die

The main question: €€€?



The answer is clear
if – and when – you offer a unique benefit

Chip projects - from idea to unique product
Learning objectives: **You** can

1. Explain the trade-offs involved in make (bake) or buy
2. Sketch the flow from idea to unique ASIC, and indicate the essential role of EDA and IP, and crucial verification and test effort
3. Apply the typical cost structure of chip project on a case with given data

Liesbet.Vanderperre@kuleuven.be

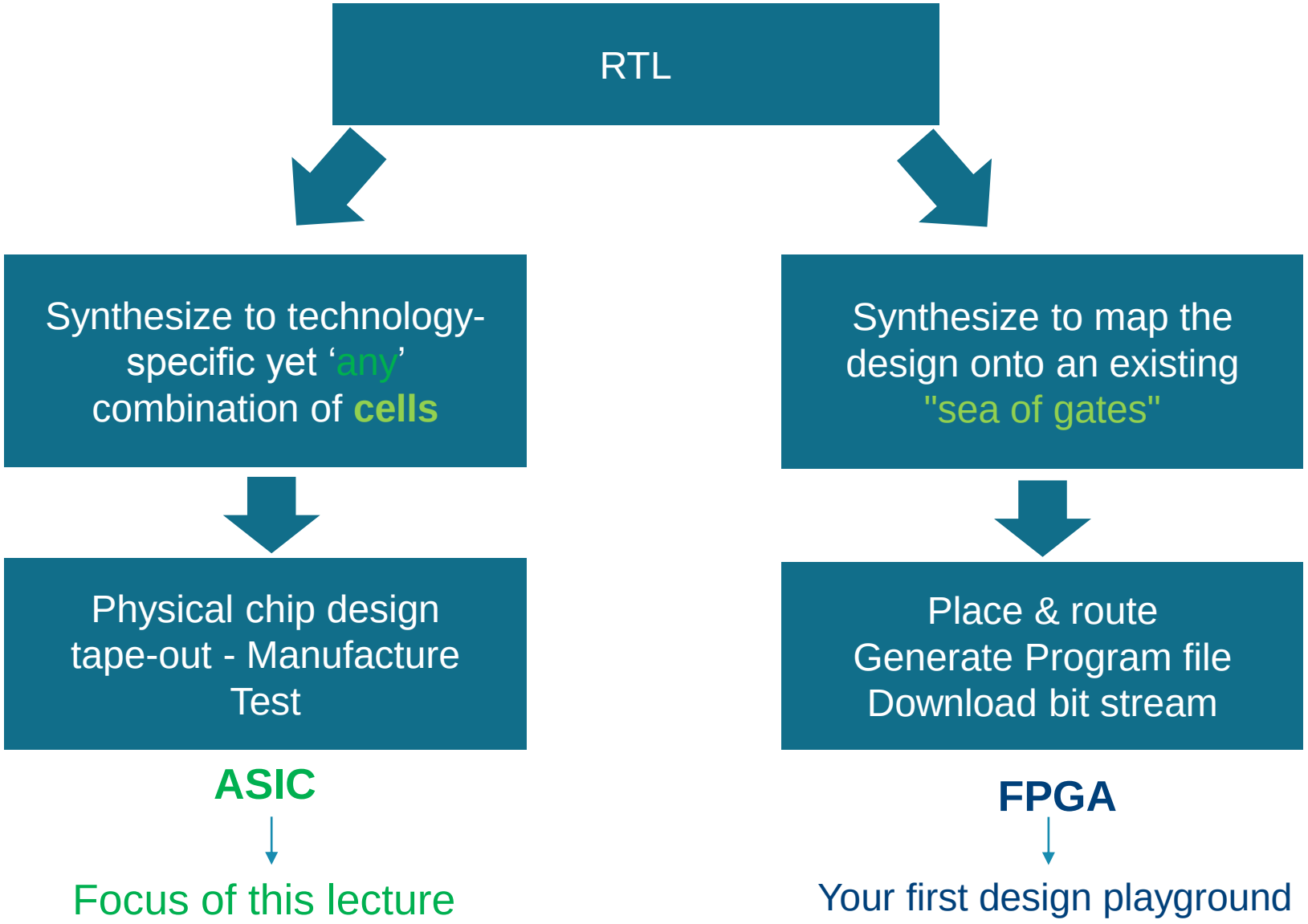
Great inputs by Steven Redant & Bas Dorren
friends & former colleagues @ imec

KU LEUVEN imec LUND UNIVERSITY

Physical platforms (targets) for digital design: Application Specific or Generic

- ASIC: Application Specific IC: Make/bake your own recipe
- PLD: Programmable Logic Device: configure your own recipe
  - Field-programmable Gate Array (FPGA): re-configurable

Different target platforms: diverging flows going from RTL



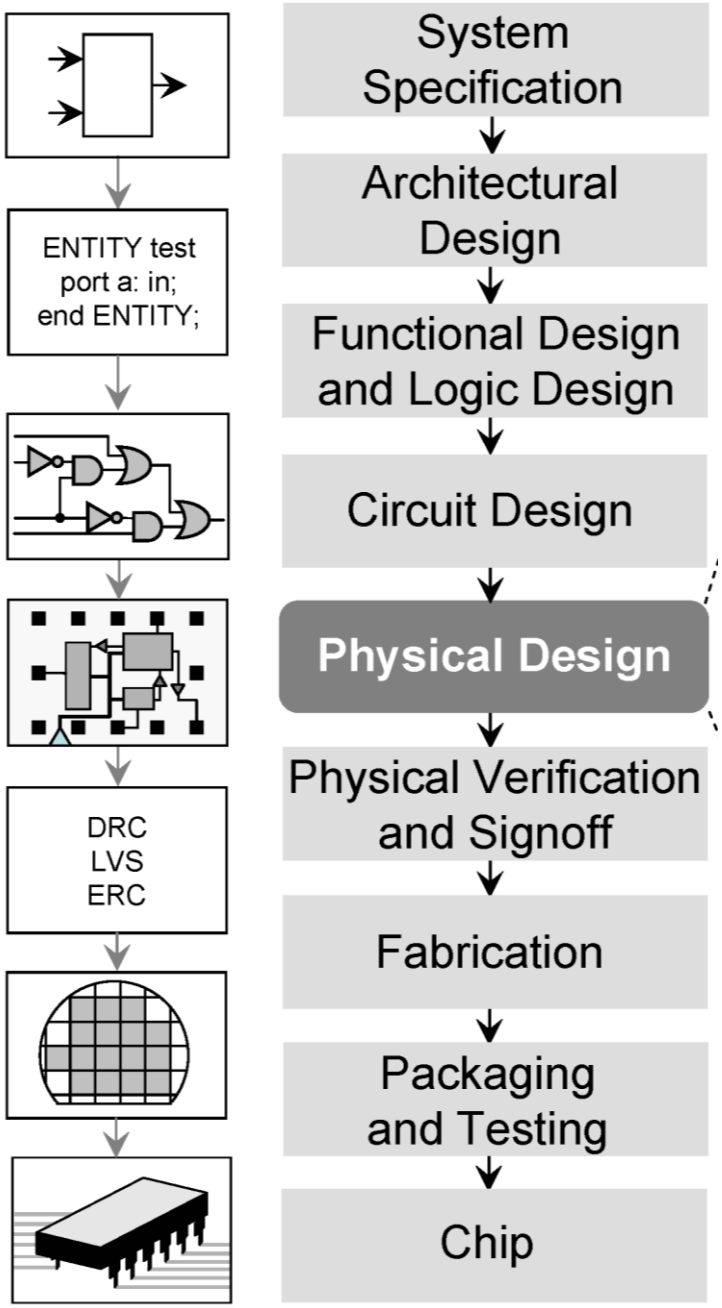
Chip projects: from idea to unique product

1. Chip projects: what is to be done?

2. Brief general semiconductor economics

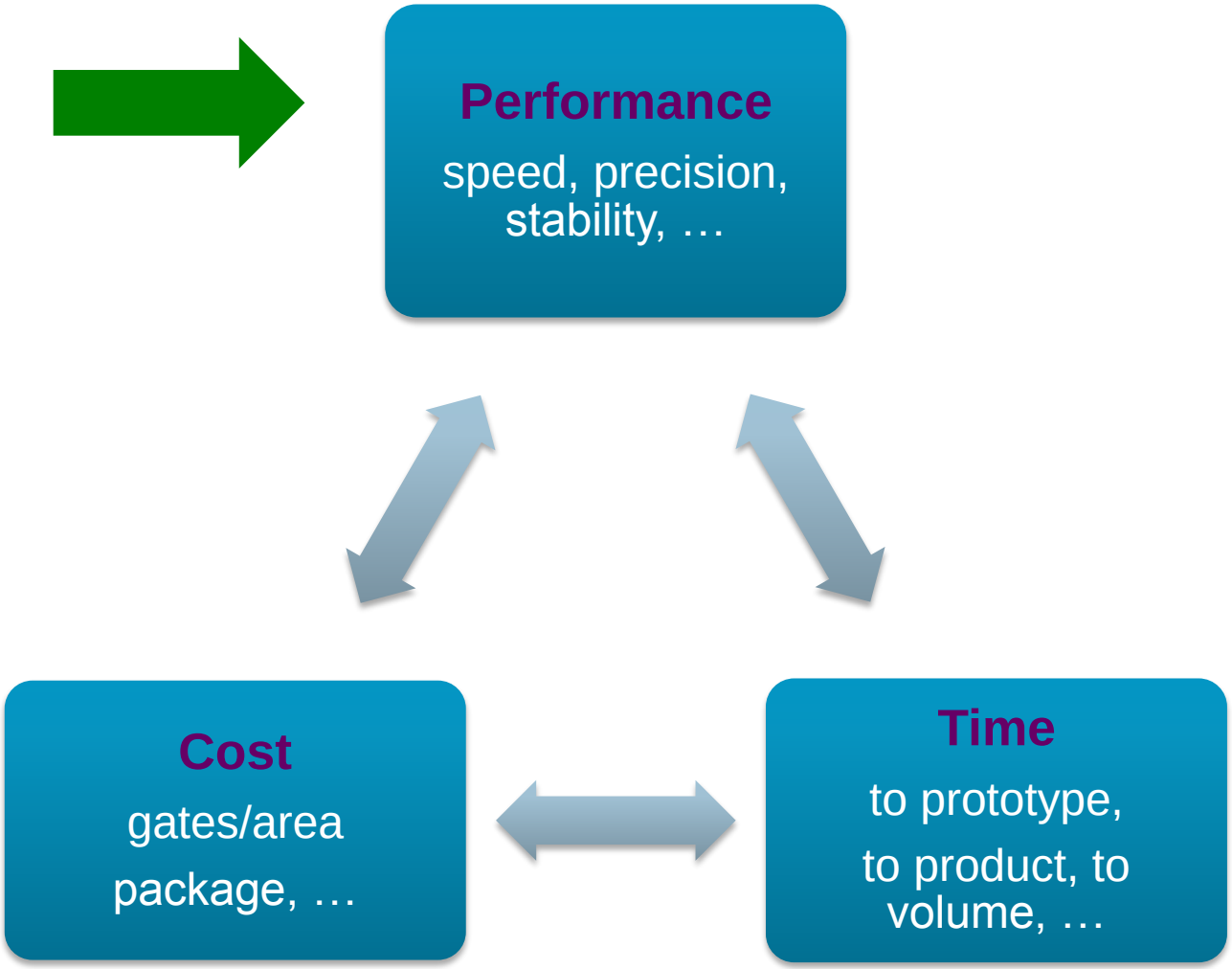
3. Exemplary project cases: bake your chip?

from idea to unique product: specification to chip

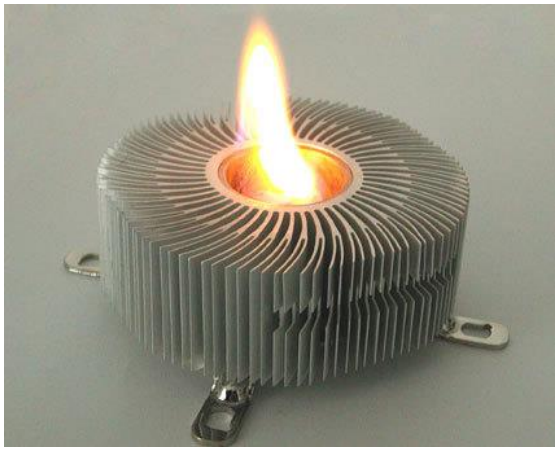


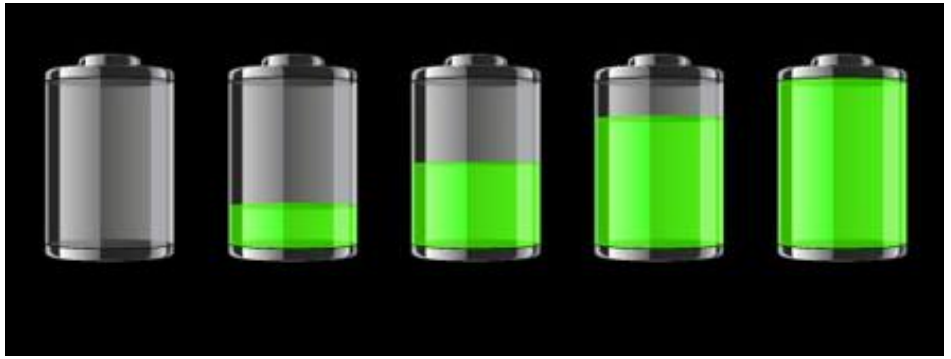

WALLY, WE DON'T HAVE TIME TO GATHER THE PRODUCT REQUIRE-MENTS AHEAD OF TIME.
I WANT YOU TO START DESIGNING THE PRODUCT ANYWAY. OTHERWISE IT WILL LOOK LIKE WE AREN'T ACCOMPLISHING ANY-THING.
OF ALL MY PROJECTS, I LIKE THE DOOMED ONES BEST.
NEWS
S. Adams
www.unitedmedia.com
5/9/97
© 1997 United Feature Syndicate, Inc.

ASIC design: the challenge

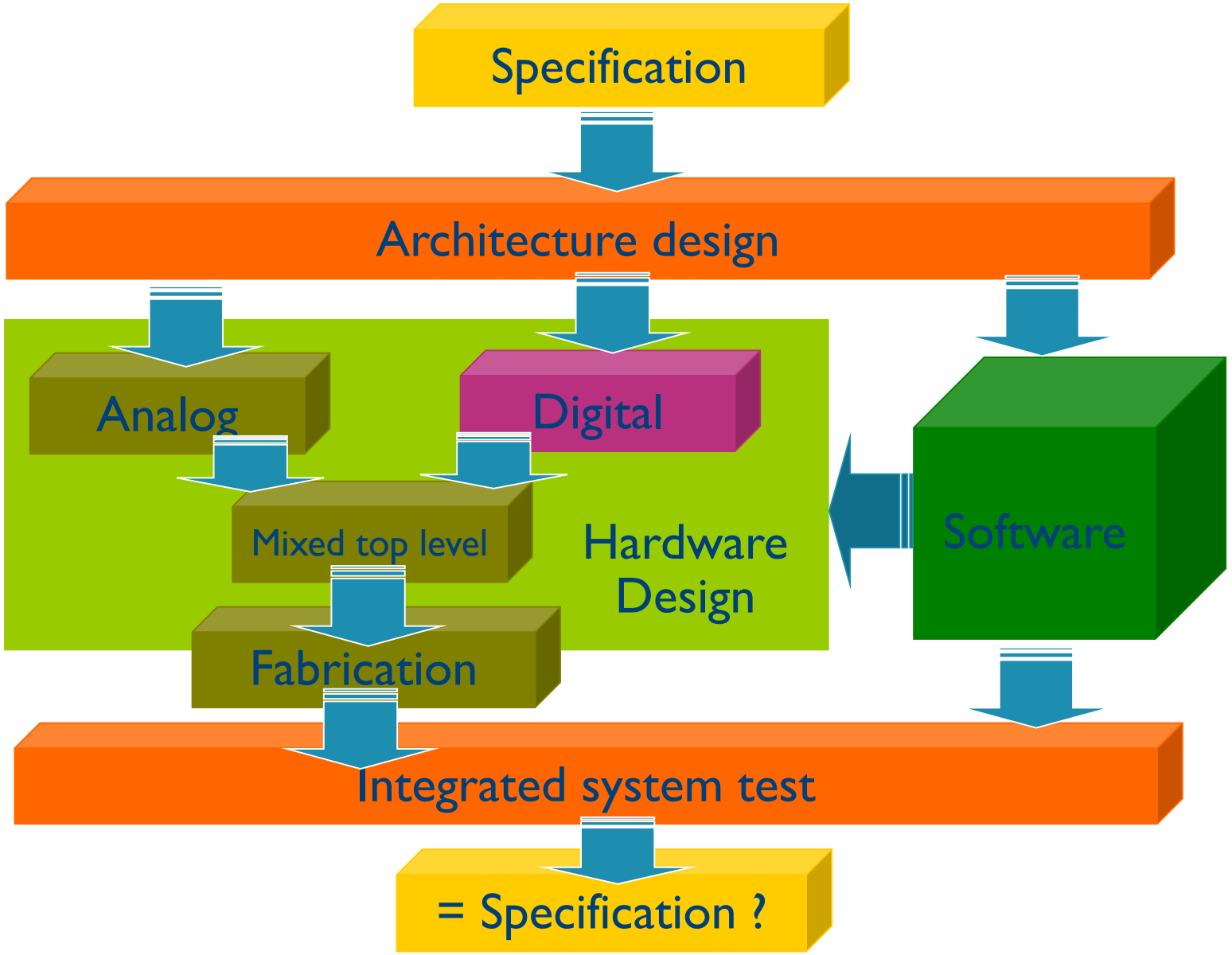


ASIC design: the art?

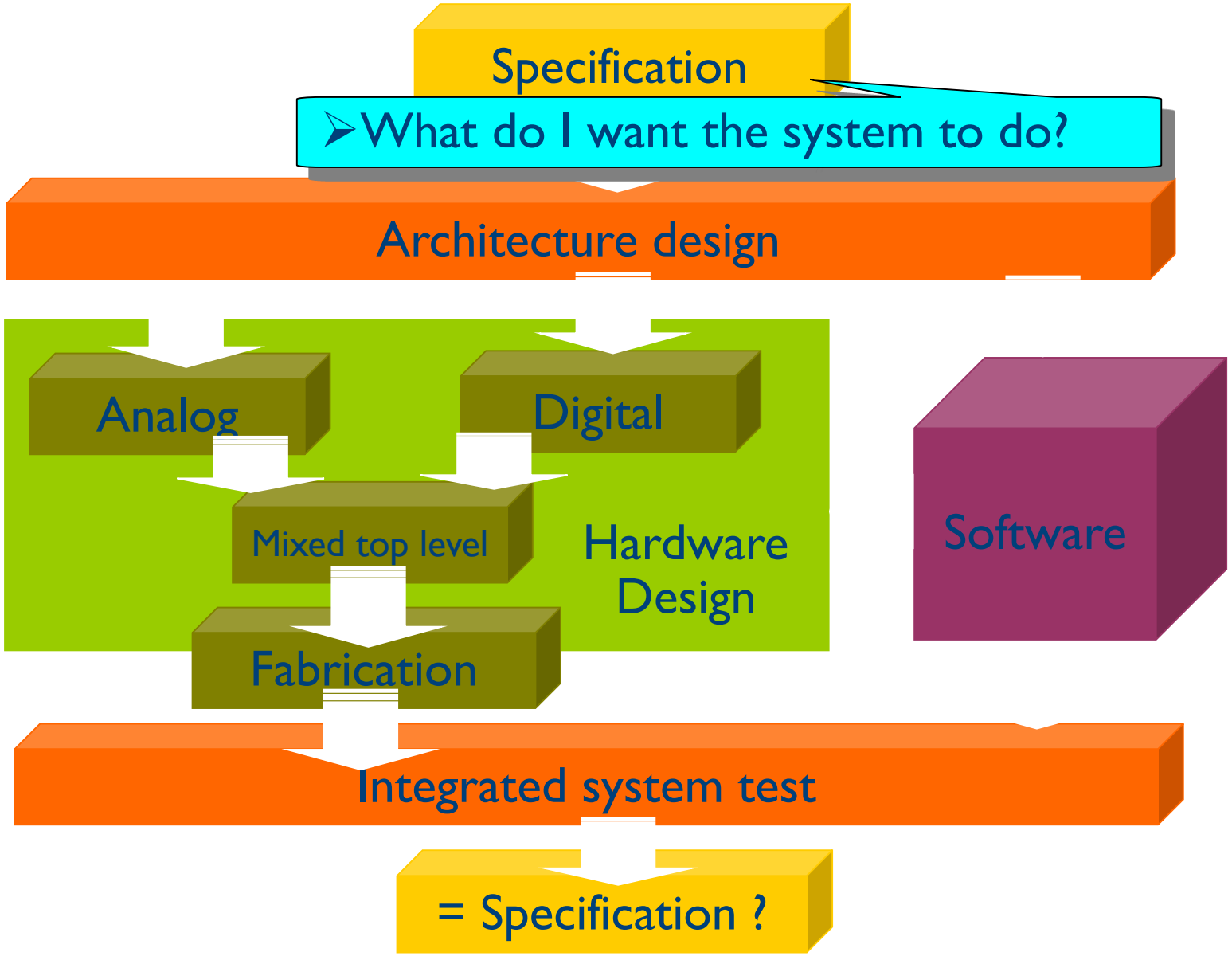




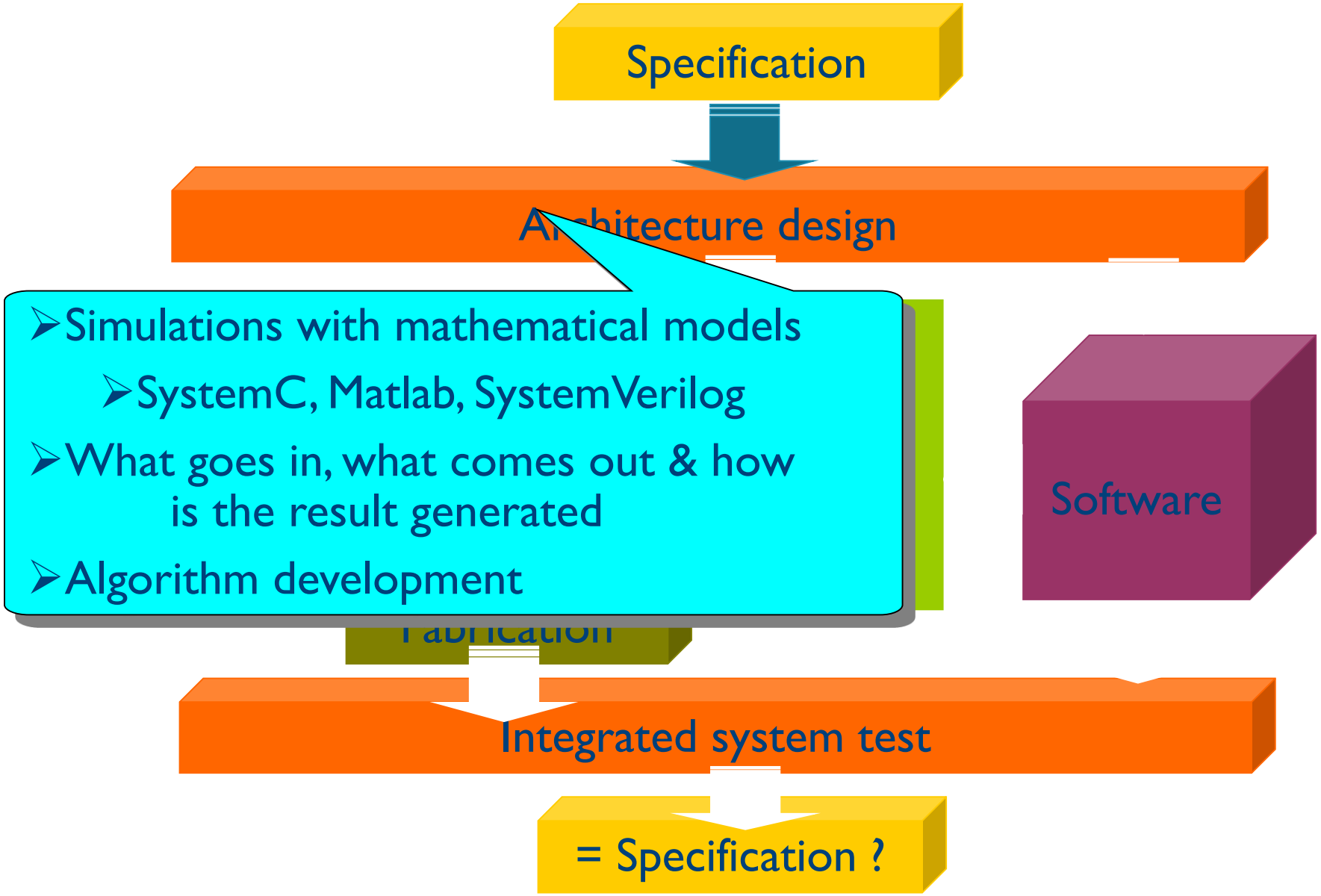
The birth of a System on chip



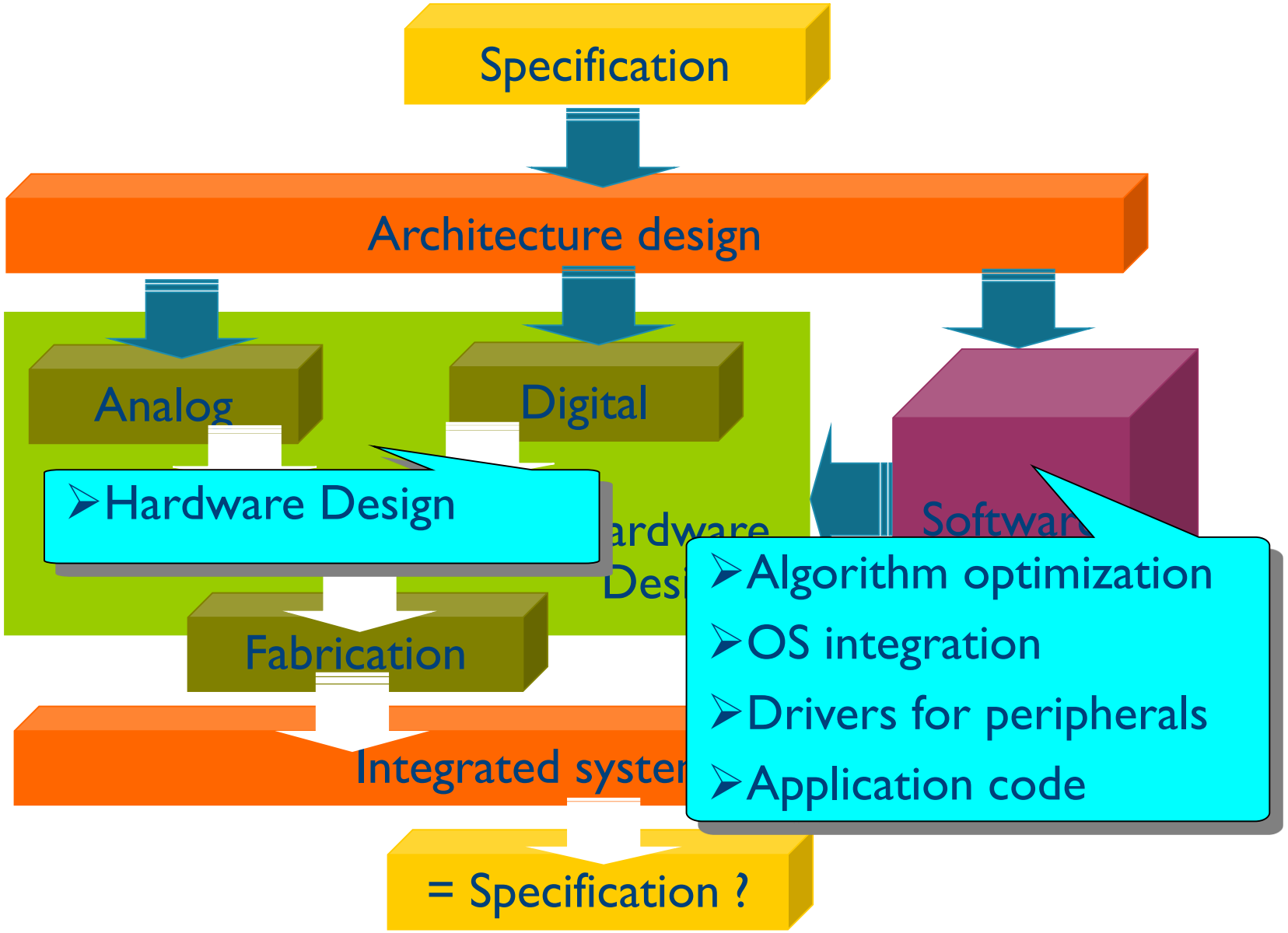
The birth of a System on chip



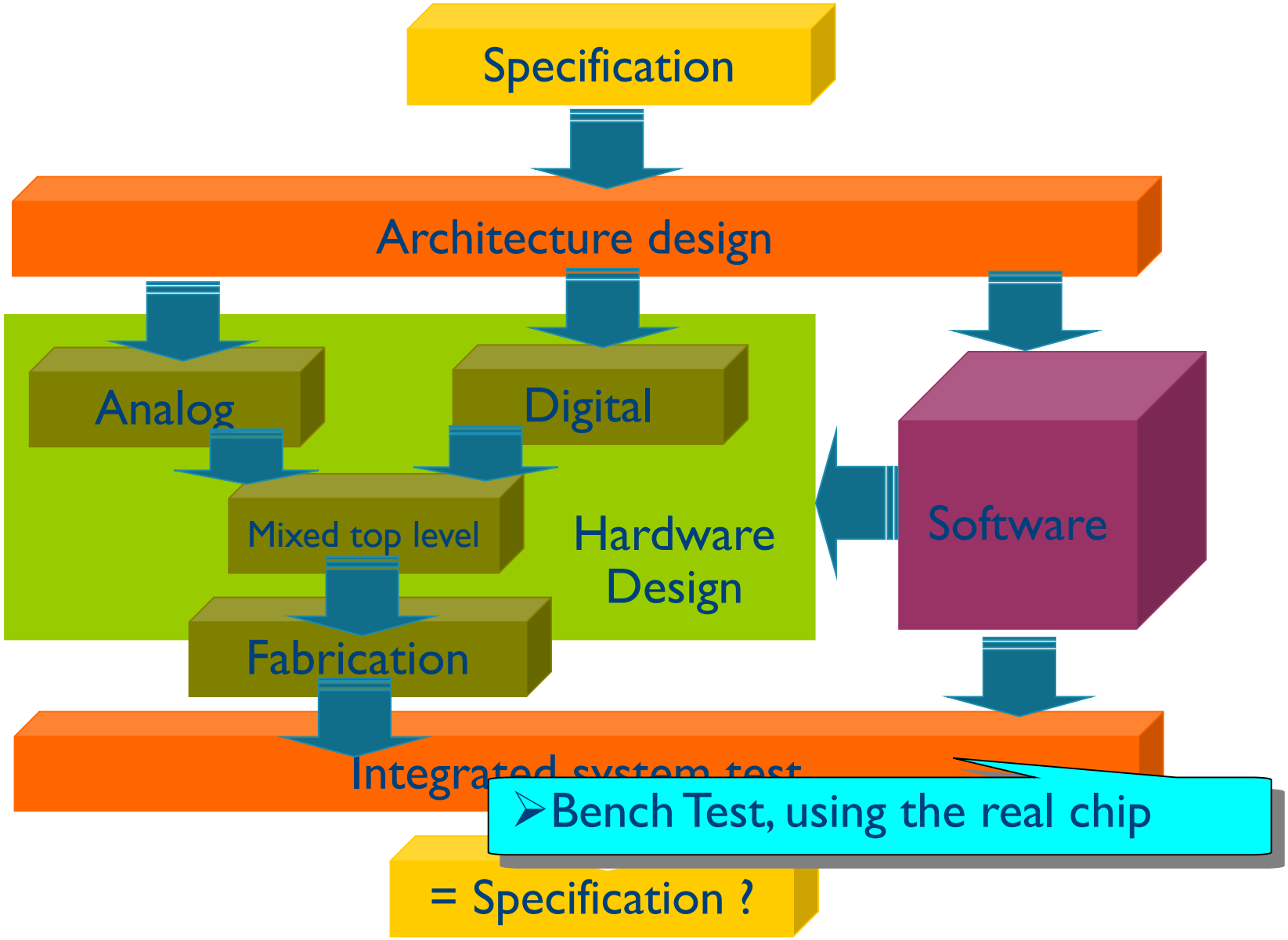
The birth of a System on chip



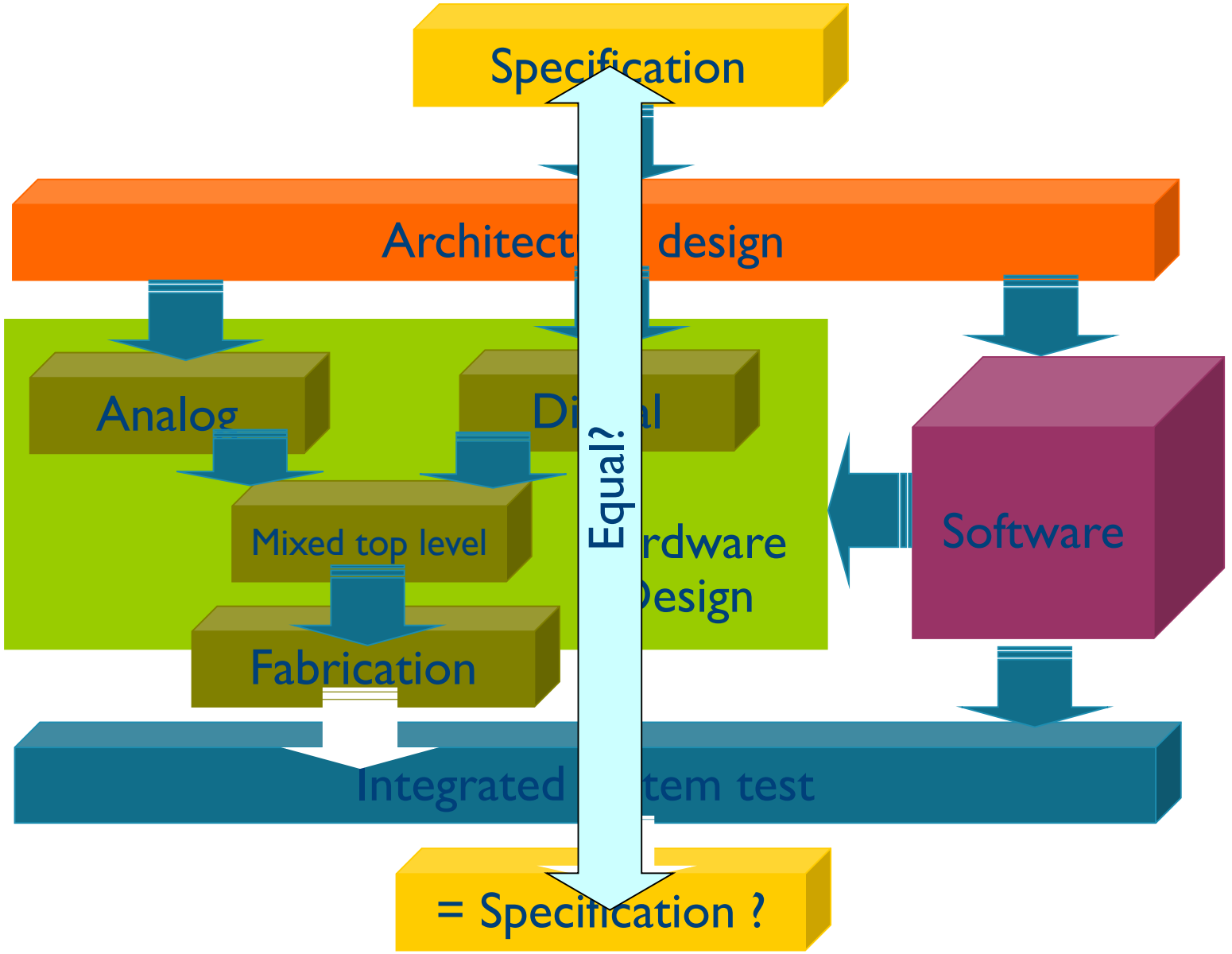
The birth of a System on chip

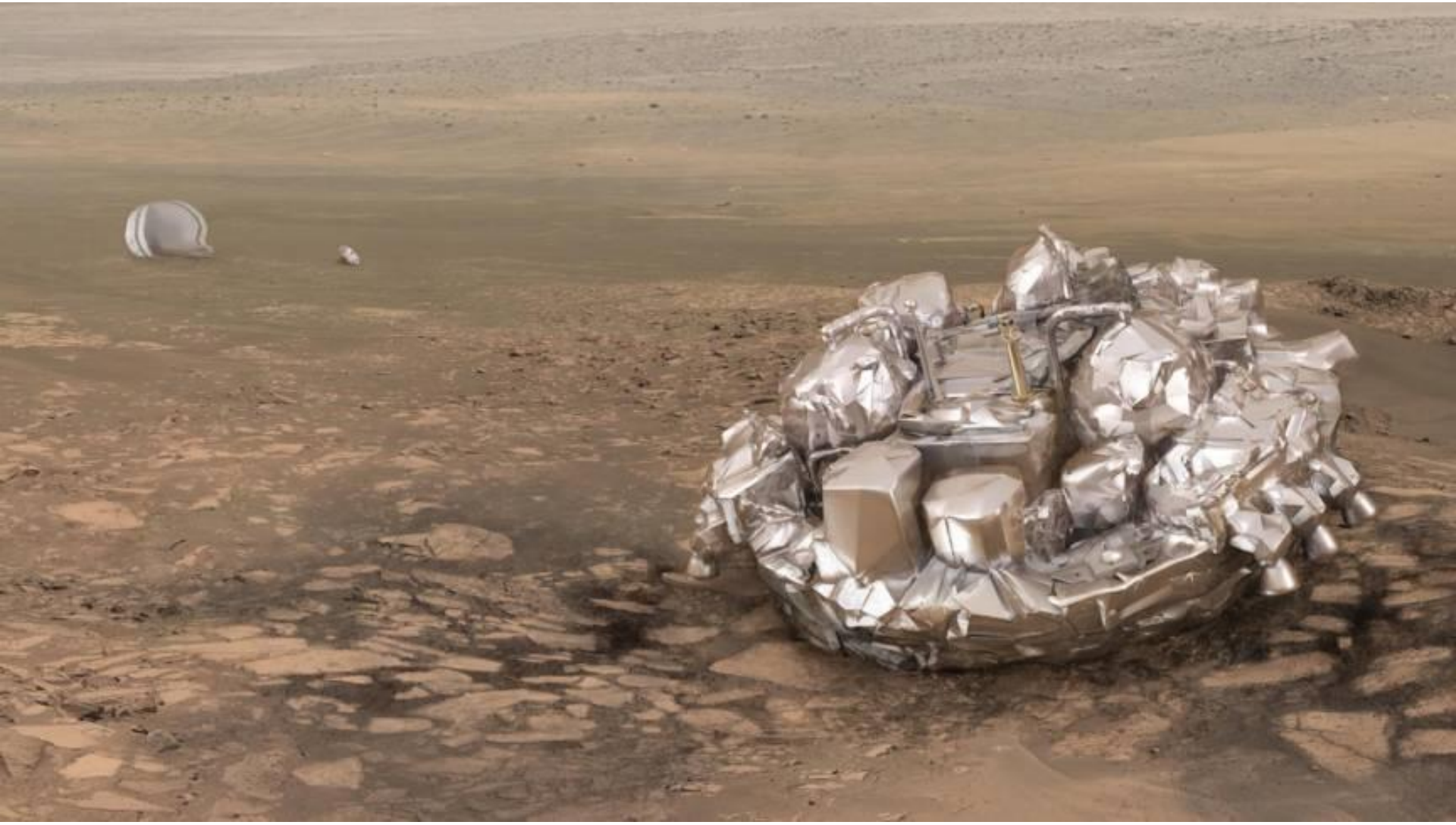


The birth of a System on chip



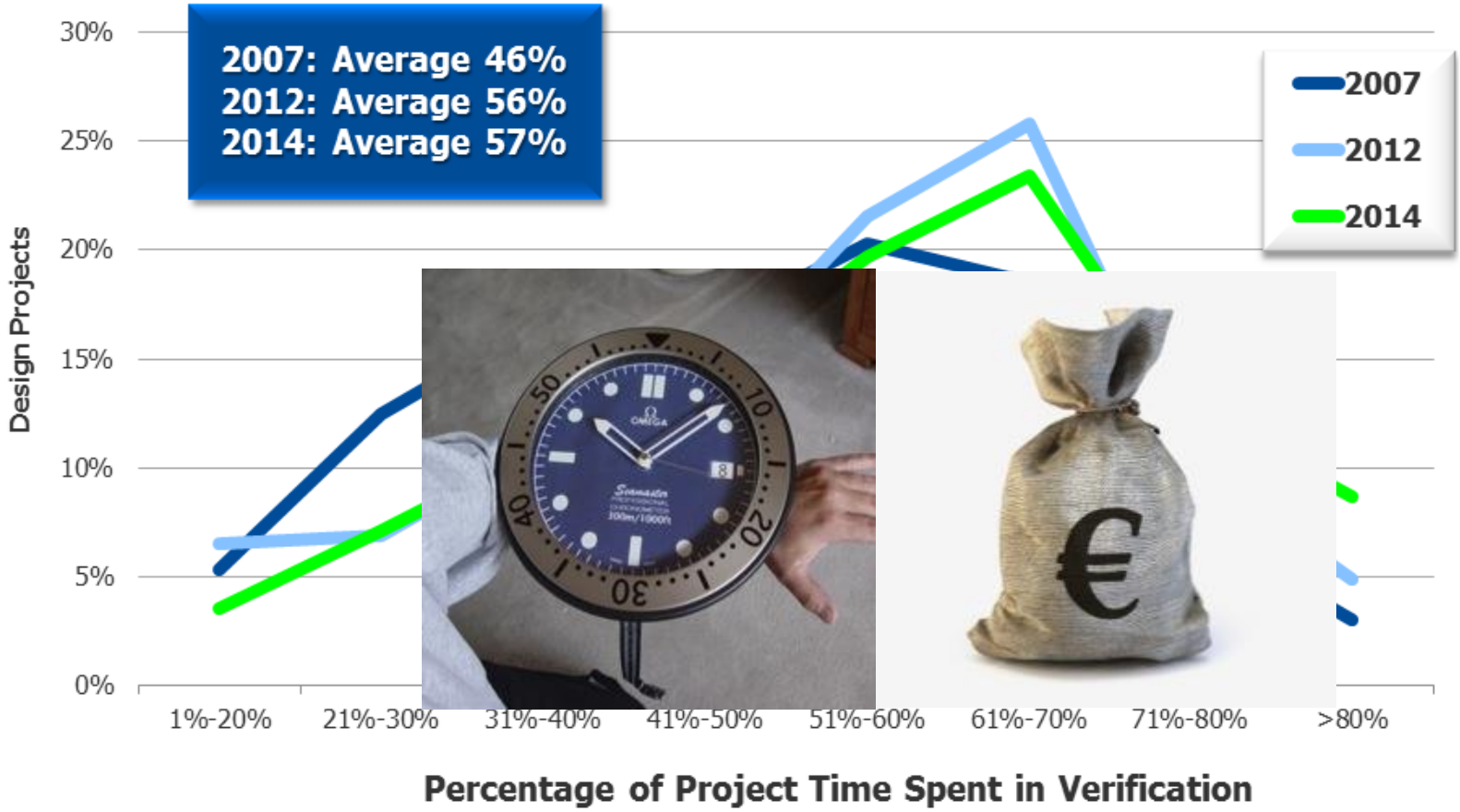
The birth of a System on chip





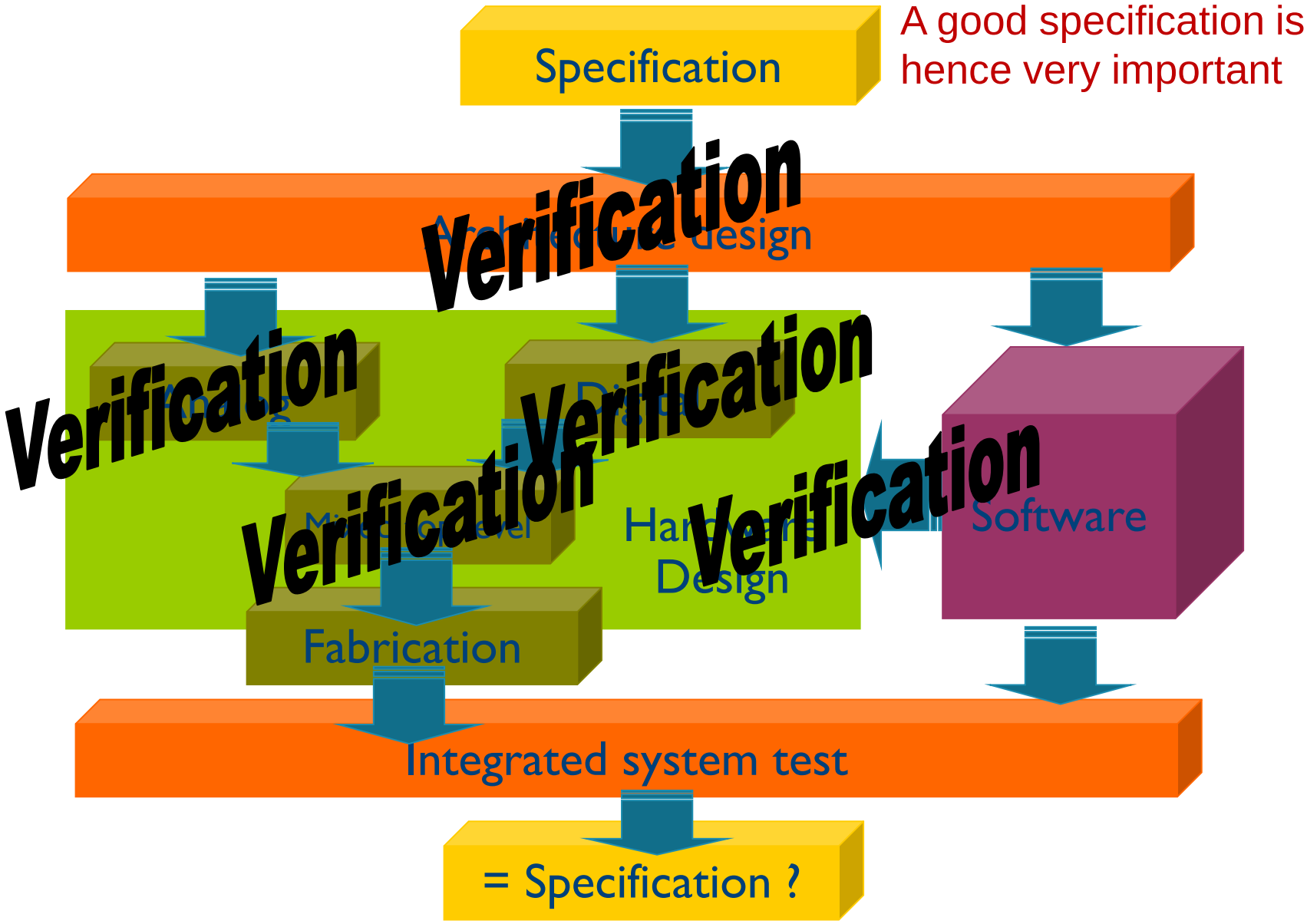
Solid verification requires a lot of effort!

Verification Consumes Majority of Project Time

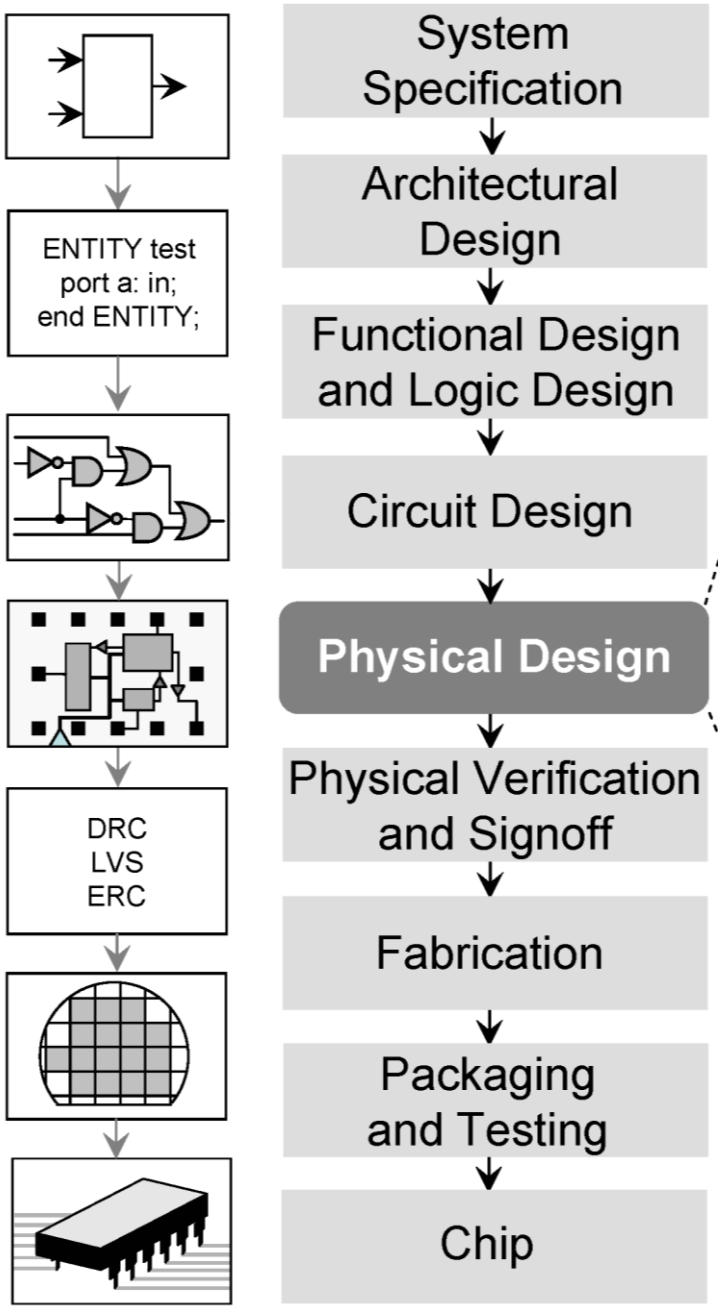


Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

Verification by Simulation



Design, Fabrication, Packaging, Test



Digital ASIC design: standard cell or (semi) custom options possible

Standard cells:

- Libraries provided capable of implementing (quasi) all digital functionality
- Logic synthesis can be tool-based
- Standard ease regular/automated physical design

Custom design:

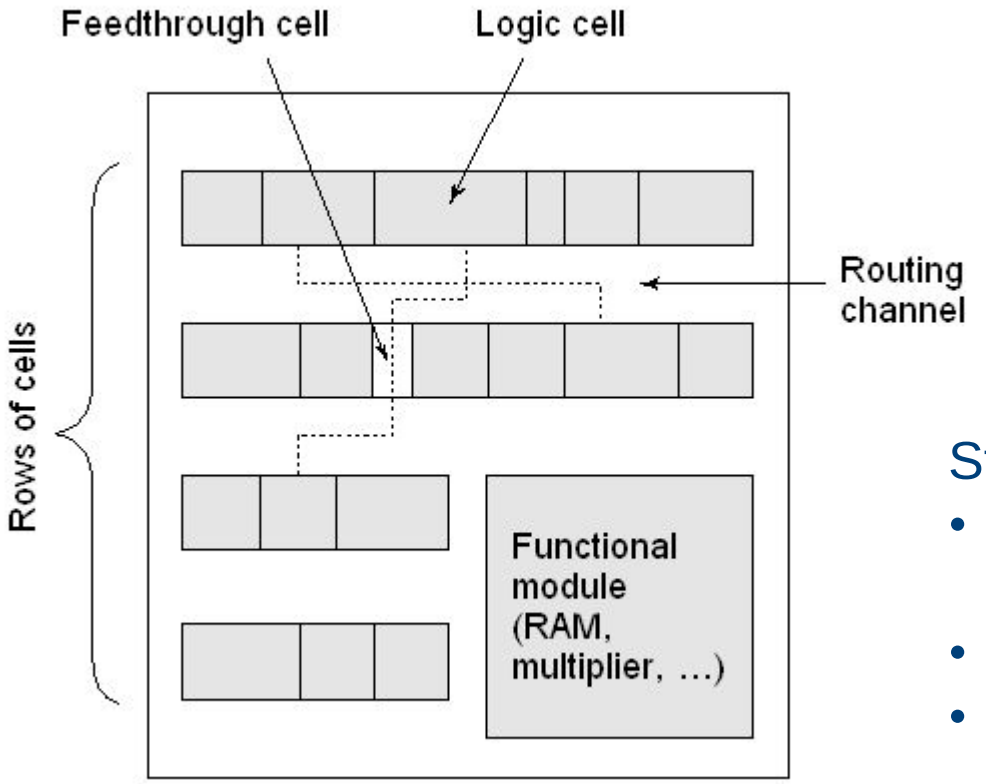
- ‘Really’ design your own (digital) hardware components
- More freedom -> better designs (performance/power) in principle possible
- More labor intensive & error prone
- Semi-custom: partly library-based

Standard cell-based design: the lego-block approach for digital design

Conceive & realize
a large variety of designs
based on library of physical ‘standard’ cells:

- Typically made available by foundry
- Limited number of components

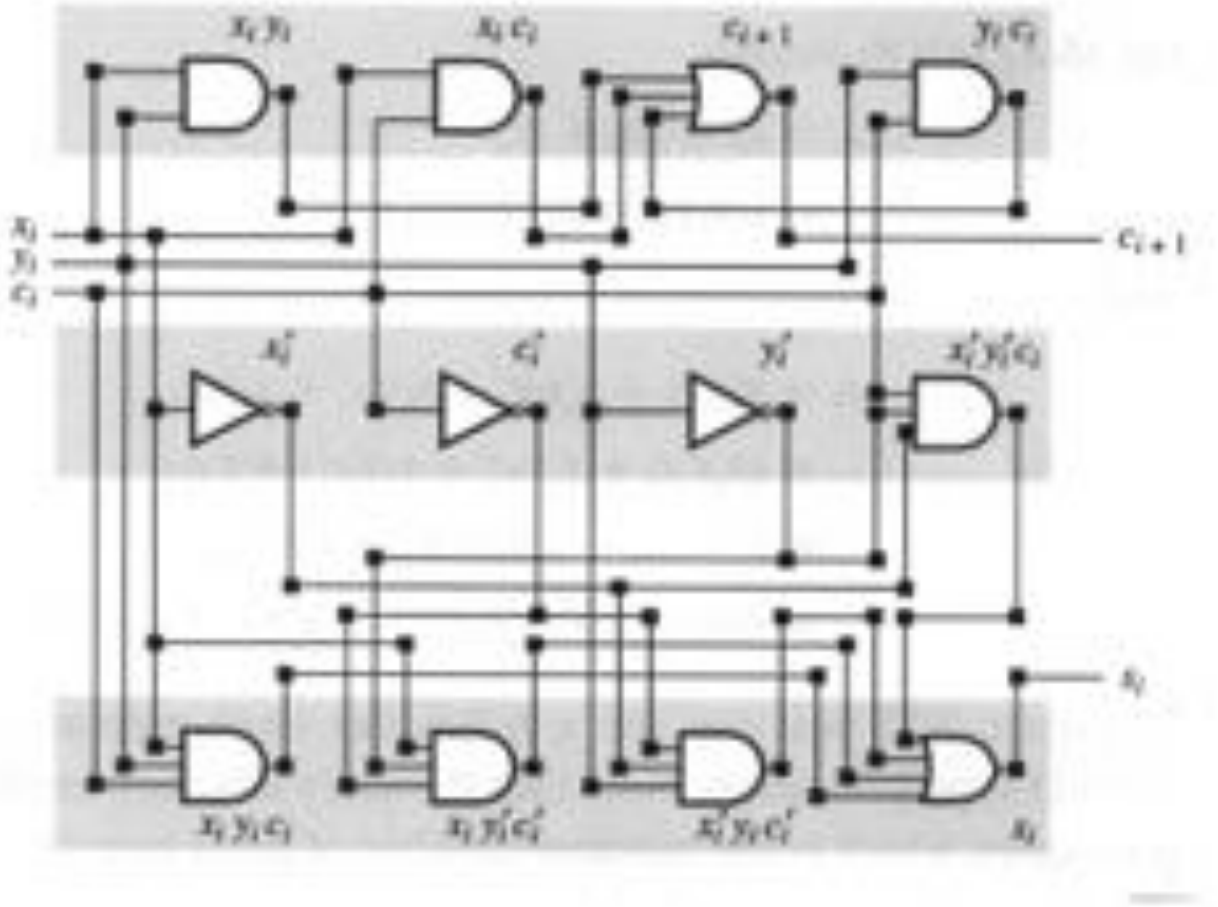
Standard cell-based design: Ready for structured layout (physical design)



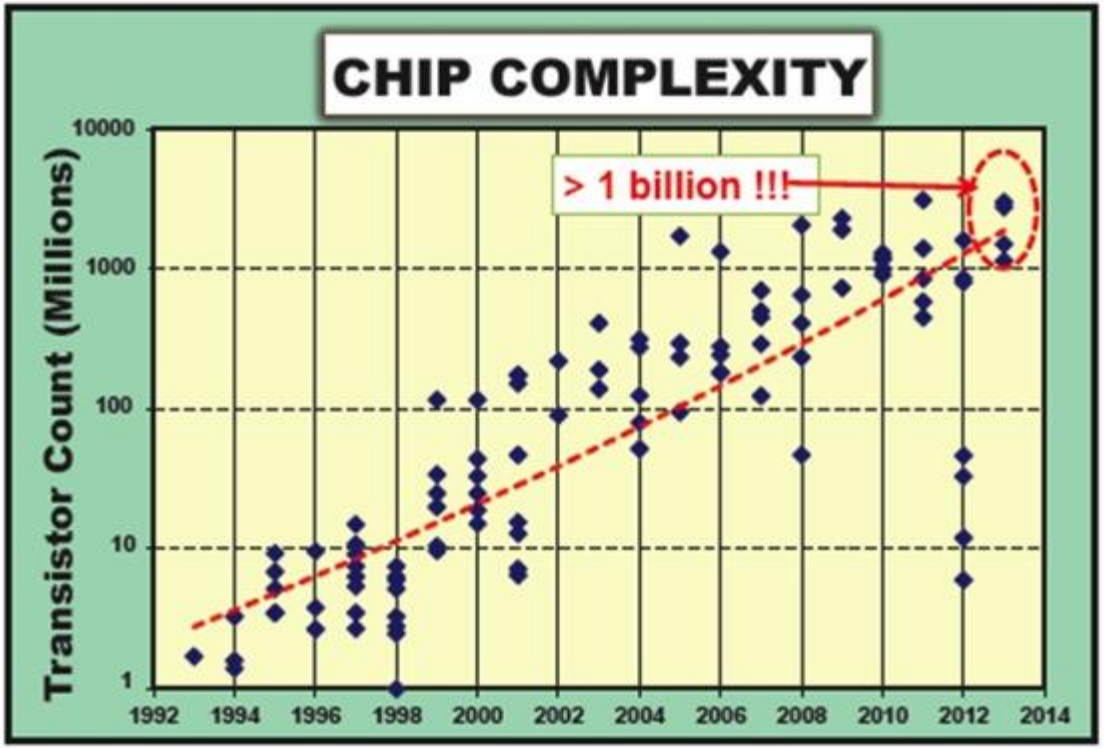
Standard cells are:

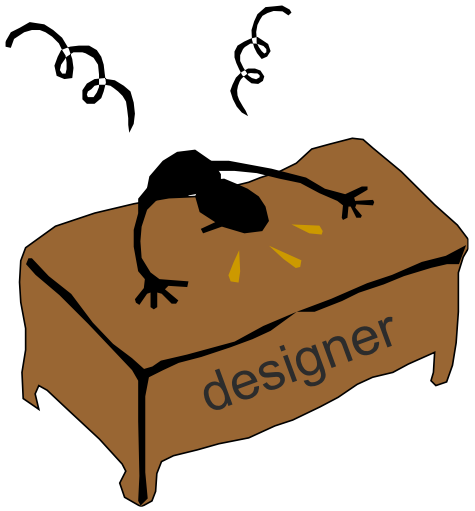
- predesigned components = building blocks
- Same height/different width,
- Inputs & outputs at bottom or top

Example: full-adder implemented in standard cells



© Daniel D. Gajski ‘Principles of digital design’



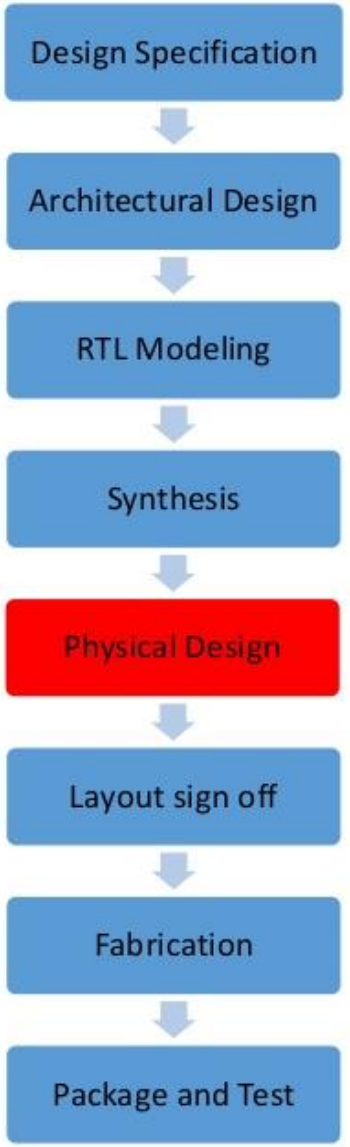


EDA: Electronic Design Automation
tools help designers to master design complexity

Hardware implementation/chip design: from specification to chip ready

Gradually lowering the abstraction level of the system until we have a chip produced

Each level of abstraction has its own design and verification methodology

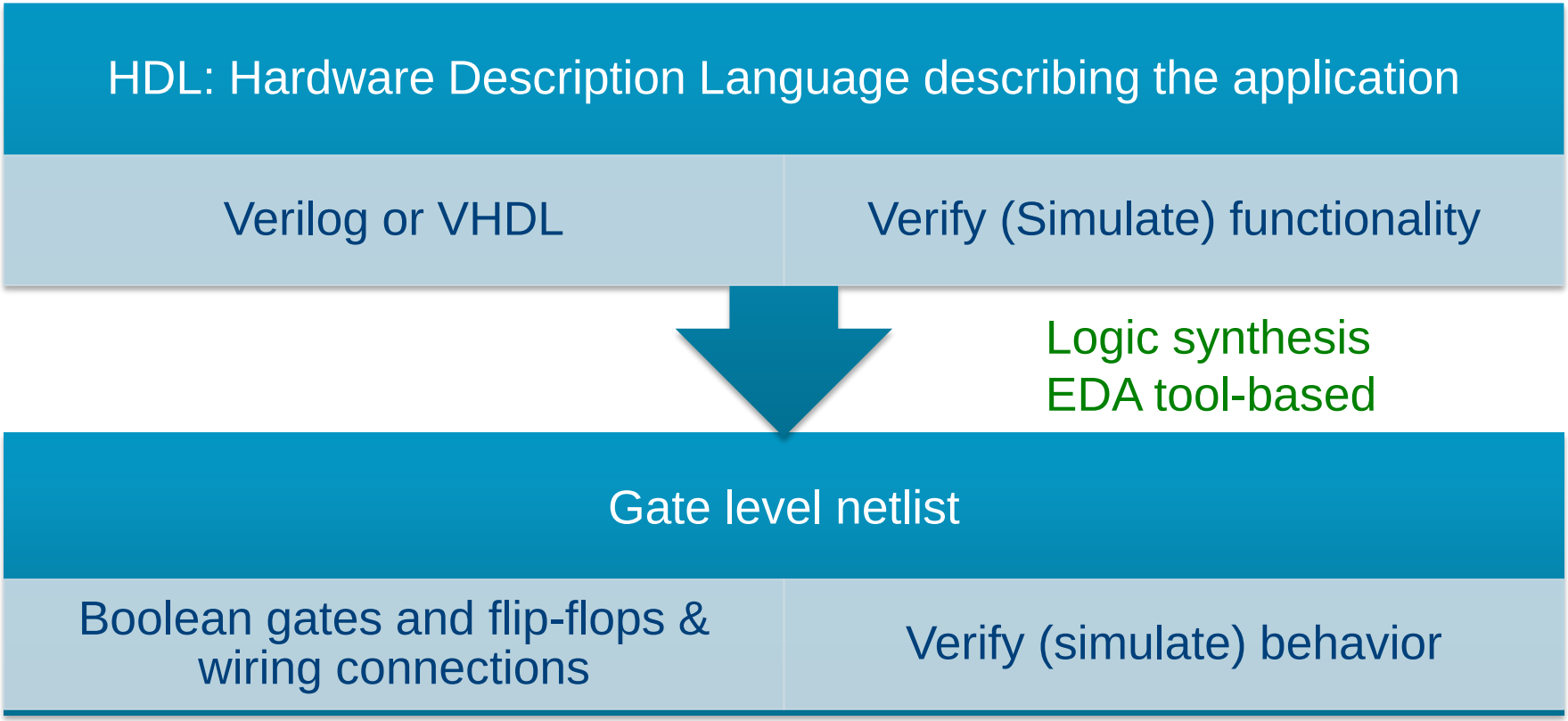


DIGITAL: semi- custom based on Hardware Description Language

- Concurrent languages: VHDL, Verilog
- Abstraction level: Register Transfer
- Which memory element
  has which value on
  which clock cycle
  & calculation of the next cycle

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

entity shifter is
port ( s_in   : in  std_logic;
       clk    : in  std_logic;
       p_load : in  std_logic;
       rst    : in  std_logic;
       p_in   : in  std_logic_vector(15 downto
0);
       p_out  : out std_logic_vector(15 downto
0));
       -- s_out = p_out(0)  !!!
end shifter;

architecture design OF shifter IS
  signal shift16 : std_logic_vector( 15 downto
0);
begin
  process(clk,rst)
  begin
    if (rst = '1') then
      shift16 <= (others => '0');
    elsif rising_edge(clk) then
      if (p_load = '1') then
        shift16 <= p_in;
      else
        shift16 <= s_in & shift16(15 downto 1);
      end if;
    end if;
  end process;

  P_out <= shift16;

end design;
```

Logic synthesis:
from HDL to logic gates level netlist



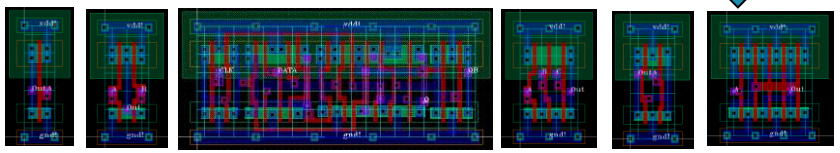
Digital ASIC Design: Synthesis to Netlist

RTL Code

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

entity shifter is
port ( s_in    : in  std_logic;
       clk     : in  std_logic;
       p_load  : in  std_logic;
       rst     : in  std_logic;
       p_in    : in  std_logic_vector(15 downto 0);
       p_out   : out std_logic_vector(15 downto 0));
       -- s_out = p_out(0)  !!!
end shifter;

architecture design OF shifter IS
  signal shift16 : std_logic_vector( 15 downto 0);
begin
  process(clk,rst)
  begin
    if (rst = '1') then
      shift16 <= (others => '0');
    elsif rising_edge(clk) then
      if (p_load = '1') then
        shift16 <= p_in;
      else
        shift16 <= s_in & shift16(15 downto 1);
      end if;
    end if;
  end process;

  P_out <= shift16;

end design;
```

Synthesis Tool

Standard Cell Library



A set of pre-designed, same height basic digital building blocks (inv, and, or, flip-flop, mux,...)
Typically several 100's of cells

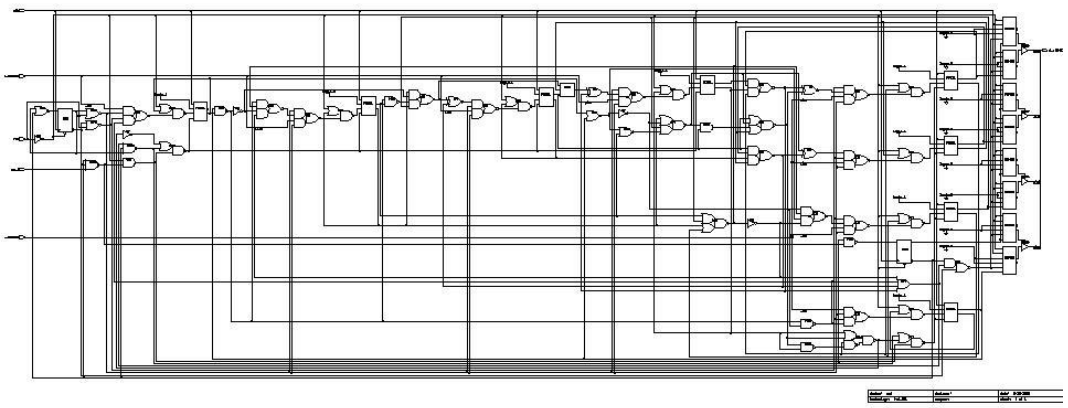
```
module shifter ( s_in, clk, p_load, rst, p_in, p_out );
  input [15:0] p_in;
  output [15:0] p_out;
  input s_in, clk, p_load, rst;
  wire   n57, n58;

  INVD1 U33 ( .I(rst), .ZN(n57) );
  SDFCNQD1 shift16_reg_15_ ( .D(p_in[15]), .SI(s_in),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[15]) );
  SDFCNQD1 shift16_reg_14_ ( .D(p_in[14]), .SI(p_out[15]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[14]) );
  SDFCNQD1 shift16_reg_13_ ( .D(p_in[13]), .SI(p_out[14]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[13]) );
  SDFCNQD1 shift16_reg_12_ ( .D(p_in[12]), .SI(p_out[13]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[12]) );
  SDFCNQD1 shift16_reg_11_ ( .D(p_in[11]), .SI(p_out[12]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[11]) );
  SDFCNQD1 shift16_reg_10_ ( .D(p_in[10]), .SI(p_out[11]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[10]) );
  SDFCNQD1 shift16_reg_9_ ( .D(p_in[9]), .SI(p_out[10]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[9]) );
  SDFCNQD1 shift16_reg_8_ ( .D(p_in[8]), .SI(p_out[9]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[8]) );
  SDFCNQD1 shift16_reg_7_ ( .D(p_in[7]), .SI(p_out[8]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[7]) );
  SDFCNQD1 shift16_reg_6_ ( .D(p_in[6]), .SI(p_out[7]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[6]) );
  SDFCNQD1 shift16_reg_5_ ( .D(p_in[5]), .SI(p_out[6]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[5]) );
  SDFCNQD1 shift16_reg_4_ ( .D(p_in[4]), .SI(p_out[5]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[4]) );
  SDFCNQD1 shift16_reg_3_ ( .D(p_in[3]), .SI(p_out[4]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[3]) );
  SDFCNQD1 shift16_reg_2_ ( .D(p_in[2]), .SI(p_out[3]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[2]) );
  SDFCNQD1 shift16_reg_1_ ( .D(p_in[1]), .SI(p_out[2]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[1]) );
  SDFCNQD1 shift16_reg_0_ ( .D(p_in[0]), .SI(p_out[1]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[0]) );
  INVD1 U51 ( .I(p_load), .ZN(n58) );
endmodule
```

Gate level netlist

The netlist describes a digital schematic

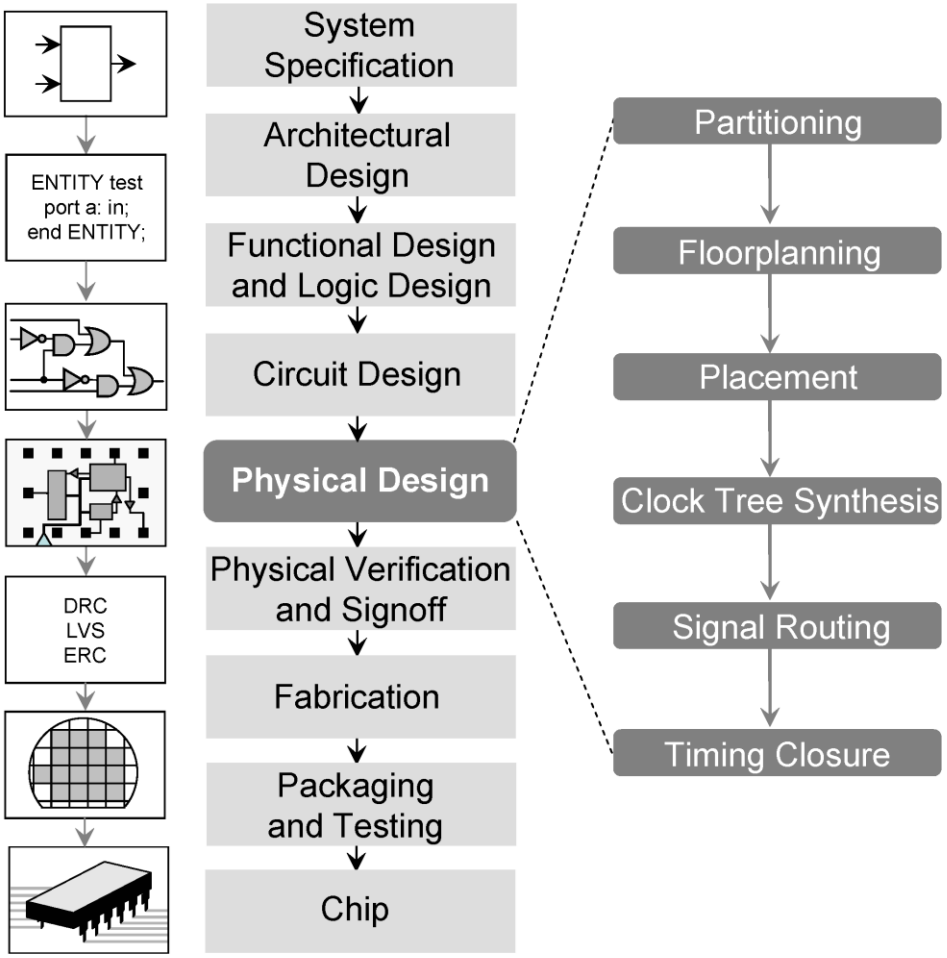
```
module shifter ( s_in, clk, p_load, rst, p_in, p_out );
  input [15:0] p_in;
  output [15:0] p_out;
  input s_in, clk, p_load, rst;
  wire   n57, n58;

  INVD1 U33 ( .I(rst), .ZN(n57) );
  SDFCNQD1 shift16_reg_15_ ( .D(p_in[15]), .SI(s_in),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[15]) );
  SDFCNQD1 shift16_reg_14_ ( .D(p_in[14]), .SI(p_out[15]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[14]) );
  SDFCNQD1 shift16_reg_13_ ( .D(p_in[13]), .SI(p_out[14]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[13]) );
  SDFCNQD1 shift16_reg_12_ ( .D(p_in[12]), .SI(p_out[13]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[12]) );
  SDFCNQD1 shift16_reg_11_ ( .D(p_in[11]), .SI(p_out[12]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[11]) );
  SDFCNQD1 shift16_reg_10_ ( .D(p_in[10]), .SI(p_out[11]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[10]) );
  SDFCNQD1 shift16_reg_9_ ( .D(p_in[9]), .SI(p_out[10]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[9]) );
  SDFCNQD1 shift16_reg_8_ ( .D(p_in[8]), .SI(p_out[9]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[8]) );
  SDFCNQD1 shift16_reg_7_ ( .D(p_in[7]), .SI(p_out[8]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[7]) );
  SDFCNQD1 shift16_reg_6_ ( .D(p_in[6]), .SI(p_out[7]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[6]) );
  SDFCNQD1 shift16_reg_5_ ( .D(p_in[5]), .SI(p_out[6]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[5]) );
  SDFCNQD1 shift16_reg_4_ ( .D(p_in[4]), .SI(p_out[5]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[4]) );
  SDFCNQD1 shift16_reg_3_ ( .D(p_in[3]), .SI(p_out[4]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[3]) );
  SDFCNQD1 shift16_reg_2_ ( .D(p_in[2]), .SI(p_out[3]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[2]) );
  SDFCNQD1 shift16_reg_1_ ( .D(p_in[1]), .SI(p_out[2]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[1]) );
  SDFCNQD1 shift16_reg_0_ ( .D(p_in[0]), .SI(p_out[1]),
.SE(n58), .CP(clk),
        .CDN(n57), .Q(p_out[0]) );
  INVD1 U51 ( .I(p_load), .ZN(n58) );
endmodule
```

=



Size of digital parts of chips: expressed in “Equivalent Gates”.
= number of basic NAND2 gates for the size of the design

Physical (back-end) design: hierarchical process



Physical design: all design components are instantiated with their geometric representations: All macros, cells, gates, transistors, .. with fixed shapes and sizes per fabrication layer
are assigned spatial locations (placement) and have appropriate routing connections (routing) completed in metal layers.
Result of physical design:
set of manufacturing specifications that must subsequently be verified.

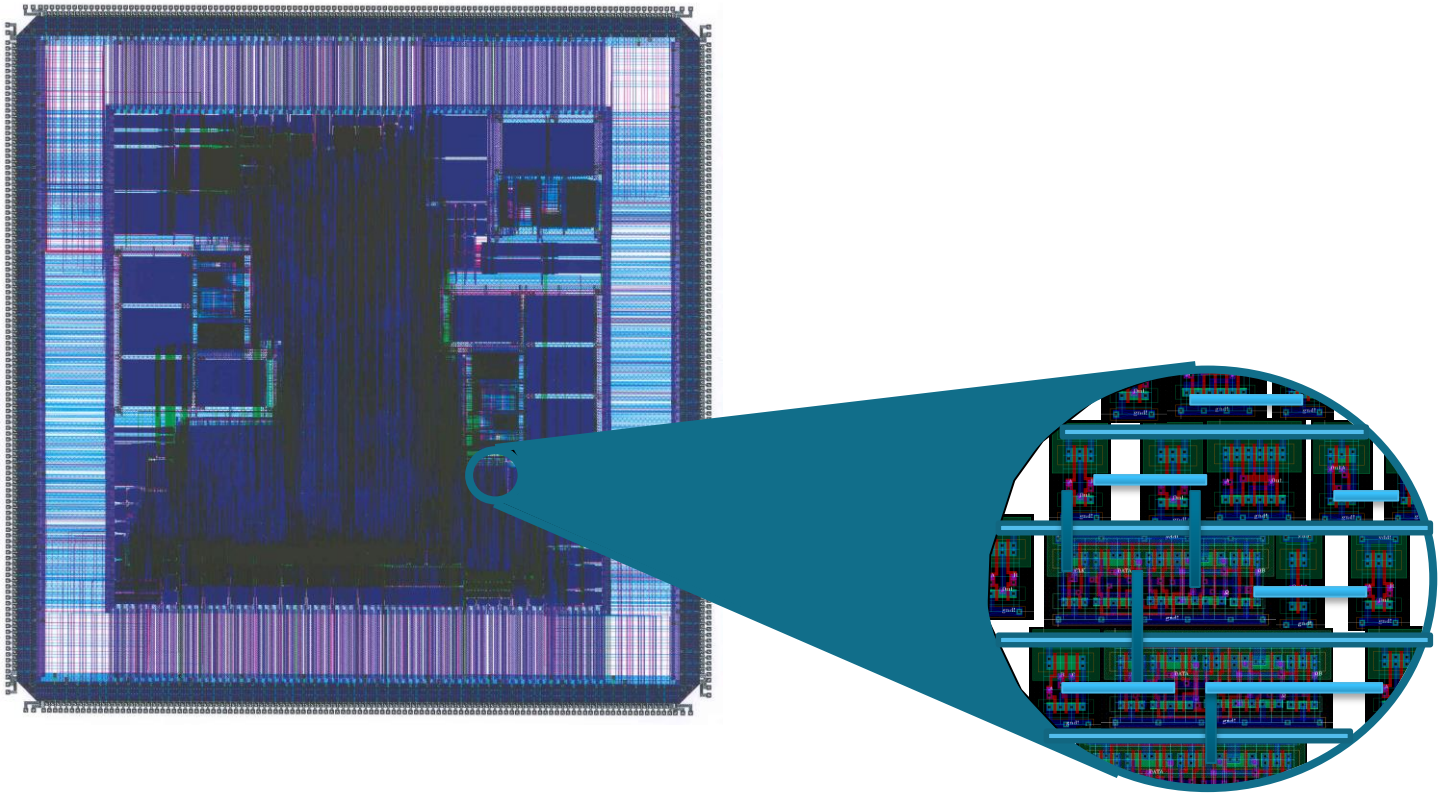
Process Design Kit (PDK): Need to get from the foundry

Specific for a technology and for a set of design tools

- Models for Transistors etc.
- Design and layout Rules
- Layer descriptions
- ‘decks’ for steps in the design flow
- …

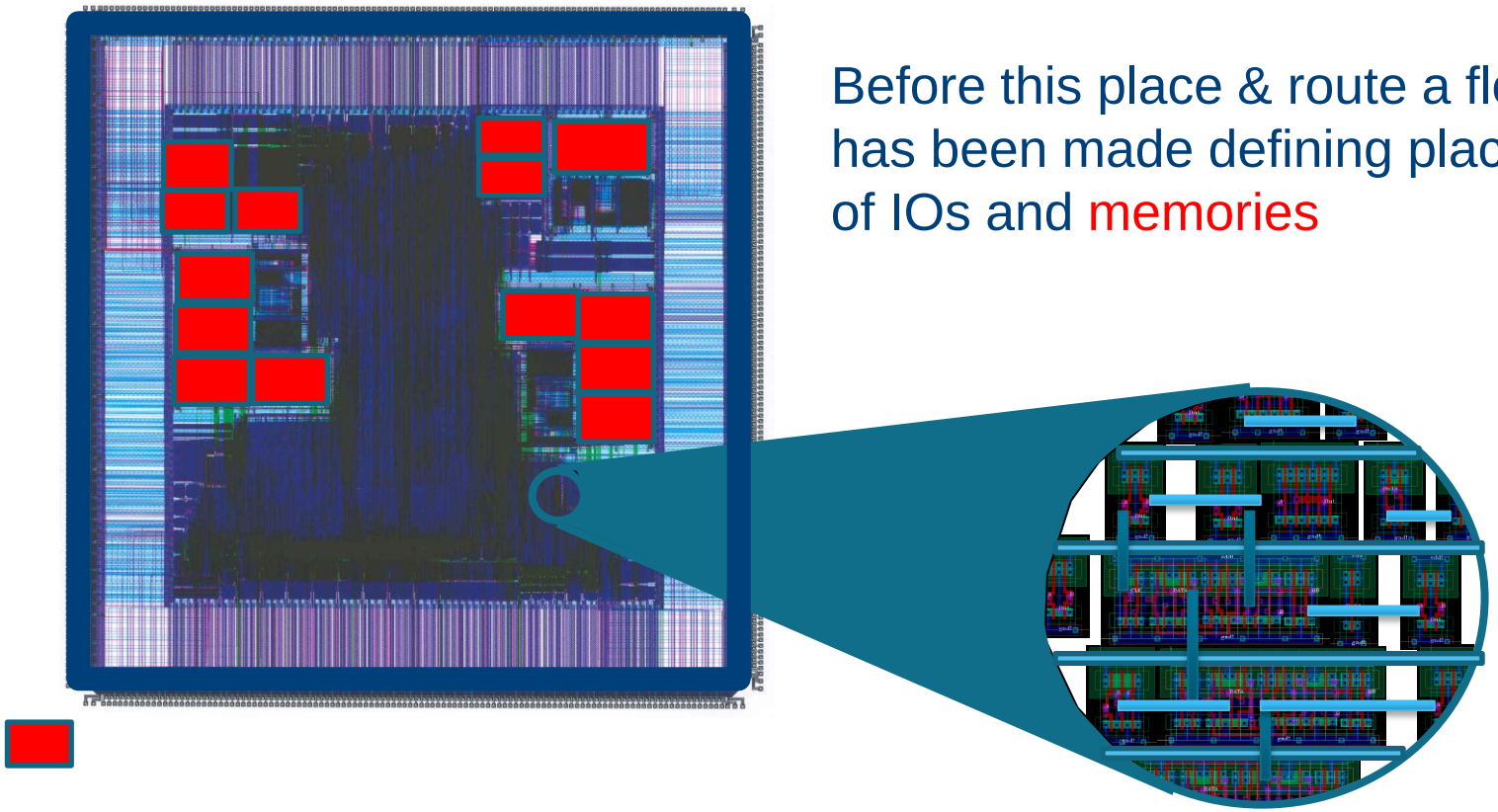
Semi-custom Layout: heavily tools aided

- The standard cells are placed in rows
- Then interconnected



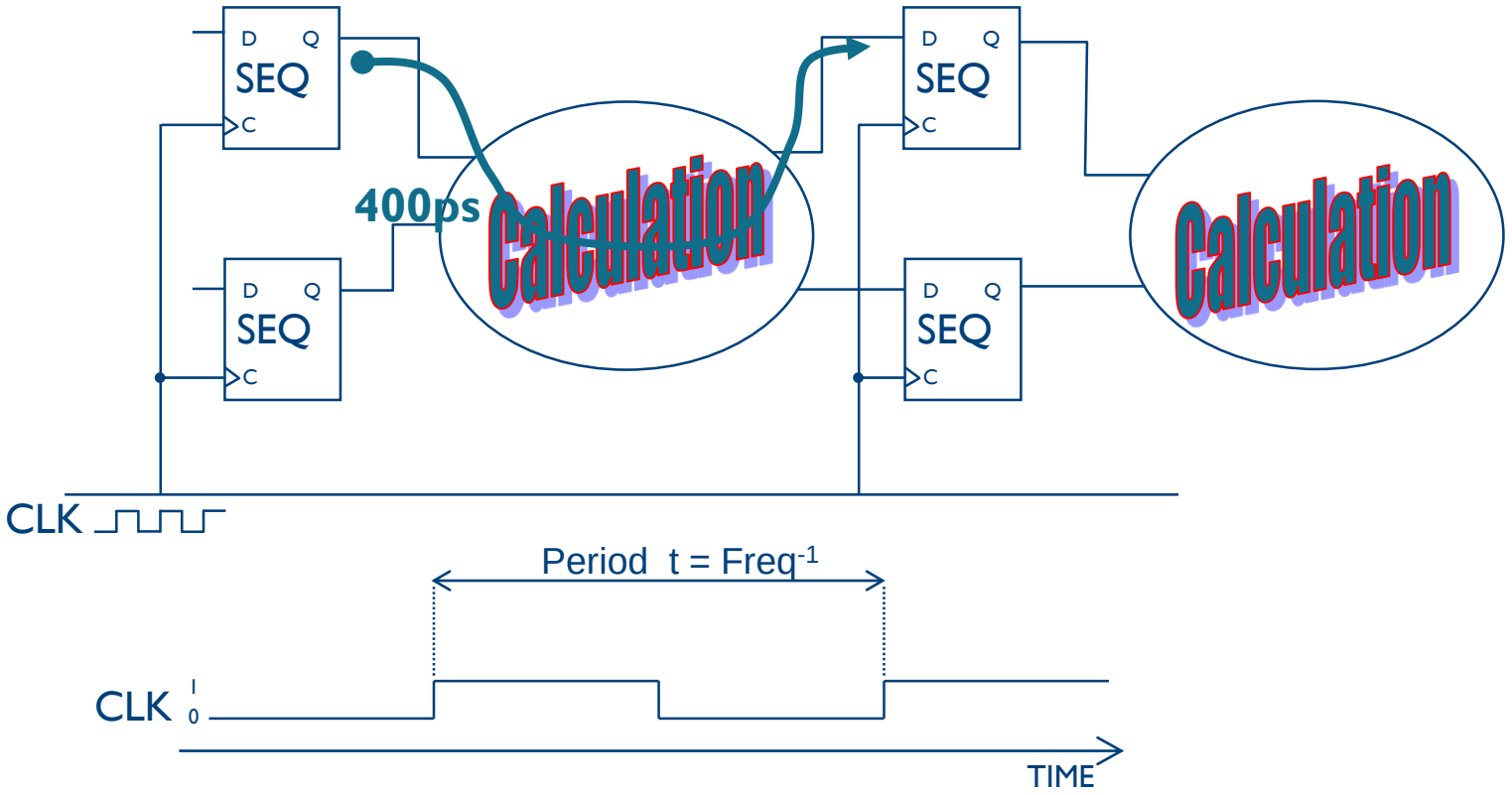
Semi-custom Layout: hierarchical approach

- The standard cells are placed in rows
- Then interconnected



Before this place & route a floorplan has been made defining placement of IOs and memories

Frequency / timing analysis: calculations need to be ready in time (clock frequency dependent)



E.g.: 2.5GHz => t = 400ps = 400 . 10^{-12}s

Tapeout checks
Before production – standard checks

- Design Rule Check (DRC)
  - Is the layout living up to the rules of the manufacturer?
- Electrical Rule Check (ERC)
  - Are there shortcircuits... in the design?
- Layout Versus Schematic (LVS)
  - Is the layout equal to the netlist?

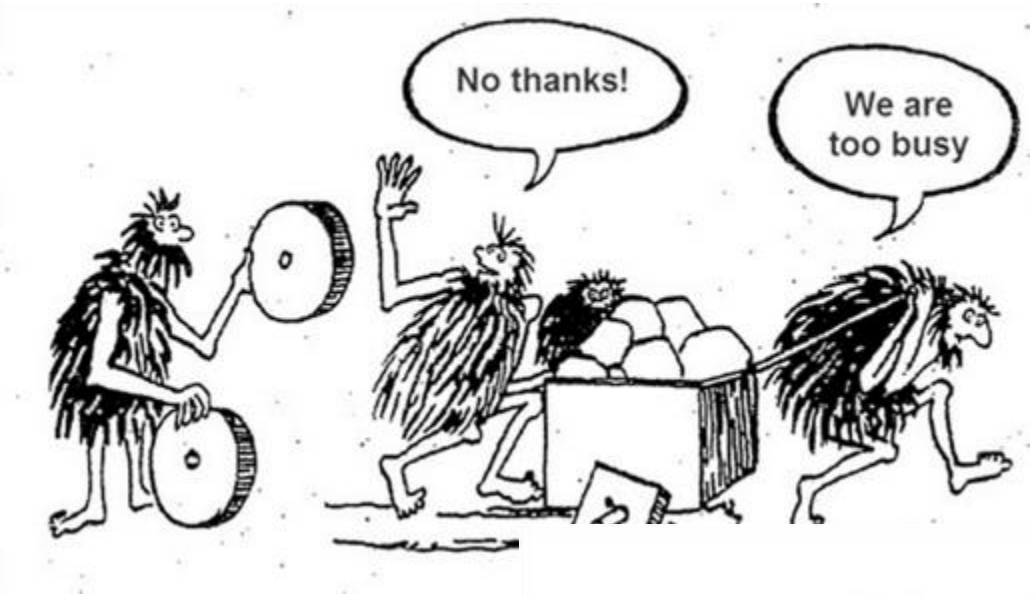
Chip design result = GDSII, a database format surviving it's parent company

- GDS II: GDS stands for Graphic Data System and is a standard for database interchange of ASIC artwork. The first version of the database, GDS, that was introduced in 1971, and GDSII was introduced later in 1978 by a US-based company Calma.

- The database is essentially a binary file format consisting of geometric shapes, labels, and additional data that a foundry can use to create a silicon chip.

- 'Tape-out':
  - Historically: tape handed over to the fab (foundry)
  - Currently means: GDSII released

Design reuse:
key to success in (complex) system design



IP Cores: available **AND** reliable for your design

- Don’t reinvent the wheel
  - Focus on the **uniqueness**
  - Designed by specialists in that field
  - Well documented
  - (Mostly) Silicon Proven
- Different kinds
  - Hard Cores: fixed to a specific technology
  - Soft Cores: Synthesizeable RTL code
- Typical IP
  - Cell libraries, RAM, ROM, ... (Hard)
  - PLL, ADC, DAC,... (Hard)
  - USB, DDR, MIPI, ... (Soft)
  - ...

From tape to market: Fry, Dice, Package and Test

yield!

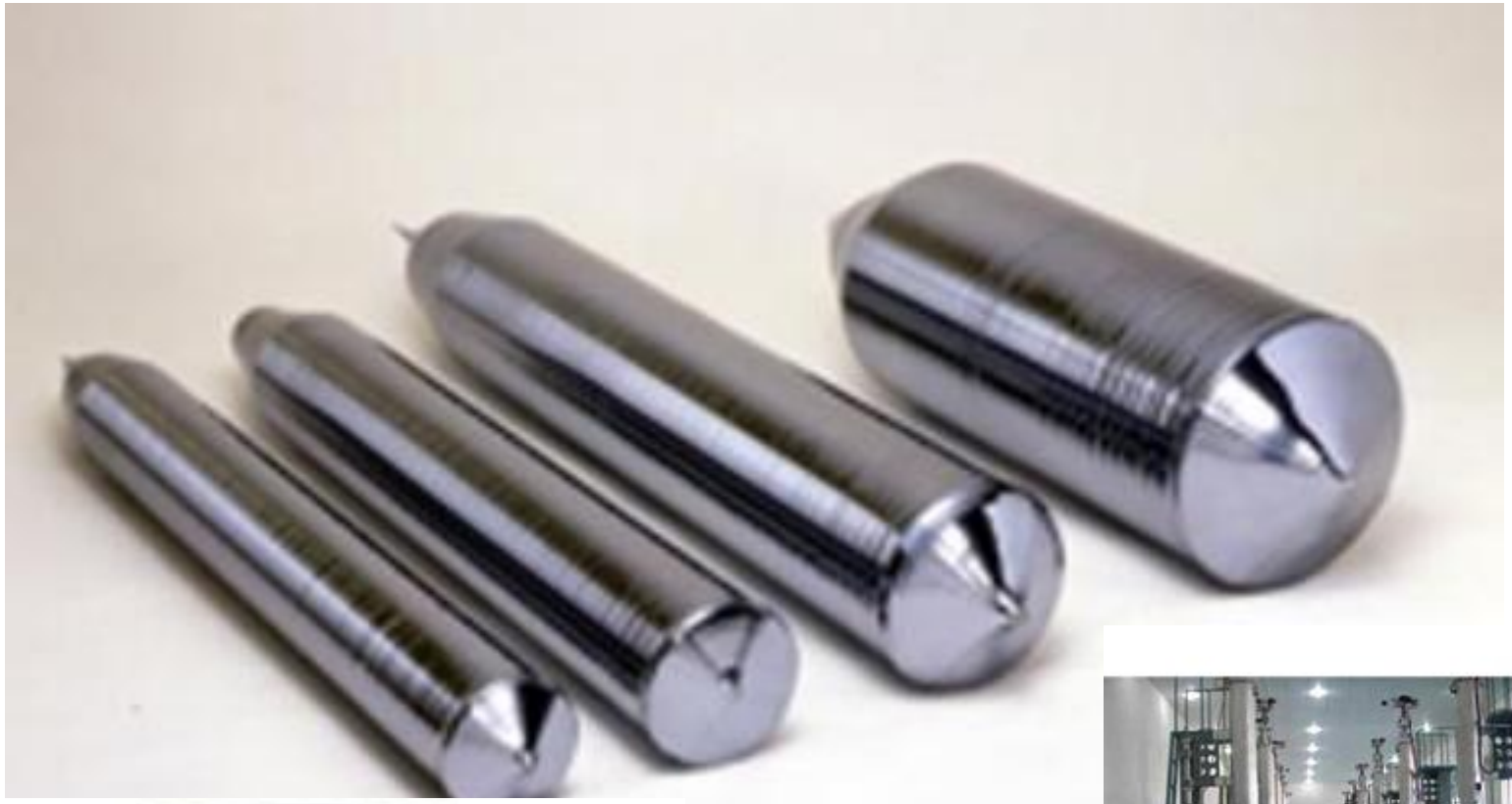
GDSII | Wafers | Chips | Good chips

Design → Manufacturing → Dicing & Packaging → Test → Market





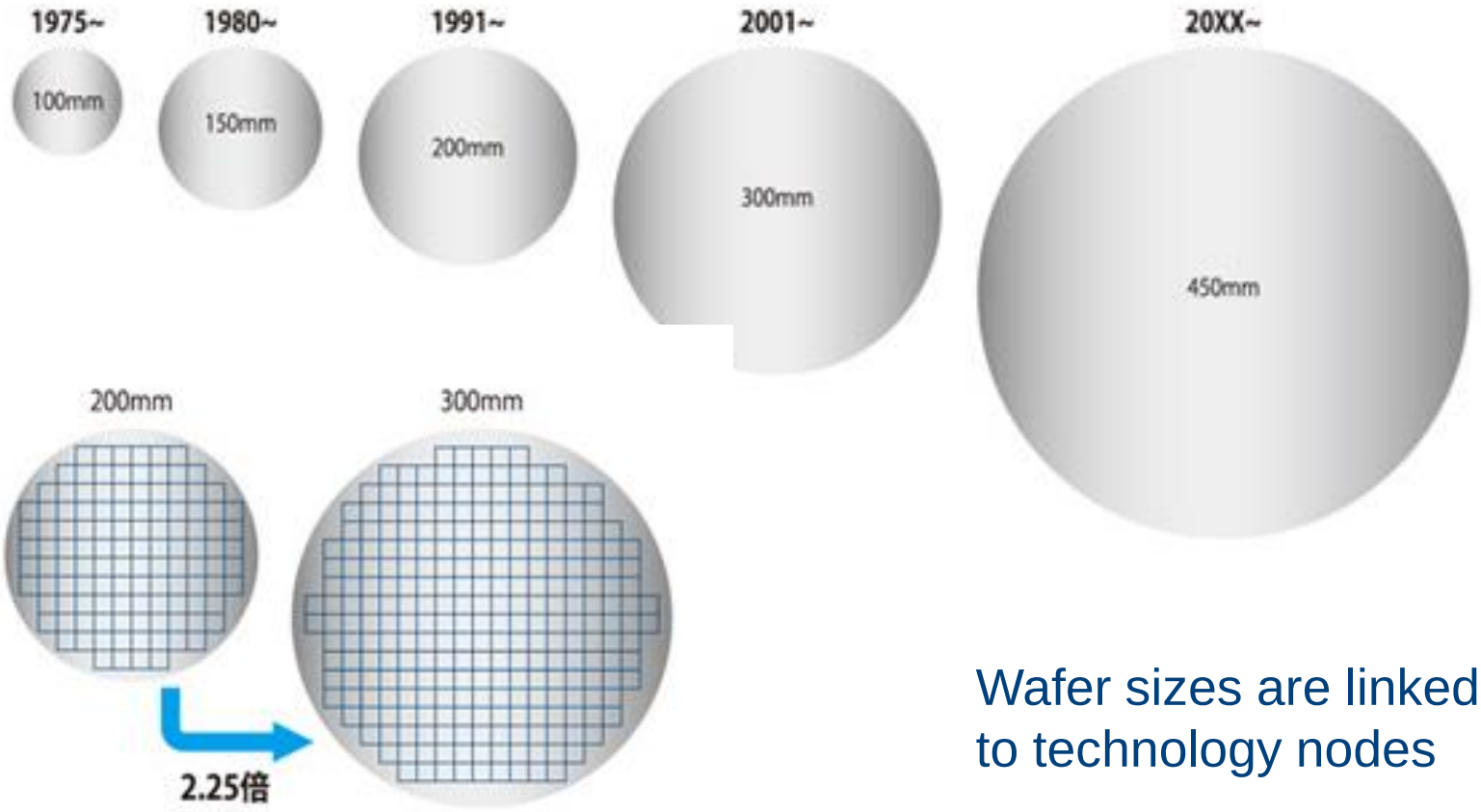

Silicon wafers are cut from ingots



An ingot sawing plant

Wafer size increased over time

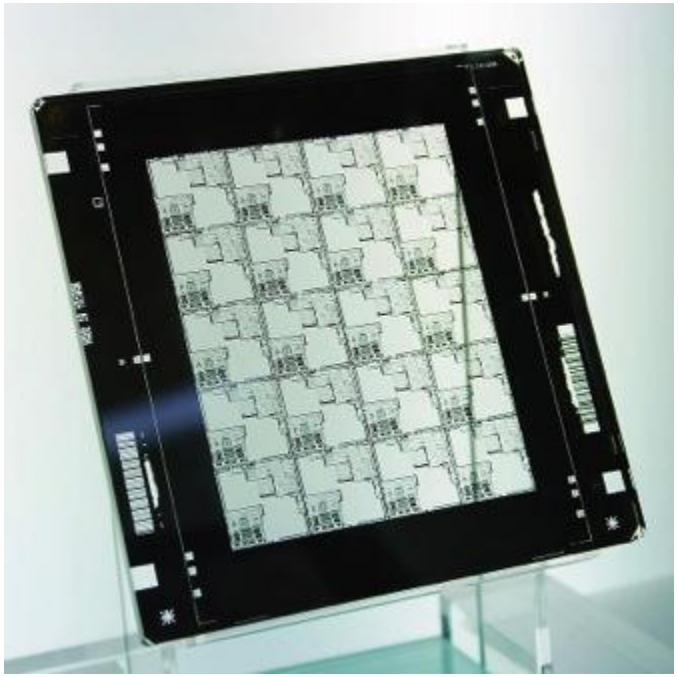
Increasing production efficiency



Wafer sizes are linked
to technology nodes

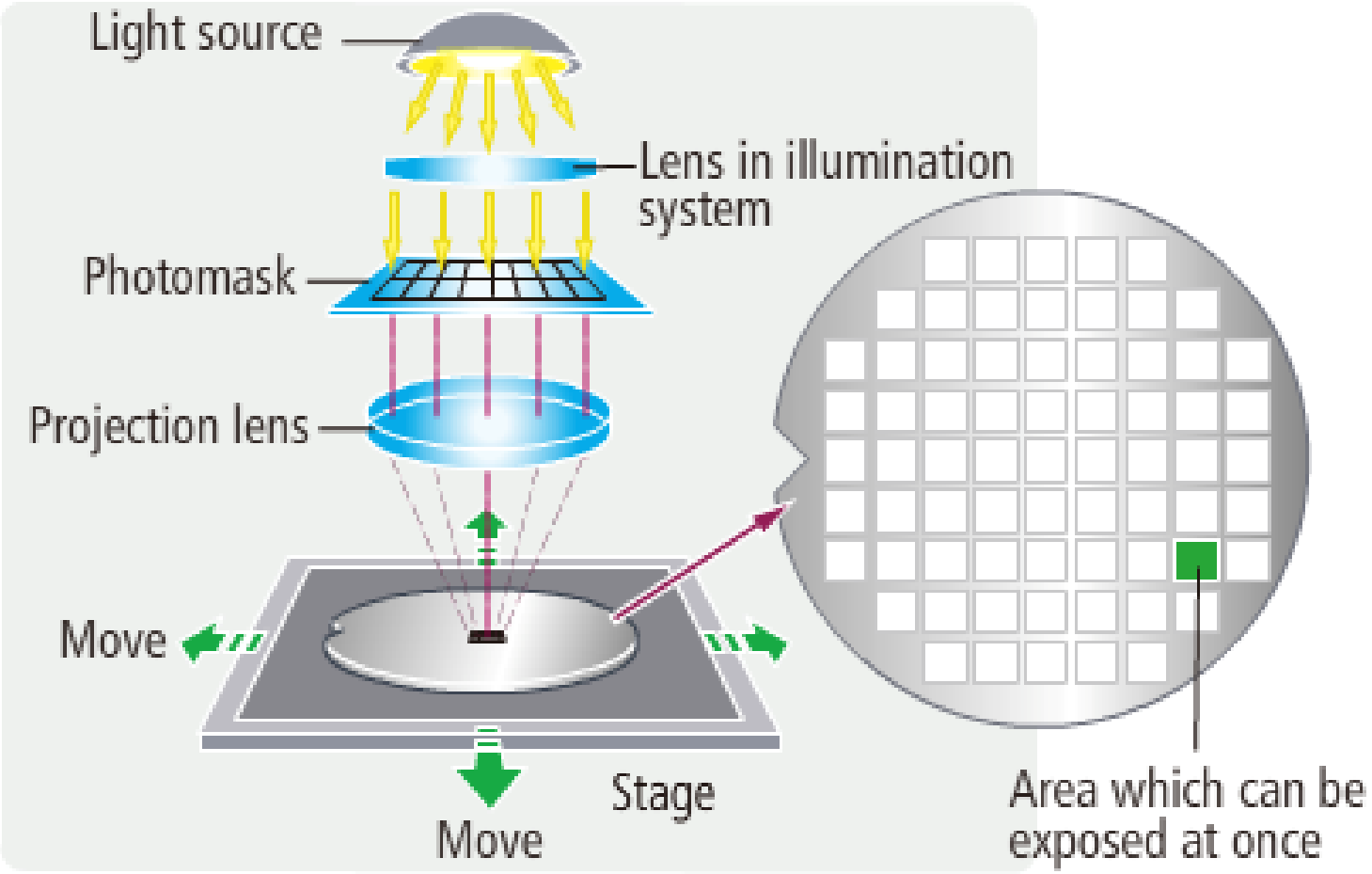
A 300mm wafer gives 2.25x
More chips of the same size

Lithography = key:
Chip layers are created by lithographical steps on the wafer using Masks



One mask

Reticle Stepping

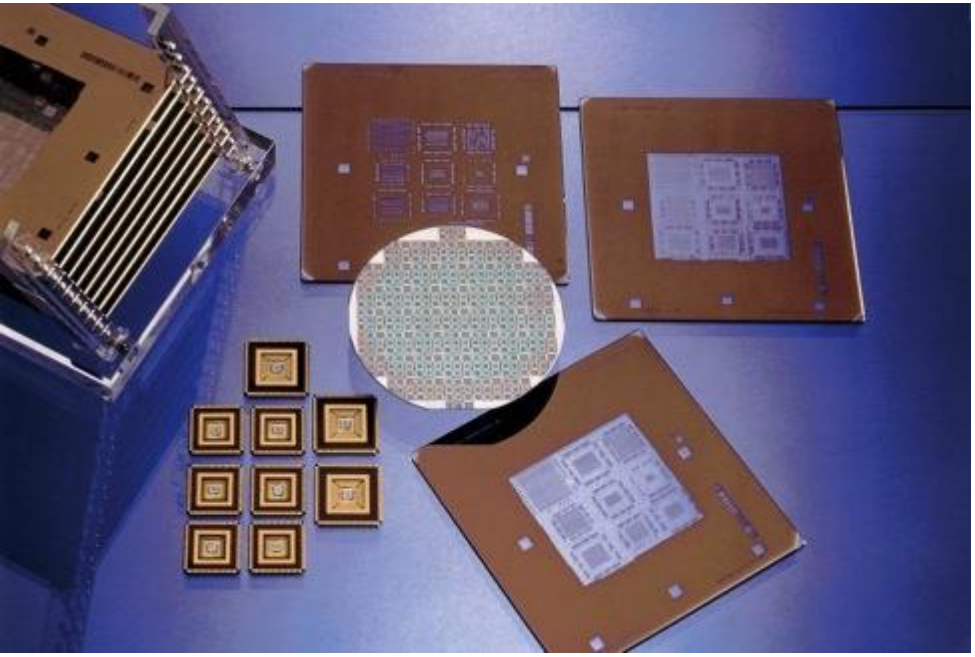


The part that is stepped is called the RETICLE

Mask SET needed to complete full processing of chips

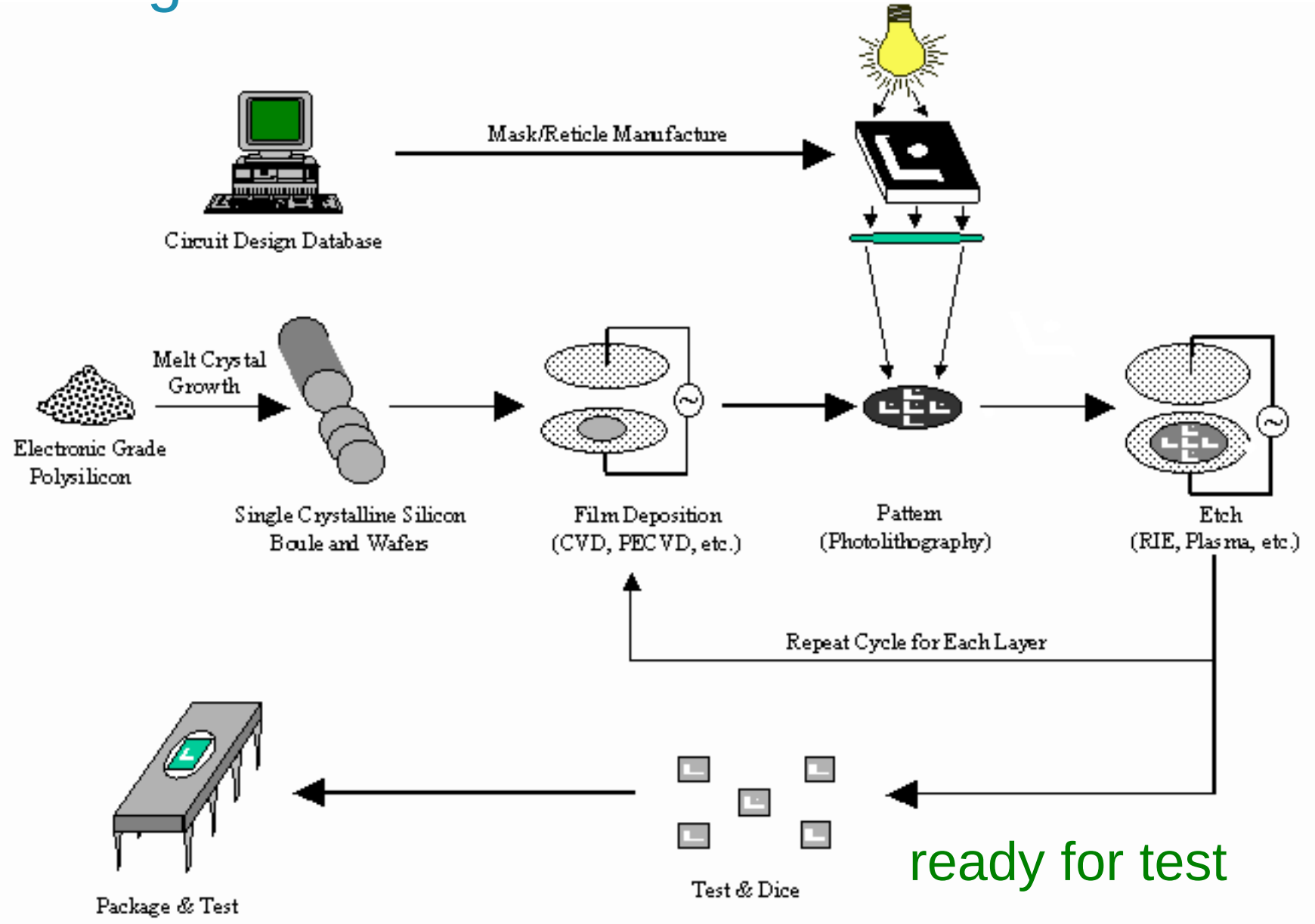
Amount of masks, intricacy & cost is a function…

- Of technology node
- Of the chosen number of metal levels
- Of the chosen options



Technology	Approx. Number of Masks
0.35 micron	26
0.25 micron	28
0.18 micron	30
0.13 micron	32
90 nm	36
65nm	39
40nm	44
28nm	44
16nm	56

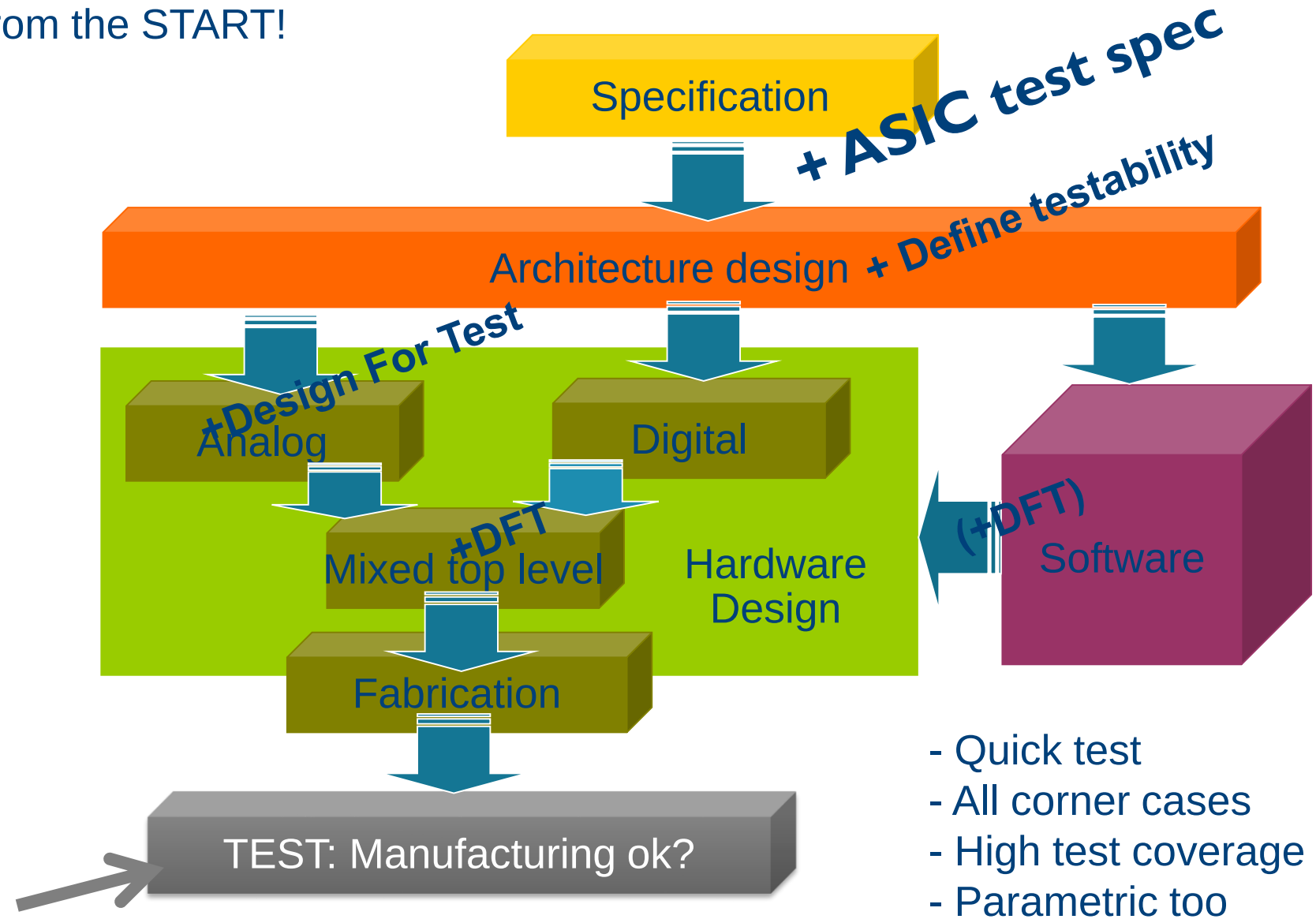
Design sent to fabrication




THEY FOUND ANOTHER NEEDLE IN A HAYSTACK...

Design For Test (DFT)

From the START!



- Quick test
- All corner cases
- High test coverage
- Parametric too

This is NOT a functional check!

Die yield: percentage of functional dies

Die = block of semiconducting material, on which a given functional circuit is fabricated.

Processed wafer is cut (“diced”) into pieces:

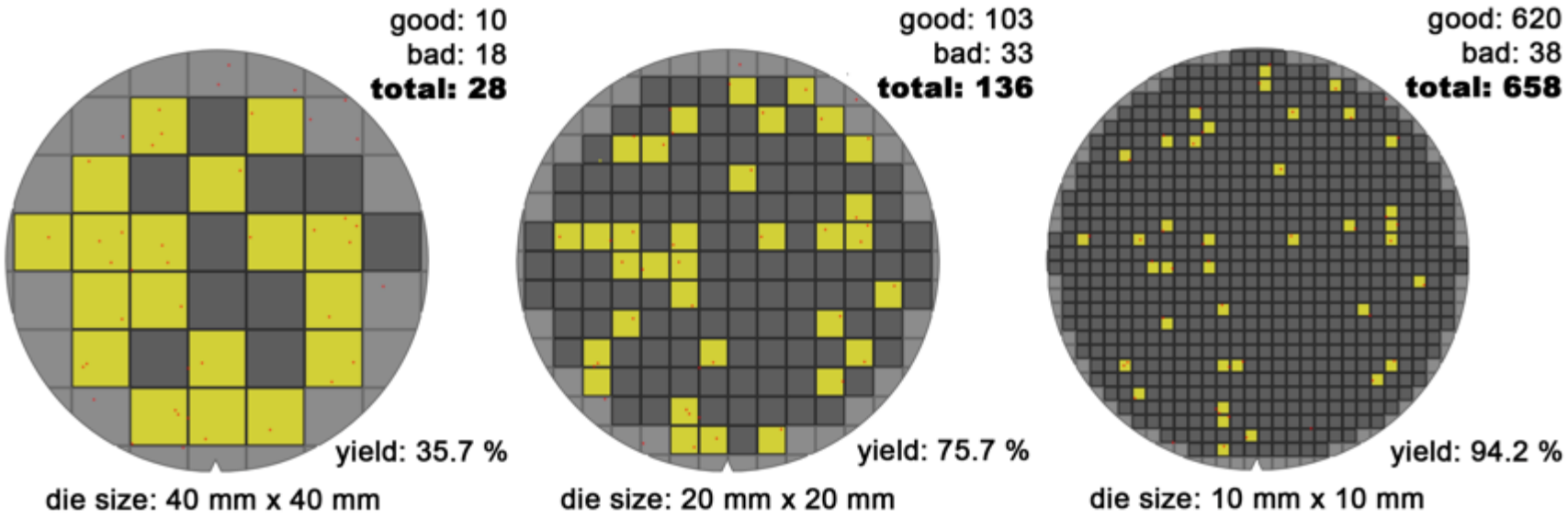
- each containing one copy of the circuit
- or a different circuit on a multi-project wafer

Each of these pieces is called a die.

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

with α = technology dependent parameter

Die yield: technology and area dependent



Bad dies can and will escape!

Today we almost often encounter the bad combination:

- Processing not perfect: yield
- Test coverage not 100%

⇒ There will be bad dies, and not all of them will be discovered in test

⇒ Bad dies will escape and go in products

How many bad dies can and will escape?

- Assume following yield formula applies:

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

- defects/unit area increases with smaller technology
- α = process complexity factor
- T = test coverage
- Bad samples escaping (in the test labeled good):

$$1 - Y^{(1-T)}$$

Bad dies escaping: how many?

Consider following parameters:

	Defects/cm^2	α (process complexity factor)
90nm	1	3
65nm	2	4

Estimate for following cases:

	Area	Yield	Bad samples escaping
Design 1, G gates, 90nm	2 mm^2		
Design 1, 65nm	1 mm^2		
Design 2, 2xG gates, 65nm	2 mm^2		

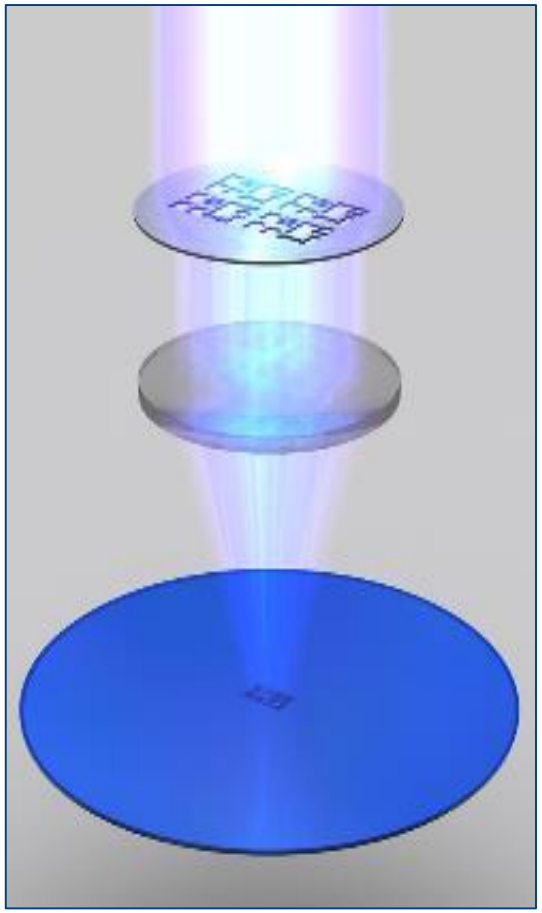
Chip projects: from idea to unique product

1. Chip projects: what is to be done?

2. Brief general semiconductor economics

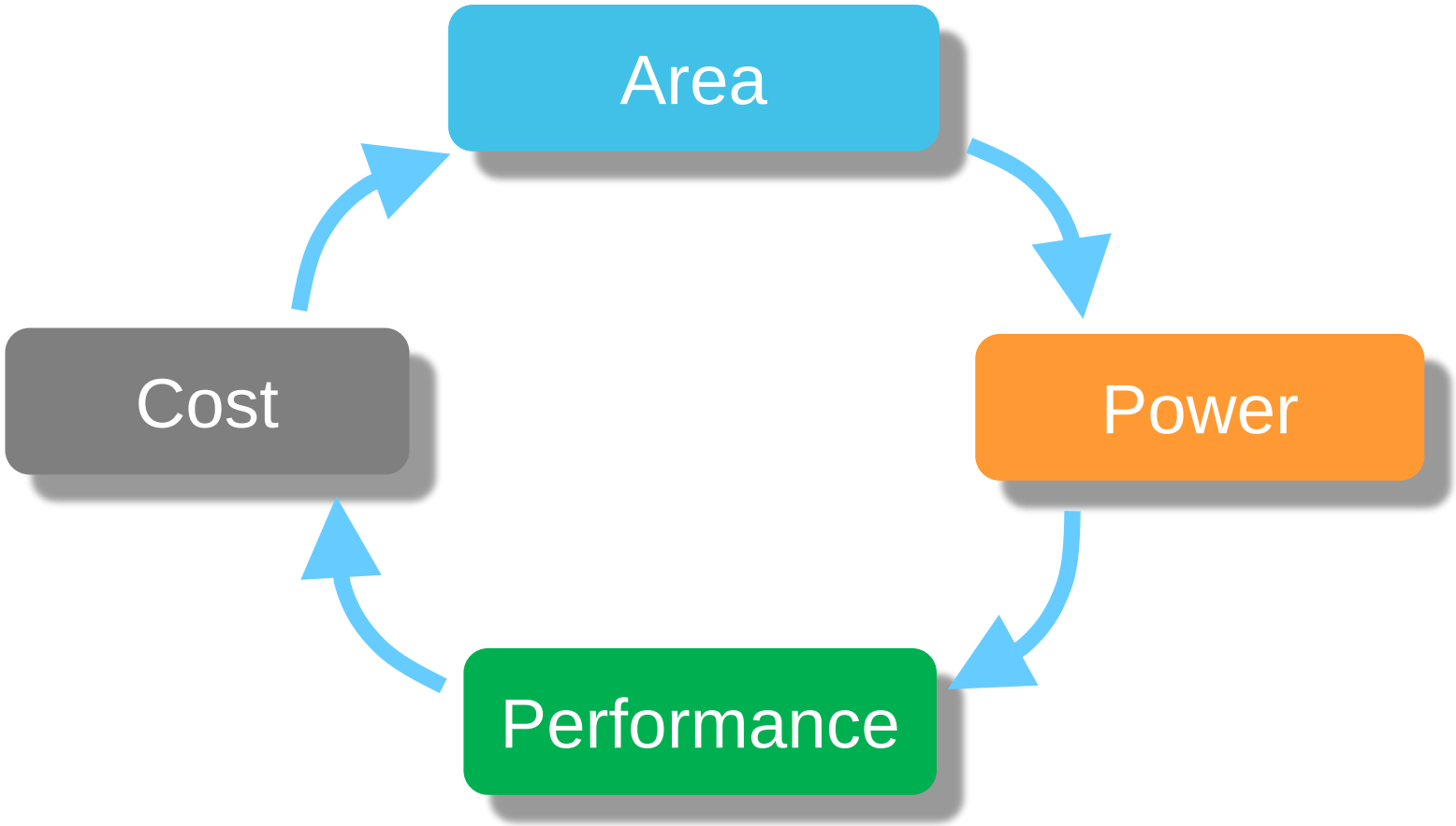
3. Exemplary project cases: bake your chip?

Why making chips is expensive



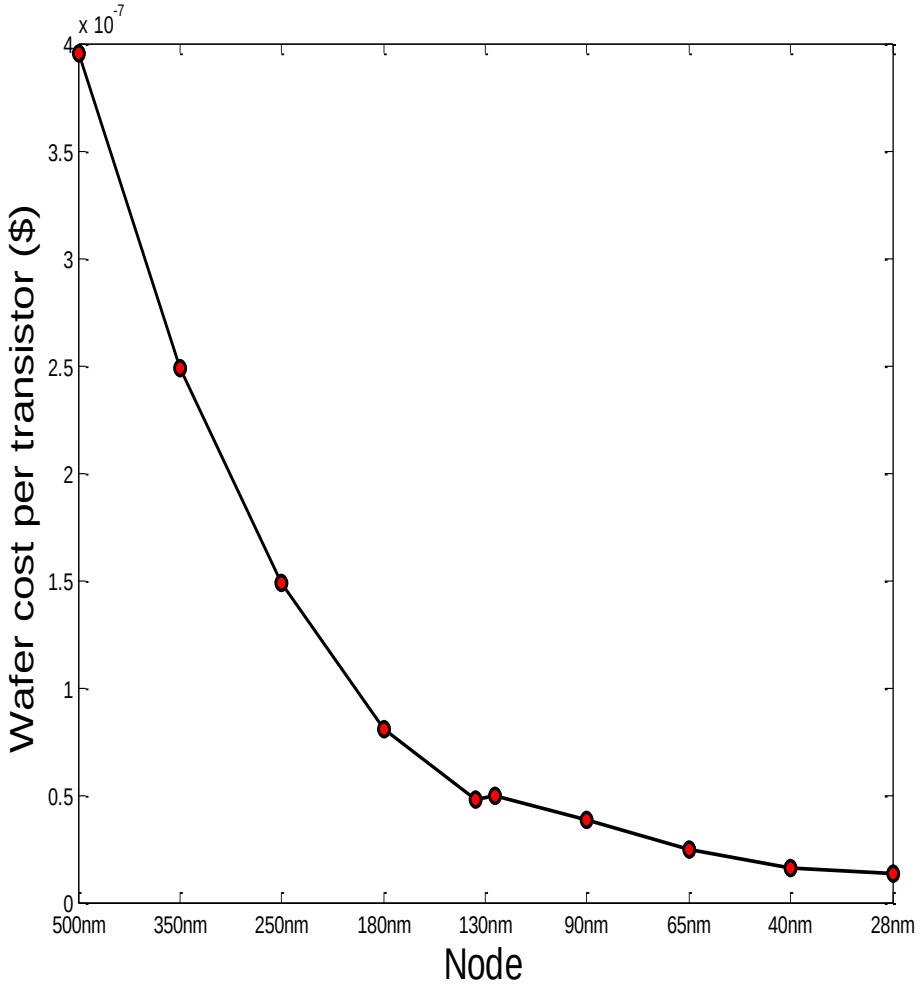
Design information sent to foundry to create optical masks

Why we scale logic



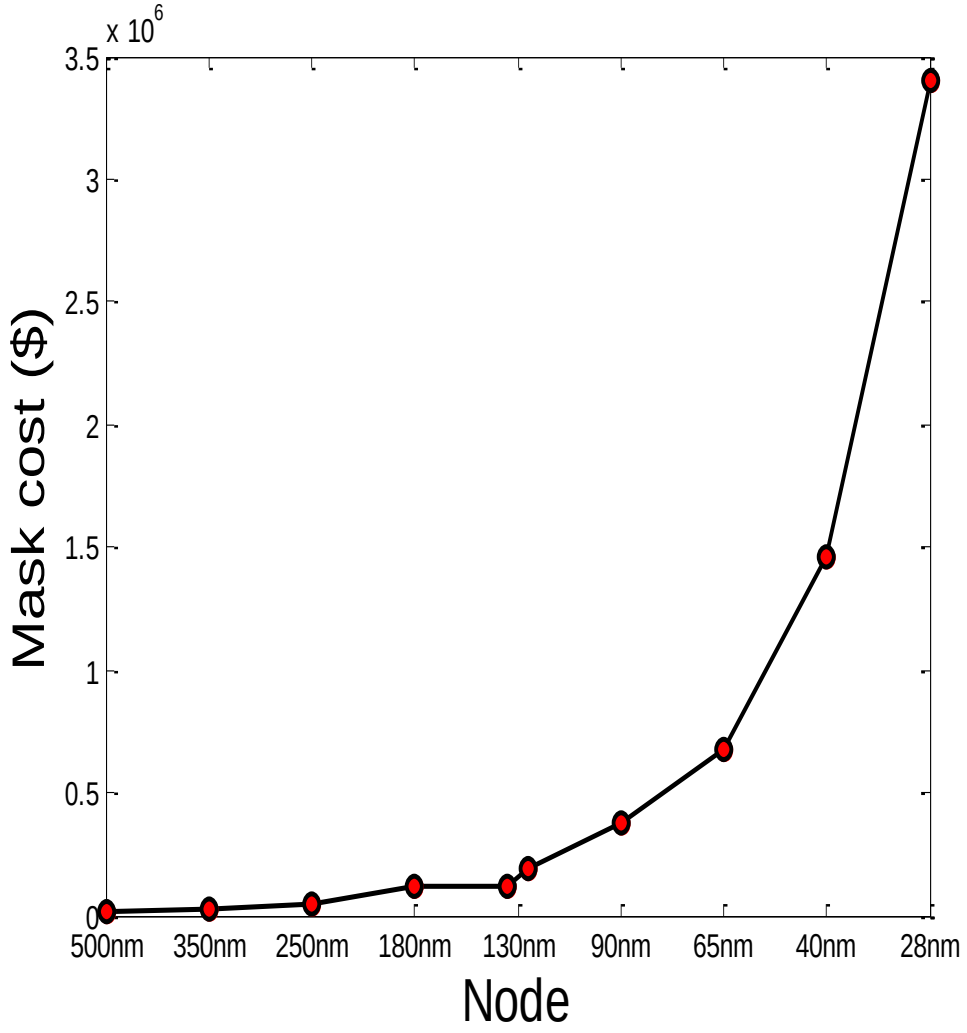
Slowing any of these drivers slows scaling

Wafer processing cost per Transistor



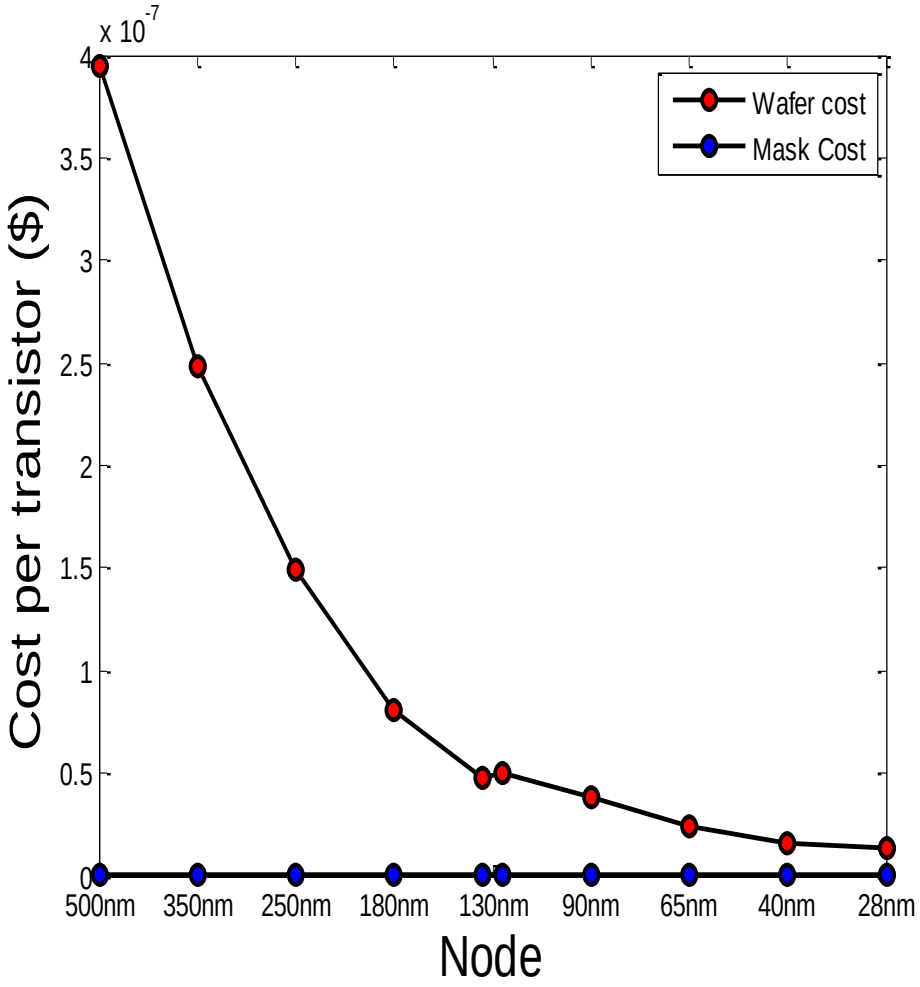
Ideally, between 2 nodes, digital logic reduces in size with a factor of 2

But mask costs from node to node …



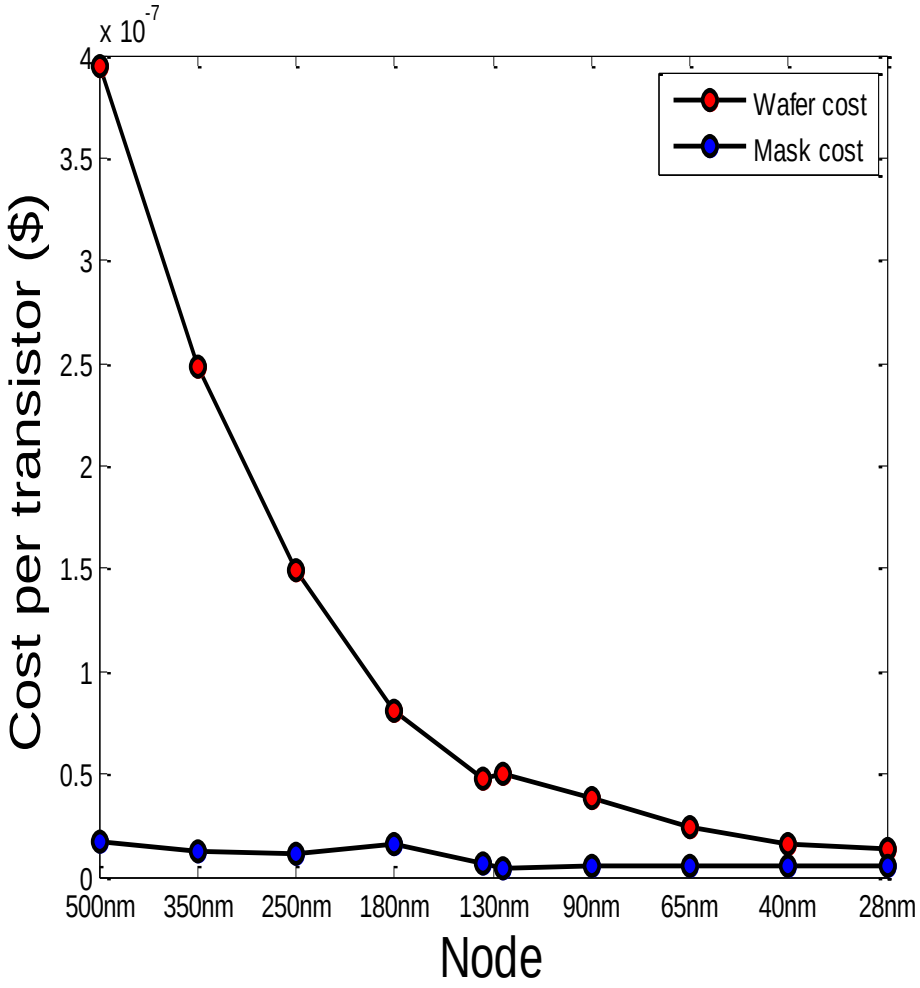
Mask and processing costs

100,000 wafers



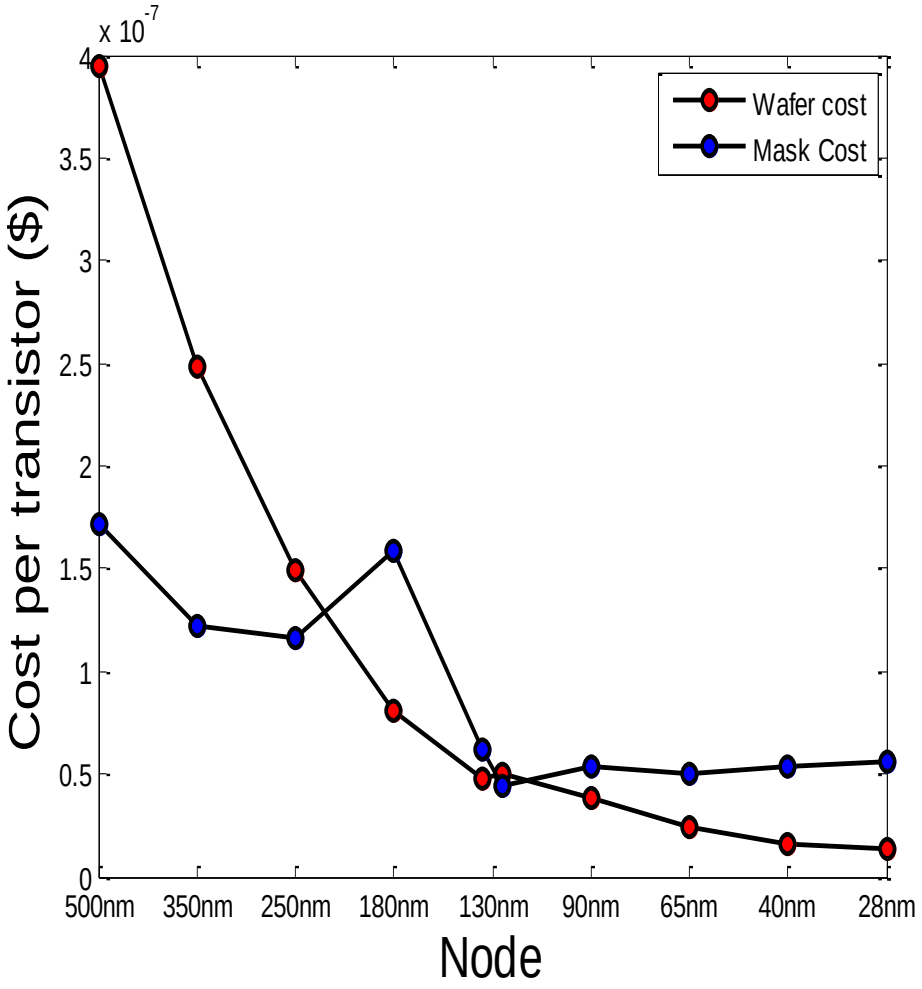
Mask and processing costs

1,000 wafers



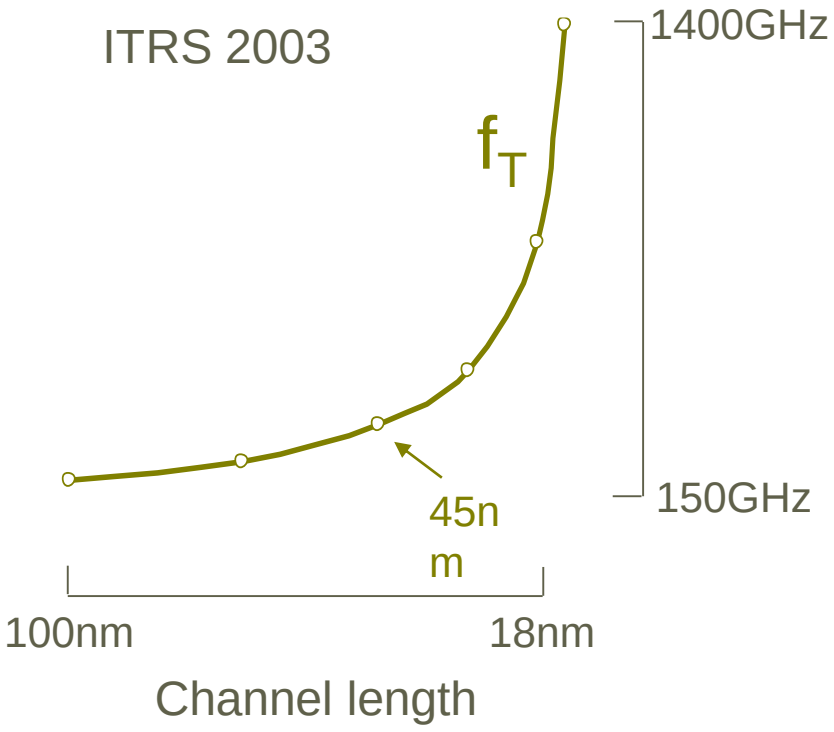
Mask and processing costs

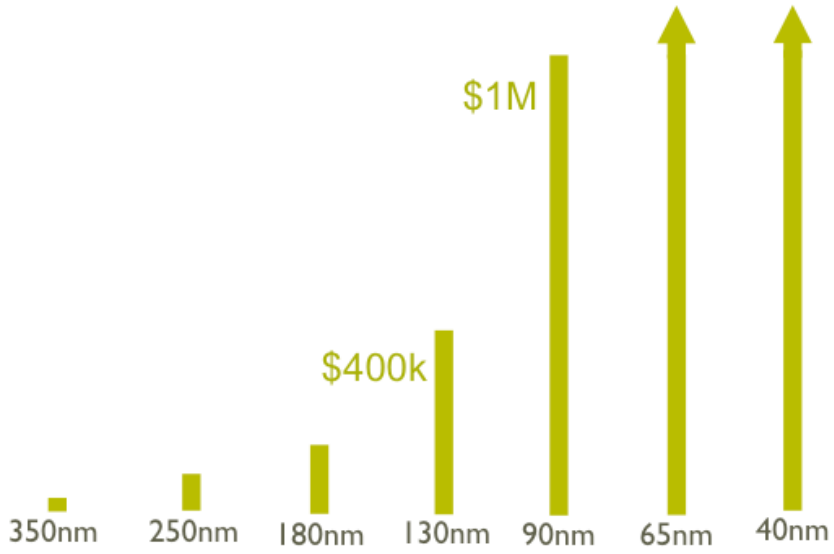
100 wafers



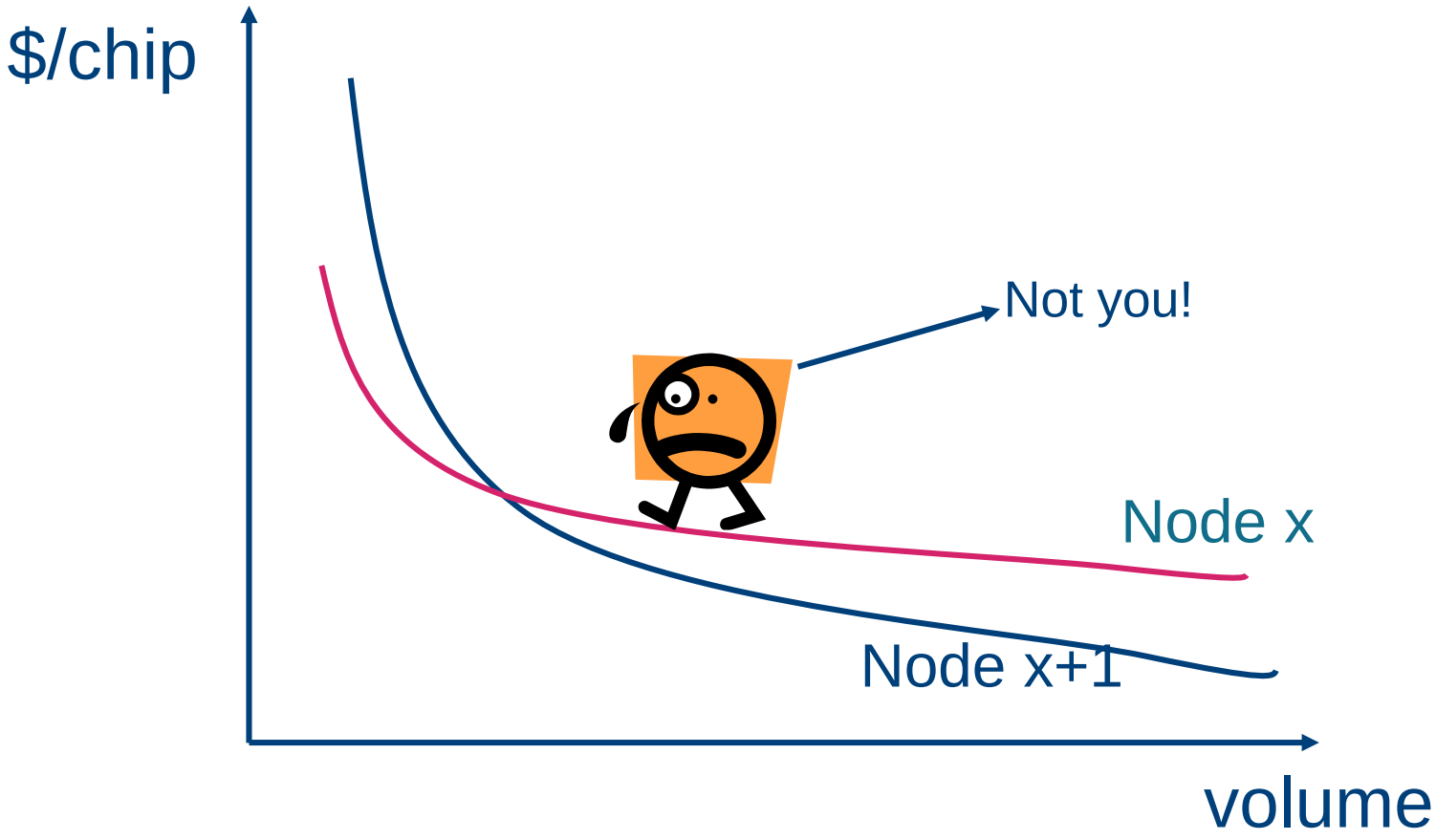
The good news:
CMOS transistors become ever faster

The bad news:
mask sets ever more expensive

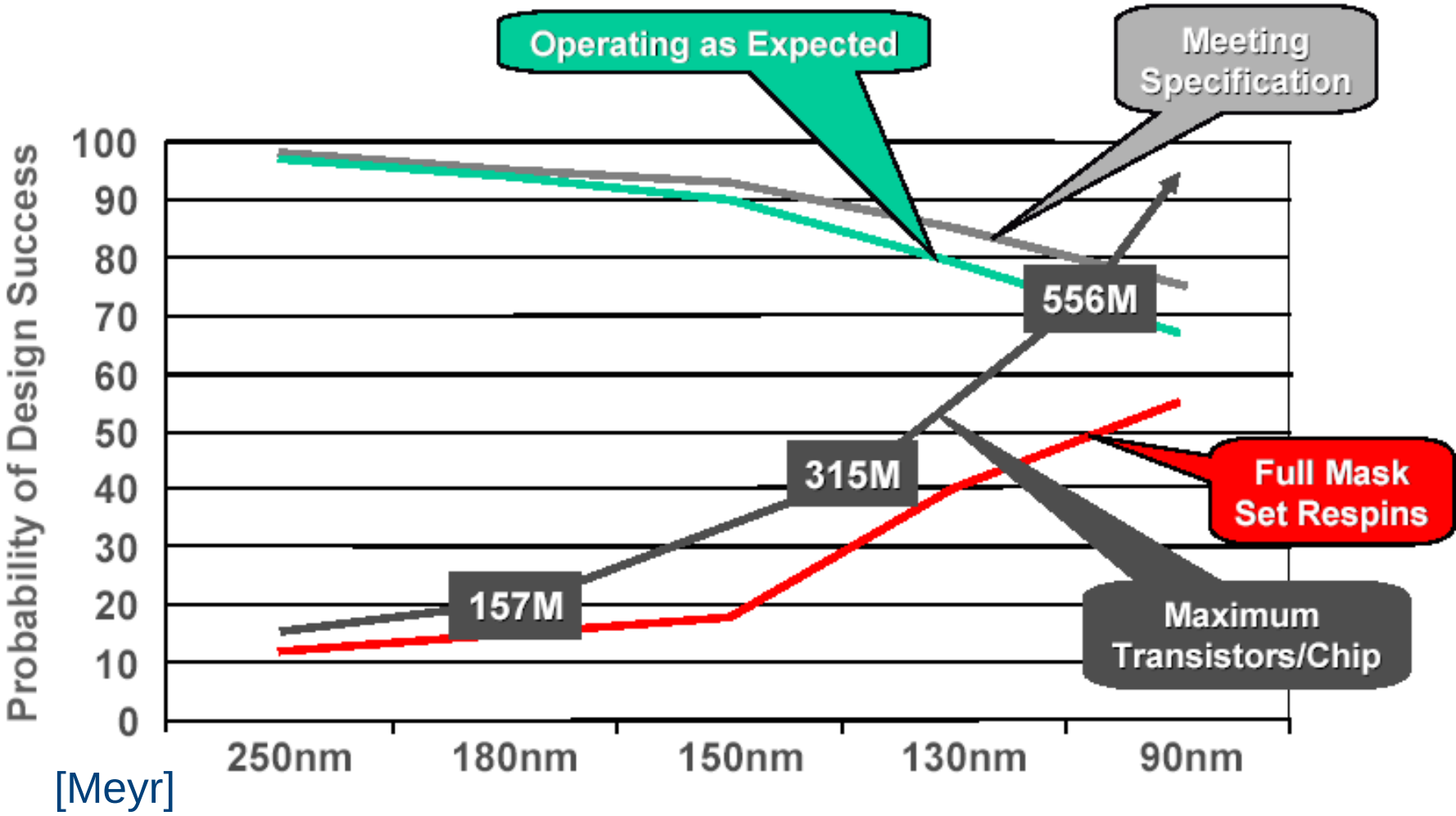




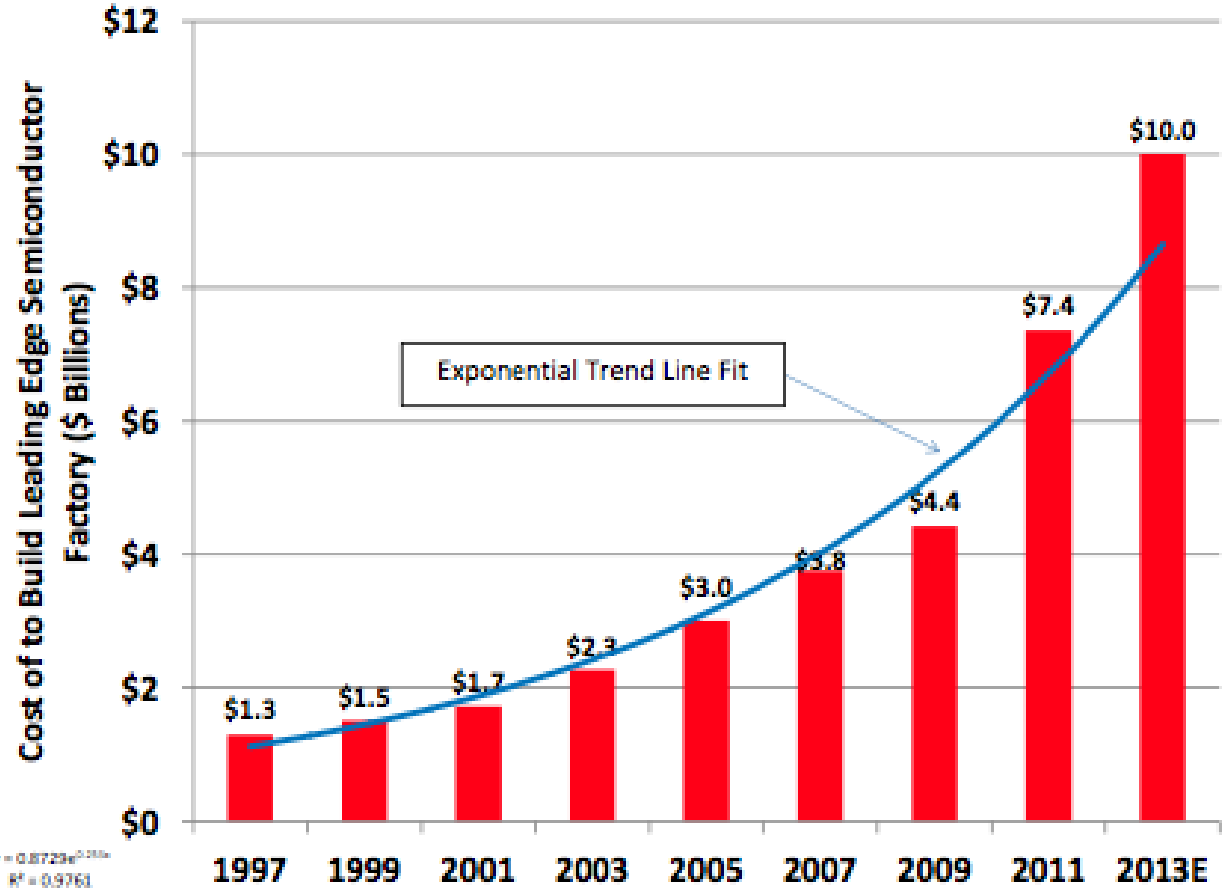
Increasing Non Recurring Engineering (NRE) cost pays off for high volume products (only)



More on chip, also bugs …

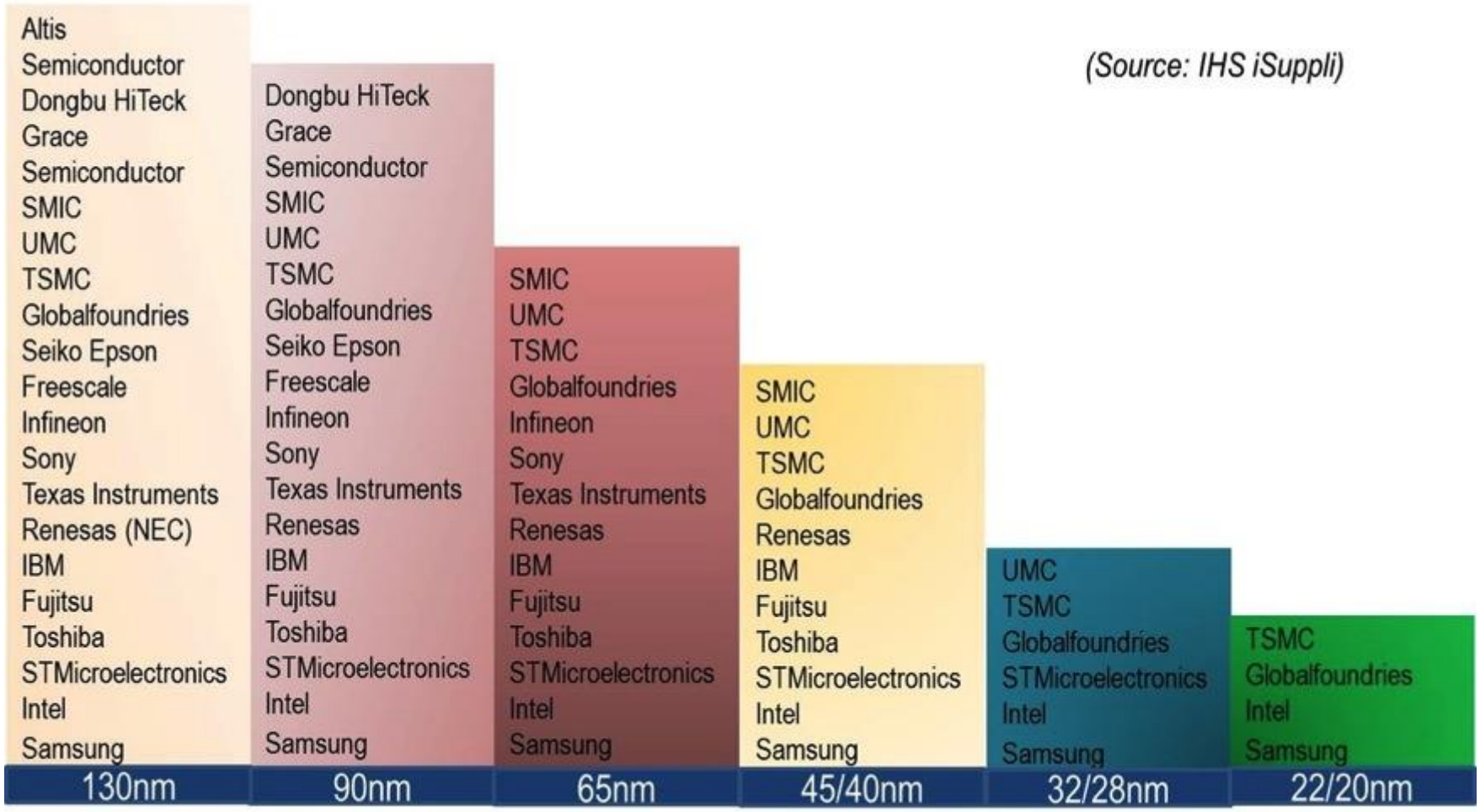


Costs allocating with scaled technology: rising with increased complexity

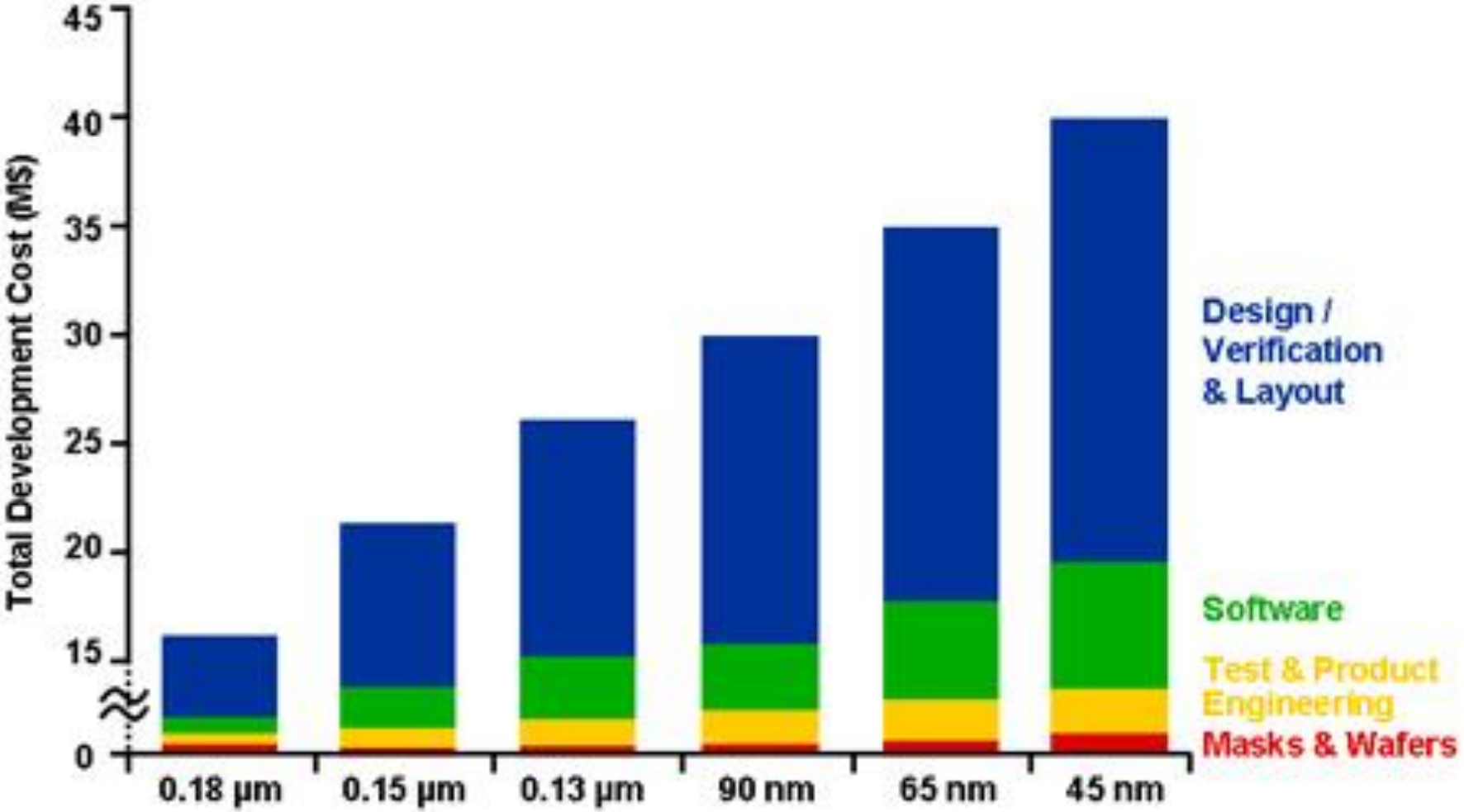


Source: Reports and press releases from Intel, TSMC and Global Foundries

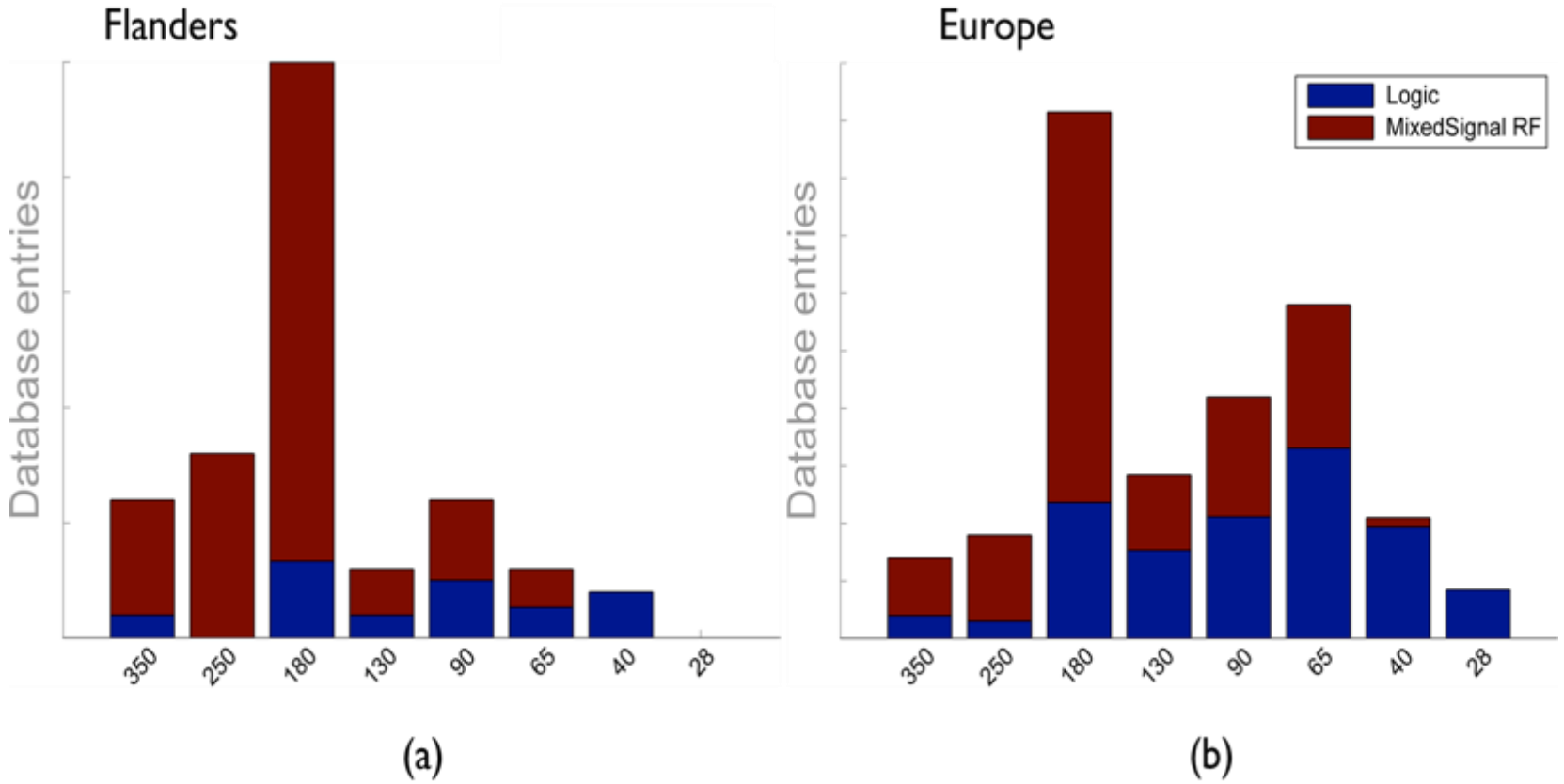
Less Foundries offer newest technologies



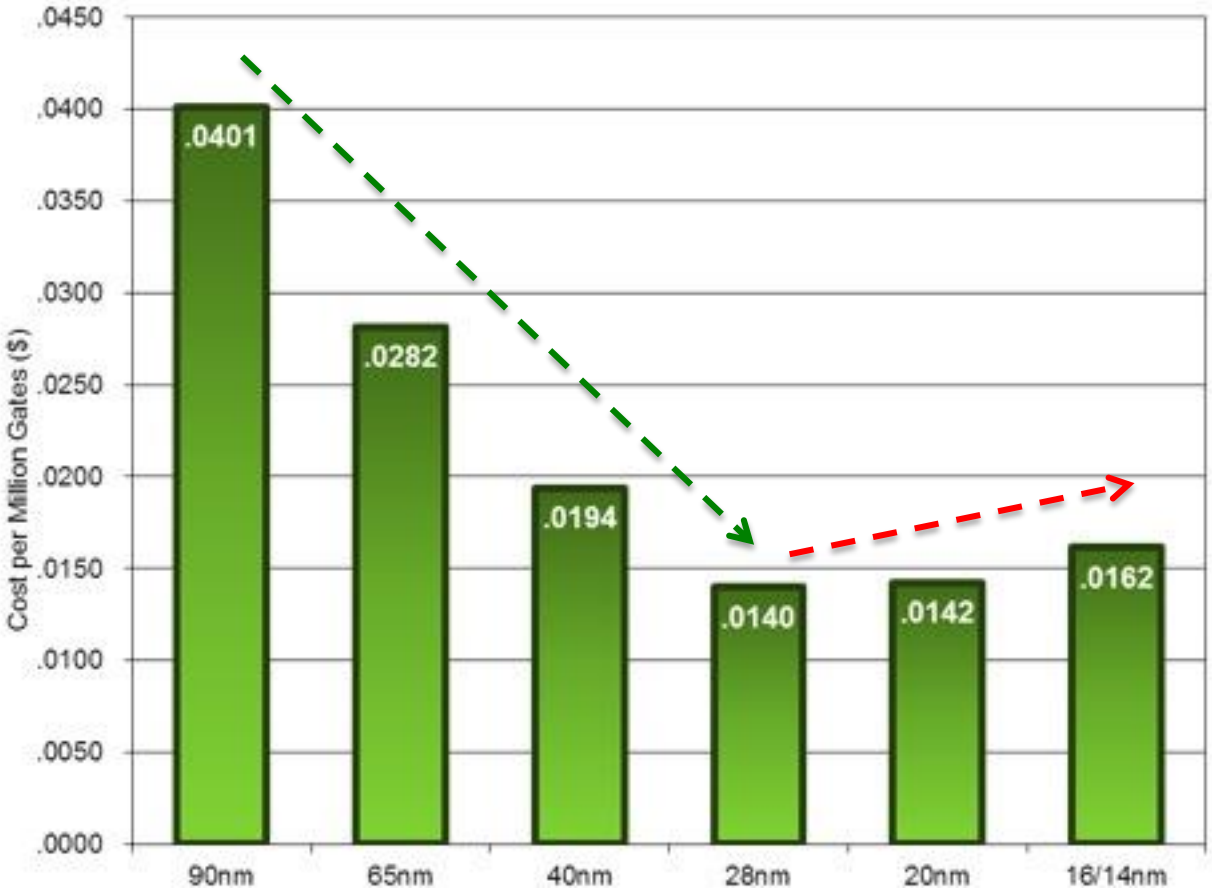
Total chip development cost: a changing image



Node usage by signal type (2015)



Source: imec IC-link data base

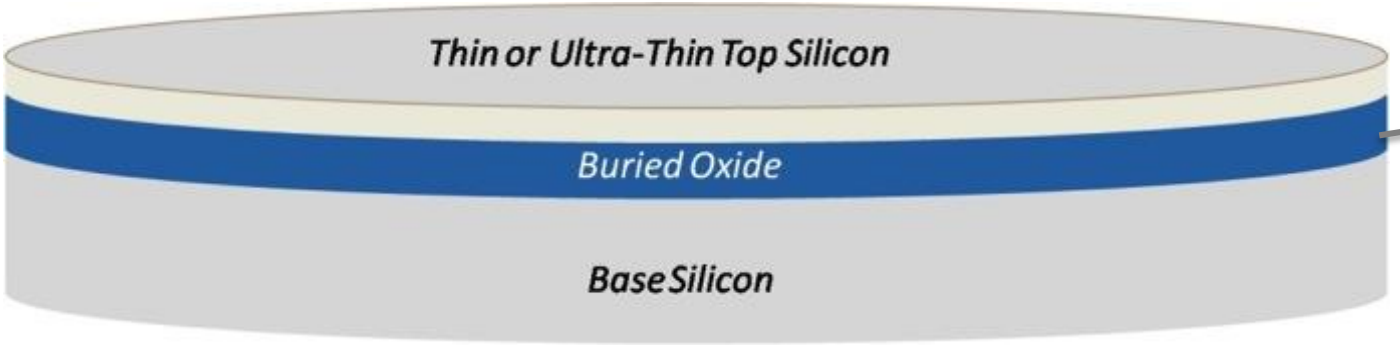


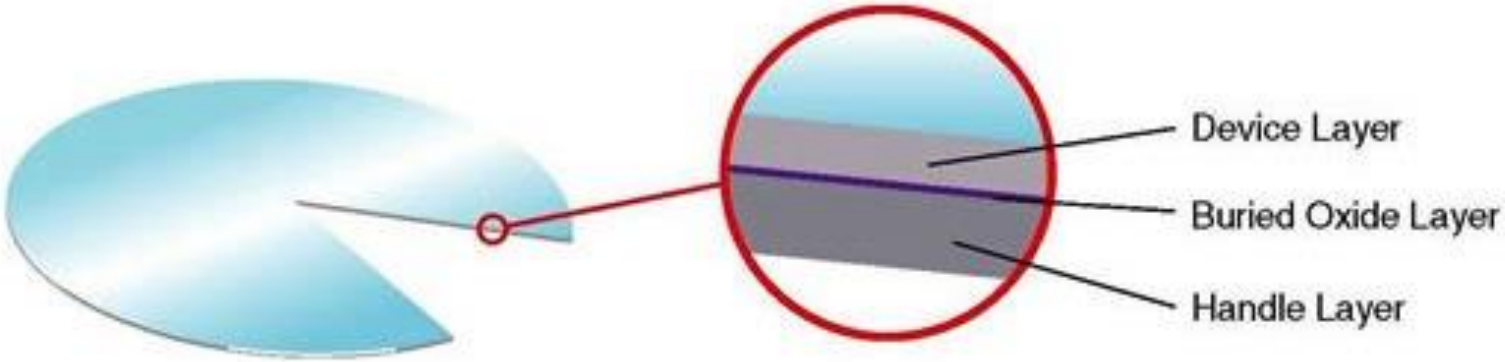
Are we willing to pay a price for …?

Can we get the slope going down again?
Clever people…
yes we can!

SoI technologies: what they are

- Technologies that use a layered silicon–insulator–silicon substrate
- Insulator can be oxide (often) or other material (for example sapphire for improved RF performance)





Logically wafers are more expensive, yet can be compensated by lower processing and/or development costs

Chip projects: from idea to unique product

1. Chip projects: what is to be done?

2. Brief general semiconductor economics

3. Exemplary project cases: bake your chip?

Challenges when doing an ASIC

- The right product idea
- Mass volume production cost
- Development cost must match capitalization
- Find best matching wafer technologies
- Best in class design techniques
- Time to market
- Use trusted partners
  - Excelling in support, also for your volume
  - Reliable foundry and models reduce risk on redesign
  - Use Si proven IP, if possible
- Managing risk

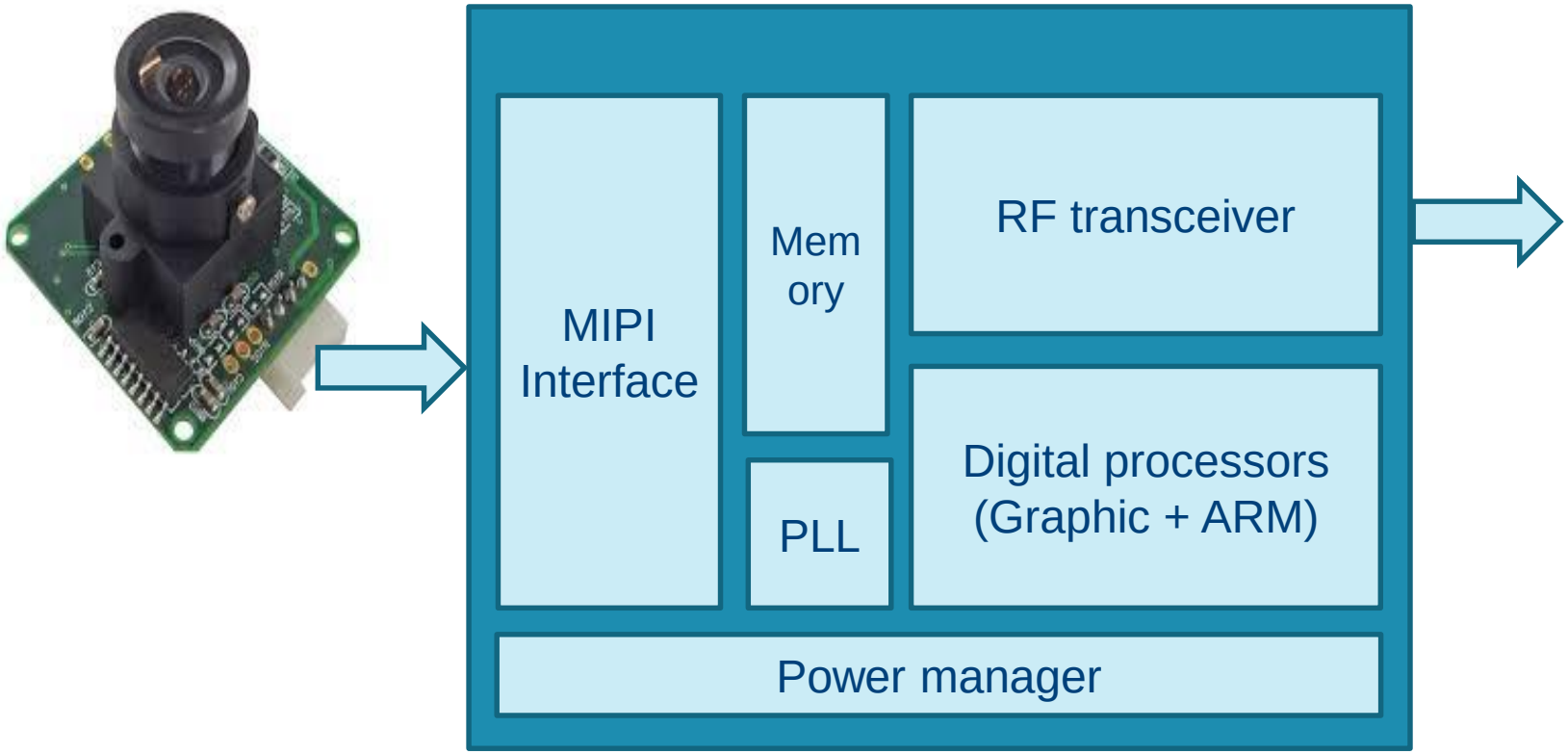
Investment breakdown for ASIC development

1. System level design; chip specification
2. Analog and/or digital design of circuits, + integration (top-level) and final lay-out
3. IP purchase (licensing)
4. Prototyping cost (MPW: Multi-Project Wafer limits processing cost for designs before production)
5. Redesign cost
6. Final mask cost
7. Package design
8. Verification test and production test design
9. Qualification and certification

Example Case 1:
wireless security Camera with data processor

- Product
  - Wireless camera
  - In camera data processing
  - Application: security and people counting in trains, busses and shops
  - Market opportunity: 100 kpieces per annum
  - Product lifetime over 5 years
- Technology needs:
  - MIPI interface
  - On chip data processing; high digital conte
  - Low jitter clock
- Chip cost target 8 $ (expectation)

Block Diagram: mixed-signal IC



Implementation choices made

- Ultra low power technology
  - 65 nm logic technology
  - Embedded SRAM memory

- Selected IP’s to be integrated: generic solutions do not contribute to the uniqueness of your product, costs can be saved if they are licensed (black box)
  - MIPI receiver
  - PLL
  - ARM core
  - Graphics processor

Item	Cost
MIPI interface	300 k$
ARM core	300 k$
GPU	500 k$
RF transceiver (developed by a local design-house)	1000 k$
Small IP: precision PLL, compilers and other some small stuff	200 k$
Digital layout	350 k$
2 MPW runs including assembly (65nm)	175 k$
Full mask-set including corner lot (65 nm including options and engineering lot)	900 k$
Package development (flipchip BGA)	85 k$
Test program development	175k$
Qualification	100 k$
Total	4.085 M$

Ball park numbers only

IP

dedicated labor

IP

dedicated labor

processing

dedicated labor

development only

Product cost (per sample)

Item	Cost
Silicon	4.25 $
Package cost (BGA)	3.00 $
Test cost	0.50 $
Total not including operation fee	7.75 $

Ball park numbers only

Example Case 2:
ASIC to drive Wired LED panels

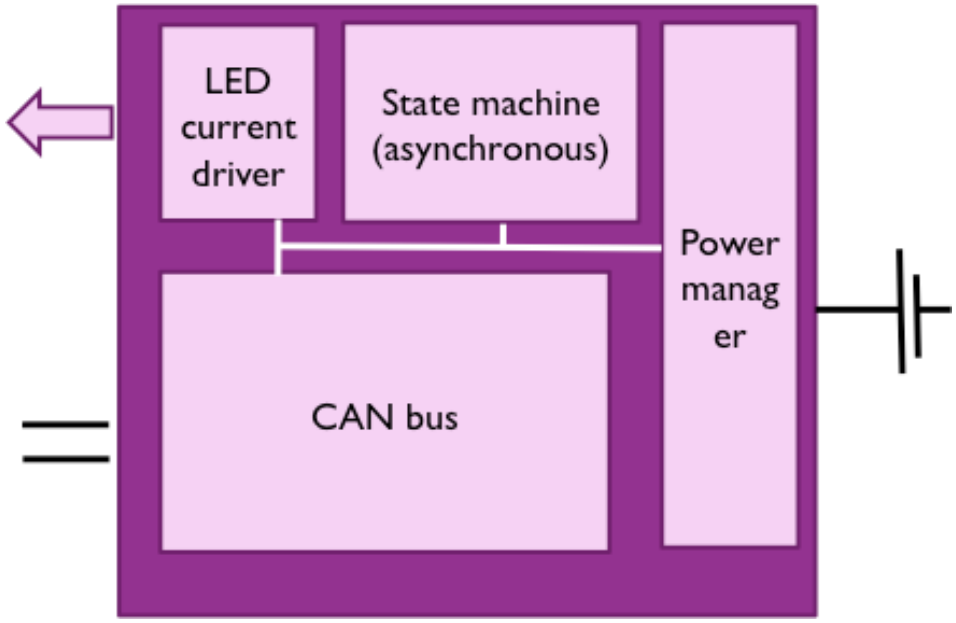
- Product
  - Chip for communication of text updates between LED panels
  - Application: industrial
  - Market opportunity: 20k pieces per annum
  - Product lifetime 10 years

- Technology needs:
  - Precision analog
  - High voltage power supply and current drivers
  - CAN bus transceiver
- Chip cost target 5$ (expectation)

Block scheme and Implementation choices

- Technology: Specialty 0.18 um HV technology

- Implementation
  - Fully integrated CAN bus transceiver
  - Integrated DC / DC converter
  - Fully integrated LED driver for 64 LEDs in strings
  - State-machine for data decoding
  - Memory for data storage



(serial databus)

Typical Development cost
(Ball park numbers only)

Item	Cost
Chip design incl. redesign (outsourced to local design house)	450 k$
Digital design and layout	30 k$
1 MPW run including assembly (0.18um BCD techno)	30 k$
Full mask-set including corner lot (0.18 um BCD MLM)	85 k$
Package development (QFN 88)	5 k$
Test program development	100k$
Industrialization	100 k$
Total	800 k$

Resulting Typical product cost breakdown
(per sample)

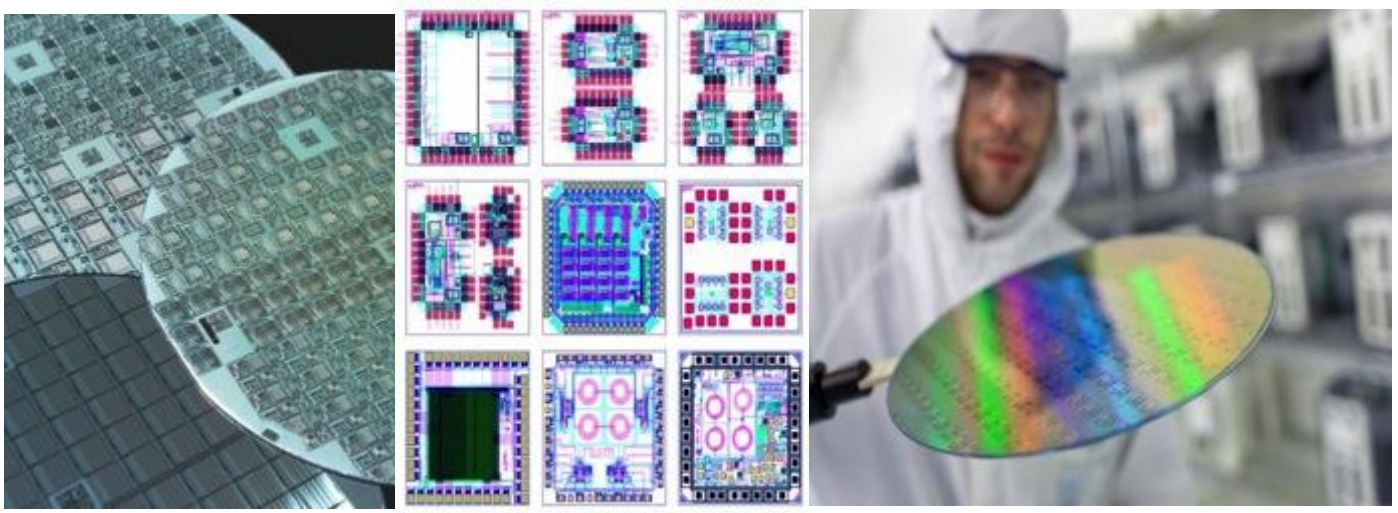
Item	Cost
Silicon cost	0.40 $
Assembly cost	0.40 $
Test cost (for small production volumes)	0. 30 $
Logistics cost	0.05 $
Total not including operation fee	1.15 $

The height of operation fee depends on:

- The services the ASIC provider offers
- The risk they take
- The business model

Ball park numbers only

Manufacturing and supply chain: A chip is made by many parties



- Design house: analog, digital
- IP vendor: digital core, memories, interfaces etc.
- Foundry
- Assembly house
- Test house
- Qualification
- Supply chain

Using the right partner reduces risk!

… Your world world has changed

Chips

ASICs

Wafers

Standard cell

Die

Gate

Masks

Chips can be too crunchy!

Chips

ASICs

Wafers

Standard cell

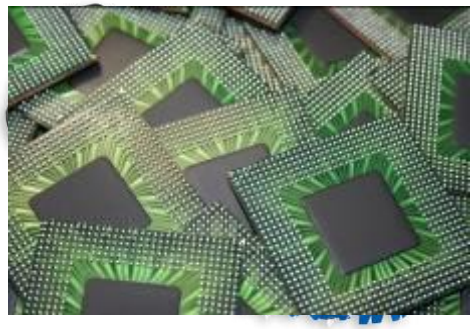
Die

Gate

Masks

You can’t wear all asics

Chips



ASICs

Wafers

Standard cell

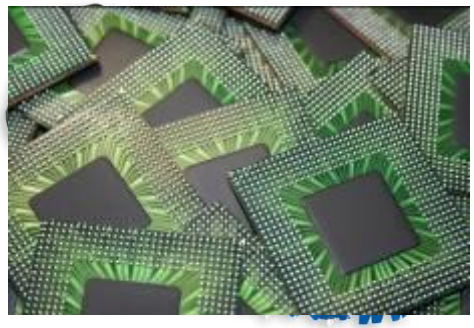
Die

Gate

Masks

You can’t eat all wafers

Chips



ASICs

Wafers

Standard cell

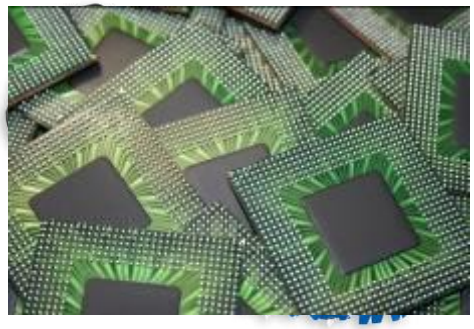
Die

Gate

Masks

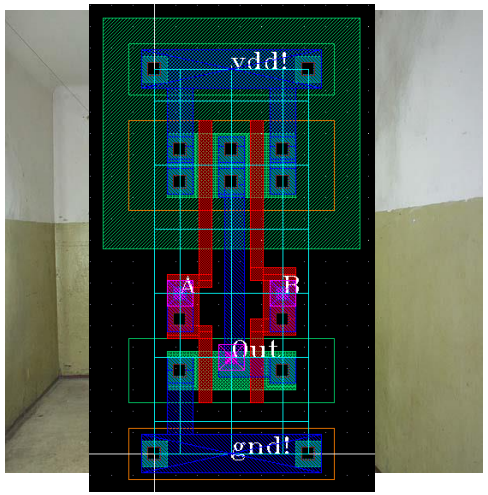
You can’t be locked in some cells

Chips



ASICs

Wafers



Standard cell

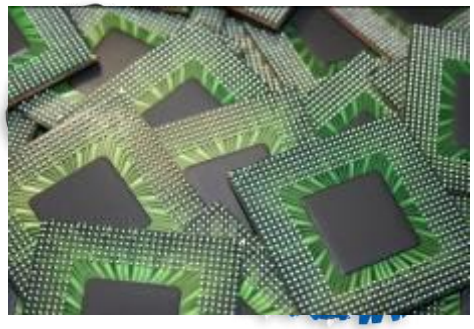
Die

Gate

Masks

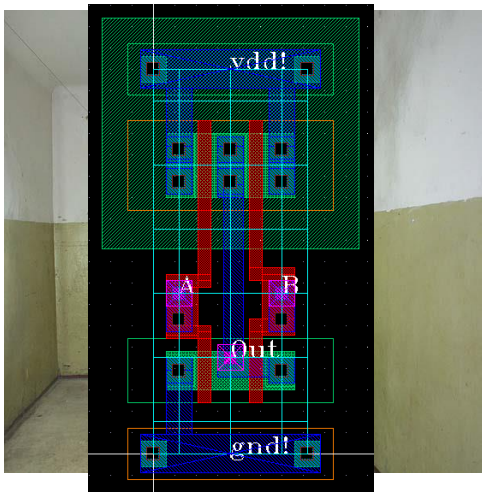
Dies are not only for gaming

Chips



ASICs

Wafers



Standard cell

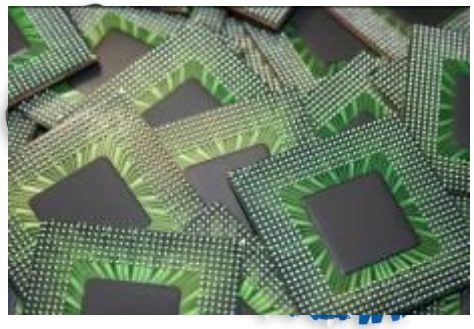
Die

Gate

Masks

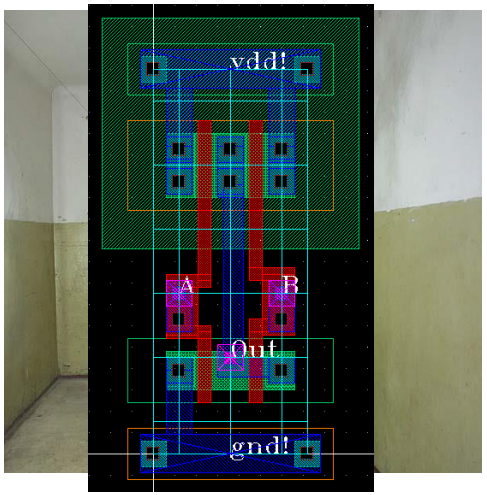
Gates can have a digital function

Chips



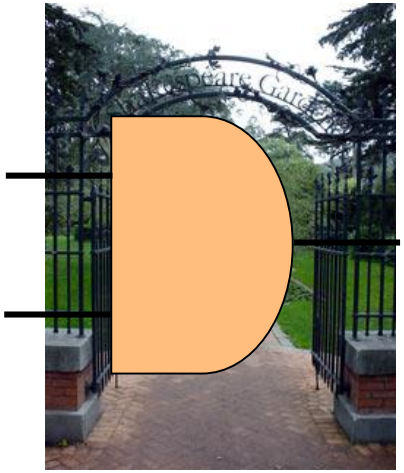
ASICs

Wafers



Standard cell

Die

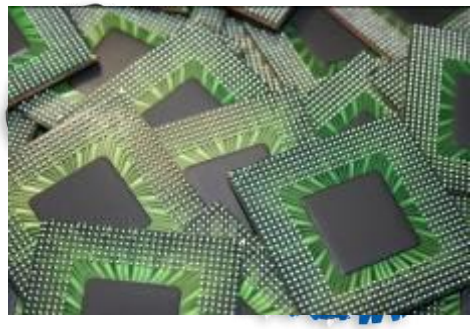


Gate

Masks

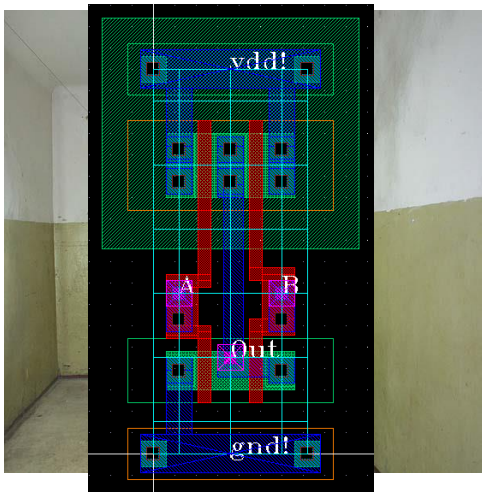
Some masks can’t be worn

Chips



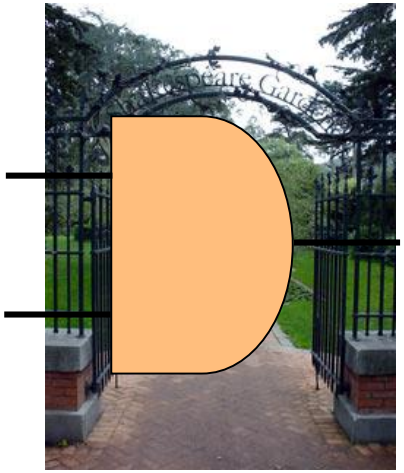
ASICs

Wafers

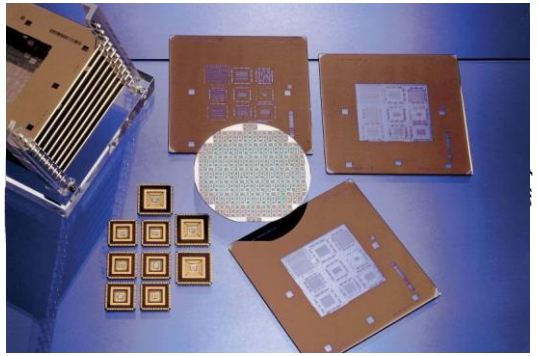


Standard cell

Die



Gate



Masks

Key conclusions before starting your own chip

1. Many successful products are based on legacy technologies: Best match between NRE and product cost

2. For low volumes advanced technologies should only be selected for functional requirements

Die yield: may have significant cost impact

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

Note! More complex chips can become much more costly:

1. Less dies per wafer (round wafers, square/rectangular dies: waste, larger for larger dies)
2. Die yield will decrease

Wishing you: ….!

friends like Steven Redant & Bas Dorren
(former colleagues @ imec)