Budapest University of Technology and Economics
Department of Electron Devices

Technology of IT Devices

Lecture 6

Digital Design 2.

Digital Design

- Last lecture was about
  - Abstraction levels, design flow, HDLs
- SystemC
- Verification
- Simulation
- Physical design (very shortly)
- IPs (Semiconductor IPs)

Budapest University of Technology and Economics
Department of Electron Devices

SystemC

History

Basic terms

Examples

SystemC

- C++ class library for digital design
  - Capable of classic hardware description
    - It can describe the operation of hardware on the “bit and latency level”
  - Capable of high level design
- ESL: Electronic System Level design
  - The design can be clear
  - Hardware and software can be designed together (HW-SW co-design)
  - Interfaces between the components can be modeled
- We can describe hardware in a C++ development tool
  - Parallel operation, timing, latency, ports, bits
  - Simulation of compiled code is very fast
  - It can be an input for high level synthesis

History of SystemC

- C++ is frequently used for hardware modelling and simulation
- **An Efficient Implementation of Reactivity for Modeling Hardware in the Scenic Design Environment (1997)**
- Standardized
  - By Synopsys, first version in 1999
- Open SystemC Initiative
  - Each of the three EDA (Electronic Design Automation) companies (Cadence, Mentor Graphics, Synopsys) take part in it
  - IEEE 1666–2011 standard
  - Open source
  - http://www.accellera.org/downloads/standards/systemc

Basic terms

- Module
  - The basic building block of SystemC
  - A module realizes a given function and can contain other modules and processes (VHDL entity or Verilog module)
  - A properly derived C++ class
- Process
  - Describes the function. Method of the class representing the module.
    - equivalent of VHDL process or Verilog always-block
- Port
  - Connect a logic signal
    - It has a direction (a variable in C does not)

NAND gate - example

```
#include <systemc.h>

SC_MODULE (nand2) // a C macro ("class nand2 : public sc_module")
{
        sc_in<bool>  a, b; // input ports
        sc_out<bool> y;    // output port

        void nand2_method() // description of operation
        {
                y.write( ! ( a.read() && b.read() )); // y= ! (a && b)
        }


        SC_CTOR (nand2) // constructor for nand2
        {
                SC_METHOD (nand2_method); // this function descibes the operation
                sensitive << a << b; // sensitivity list
        }
};
```

D-flipflop - Example

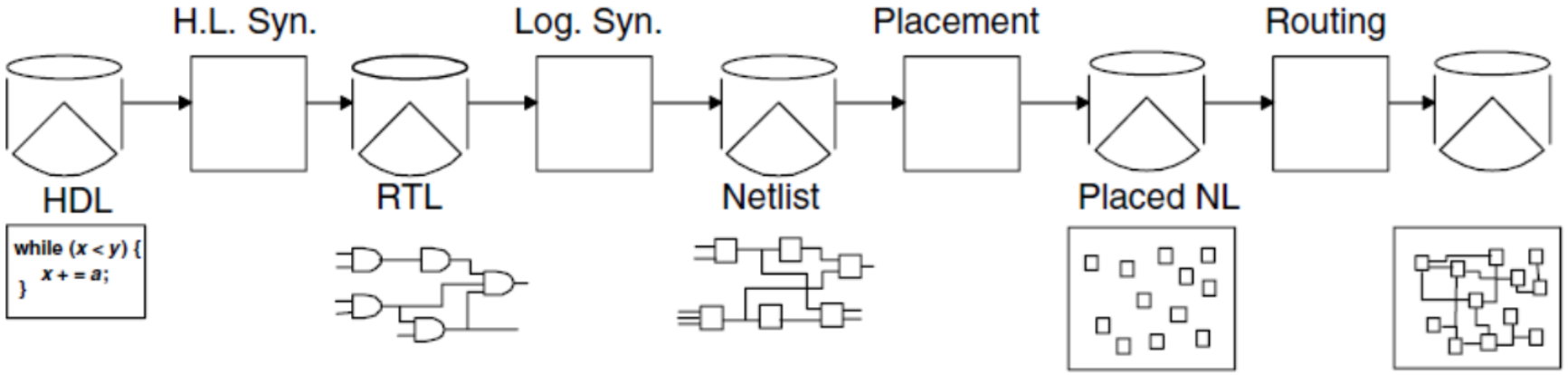
```
#include <systemc.h>

SC_MODULE (dff)
{
    sc_in<bool>  clk;
    sc_in<bool>  din;
    sc_out<bool> dout;

    void dff_method()
    {
         dout= din; // dout.write( din.read())
    }

    SC_CTOR (dff)
    {
        SC_METHOD (dff_method);
        sensitive_pos << clk;
    }
};
```

Simulation kernel

- SystemC contains an embedded event driven simulation kernel
- The excitations are in the main function
  - It is more elegant to create a testbench (like in HDLs) and only define the interconnections in the main function
- The built code is a simulator
  - This results in a high simulation speed
- The whole C/C++ toolkit and system libraries are available for analyzing the output of the simulation
  - It can be handled in modern HDLs as well, but it is not easy (VPI/VHPI)
  - The possibilities of even std::cout are wider than in the case of a simple waveform
  - Here you can find a graphic card design solution based on SDL
    - http://www.eet.bme.hu/~czirkos/icterv/syscvideo-0.6.zip

Verification and simulation

Digital design flow



- Stepping from a higher to a lower abstraction level can be done by automatic or manual synthesis
- We have to make sure that the lower level representation is completely equivalent to the higher level one
  - It is essential in the case of manual synthesis
- This is called verification
- Tools are simulations, and other checker software

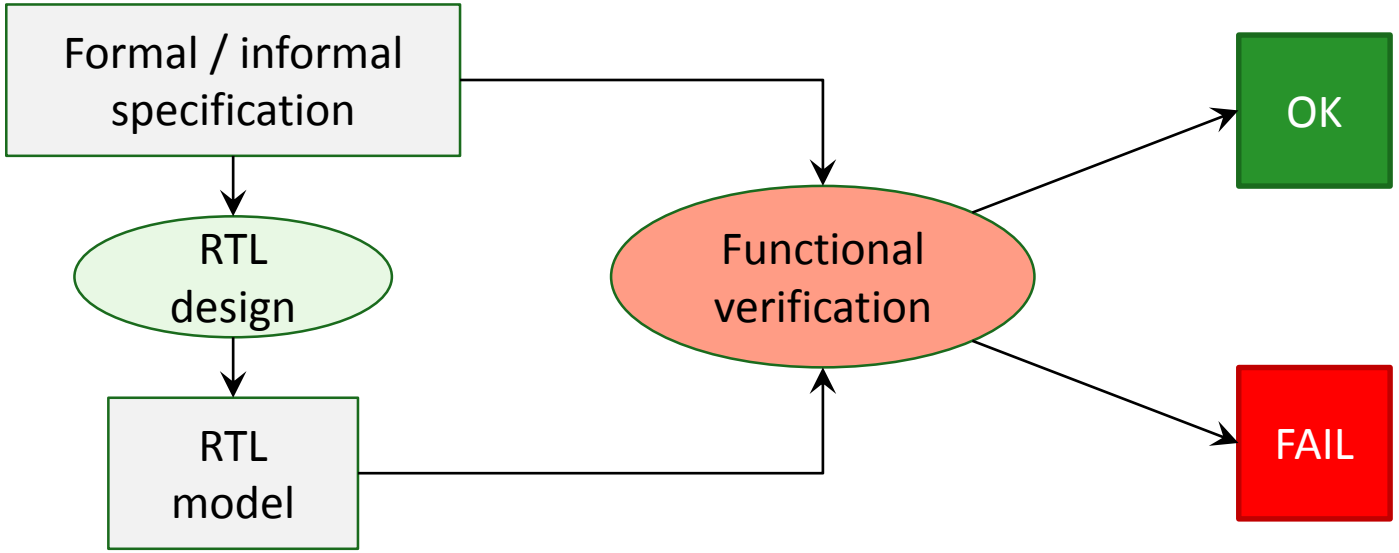
Timing and performance on different abstraction levels

Level	Timing and performance
System level	Requirements defined for the whole system
Algorithmic Level	Predicted by the designer
RTL	Predicted by software
Logic level	Gates: exact; Wiring: predicted
Circuit level	Exact (simulated)

- As the abstraction level decreases, the information regarding the timing and performance is more and more precise.
- The propagation delay caused by interconnections can be determined during physical design
  - The design process is iterative
  - The calculated delay and performance parameters can be “back annotated” to a higher level of abstraction (back annotation).

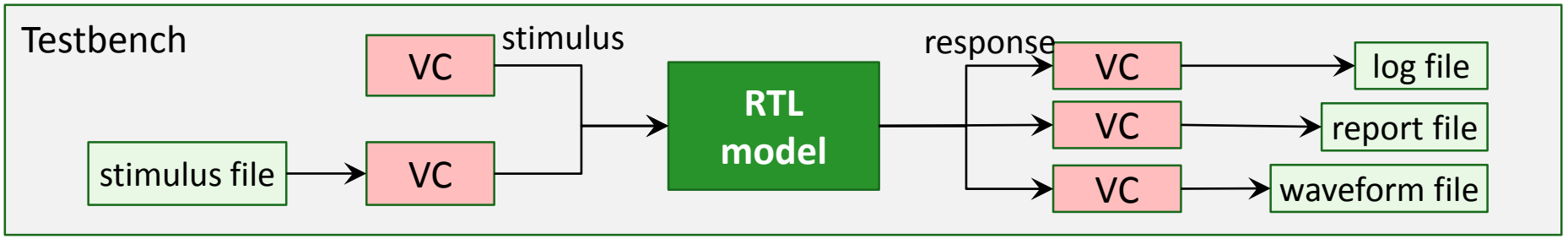
Functional verification

- Checking (proving) that the **HDL model** fulfills the functional specification
- **HDL model**: Checking the validity of the RTL model (not the result of the RTL synthesis)
- Functional verification: Only the functions are verified (delays, power consumption, heating issues, etc. are not)



Testbench

- The testbench provides the input data (stimulus), and observes the outputs
- Elements of a test environment
  - **"Testbench"**: It models the **circuit environment** of the Design Under Verification (*DUV)*
    - Instantiates the ***DUV***
    - Instantiates the **verification components**
  - **Verification components**
    - They generate the input signals and data
    - Observe and log the output values



Relevance of functional verification

- Increasing complexity of the specification results in increasing complexity of the testbench
  - The complexity of the testbench is growing faster than the complexity of RTL design
- Complex IPs *(Intellectual Property)*, in the case of SoC *(System-on-Chip)*
  - 70% of the design effort is related to verification
  - There are two times more verification engineers than design engineers
  - 80% of the codebase is connected to verification
    - (testbench, verification components)

Creating the stimulus

- **Directed** test
  - It can only reveal errors which the designer can think of.
  - The number of tests is increasing with the complexity of HDL model
- **Random** test
  - The probability of revealing hidden errors is higher
  - Time consuming
- **Constrained random** test
  - The verification space is partitioned and random tests are performed inside the partitions.
  - The efficiency of random testing is higher in a limited verification space.
- In practice: directed test (extremes) + constrained random test

Metrics for the efficiency of a test environment

- Code coverage: Quantitative metrics expressing how efficiently a test environment activates the RTL model
  - The test environment is qualified, not the RTL model
  - We only want to get through the RTL model, not the gates realizing the described function.
- Coverage types
  - Statement coverage
  - Branch coverage
  - Condition coverage
  - Toggle coverage
  - FSM coverage

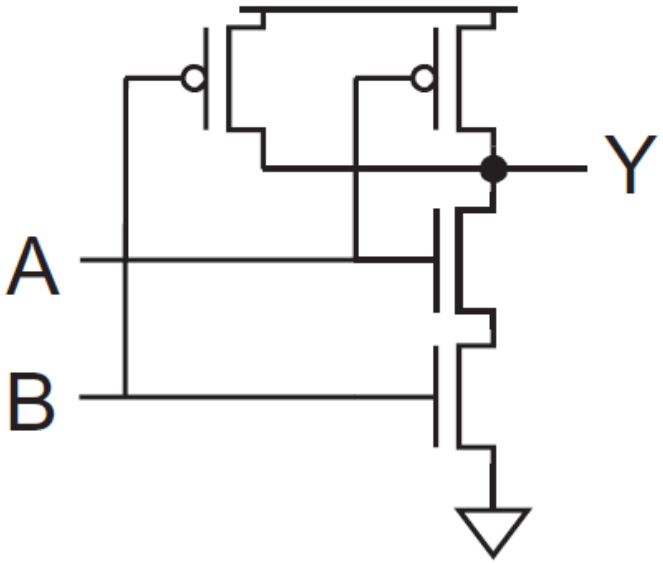
Simulation

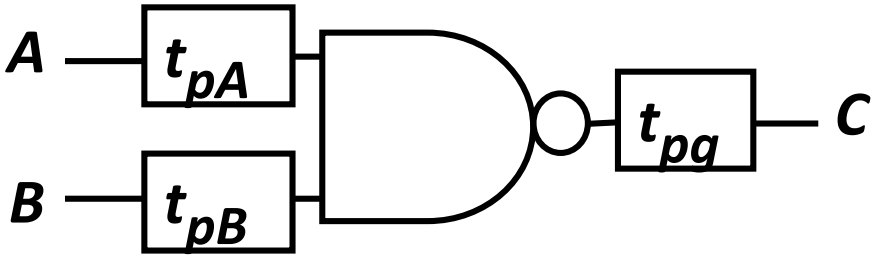
- A tool of verification
  - It is easier to reveal the error using a simulator
- For lower abstraction levels, the simulations are more complicated and have higher resource requirements:
  - System level simulation
  - Logic simulation
  - Circuit simulation
  - Technology simulation
  - Etc.
- By performing low-level simulations, we can create more abstract models of subcircuits. These models can be used to improve the speed of higher level simulations.

Logic simulation

- Logic simulation is an approximation of reality
  - The physical phenomena appear in a very simplified manner.
  - We use logic values (0, 1, X, Z) instead of continuous voltage values
    - Abstract concept of an event: a value changes instantaneously.
    - We only take the effect of an event into consideration, the behavior of the circuit between the events is not important.
    - The simulation can only reveal errors which have a meaning after these simplifications (logical, not circuit-level errors).

Model of a CMOS NAND gate





- t_{pA} and t_{pB} are the delay of the A and B inputs (caused by the interconnections)
- t_{pg}: the delay of the gate

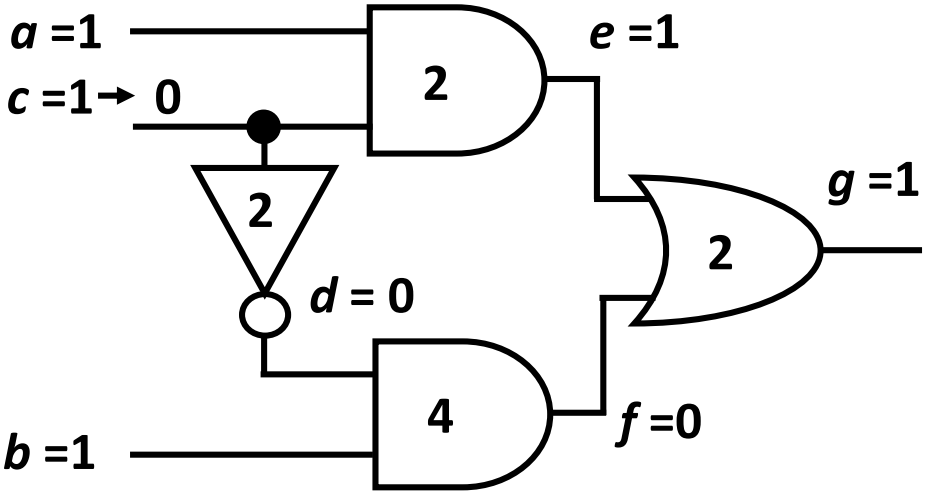
Logic simulation – zero delay

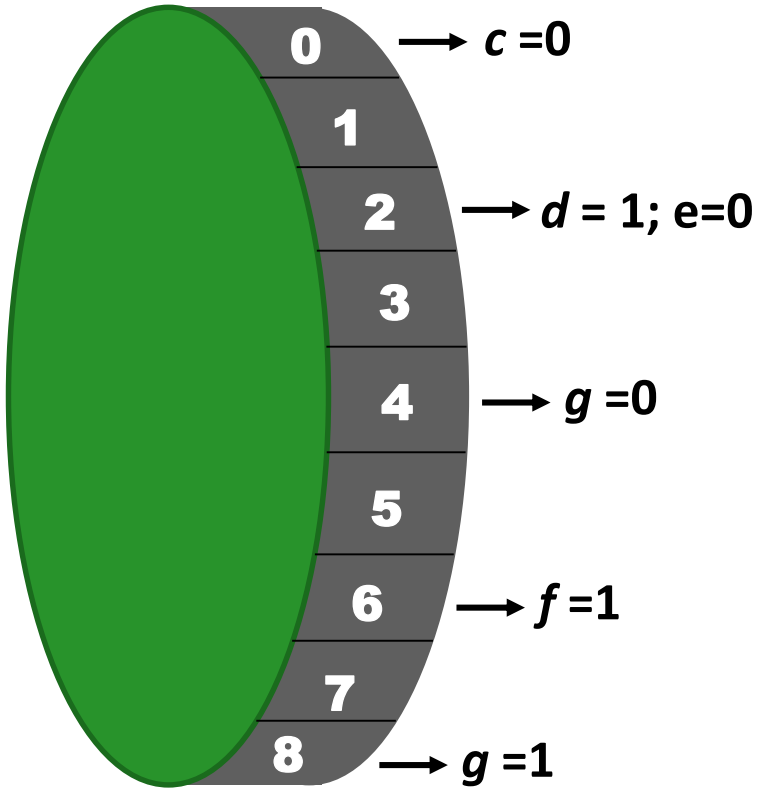
- Without delay (zero delay)
  - On algorithmic or RT level
  - On the output of the combinational network the result appears right after the input change
  - In synchronous logic networks the new values appears right after the active edge of the clock signal
- Practically, the solution of Boolean equations
  - Fast, but we won't get any information about the delays
  - The logic operation can be checked in this way

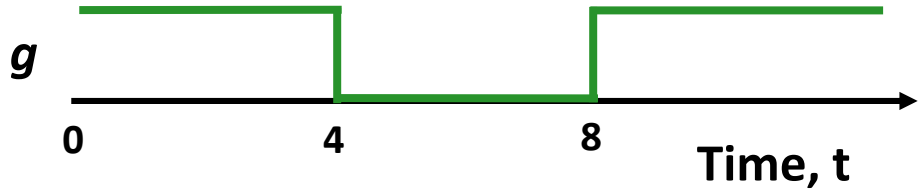
Event driven logic simulation

- Event driven logic simulation
  - Only the effects of changes should be analyzed.
  - It is usually less than 10% of the whole network.
- An event is any change on the input of a gate
- This event has to be evaluated
  - If an input of a gate changes, it generates another event, which has to be inserted into the priority queue with the proper delay
  - If every event is evaluated and the result is done, we can go further
  - This is called the time wheel.

Example

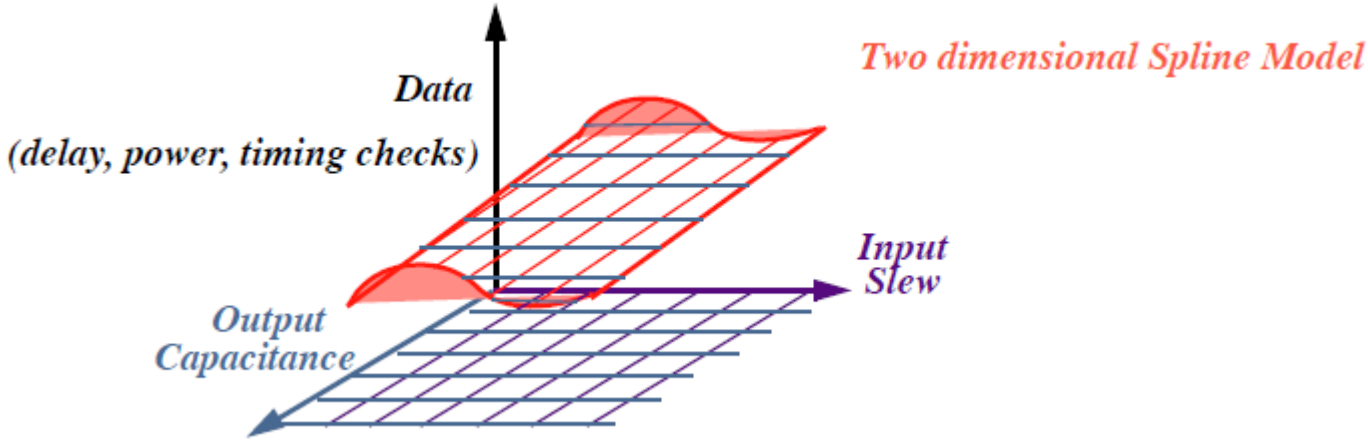






Delay and performance parameters of logic gates

- The dimensions and delay data of standard cells are stored in databases
- The timing depends on the slope of the rising and falling edges of the input signal, and the load connected to the output (number of gates, length of the interconnections)
  - It is created by circuit simulation of the given cell
  - This is a 2D table
  - $t_{PD} = f(t_t, C)$
- It can be calculated after placement



Circuit simulation

- The most detailed simulation
- The precision depends on the component models
- It has a high resource requirement
- Method: nodal analysis
  - We select a node from the circuit which will be the ground (0V), others are unknown.
  - The branch currents are expressed using the nodal potentials
  - The resulting system of equations comprises *n* equations and has *n* unknowns
    - We can solve the equation system with e.g. Newton-Raphson iteration
    - In time-domain cases we have a differential equation system, and it can be solved by a numeric method.

Types of circuit analysis

- DC or OP (operating point) simulation
  - Finding the operating point
- DC transfer characteristic
  - It changes a value of a source or a parameter in the given limits
  - E.g.: the transfer function of an inverter
- Transient
  - Time-domain analysis

Physical design

Physical design

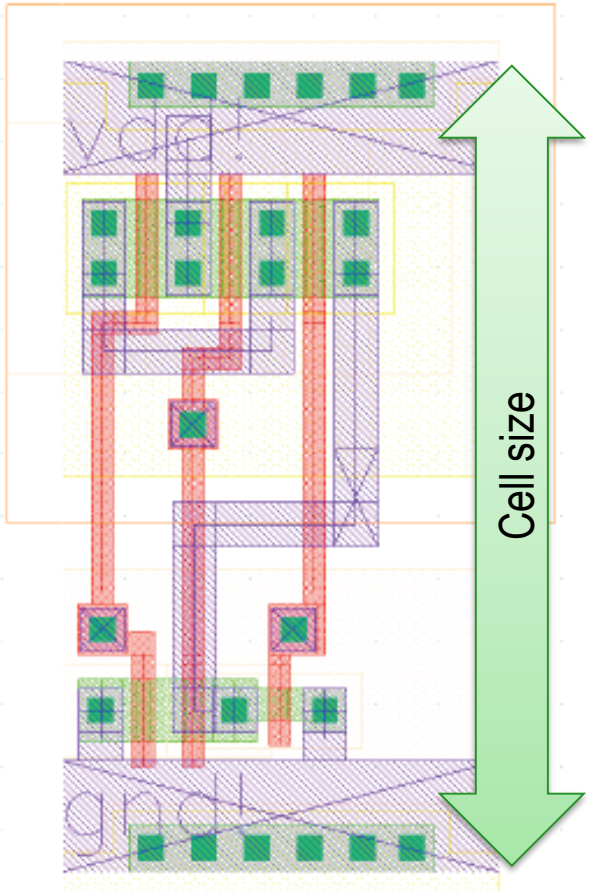
- So far the design steps are independent of the technology
  - But, in fact, on RT level we have to consider the technology, because we have to choose the optimal microarchitecture for the given technology
    - E.g. in FPGAs there are DSP blocks, so implementing multiplication is much easier than in full custom ICs
- Manufacturing methods
  - Full custom on circuit (transistor) level
    - For critical blocks only
  - Standard cell design
    - Using standard (sized) elements
  - Using programmable logic devices
    - FPGA, CPLD etc.
    - Details later…

Cell library

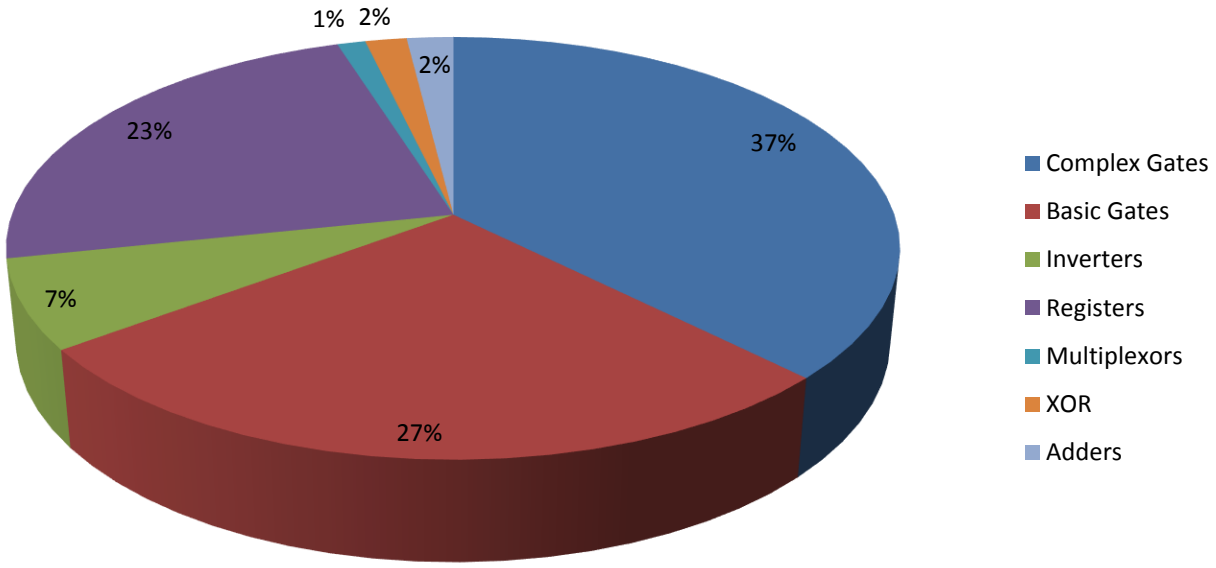
- Collection of basic logic circuits
  - Simple logic gates, complex gates, and flip-flops etc.
  - With different fan-outs
    - The fan-out is higher, the logic gate is faster (same load)
    - But the physical dimensions and the power consumption will be different
- The height of a standard cell is fixed, but the width can differ
  - Cells are placed in cell rows
  - On the top of the cell the power supply wire (rail) can be found
  - At the bottom of the cell the ground wire (rail) can be found
  - Inside the cell only the lowest metal layer is used, so wiring can be placed on higher metal layers

Example: a cell of a complex gate

- The different colors represent the geometric shapes of the wires and the transistors
  - The masks for lithography are based on this plan
- This is called the layout.
  - It contains all the components and wires
- E.g. Layout design of $Y = \overline{AB + C}$ complex gate
  - Blue: Metal 1 layer
  - Red: polysilicon, gate of the transistors (and its interconnections)
  - Green: active zone, (S, D, and the channel)
  - Dark green: contacts between layers



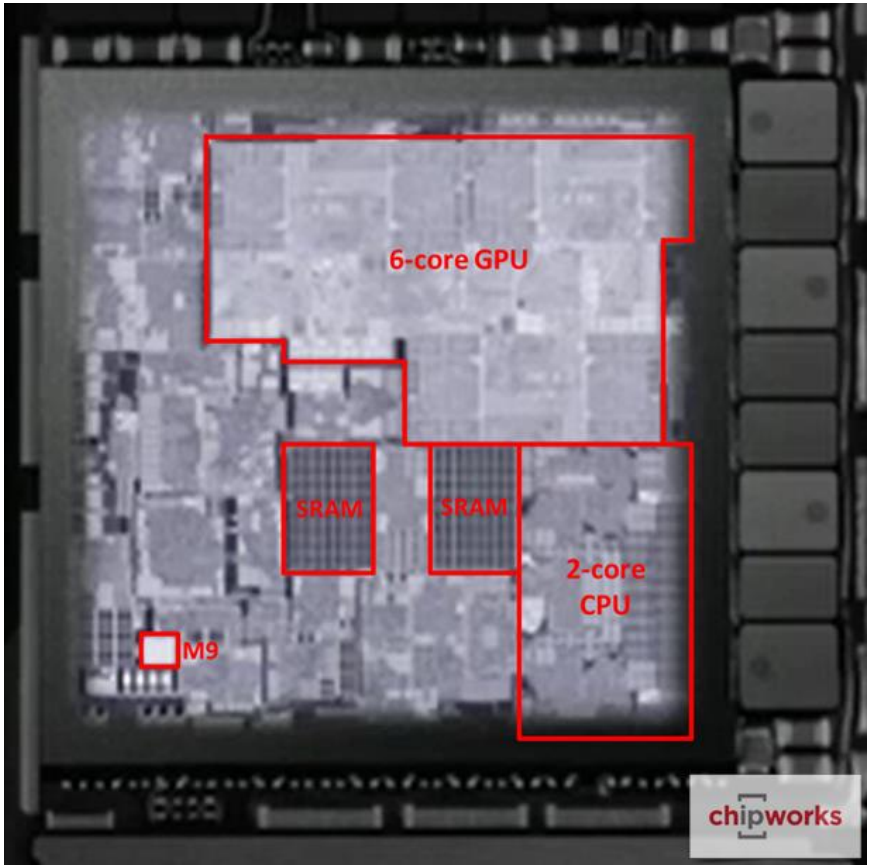
Using a cell library - Example



- 6502 processor core (OpenCores.org)
- AMS 350nmtechnology, logic synthesis: RTL compiler (with default settings)
- 1679 cells,0,14mm^2, 12088 db. transistors

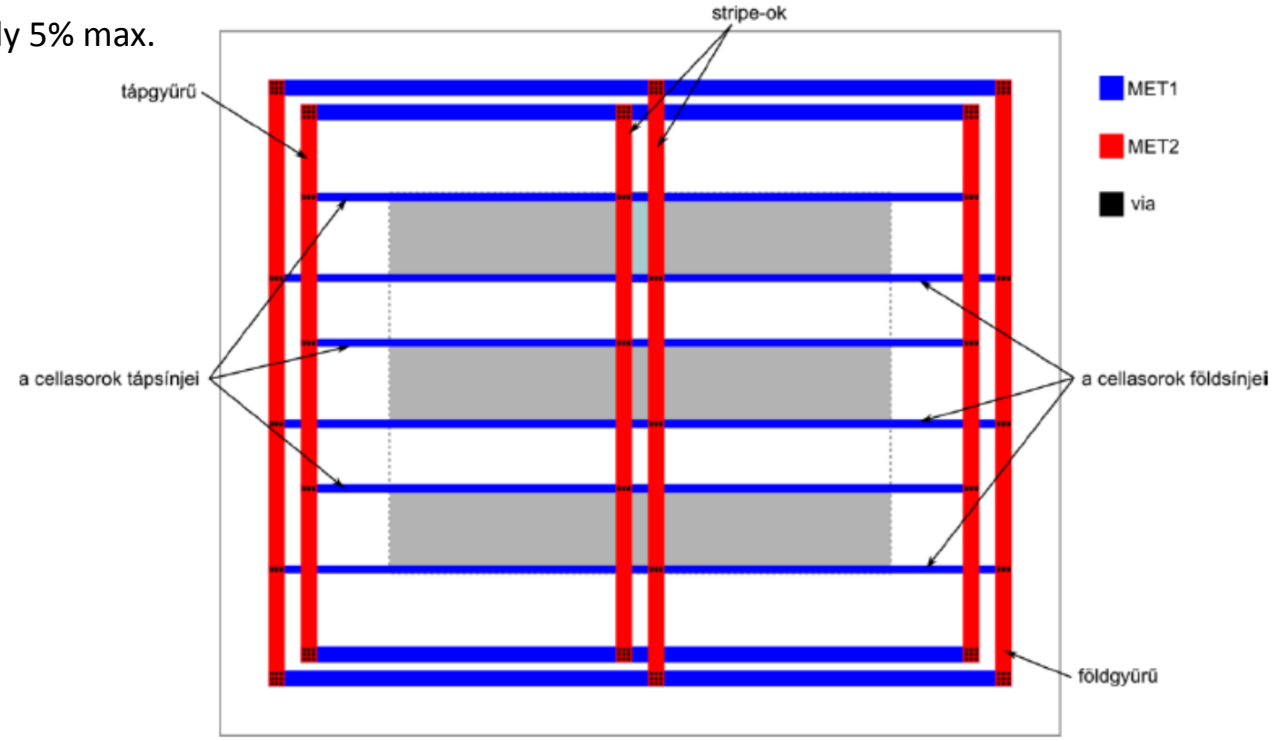
Step 1 – Floorplan

- Placing the main blocks, input and output on the surface of the chip
  - Core: the main block
  - Pad: inputs and outputs
    - The pads surround the core, hence the name: pad-ring.
  - (Apple A9 – a simple floorplan)



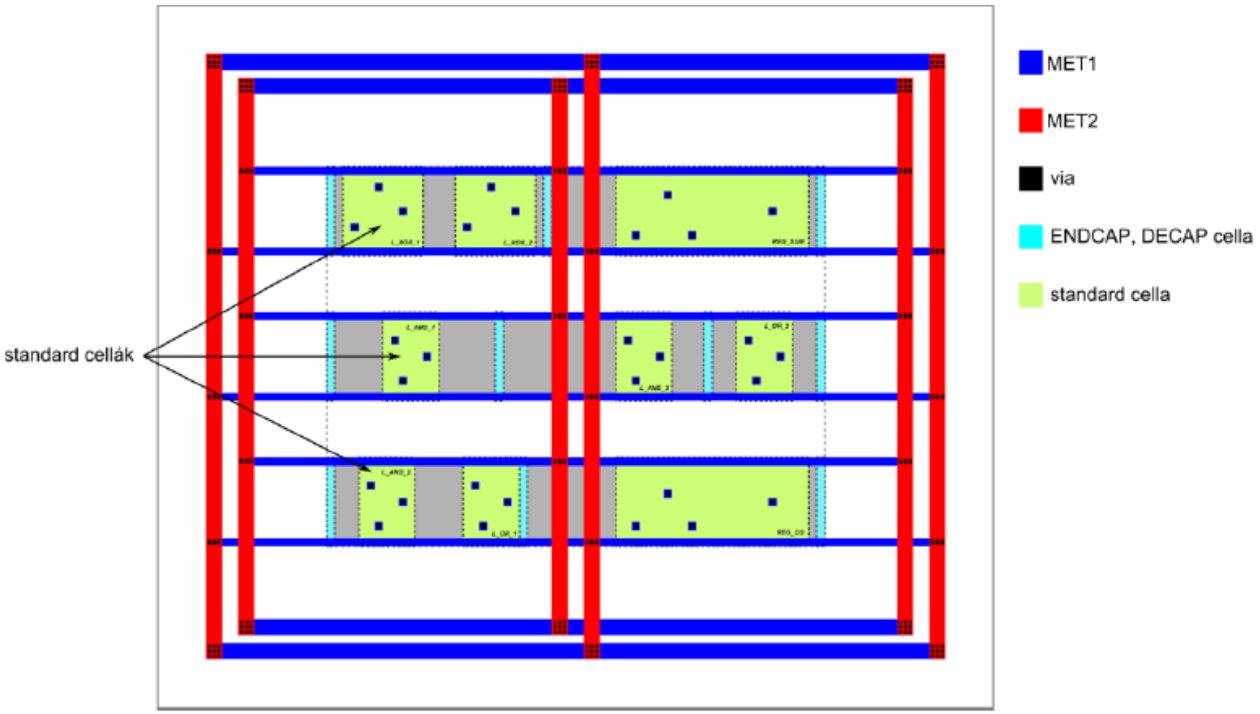
Step 2 – Power plan

- Determine the static and dynamic current consumption
  - In the case of CMOS circuits, dynamic current consumption is dominant
  - Switching activity of logic gates has to be calculated using logic simulations
- Based on these values the power supply network can be designed
  - IR drop: max. allowed voltage drop on the wire at the highest current
  - Usually 5% max.



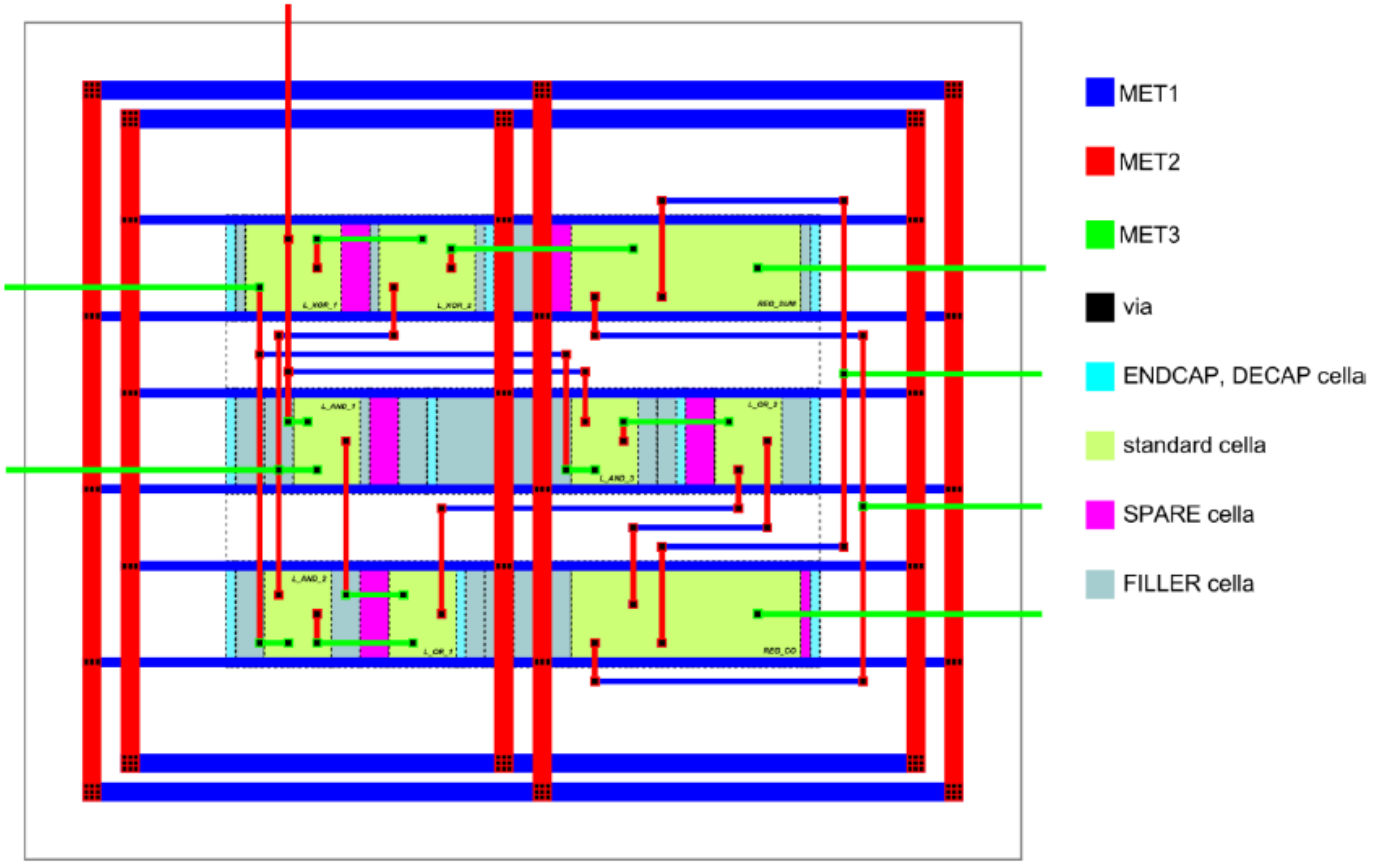
Step 3 – Placement (of the cells)

- Placement of the logic cells

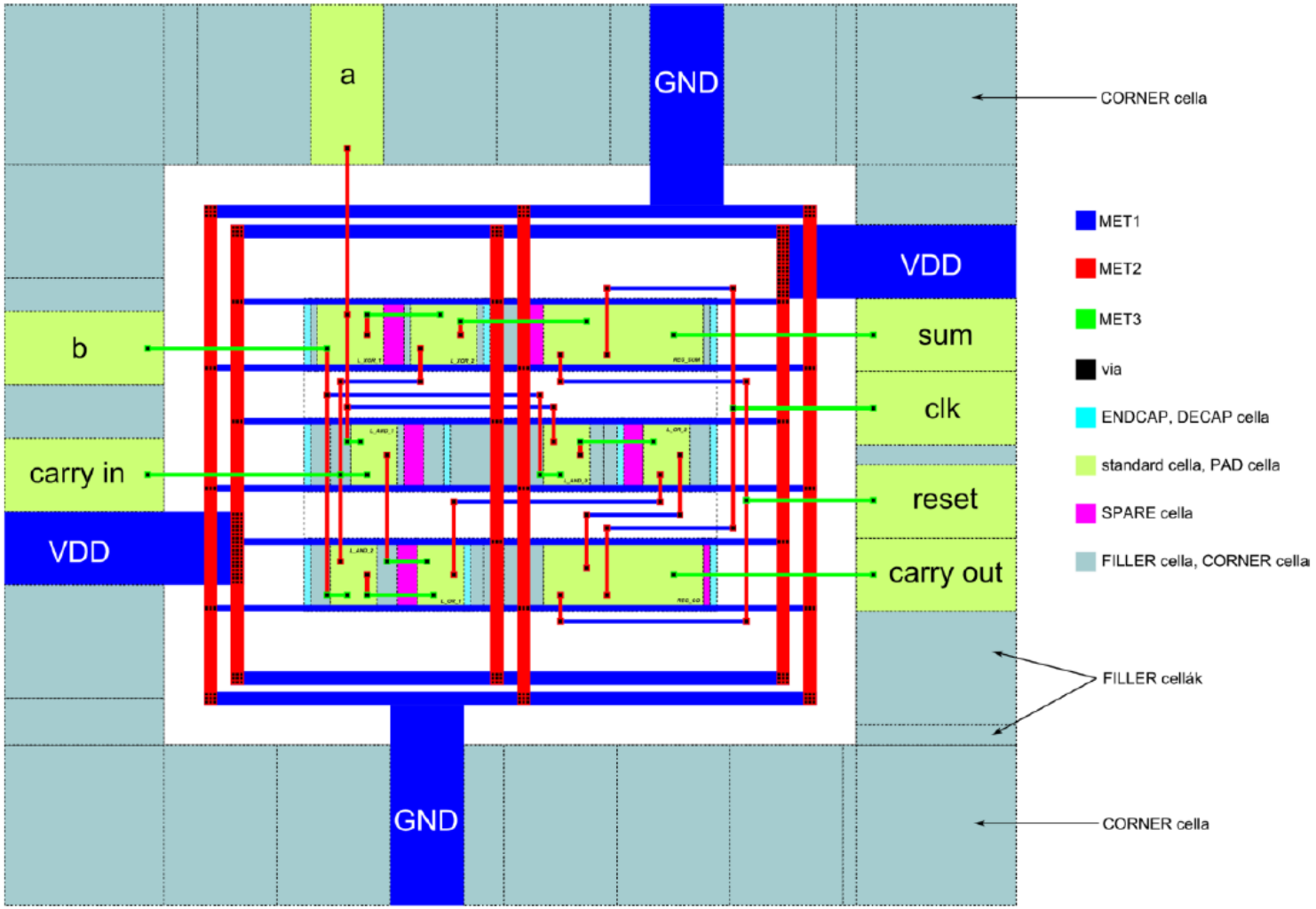


Step 4 – Route

- Creating interconnections between cells



Step 5 – Creating the pad-ring



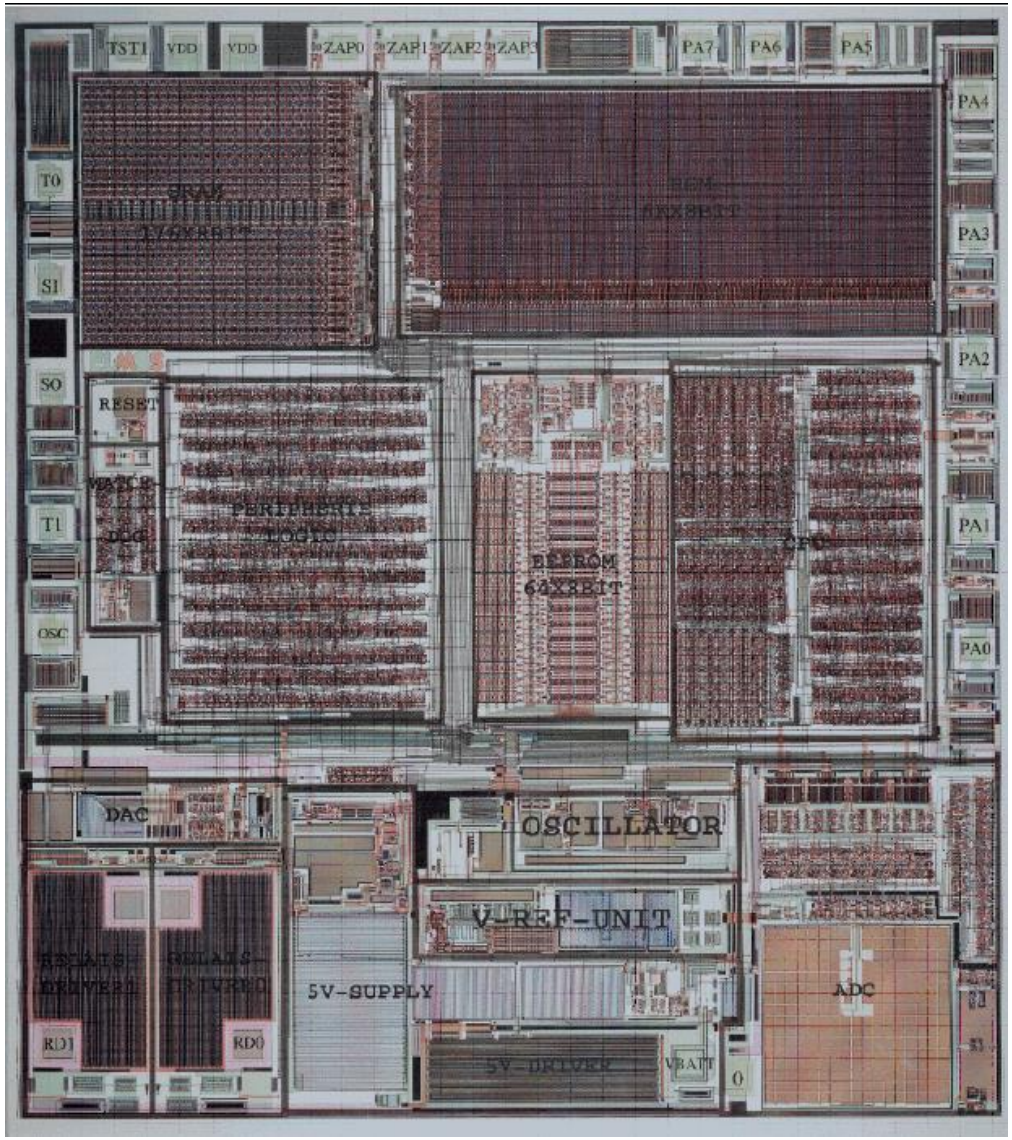
Post layout simulation

- After physical design, the output load of the gates is well-known
  - Based on the physical design (layout), the geometries and thus the logic delays may be determined.
- The exact timing information is available
  - The post-layout (timing) simulation may be performed.
  - The results should be the same as those obtained during RTL simulation.
  - In the case of any differences, the RTL model or the layout should be refined.

Semiconductor IP (Intellectual Property)

- IP core or IP block: reusable unit, created by another designer
- A subcircuit, which can be bought and used as a predefined component.
  - It is a virtual component
    - Similar to a software library.
  - The user must pay a license fee.
- Because of the complexities characteristic of today's designs, it is impossible to do everything from scratch.
  - It is worth it to buy successful IPs.
  - It is a successful business model (see ARM)

Example: SoC


TST1
VDD
VDD
ZAP0
ZAP1
ZAP2
ZAP3
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
T0
S1
SO
T1
OSC
RESET
SRAM
EEPROM
64X8BIT
CPU
DAC
OSCILLATOR
V-REF-UNIT
5V-SUPPLY
ADC
RD1
RD0
VBATT

Soft IP Core

- Synthesizable RTL model (Verilog, VHDL)
  - The source code may be encrypted. The provided synthesis tool can decrypt it using a valid license key.
- Generic netlist
  - It only includes generic logic gates.
- Both representations are independent of the underlying technology.
  - The model is portable.
- Technology optimization is performed by the user. It is a difficult task.
  - The timing and the geometries are not known, so the IP vendor cannot guarantee anything.

Hard IP Core

- The output of the physical design: the layout
- The IP is not technology-independent any more
  - The sizes, the timing, and the performance are known and guaranteed.
  - It contains hand-optimized portions.
  - Therefore, the quality (of the parameters) is often higher than those provided by soft-IPs
  - But it is technology-dependent.
    - When switching to another technology vendor, a new license must be bought.